A Project Report

FOREST FIRE PREVENTION SYSTEM

The Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Amitkumar Naidu (6427)

Prabhat Sharma (6430)

Tanishq Kundu (6496)

&

Aditya Vishwakarma (6443)

Under the esteemed Guidance of

Mrs. ARCHANA BHIDE

DEPARTMENT OF INFORMATION TECHNOLOGY

RAMNIRANJAN JHUNJHUNWALA COLLEGE (AUTONOMOUS)

(Affiliated to University of Mumbai)

GHATKOPAR (MUMBAI - 400086, MAHARASHTRA

RAMNIRANJAN JHUNJHUNWALA COLLEGE (AUTONOMOUS)

(Affiliated to University of Mumbai)

MUMBAI - MAHARASHTRA - 400086

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, "**FOREST FIRE PREVENTION SYSTEM**" is Bonafide work of **Tanishq Kundu, Amitkumar Naidu, Prabhat Sharma, & Aditya Vishwakarma** bearing **Roll No:. 6496, 6427, 6430, 6443** submitted in partial fulfillments of the requirement for the degree of **BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY** from University of Mumbai the academic year 2023-24.

_____ _____

Internal Guide **Co-coordinator**

_____

External Examiner

College Seal **Date :-** _____

ABSTRACT

The **Forest Fire Prevention System** is a groundbreaking new system that uses cutting-edge technology to detect and prevent forest fires. It combines sensors, data communication technologies, and intelligent analysis algorithms to create a comprehensive and proactive fire detection and prevention network.

The system uses devices such as Arduino Nano with GSM/SIM module 4G, GPS/GPRS modules, and environmental sensors to collect real-time data on temperature, humidity, smoke, and other important factors. This data is then sent to an IoT server, where it is processed and analysed using advanced algorithms to identify potential fire outbreaks and fire-prone areas.

The IoT server acts as the central hub for the system. It uses rule-based decision-making to automatically trigger alerts and coordinate responses to potential fire threats. The server also allows for remote monitoring of the system and integration with external systems, such as fire department dispatch systems. Data visualisation tools and reports generated by the system provide valuable insights to help decision-makers understand fire risk levels, identify trends, and optimise the system's performance.

ACKNOWLEDGEMENT

The success of this project is due to the guidance and support of many people. I am truly grateful for the opportunity to achieve this milestone. It required the dedicated efforts of every team member, and I sincerely thank everyone who contributed. This achievement is a testament to our collective commitment and hard work.

I appreciate and thank **HoD Mrs. Archana Bhide**, for granting me an opportunity to do the project activity and providing us with all support and leadership, which made me finish the project duly. I would like to give a very special honours and respect to our guide, **Prof. Prachi Surve, who** took keen interest in checking the minute details of the project work and guided us throughout the same.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance and have been helpful in various phases of the completion of the project.

INDEX

DECLARATION

I hereby declare that the project entitled, "**FOREST FIRE PREVENTION SYSTEM**" has not been in any case duplicated or submitted to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY** to be submitted as the final semester project as part of our curriculum.

Tanishq Kundu, Amitkumar Naidu, Prabhat Sharma, Aditya Vishwakarma

CHAPTER 1 - INTRODUCTION

Chapter Outline

The **Forest Fire Prevention System** is an innovative Internet of Things (IoT) project aimed at mitigating the devastating impact of forest fires. This project combines advanced sensor technology, data analytics, and real-time communication to detect, monitor, and prevent forest fires efficiently. The system is designed to provide early warnings, facilitate rapid response, and assist firefighting efforts, ultimately minimizing property damage and saving lives.

1.1] Background

Forest fires can be devastating, causing widespread damage to the environment, wildlife, and human settlements. They are often difficult to detect and can spread rapidly, making it challenging for authorities to respond promptly.

Traditional fire monitoring techniques heavily rely on human observation. This can be slow, inaccurate, and limited in coverage. Human observers may not be able to detect fires in remote areas or during poor weather conditions. Additionally, human observation is subjective and can be prone to errors.

The Forest Fire Prevention System leverages IoT technology to address the challenges of traditional fire monitoring techniques. IoT devices, such as sensors, can be deployed in remote areas and in all weather conditions. They can collect real-time data on environmental conditions, such as temperature, humidity, smoke, and wind speed. This data can then be transmitted to a central cloud-based server, where it is analysed using advanced algorithms to identify potential fire hazards.

If the system detects a potential fire hazard, it can immediately send an alert to relevant authorities and firefighting teams. This allows for a swift and effective response, which can help to minimise the damage caused by the fire.

In addition to early detection, the Forest Fire Prevention System can also provide real-time monitoring of fire incidents. This can help authorities to track the progress of fires and deploy firefighting resources more

effectively. Additionally, the system can collect valuable data on forest fire behaviour and patterns. This data can be used to improve the design and development of future forest fire prevention systems.

Here are some specific examples of how the Forest Fire Prevention System can be used to improve forest fire prevention and response:

- Early detection: The system can detect fires in remote areas and during poor weather conditions, which is difficult or impossible with traditional fire monitoring techniques.
- Real-time monitoring: The system can provide real-time monitoring of fire incidents, which can help authorities to track the progress of fires and deploy firefighting resources more effectively.
- Data collection: The system can collect valuable data on forest fire behaviour and patterns. This data can be used to improve the design and development of future forest fire prevention systems.
- Improved coordination: The system can help to improve coordination between different agencies and organisations involved in forest fire prevention and response.
- Public awareness: The system can be used to raise public awareness of forest fire risks and promote preventive measures.

1.2] Objective

The primary objectives of the Forest Fire Prevention System are as follows:

- **Early Detection**: Implement a network of smart sensors strategically placed in high-risk forest areas to detect fire incidents as early as possible.

- **Real-time Monitoring**: Establish a real-time monitoring and data collection system to track environmental conditions, such as temperature, humidity, and wind speed, to assess fire-prone conditions.

- **Prompt Alerting**: Develop an efficient alerting mechanism to notify relevant authorities, nearby residents, and firefighting teams instantly when a potential fire is detected.

- **Data Analytics**: Analyze the collected data to identify patterns, trends, and potential fire hazards to improve predictive capabilities.

- **Remote Control**: Integrate remote control capabilities to enable firefighting teams to deploy preventive measures and contain fires using automation.

1.3] Purpose and scope

1.3.1] Purpose

Forest fires are a major global problem, causing widespread environmental damage and economic losses. In recent years, the frequency and intensity of forest fires have increased due to climate change, human activity, and other factors. This has made it more important than ever to develop effective forest fire prevention and detection systems.

The Forest Fire Prevention System IoT project is a cutting-edge initiative that leverages the power of IoT technology to achieve this goal. The system uses a network of sensors to collect real-time data on environmental conditions, such as temperature, humidity, smoke, and wind speed. This data is then transmitted to a central cloud-based server, where it is analysed using advanced algorithms to identify potential fire hazards.

If the system detects a potential fire hazard, it will immediately send an alert to relevant authorities and firefighting teams. This allows for a swift and effective response, which can help to minimise the damage caused by the fire.

The Forest Fire Prevention System IoT project has the potential to make a significant impact on the global fight against forest fires. By providing early detection, real-time monitoring, and prompt alerts, the system can help to save lives, protect ecosystems, and reduce economic losses.

In addition to the benefits mentioned above, the Forest Fire Prevention System IoT project can also help to:

- Improve the efficiency and effectiveness of firefighting operations.
- Identify fire-prone areas and develop targeted prevention strategies.
- Raise public awareness of forest fire risks and promote preventive measures.
- Collect valuable data on forest fire behaviour and patterns, which can be used to improve the design and development of future forest fire prevention systems.

1.3.2] Scope

The scope of the Forest Fire Prevention System IoT project includes the following aspects:

- **Sensor Prototyping:** Design and develop IoT-based smart sensors equipped with infrared cameras, smoke detectors, and temperature sensors for early fire detection.

- **Hardware Implementation:** Assemble and integrate the developed sensors with microcontrollers and communication modules to enable data transmission.

- **Data Collection and Aggregation:** Implement a data aggregation hub to gather information from multiple sensors and organize it for further analysis.

- **Real-time Data Analytics:** Utilize data analytics techniques to process and analyze the collected data, identifying potential fire hazards and patterns of fire occurrence.

- **Alerting Mechanism:** Develop an efficient alerting system that notifies relevant authorities and users through mobile applications or email when a potential fire is detected.

- **Remote Control and Automation:** Integrate remote control capabilities to allow authorised users to activate preventive measures and automate response actions to control fires.

- **User Interface Development:** Create a user-friendly web or mobile application that enables authorities and users to access real-time data, receive alerts, and control preventive measures remotely.

- **Communication Network Setup:** Establish a reliable and secure communication network to facilitate seamless data transmission between sensors, data aggregation hub, central server, and user interfaces.

- **Testing and Validation:** Conduct thorough testing of the Forest Fire Prevention System to ensure its accuracy, responsiveness, and reliability under various scenarios and environmental conditions.

- **Documentation and Reporting:** Document the entire project development process, including system design, implementation, testing, challenges faced, and lessons learned. Prepare a comprehensive report showcasing the project's objectives, methodology, results, and future recommendations.

- **Environmental Considerations:** Address environmental concerns related to the project, such as sustainable deployment of sensors and minimizing the system's ecological impact.

- **Community Outreach:** Raise awareness about forest fire prevention through workshops, demonstrations, and educational materials, encouraging community engagement and participation.

1.3.3] Applicability

The Forest Fire Prevention System using IoT offers a wide range of applications and benefits across various sectors. The system's adaptability and effectiveness make it a valuable tool for addressing the challenges posed by forest fires. The following are the key applicability areas of the project:

- **Environmental Conservation and Protection**
  The project significantly contributes to environmental conservation and protection by proactively detecting and preventing forest fires. By minimizing the scale and impact of wildfires, the system helps preserve biodiversity, protects ecosystems, and ensures sustainable management of forests.

- **Human Safety and Property Protection**
  Implementing the Forest Fire Prevention System in residential areas and communities near forests enhances the safety of residents and protects properties from potential fire hazards. Early fire detection and rapid response enable timely evacuation and prevent loss of lives and assets.

- **Wildlife Habitat Preservation**
  Wildlife sanctuaries, conservation areas, and national parks can utilize the system to safeguard animal habitats and prevent harm to wildlife populations. Detecting and suppressing fires promptly ensures that natural habitats remain intact, allowing animals to thrive undisturbed.

- **Infrastructure Protection**
  Industries located near forested areas, such as power plants, factories, and communication infrastructure, can mitigate the risk of fire damage by implementing the system. Protecting critical infrastructure ensures the continuity of operations and reduces economic losses.

- **Remote and Inaccessible Regions**
  The Forest Fire Prevention System is particularly valuable for monitoring and safeguarding remote and inaccessible forest regions. Traditional monitoring methods may be challenging in such areas, making the IoT-based system an ideal choice for proactive fire prevention.

- **Government and Forest Management**
  Forest fire prevention is a critical concern for government and forest management authorities. Implementing the IoT project empowers them with real-time data and analytics to make informed decisions, allocate firefighting resources efficiently, and devise effective fire prevention strategies.

- **Public Awareness and Education**
  The project serves as an educational tool to raise public awareness about the importance of forest fire prevention and the impact of wildfires on the environment and communities. Public participation and support are essential for the success of forest fire prevention efforts.

1.4] Achievement

➢ The Forest Fire Prevention System project's achievement lies in its ability to:

- Early detection of potential forest fires.
- Enable proactive fire management.
- Reduce the environmental impact.
- Enhance community safety.
- Optimise resource allocation.
- Provide data for informed decision-making.
- Scale to different regions.
- Integrate advanced technologies for forest fire prevention.

1.5] Organization of Report

In the 2nd chapter - Survey of Technologies, what all technologies have been used in the project is discussed.

In the 3rd chapter - Requirements and Analysis, requirements of the topic, planning and scheduling of modules creation, software and hardware required to work within are discussed.

In the 4th chapter - System Design, modules, Data flow, UML diagrams representing relationships, Security issues and Test cases performed.

In the 5th chapter - Results and Discussions, the Programming part is discussed and what approach the code is for and the UI designs.

In the 6th chapter - Conclusion, what's the significance, limitations, and future scope of the system and project.

In the 7th chapter - Reference, list of all the sources and references that have been consulted or refered during the development of our understanding or implementation of the project.

CHAPTER 2 - SURVEY OF TECHNOLOGY

➢ **Arduino IDE:**

- The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macros, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino-compatible boards.

- Arduino IDE is an open-source software that is mainly used for writing and compiling the code into the Arduino Module.

- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.

- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded to the controller on the board.

Writing sketches:

- Programs written using Arduino IDE are called sketches. These sketches are written in the text editor and are saved with the file extension '.ion'. The editor has features for cutting/pasting and searching/replacing text.

- The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino IDE., including complete error messages and other information. The bottom right corner of the window displays the configured board and serial port.

➢ **Alternative Technologies: -**

1. Raspberry Pi: -

- The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.

- The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards and mice) or cases.

However, some accessories have been included in several official and unofficial bundles.

➢ **Advantages of Arduino over Raspberry Pi:**

- **Simplicity:**

It's very easy to interface analogue sensors, motors and other electronic components with Arduino, with

just a few lines of code. While in Raspberry Pi, there is much overhead for simply reading those sensors, we need to install some libraries and software for interfacing these sensors and components.

- **Robustness:**

Raspberry Pi runs on an OS so it must be properly shut down before turning OFF the power, otherwise OS & applications may get corrupted and Pi can be damaged. Arduino is just a plug-and-play device which can be turned ON and OFF at any point in time, without any risk of damage. It can start running the code again upon resuming the power.

- **Power consumption: -**

Pi is powerful hardware, it needs a continuous 5v power supply and it is difficult to run it on Batteries, while Arduino needs less power and can easily be powered using a battery pack.

- **Price:-**

Obviously, Arduino is cheaper than Raspberry Pi, Arduino costs around $10-20 depending on the version, while the price of Raspberry is around $35-40.

➢ **Operating System:-**

- For many embedded applications, it is obvious that an OS is needed. If the application is complex and is running on a high-end processor, it is almost certain that an OS would be beneficial. At the other end of the scale, simple software running on a low-end chip has no need for an OS at all.

- A microcontroller will run just one program repeatedly — not a full operating system. This flexibility combined with the fact that the Arduino software is free, The hardware boards are pretty cheap, and both the software and hardware are easy to learn.

- Microcontrollers also don't have the same amount of computing power or resources as most single-board computers.

- Arduino is a micro-controller board which runs a dedicated program, there's no OS, just your code.

CHAPTER 3 - REQUIREMENT AND ANALYSIS

Chapter Outline

3.1] Problem Definition

3.2] Functional Requirement

3.3] Non-Functional Requirements

3.4] Planning and Scheduling

3.5] Software and Hardware Requirements

3.6] Conceptual UML Models (Use case, Class diagram, Sequence diagram)

3.1] Problem Definition

The primary challenge is to develop a comprehensive IoT-based solution that can reliably and effectively prevent forest fires through early detection and rapid response. This involves addressing several key issues:

1. **Early Detection:** Current forest fire detection methods often suffer from delays, leading to the fires spreading before appropriate measures are taken. The system needs to identify potential fire outbreaks in their earliest stages, reducing response times.

2. **Remote Monitoring:** Traditional monitoring methods lack real-time remote capabilities. The system should enable continuous monitoring of remote forest areas, providing accurate data for timely decision-making.

3. **Data Accuracy:** The system must differentiate between normal environmental fluctuations (e.g., temperature changes) and actual fire events, ensuring a low false-positive rate to avoid unnecessary alarm triggers.

4. **Scalability:** The solution should be scalable, capable of covering large forested areas with a network of interconnected IoT devices without overwhelming maintenance demands.

5. **Energy Efficiency:** IoT devices deployed in remote locations should operate with minimal energy consumption, maximising their operational lifespan and minimising environmental impact.

6. **Interconnectivity:** The system should establish seamless communication between various components such as sensors, data processing units, communication modules, and the central monitoring station.

3.2] Functional Requirement

Characteristic	Priority	Requirement
User Authentication	High	Users must be logged in to the system, using secure and encrypted methods.
Real-time Data Collection	High	The system shall continuously collect environmental data (temperature, humidity, smoke levels, etc.) from IoT sensors in forested areas.
Data Processing	High	Collected data must be processed in real-time to detect anomalies and potential fire events.
Alarm Generation	High	If fire-related anomalies are detected, the system shall trigger alarms at the central monitoring station.
Alert Notification	High	The system shall send real-time alerts via SMS, email, or mobile app notifications to relevant authorities and firefighting teams.
Remote Monitoring	Medium	Users should be able to remotely monitor the status of the forested areas through a web or mobile interface.
Data Visualization	Medium	The system should provide graphical representations of historical and real-time data for analysis.

Automated Response	High	In the event of a confirmed fire, the system shall activate water sprinklers and alert nearby firefighting teams.
Expandability	Medium	The architecture should allow for easy integration of additional IoT devices and sensors in the future.
Energy Efficiency	Medium	IoT devices should be designed to minimise energy consumption while ensuring consistent data collection.
User Roles and Permissions	Medium	Different user roles (admin, firefighter, observer) should have varying levels of access and control.
Historical Data Storage	Low	The system should store historical data for analysis and reporting purposes.(Specifically for App,Website)
User-Friendly Interface	Medium	The web and mobile interfaces should be intuitive and user-friendly for efficient system management.(Specifically for App,Website)

3.3] Non-Functional Requirement

Characteristic	Priority	Requirement
Security and Privacy	High	All communication between IoT devices, the central monitoring station, and user interfaces must be encrypted and secure.
Scalability	High	The system should support the addition of new IoT devices and handle increased data load without performance degradation.
Reliability and Availability	High	The system should have a high uptime, with minimal downtime for maintenance or repairs.
Response Time	High	Alarms and alerts should be generated and sent within seconds of detecting a potential fire event.
Accuracy	High	The system's fire detection algorithms must have a high accuracy rate, with minimal false positives and false negatives.
Interoperability	Medium	The system should be compatible with various types of IoT sensors and communication protocols.
Data Storage and Retention	Medium	Historical data should be stored for a specified period for analysis and reporting purposes.

User Experience	Medium	User interfaces should be responsive and provide a smooth and intuitive experience for users.
Energy Efficiency	High	IoT devices should be designed to maximise energy efficiency, extending their operational lifespan.
Regulatory Compliance	High	The system must adhere to relevant environmental, data protection, and safety regulations.
Disaster Recovery	Medium	A robust disaster recovery plan should be in place to minimise data loss and ensure system continuity in case of failures.
Maintenance and Support	Medium	The system should be easy to maintain, and support resources should be available to address issues promptly.
Data Backup and Redundancy	Medium	Data should be regularly backed up, and redundant systems should be in place to prevent data loss.
Accessibility	Low	The user interfaces should follow accessibility guidelines to ensure usability by individuals with disabilities.
Documentation	Medium	Comprehensive documentation should be provided for system installation, configuration, and usage.
Performance	High	The system should handle data processing and communication with low latency, even during peak usage periods.

3.4] Planning and Scheduling

Planning and scheduling is a complicated part of software development.

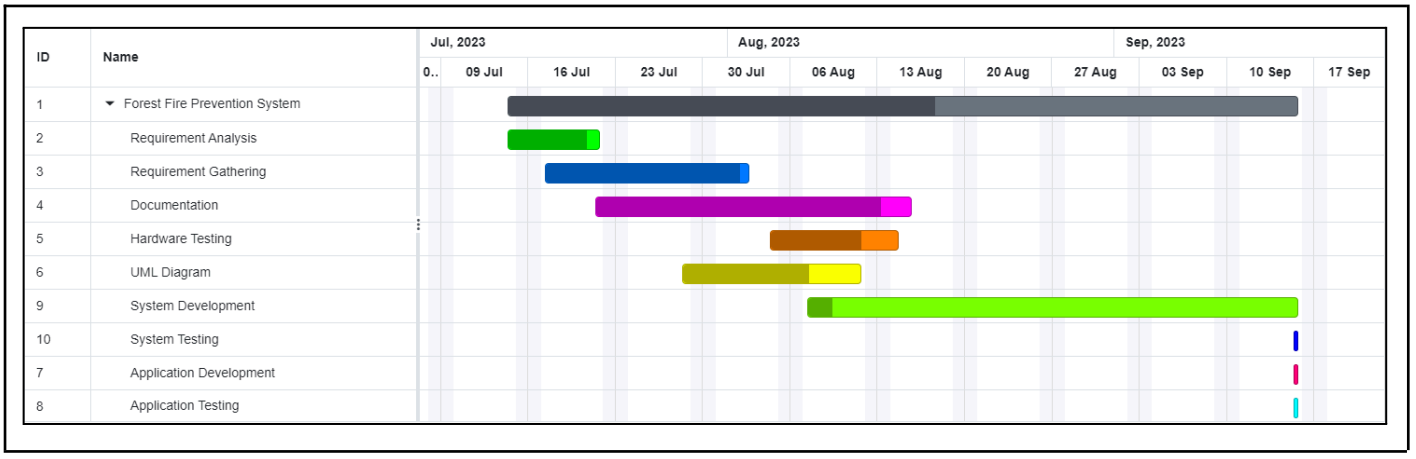
Planning: Planning, can be thought as determining all the small tasks that must be carried out in order to accomplish the goal. Planning also takes into account, rules known as constraints, which, control when certain tasks can or cannot happen.

Scheduling: Scheduling can be thought of as determining whether adequate resources are available to carry out the plan. For how much time a function should be given time so that it can be created properly.

A project with no starting and finishing date is less likely to succeed objective of developing our project schedule are:

1. To identify project activities that need to be completed.
2. To estimate the duration of the project activities.
3. To determine the sequence and time when the project activities will need to happen.
4. To monitor and control the project schedule.

➢ **Gantt chart:**



ID	Name	Start Date	End Date	Duration	Progress %
1		Jul 14, 2023	Sep 15, 2023	46 days	54
2	Requirement Analysis	Jul 14, 2023	Jul 21, 2023	6 days	85
3	Requirement Gathering	Jul 17, 2023	Aug 02, 2023	13 days	95
4	Documentation	Jul 21, 2023	Aug 15, 2023	18 days	90
5	Hardware Testing	Aug 04, 2023	Aug 14, 2023	7 days	70
6	UML Diagram	Jul 28, 2023	Aug 11, 2023	11 days	70
9	System Development	Aug 07, 2023	Sep 15, 2023	30 days	5
10	System Testing	Sep 15, 2023	Sep 15, 2023	1 day	0
7	Application Development	Sep 15, 2023	Sep 15, 2023	1 day	0
8	Application Testing	Sep 15, 2023	Sep 15, 2023	1 day	0

3.5] Software and Hardware Requirements

➢ **Softwares:**

1. Arduino IDE

The Arduino IDE (Integrated Development Environment) is a software platform designed to simplify the programming and development of applications for Arduino microcontroller boards. It provides a user-friendly interface for writing, uploading, and debugging code for Arduino projects. The Arduino IDE supports the C/C++ programming languages and offers a wide range of libraries and example codes to facilitate the development of various electronic projects. It's a versatile tool used by makers, hobbyists, and professionals for prototyping and creating interactive electronic devices and systems.

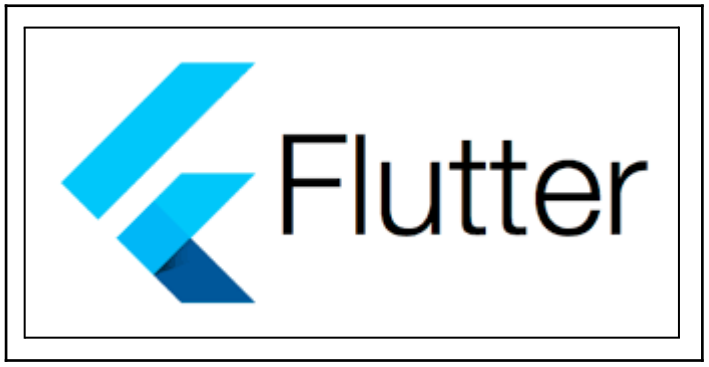
- **Image**

- **Specification:**
    1. Arduino Uno ATmega328P – 8-bit AVR family microcontroller, Operating Voltage 6-20V, DC Current on I/O Pin – 40ma
    2. Development Platform**:** IA-32, x86-64, AR
    3. Language Used**:** Arduino C++
    4. Code development**:** Arduino IDE

2. Flutter

Flutter is a user-friendly, open-source UI software development kit created by Google, enabling developers to build natively compiled applications for mobile, web, and desktop from a single codebase. It utilises a reactive framework and a rich set of customizable widgets, allowing for fast development and expressive user interfaces. With its hot reload feature, developers can instantly see changes in their code reflected in the app, facilitating rapid iteration and experimentation. Flutter's comprehensive documentation and vibrant community make it an ideal choice for developers seeking efficiency and flexibility in app development across multiple platforms.

- **Image:**



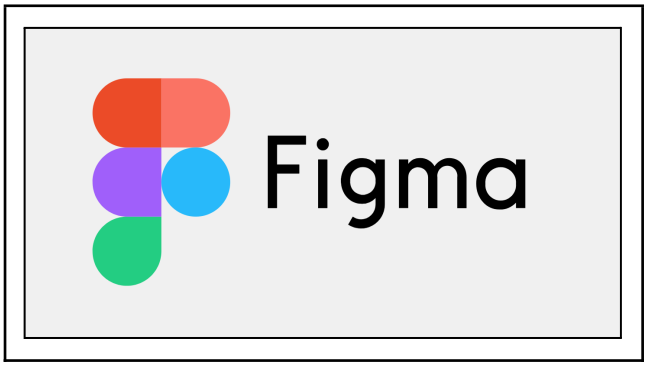
- **Specification:**

    - In the context of Flutter, developers can leverage various tools and technologies to connect apps to IoT devices, including Bluetooth, Wi-Fi, and cloud platforms.
    - Using Flutter, developers can create an app interface to display data from IoT devices, such as temperature and humidity readings, by integrating appropriate communication protocols and APIs.
    - This allows for seamless interaction between the app and IoT devices, enabling users to monitor and control connected systems efficiently.

3. Figma

Figma is a collaborative interface design tool used to create visually appealing and interactive app prototypes. It allows users to design app interfaces, including those for IoT devices, with ease and precision. With Figma, users can create intuitive layouts and incorporate elements to display data such as temperature and humidity readings. Its collaborative features enable teams to work together seamlessly, streamlining the design process for projects.

- **Image:**



- **Specification:**

1. Integration Capabilities: Developers can integrate IoT device data display into app interfaces designed in Figma by incorporating appropriate UI components and interactive elements.

2. Prototyping: Figma allows developers to create interactive prototypes of IoT data display interfaces, enabling stakeholders to visualise the user experience and functionality before implementation.

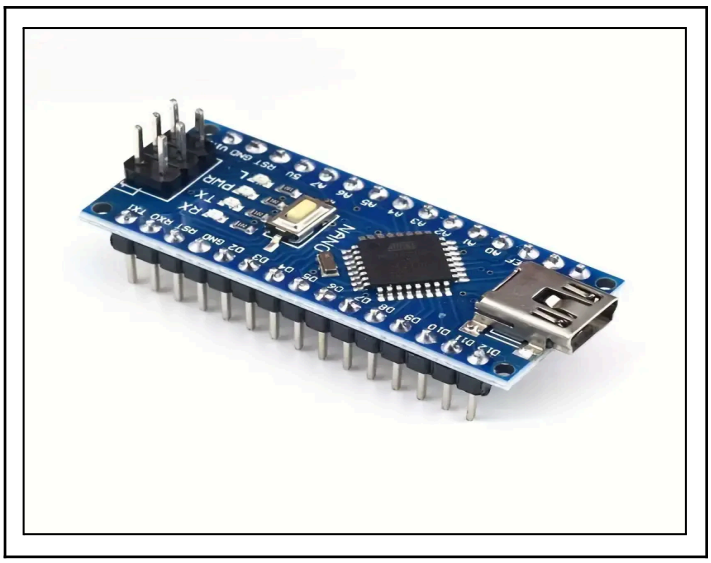
3. Design Consistency: Figma's component libraries and design systems ensure consistency in the design of interfaces for displaying IoT data across multiple screens and platforms.

➢ **Hardware Components:**

1. Arduino Nano:

The Arduino Nano is a compact and versatile microcontroller board based on the ATmega328P microcontroller chip. It offers similar functionalities to the Arduino Uno but in a smaller form factor, making it ideal for projects with space constraints or where portability is important. The Nano features a USB interface for programming and power supply, digital and analog pins for connecting sensors, actuators, and other components, and onboard voltage regulation for stable operation. It's widely used in various DIY electronics projects, robotics, prototyping, and educational activities due to its ease of use, affordability, and compatibility with the Arduino ecosystem.

- **Image:**



- **Specification:**

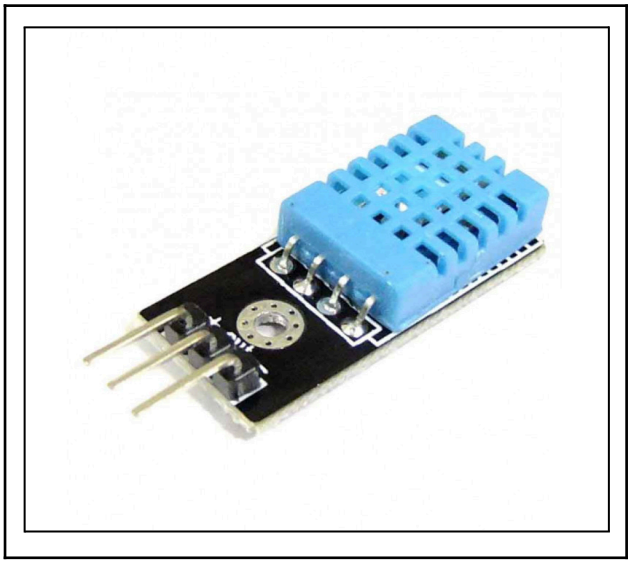
1. Microcontroller: ATmega328P
2. Operating Voltage: 5V
3. Input Voltage (recommended): 7-12V

4. Digital I/O Pins: 14 (including 6 PWM outputs)
5. Analog Input Pins: 8
6. DC Current per I/O Pin: 40 mA
7. Flash Memory: 32 KB (ATmega328P) of which 2 KB used by bootloader
8. SRAM: 2 KB (ATmega328P)
9. EEPROM: 1 KB (ATmega328P)
10. Clock Speed: 16 MHz
11. USB Interface: CH340G or Atmega16U2 (depending on the version)
12. Dimensions: Approximately 45mm x 18mm
13. Operating Temperature: -40°C to +85°C
14. Power Consumption: 19mA (idle), 500mA (maximum)
15. Input Voltage (limits): 6-20V

2. DHT11 (Temperature/Humidity Sensor):

The DHT11 is a basic, low-cost sensor that measures temperature and humidity levels in the surrounding environment. It provides accurate and real-time data, making it a popular choice for various applications, including weather monitoring, home automation, and climate control systems. The sensor is easy to use and is suitable for both hobbyist and professional projects, offering a simple digital output for temperature and humidity readings.

- **Image:**



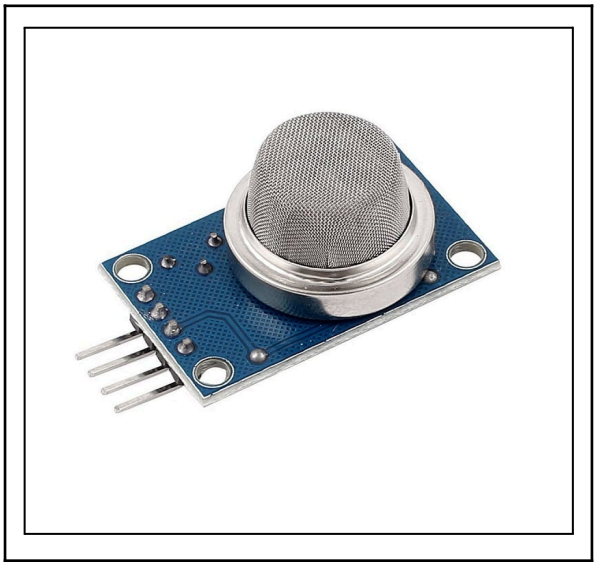
- **Specification:**

1. Measurement: Measures both temperature and humidity.
2. Accuracy: Basic accuracy suitable for general use.
3. Output: Digital output for easy integration.
4. Operating Voltage: Typically 3.3V to 5V.
5. Measurement Range: Temperature from 0°C to 50°C, Humidity from 20% to 80%.
6. Compact Size: Small and easy to integrate into various projects.
7. Cost: Low-cost and widely available.

3. **MQ-7 Gas Sensor:**

A gas sensor is a device designed to detect and measure the concentration of specific gases in the surrounding environment. These sensors are crucial for various applications, including air quality monitoring, industrial safety, and gas leak detection. They work by detecting changes in electrical conductivity, chemical reactions, or other physical properties when exposed to target gases. Gas sensors play a vital role in ensuring safety and environmental monitoring in both residential and industrial settings.

- **Image:**



- **Specification:**

1. Gas Detection: Detects specific gases like methane, carbon dioxide, or carbon monoxide.
2. Sensitivity: Measures gas concentration in parts per million (ppm) or percentage (%).
3. Detection Method: Utilises chemical reactions or physical properties.
4. Output: Typically provides analog or digital signals.
5. Operating Voltage: Usually 3.3V to 5V.
6. Response Time: Quick response to gas presence.
7. Measurement Range: Specifies the range of gas concentrations it can detect.
8. Applications: Used in gas leak detection, air quality monitoring, and industrial safety.
9. Calibration: May require periodic calibration for accuracy.
10. Size: Compact and suitable for integration into various systems.

4. **GSM/SIM module 4G**

A GSM module works by connecting to the GSM network through a SIM card. The SIM card provides the module with a unique identification number, which is used to identify the device on the network. The GSM module then communicates with the network using a set of protocols, which allows it to send and receive data.

- **Image:**

- **Specification:**

1. Communication: GSM (Global System for Mobile Communications).
2. Function: Enables mobile network connectivity for data and messaging.
3. Antenna: Typically includes an antenna for signal reception.
4. Operating Voltage: Usual range is 3.3V to 5V.
5. SIM Card: Requires a standard SIM card for network access.
6. Interface: Commonly interfaces with microcontrollers via UART.
7. Data Transmission: Supports SMS, GPRS, and sometimes calls.
8. Size: Compact and suitable for integration into various projects.
9. Applications: Used in IoT, tracking devices, and remote monitoring.

5. Flame Sensor

A flame sensor is a small electronic device designed to detect the presence of an open flame or fire. It typically works by sensing the infrared radiation emitted by flames. When a flame is detected, the sensor sends a signal, making it useful for applications such as fire safety systems, gas appliances, and industrial processes where the presence of flames needs to be monitored for safety and control purposes.

- **Image:**

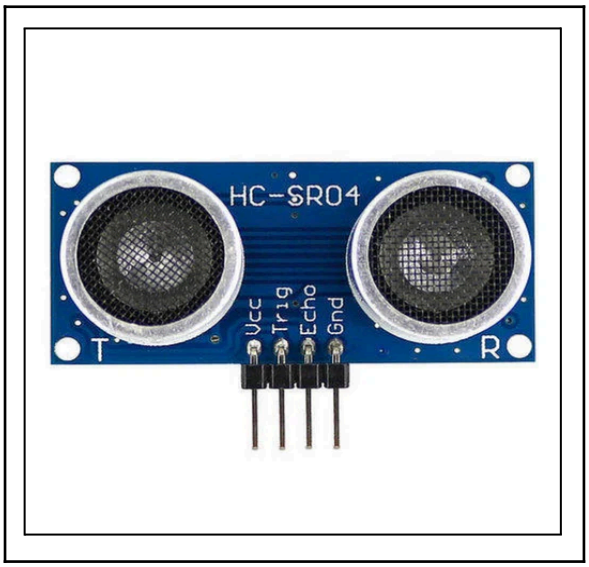
- **Specification:**

1. Detection Method: Infrared (IR) or Ultraviolet (UV) flame detection.
2. Sensitivity: Adjustable for varying flame intensities.
3. Response Time: Typically in milliseconds.
4. Operating Voltage: Commonly 3.3V to 5V.
5. Output Type: Digital or analog signal (relay, transistor, voltage).
6. Detection Range: Varies but often within a few metres.
7. Operating Temperature: -20°C to +85°C.
8. Dimensions: Compact form factor for easy integration.
9. Certifications: May have UL, CE, or other safety certifications.

6. Sonar Sensor:

A sonar sensor, short for "Sound Navigation and Ranging," is a device that uses sound waves to measure distances between the sensor and objects in its vicinity. It emits high-frequency sound pulses and calculates the time it takes for these pulses to bounce off objects and return. This data is then used to determine the distance to the objects, making sonar sensors valuable for applications such as obstacle detection, navigation, and measurement in various fields, including robotics and underwater exploration.

- **Image:**



- **Specification:**

1. Principle: Utilises sound waves for distance measurement.
2. Operating Range: Determines distances by measuring sound wave travel time.
3. Applications: Valuable for obstacle detection, navigation, and measurement.
4. Frequency: Emits and receives high-frequency sound pulses.
5. Use Cases: Commonly used in robotics, underwater exploration, and more.
6. Accuracy: Provides accurate distance measurements in real-time.
7. Compact: Typically small in size for easy integration into various devices.
8. Versatile: Suitable for a wide range of applications requiring distance sensing.

7. Jumper WIres:

Jumper wires are simple, flexible electrical wires with connectors at both ends, typically used for connecting electronic components on a breadboard or between various points on a circuit. They provide a convenient and removable way to establish electrical connections in prototyping and electronics projects, allowing for quick and flexible experimentation and circuit design without the need for soldering. Jumper wires come in various lengths and colours, making it easy to organise and distinguish connections in a project.

- **Image:**

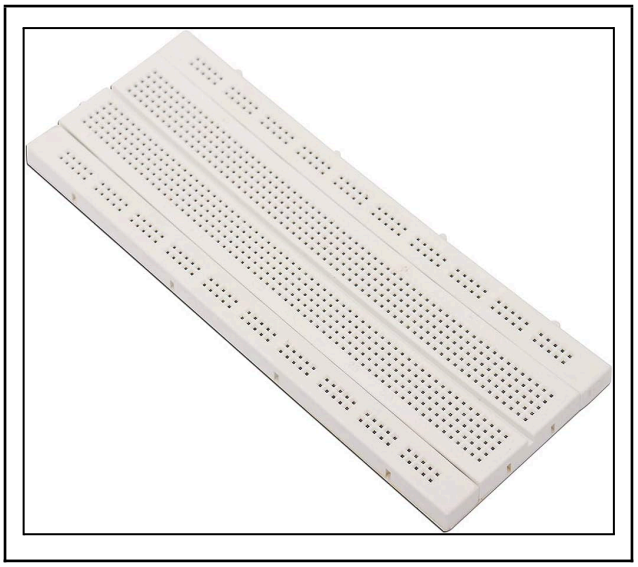
- **Specification:**

1. Wire Type: Stranded or solid-core copper wire.
2. Connector Type: Male-to-male, male-to-female, or female-to-female connectors.
3. Length: Available in various lengths, commonly 3, 6, 12, or 20 inches.
4. Wire Gauge: Typically 22 AWG (American Wire Gauge) or similar.
5. Colour Coding: Multiple colours for easy identification and organisation.
6. Material: Insulated with PVC or similar material for electrical safety.
7. Flexibility: Flexible and easy to bend for convenient connections.
8. Use: Designed for prototyping, breadboarding, and temporary circuit connections.
9. Compatibility: Compatible with common electronic components and breadboards.

8. Bread Board:

A breadboard is a reusable, solderless electronic prototyping tool used for creating and testing electrical circuits. It consists of a grid of holes, typically arranged in rows and columns, where electronic components and wires can be inserted to build and experiment with circuits. Breadboards are invaluable for rapid prototyping and experimentation in electronics, allowing components to be easily connected and rearranged without the need for soldering. They are a staple tool for both beginners and experienced engineers and are widely used in electronics education and development.

- **Image:**



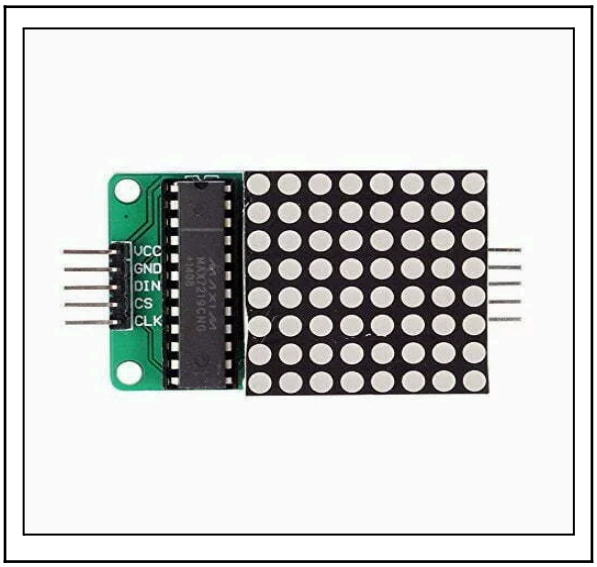
- **Specification:**

1. Layout: A grid of holes arranged in rows and columns for component placement.
2. Contact Material: Metal clips or spring-loaded contacts for electrical connections.
3. Size: Commonly available in various sizes, including mini (170 or 400 tie-points), half-sized (830 or 840 tie-points), and full-sized (1,200 or 1,360 tie-points).
4. Material: Often made from plastic with a self-healing property to withstand multiple insertions.
5. Binding Posts: Some breadboards have binding posts for external connections.
6. Bus Strips: Typically feature bus strips for power and ground connections.
7. Hole Diameter: Standard hole diameter to accommodate various electronic components.
8. Reusable: Solderless design allows for component reusability and experimentation.
9. Applications: Ideal for prototyping and testing electronic circuits.
10. Compatibility: Designed for use with common electronic components, including resistors, capacitors, and integrated circuits.

9. Dot Matrix Board:

A dot matrix board is a display or input device made up of a grid of tiny individually controllable LEDs or similar elements. It's used for various applications, including text displays, scrolling messages, and simple graphics. These boards are versatile and customizable, making them suitable for electronic signs, information panels, and DIY projects.

- **Image:**



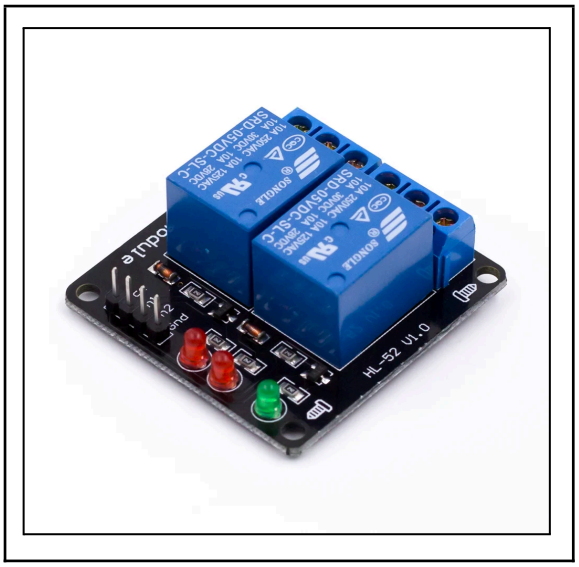
- **Specification:**

1. Matrix Size: Varies, typically available in sizes like 8x8, 16x16, or larger.
2. LED Type: Commonly uses single-colour LEDs (e.g., red, green, or yellow).
3. Resolution: Determined by the matrix size, e.g., 64 LEDs in an 8x8 matrix.
4. Control: Controlled via a microcontroller or dedicated driver ICs.
5. Display Content: Capable of displaying text, symbols, and simple graphics.
6. Interface: Connects to a microcontroller or computer via standard interfaces like SPI, I2C, or UART.
7. Brightness: Adjustable brightness levels for LEDs.
8. Colour Options: Some models offer multi-color or RGB LEDs for enhanced display versatility.
9. Power Supply: Typically 5V DC, but can vary.
10. Applications: Used in electronic signs, message displays, clocks, and DIY projects.
11. Customization: Allows users to program and customise displayed content.
12. Mounting: Designed for easy integration into various projects and enclosures.
13. Compatibility: Compatible with common microcontrollers and programming languages.

10. Relays:

A relay is an electromagnetic switch that controls the flow of electricity in an electrical circuit. It uses a low-voltage signal to control a high-voltage circuit, allowing it to isolate or amplify electrical signals. Relays are commonly used for various purposes, including switching high-power devices, protecting sensitive electronics, and enabling remote control in industrial, automotive, and electronic applications. They are essential components for automating processes, ensuring safety, and managing electrical connections in a wide range of systems.

- **Image:**



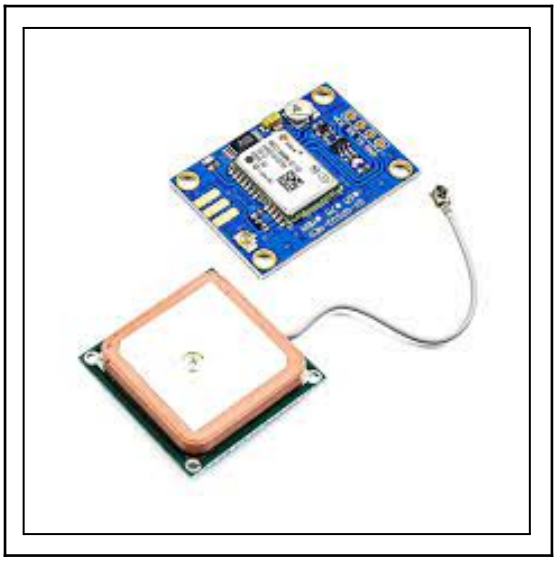
- **Specification:**

1. Type: Electromagnetic switch.
2. Contact Rating: Specifies the maximum current and voltage it can handle.
3. Coil Voltage: The voltage required to activate the relay.
4. Switching Speed: Indicates how quickly it can open or close the circuit.
5. Contact Configuration: Defines the number and type of contacts (e.g., SPST, SPDT).
6. Mounting: Options for PCB, socket, or panel mounting.
7. Lifecycle: Specifies the number of mechanical and electrical operations it can endure.
8. Applications: Used for switching high-power devices, automation, and safety circuits.
9. Size: Available in various sizes, such as miniature and power relays.
10. Operating Temperature: Specifies the temperature range for reliable operation.
11. Protection: May include features like diode suppression for coil protection.
12. Control Voltage: The voltage required to activate the relay coil.
13. Dimensions: Dimensions of the relay, including pin spacing.
14. Contact Material: Material used for the relay's switching contacts.

11. GPS module -

A GPS (Global Positioning System) module is a compact electronic device that receives signals from satellites orbiting the Earth to determine the device's precise location. It typically consists of a receiver chip, antenna, and associated circuitry. The module calculates latitude, longitude, altitude, and sometimes speed and time, providing accurate positioning information. GPS modules are commonly used in various applications such as navigation systems, asset tracking, vehicle tracking, and location-based services. They interface with microcontrollers or other devices, providing real-time location data for integration into applications. With their small form factor and low power consumption, GPS modules offer reliable and efficient positioning solutions for a wide range of applications.

Image -



Specification -

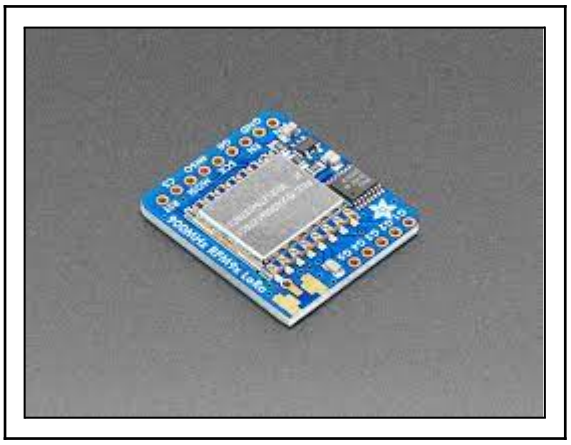
1. GPS Chipset: Incorporates a high-quality GPS chipset for accurate positioning and navigation.
2. Operating Voltage: Typically operates within a range of 3.3V to 5V.
3. Communication Interface: Supports standard communication protocols such as UART (Universal Asynchronous Receiver-Transmitter) for interfacing with microcontrollers or other devices.
4. Antenna: Includes an integrated or external antenna for receiving signals from GPS satellites.
5. Positioning Accuracy: Provides positioning accuracy typically within a few metres under optimal conditions.
6. Update Rate: Offers a configurable update rate for obtaining location data at desired intervals.
7. Protocols Supported: Compatible with standard GPS protocols such as NMEA (National Marine Electronics Association) for data output.
8. Sensitivity: High sensitivity for acquiring and tracking satellite signals even in challenging environments.
9. Form Factor: Compact module design for easy integration into various electronic devices or systems.
10. Dimensions: Typically measures approximately [specify dimensions].
11. Operating Temperature: Designed to operate reliably within a wide temperature range, typically from -40°C to +85°C.
12. Power Consumption: Low power consumption for efficient operation, typically [specify mA or µA] during normal operation.
13. Input Voltage (Limits): Accepts input voltages within a specified range, typically [specify voltage range].

12. RMF95W (LoRa) -

The RMF95W LoRa module is a wireless communication device that utilises the LoRa (Long Range) modulation technique for long-distance, low-power communication in Internet of Things (IoT) applications. It operates in the ISM (Industrial, Scientific, and Medical) frequency bands and is commonly used for remote

sensor monitoring, asset tracking, and smart city deployments. The RMF95W module features compact size, low power consumption, and high sensitivity, making it suitable for battery-operated devices and applications requiring extended range communication.

Image -



Specification -

1. Frequency Range: Supports various frequency bands such as 433MHz, 868MHz, or 915MHz, depending on region and regulatory requirements.
2. Modulation Technique: Utilises LoRa modulation for robust long-range communication, providing reliable connectivity even in challenging environments.
3. Communication Interface: Offers UART (Universal Asynchronous Receiver-Transmitter) or SPI (Serial Peripheral Interface) for interfacing with microcontrollers or other devices.
4. Transmit Power: Adjustable transmit power levels to optimise range and power consumption for different applications.
5. Sensitivity: High sensitivity receiver for receiving weak signals, enabling long-range communication even in noisy environments.
6. Data Rate: Supports configurable data rates, allowing for flexible trade-offs between communication range and throughput.
7. Encryption: Provides optional encryption features for secure communication between devices, ensuring data privacy and integrity.
8. Operating Voltage: Typically operates within a range of 3.3V to 5V, compatible with common power sources and microcontroller platforms.
9. Power Consumption: Low power consumption for extended battery life in battery-operated devices, ideal for remote and IoT applications.
10. Dimensions: Compact module size for easy integration into electronic devices or systems.
11. Operating Temperature: Designed to operate reliably within a wide temperature range, suitable for outdoor and industrial environments.

13. 18 Watt Water Lifting Submersible Pump for Aquarium, Fountains - 180V-230V

The 18 Watt Water Lifting Submersible Pump is a compact and efficient pump designed for use in aquariums, fountains, and other water features. It operates submerged in water, lifting water from the bottom and

circulating it to create aeration, filtration, or decorative effects.

Image -

Specification -

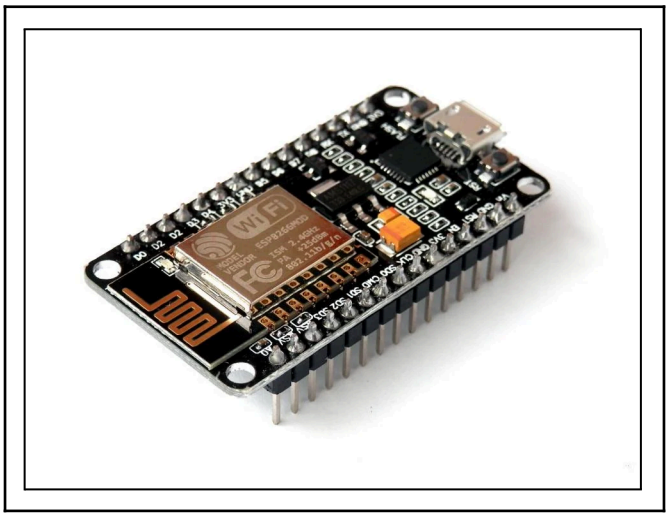
1. Power Consumption: 18 Watts, providing efficient water circulation while minimising energy consumption.
2. Voltage Compatibility: Operates within a voltage range of 180V to 230V, suitable for standard electrical systems.
3. Submersible Design: Designed to operate submerged underwater, allowing for versatile placement within aquariums, fountains, or other water features.
4. Flow Rate: Provides a specific flow rate, typically measured in litres per hour (LPH) or gallons per hour (GPH), indicating the volume of water circulated per unit of time.
5. Maximum Head Height: Specifies the maximum vertical distance the pump can lift water, commonly measured in metres (m) or feet (ft), indicating its suitability for different water feature designs.
6. Outlet Size: Features a specific outlet size, typically measured in millimetres (mm) or inches (in), determining compatibility with tubing or fittings for water distribution.
7. Material: Constructed from durable and corrosion-resistant materials suitable for continuous submersion in water, ensuring long-term reliability and performance.
8. Noise Level: Produces minimal noise during operation, maintaining a peaceful environment in aquariums or indoor spaces.
9. Safety Features: Includes built-in safety features such as overheat protection or automatic shut-off mechanisms to prevent damage or accidents.
10. Application: Suitable for various applications, including aquariums, fountains, hydroponic systems, and water features in gardens or landscapes.

14. Node MCU - Wifi module

The NodeMCU is a development board that utilises the ESP8266 Wi-Fi module, which is a low-cost, highly integrated chip designed for IoT applications. It combines a microcontroller with Wi-Fi capabilities, enabling easy connectivity to the internet for various projects. NodeMCU boards are popular among hobbyists and professionals alike due to their simplicity, affordability, and versatility. They support programming in Lua or using the Arduino IDE, making them accessible to a wide range of developers. With built-in Wi-Fi

functionality, NodeMCU boards can be used for tasks such as sensor data collection, remote control systems, and IoT applications.

Image :

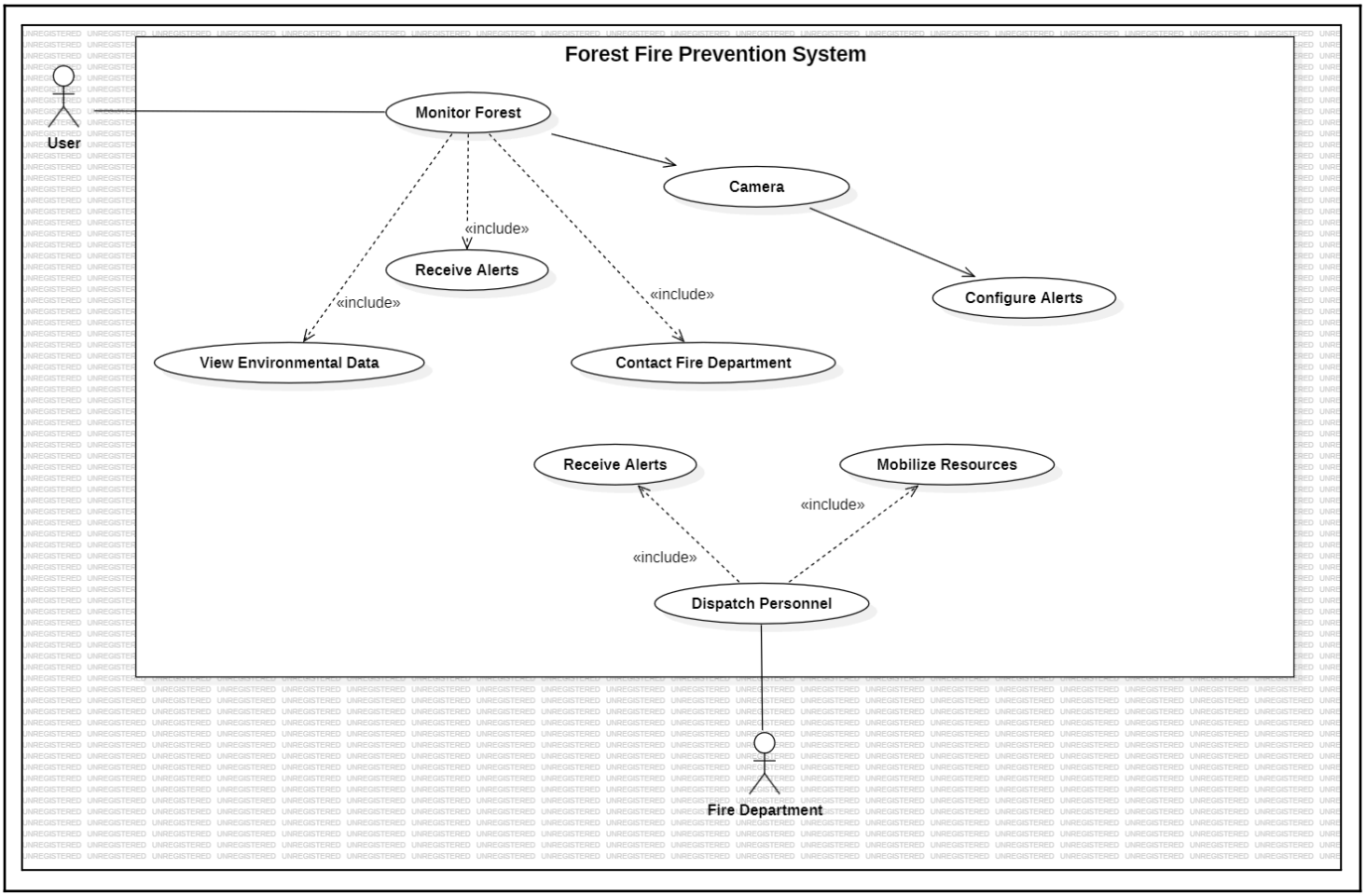


Specification -

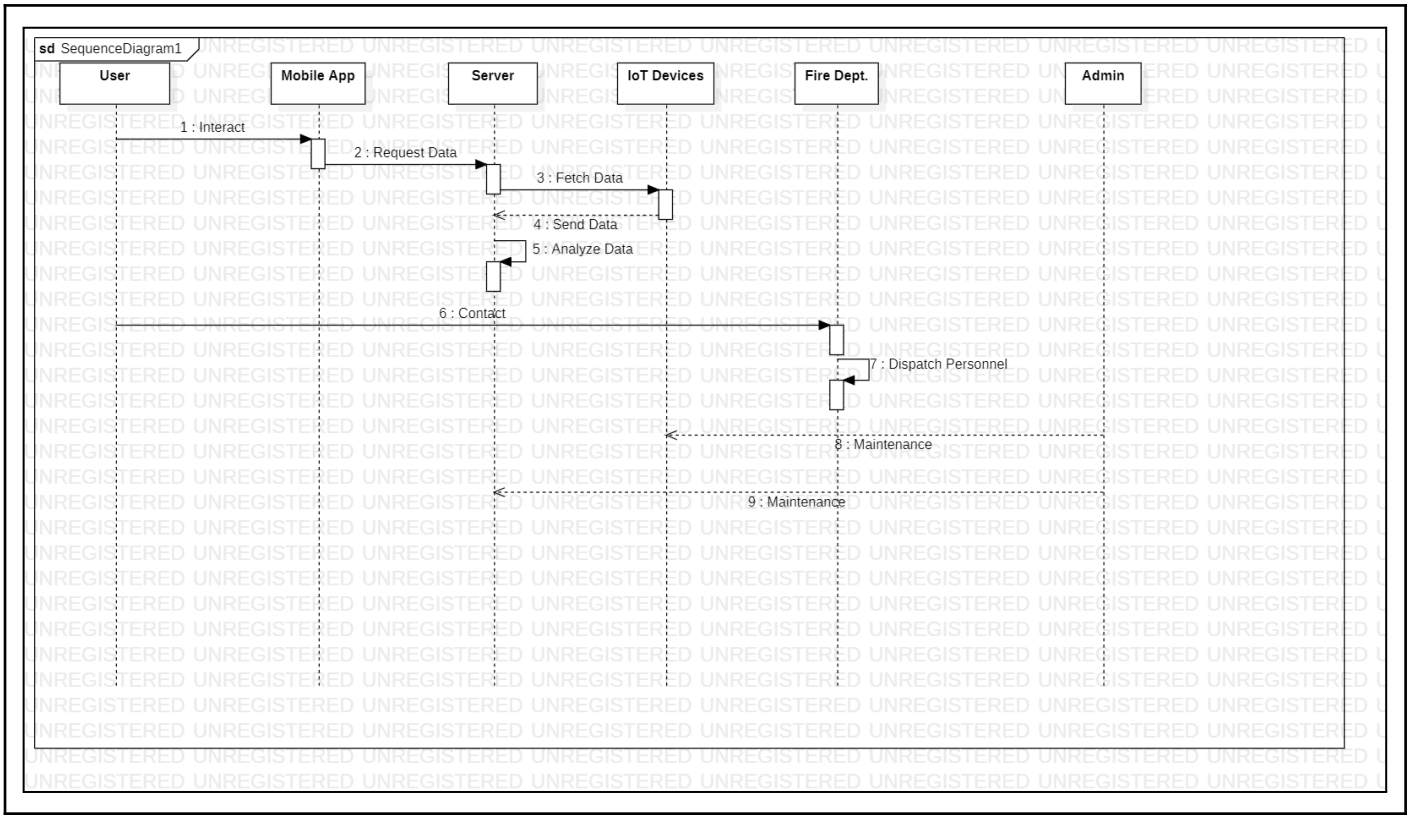
1. Microcontroller: ESP8266
2. Operating Voltage: 3.3V
3. Input Voltage (recommended): 5V
4. Digital I/O Pins: 11 (can be used as PWM outputs)
5. Analog Input Pins: 1 (3.3V max input)
6. DC Current per I/O Pin: 12 mA
7. Flash Memory: 4 MB
8. SRAM: 64 KB
9. Clock Speed: 80 MHz
10. Wi-Fi Standard: 802.11 b/g/n
11. Wi-Fi Frequency: 2.4 GHz
12. USB Interface: Micro USB
13. Dimensions: Approximately 49mm x 24mm
14. Operating Temperature: -40°C to +125°C
15. Power Consumption: ~70mA (active Wi-Fi), ~20µA (deep sleep)
16. Input Voltage (limits): 0-3.6V

3.6] Conceptual UML Models (Use case, Class diagram, Sequence diagram)

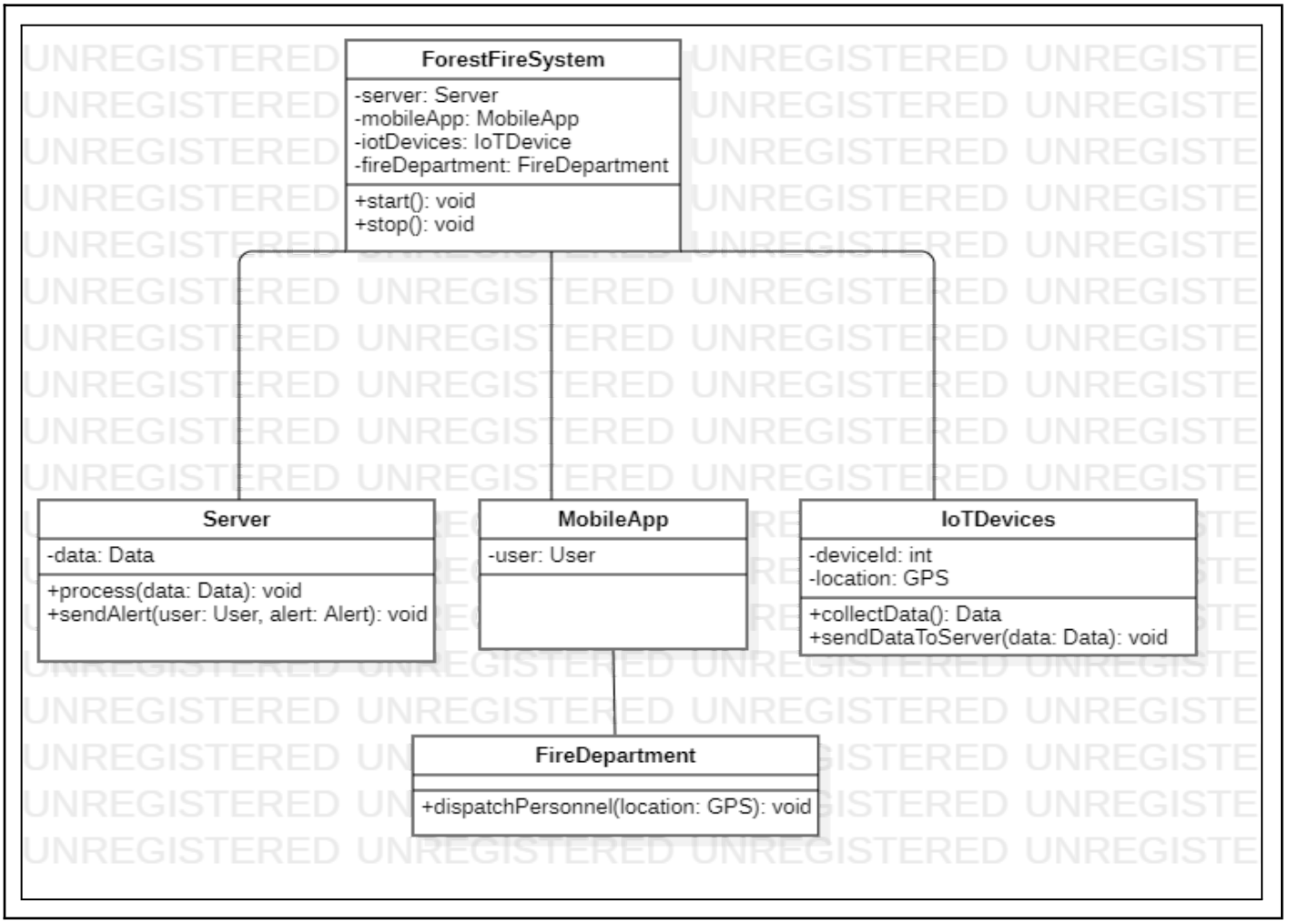
1. Use Case Diagram:



2. Sequence Diagram:



3. Class Diagram:



CHAPTER 4 - SYSTEM DESIGN

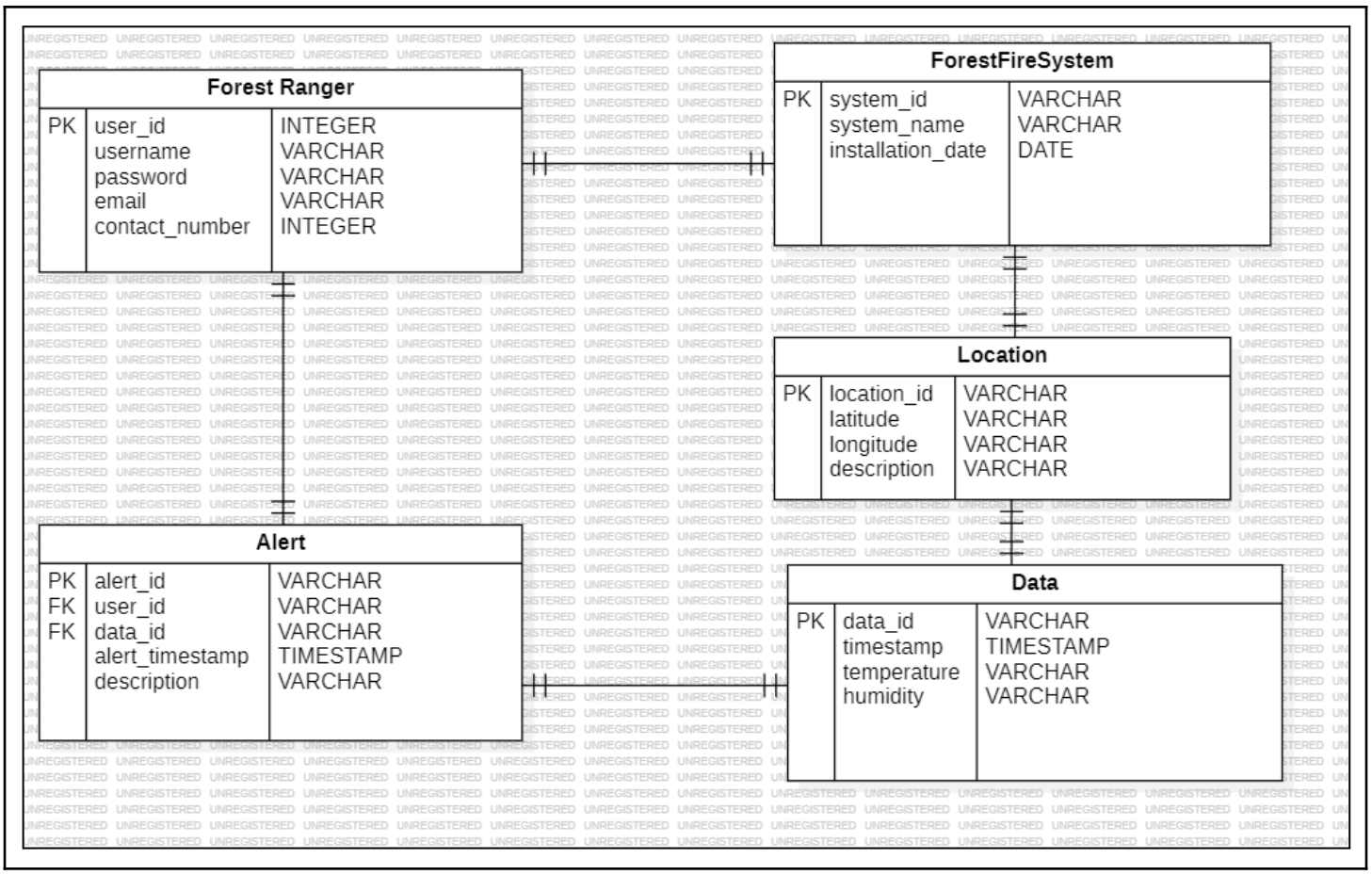
Chapter Outline

4.1] Data Design:

4.1.1] ER Diagram:

An Entity-Relationship (ER) diagram is a visual representation of the entities and relationships within a system or project. While traditionally used in database design, ER diagrams can also be valuable in non-database projects to clarify the structure and interactions within the system. Here's a short note on how ER diagrams can be beneficial in such projects:

In non-database projects, an ER diagram serves as a blueprint for understanding the various components of the system and how they relate to each other.

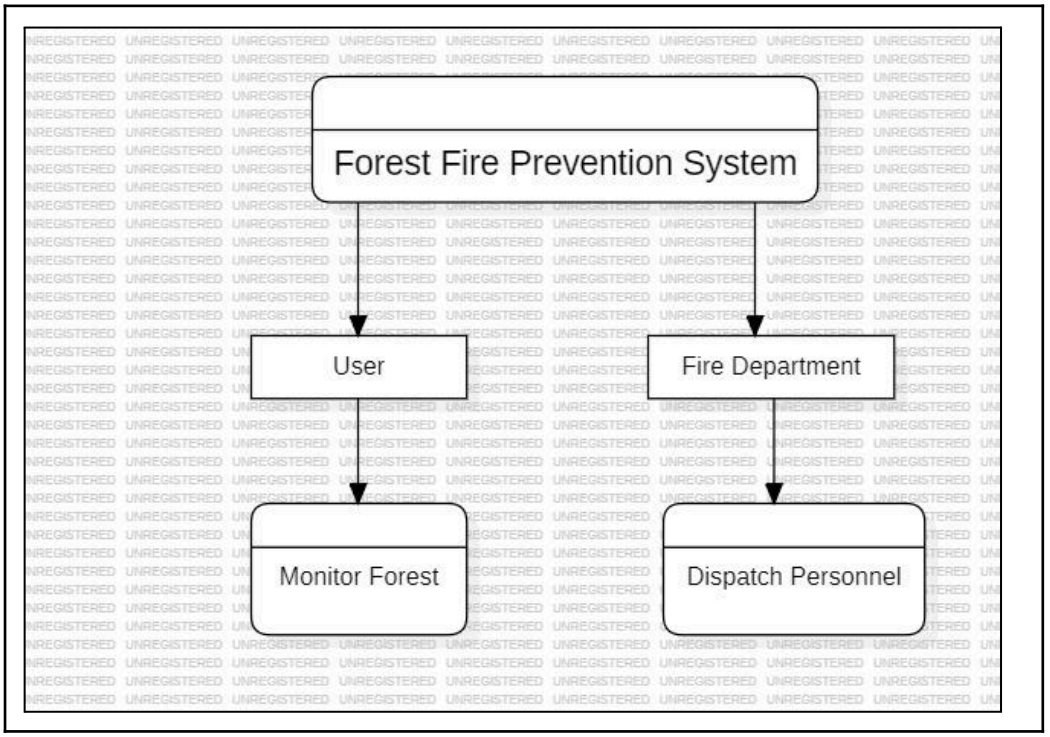

Forest Ranger
PK user_id INTEGER
username VARCHAR
password VARCHAR
email VARCHAR
contact_number INTEGER
ForestFireSystem
PK system_id VARCHAR
system_name VARCHAR
installation_date DATE
Location
PK location_id VARCHAR
latitude VARCHAR
longitude VARCHAR
description VARCHAR
Alert
PK alert_id VARCHAR
FK user_id VARCHAR
FK data_id VARCHAR
alert_timestamp TIMESTAMP
description VARCHAR
Data
PK data_id VARCHAR
timestamp TIMESTAMP
temperature VARCHAR
humidity VARCHAR

4.1.2] DFD Diagram:

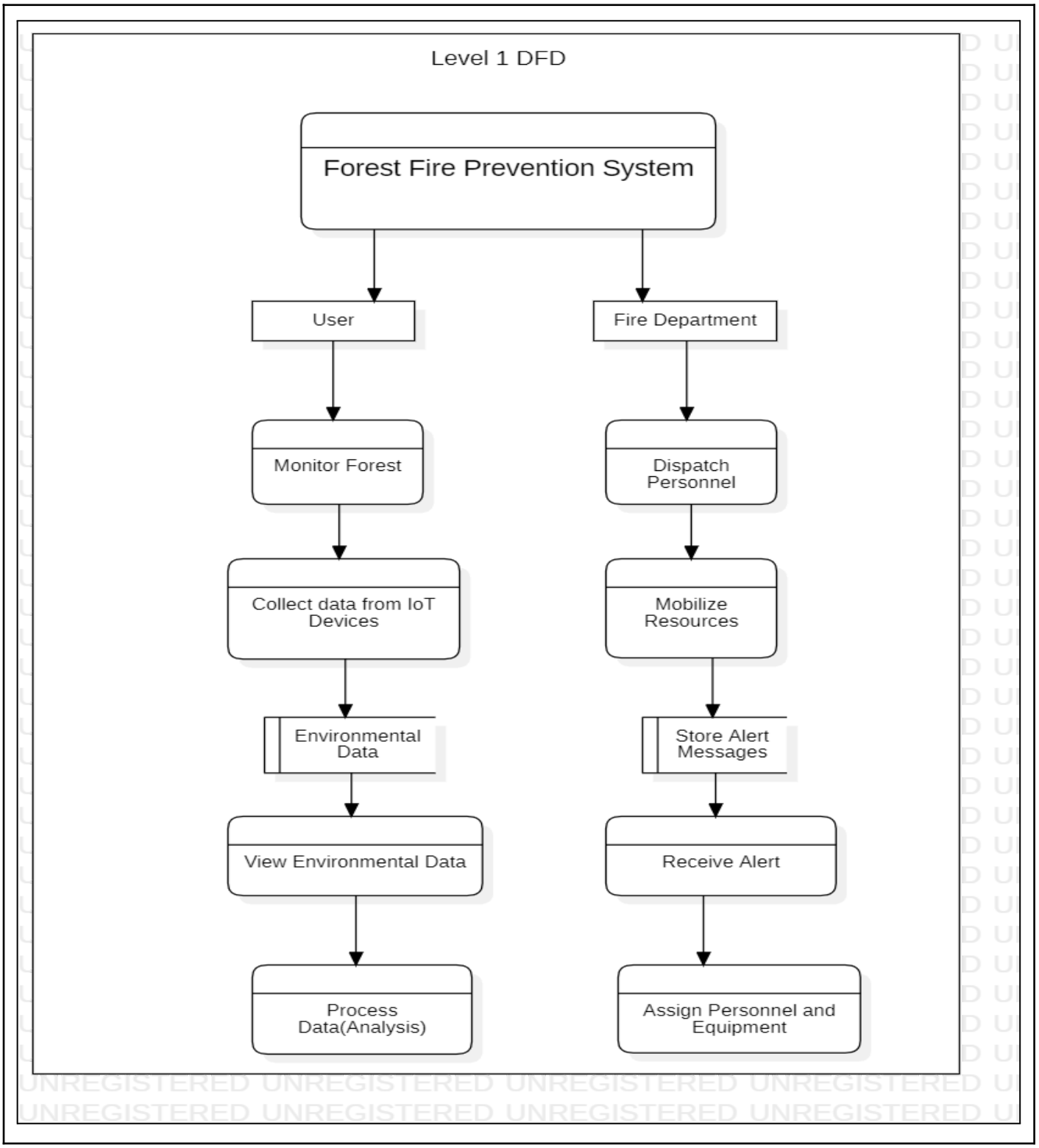
DFD stands for Data Flow Diagram, which is a graphical representation of the flow of data within a system. It illustrates how data moves through processes and data stores in a system.

DFDs are valuable tools for understanding the flow of data in a system, identifying potential bottlenecks or inefficiencies, and communicating system architecture and requirements to stakeholders and developers. They facilitate clearer understanding and effective communication during the design and development process. There are three levels i.e. Level 0, Level 1, & Level 2

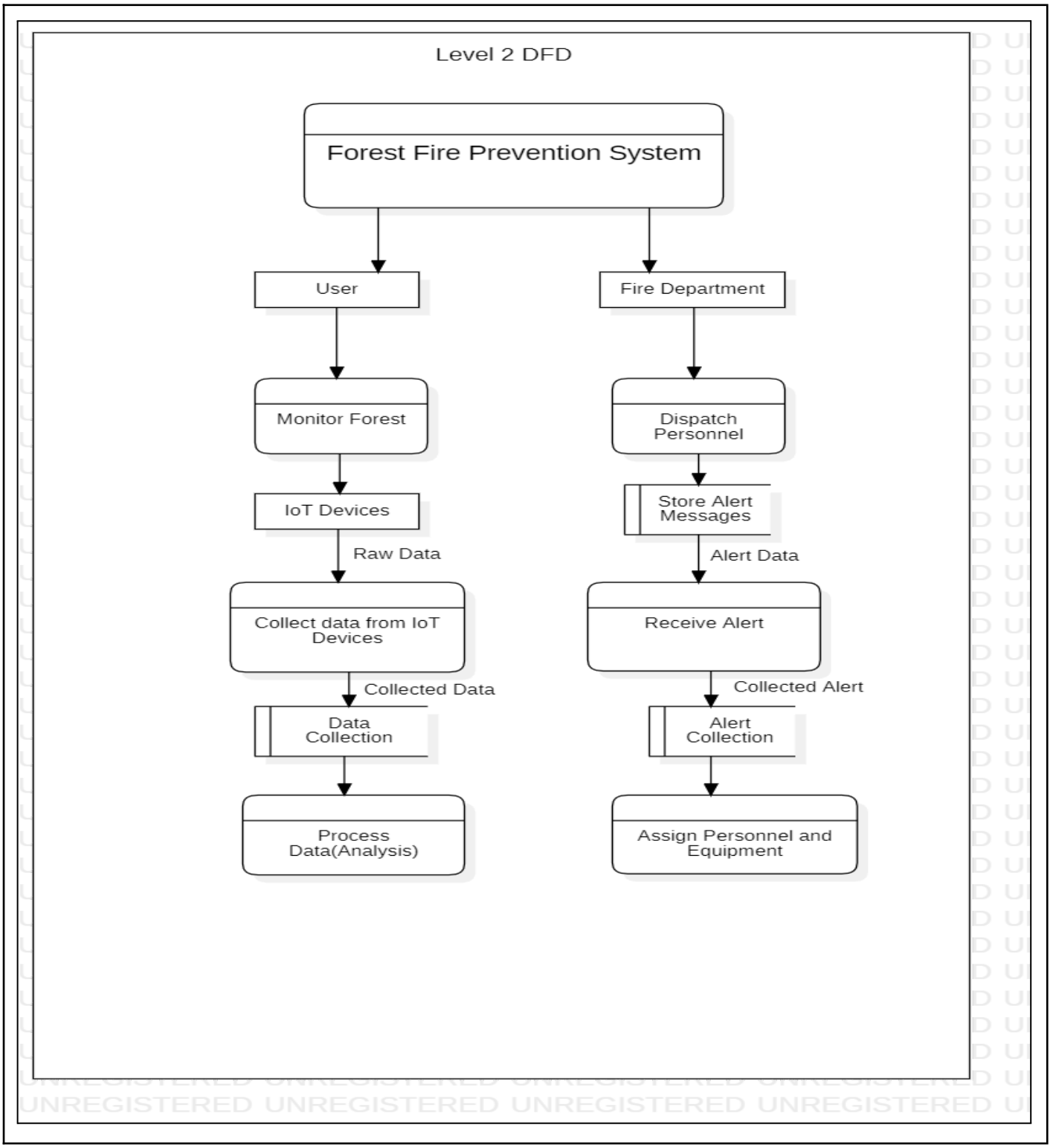
Level 0:



Level 1:


Level 1 DFD
Forest Fire Prevention System
User
Fire Department
Monitor Forest
Dispatch Personnel
Collect data from IoT Devices
Mobilize Resources
Environmental Data
Store Alert Messages
View Environmental Data
Receive Alert
Process Data(Analysis)
Assign Personnel and Equipment

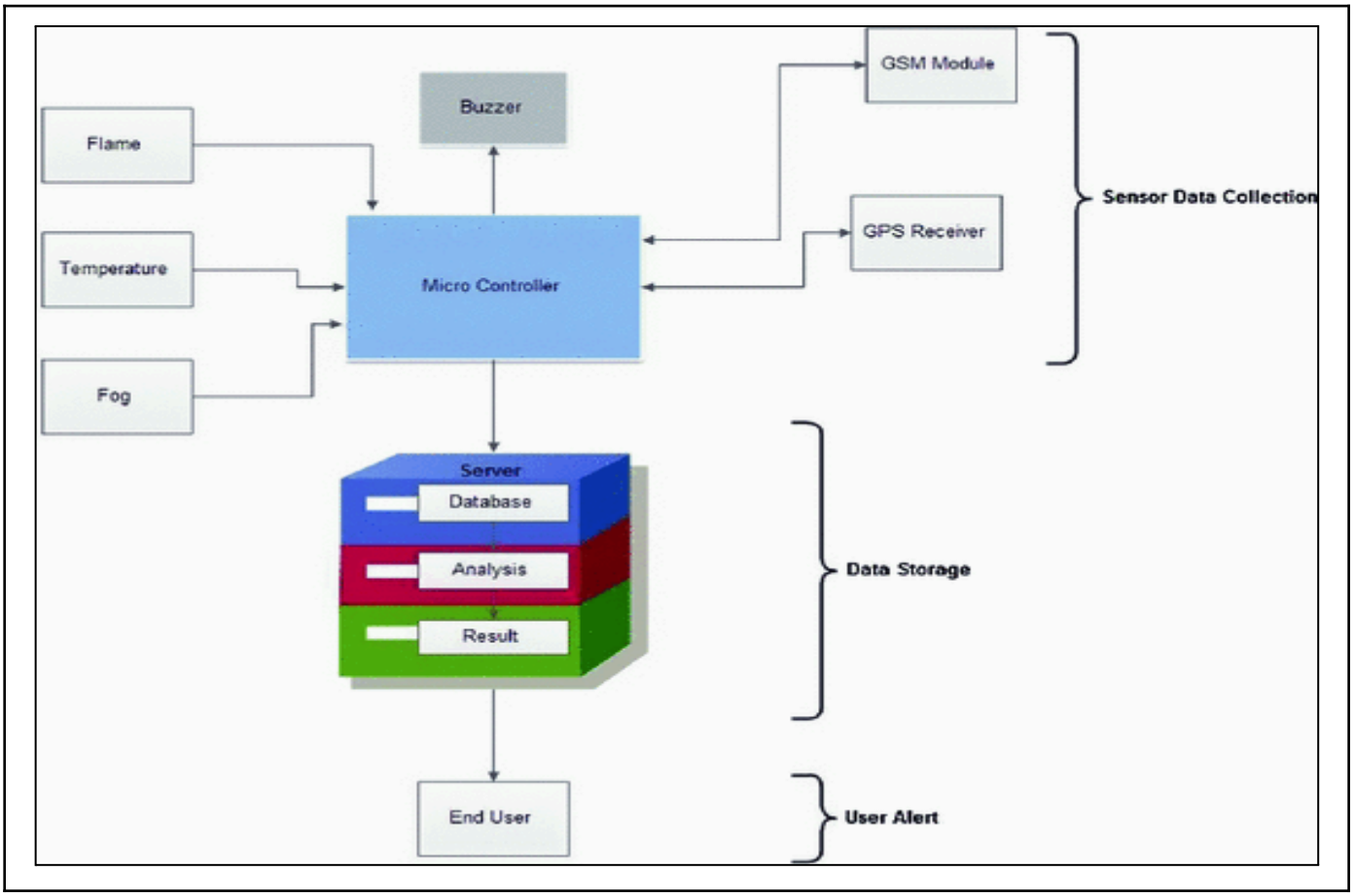
Level 2:


Level 2 DFD
Forest Fire Prevention System
User
Fire Department
Monitor Forest
Dispatch Personnel
IoT Devices
Store Alert Messages
Raw Data
Alert Data
Collect data from IoT Devices
Receive Alert
Collected Data
Collected Alert
Data Collection
Alert Collection
Process Data(Analysis)
Assign Personnel and Equipment

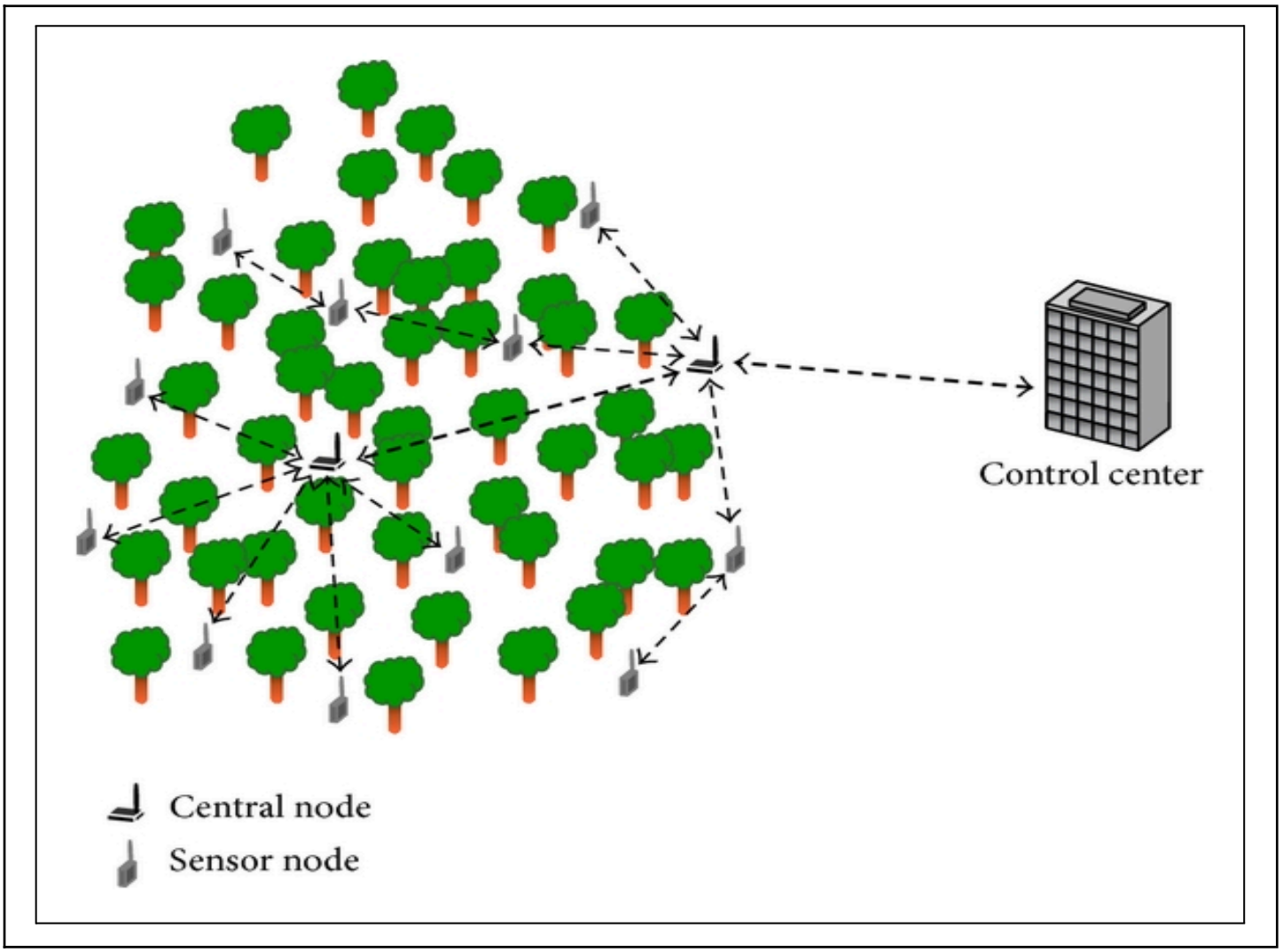
\

4.2] Flow Charts:-

4.2.1] Data Flow



4.2.3] Communication between Sensors (System) and Server:



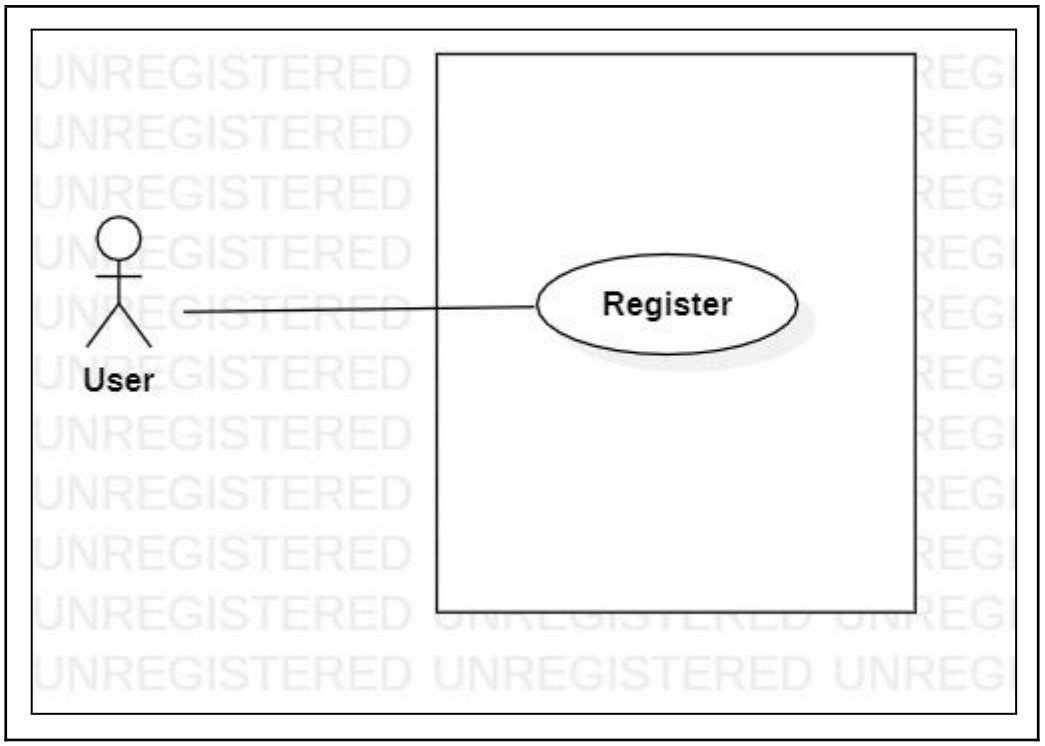
4.3] Test Cases:-

4.3.1] Functional requirements- Use case scenarios

Use Case ID	Use Case Name	Use By	Description
UC1	Register	New User	New user can register themselves in the application for getting services
UC2	Verify Account	New User	New user verifies them account by OTP
UC3	Login	User	Users can login to access the Services
UC4	Forgot Password	User	Users can login to access the Services

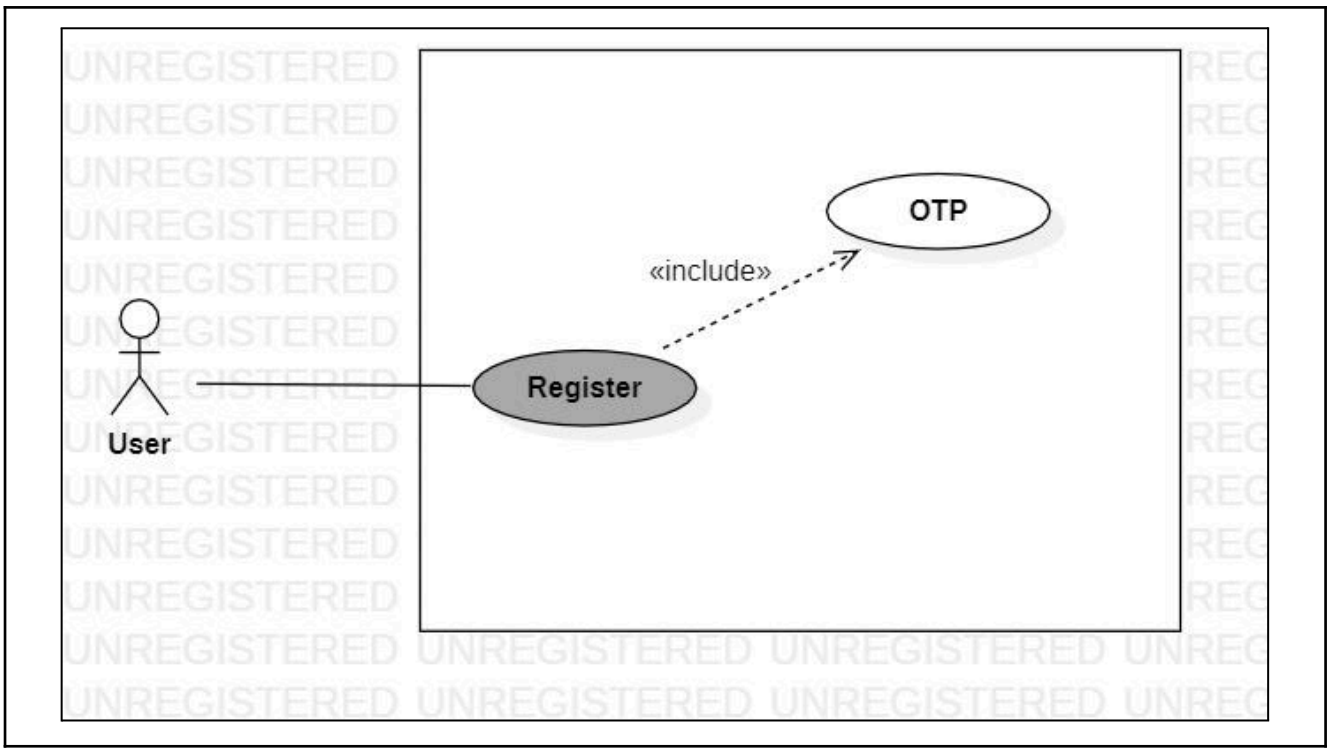
Brief Description: -
UC1:

Use Case ID	UC1
Use Case Name	Register
Relating Requirements	None
Initiating Actor	New User
Actor's Goal	To successfully Register to the Application
Pre-condition	None
Post-condition	The Profile is created successfully
Conclusion	User profile created
Flow of Event	The new user is created.



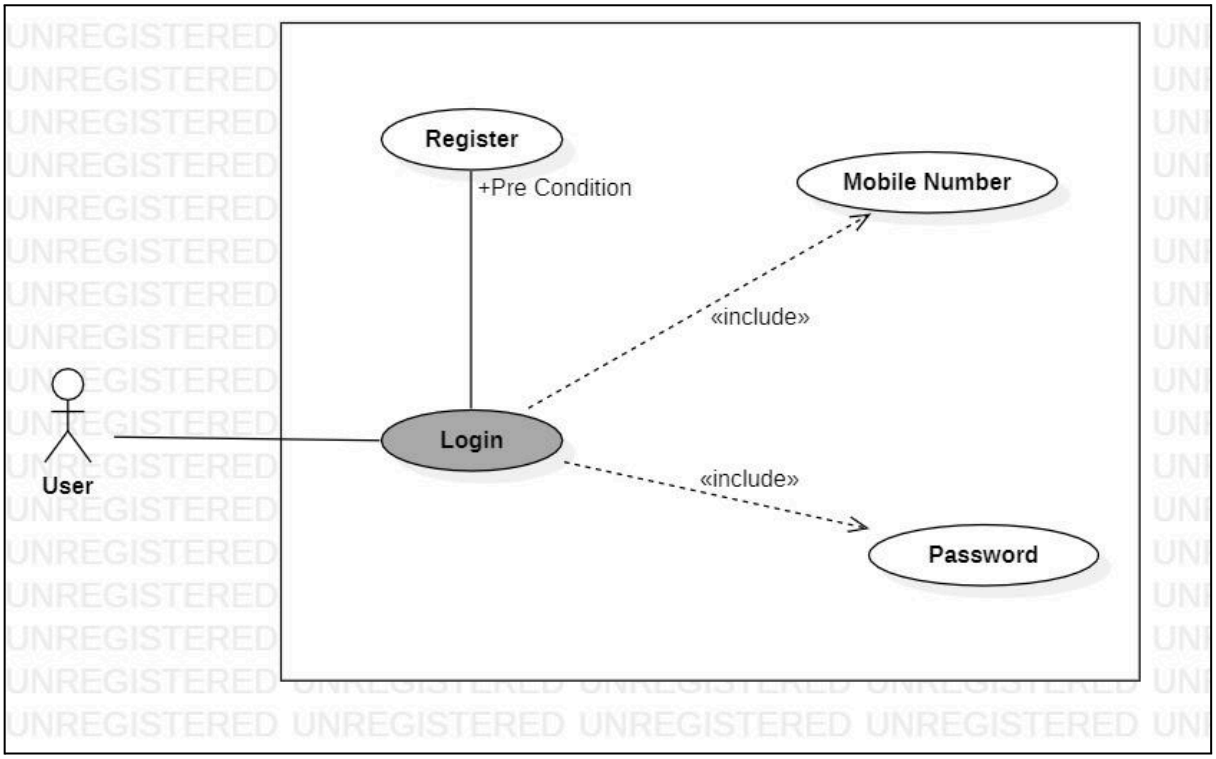
UC2:

Use Case ID	UC2
Use Case Name	Verify account.
Relating Requirements	OTP for verification.
Initiating Actor	New User
Actor's Goal	To successfully Register to the Application
Pre-condition	Users must be registered.
Post-condition	The Profile is created successfully
Conclusion	User profile created
Flow of Event	New user created.



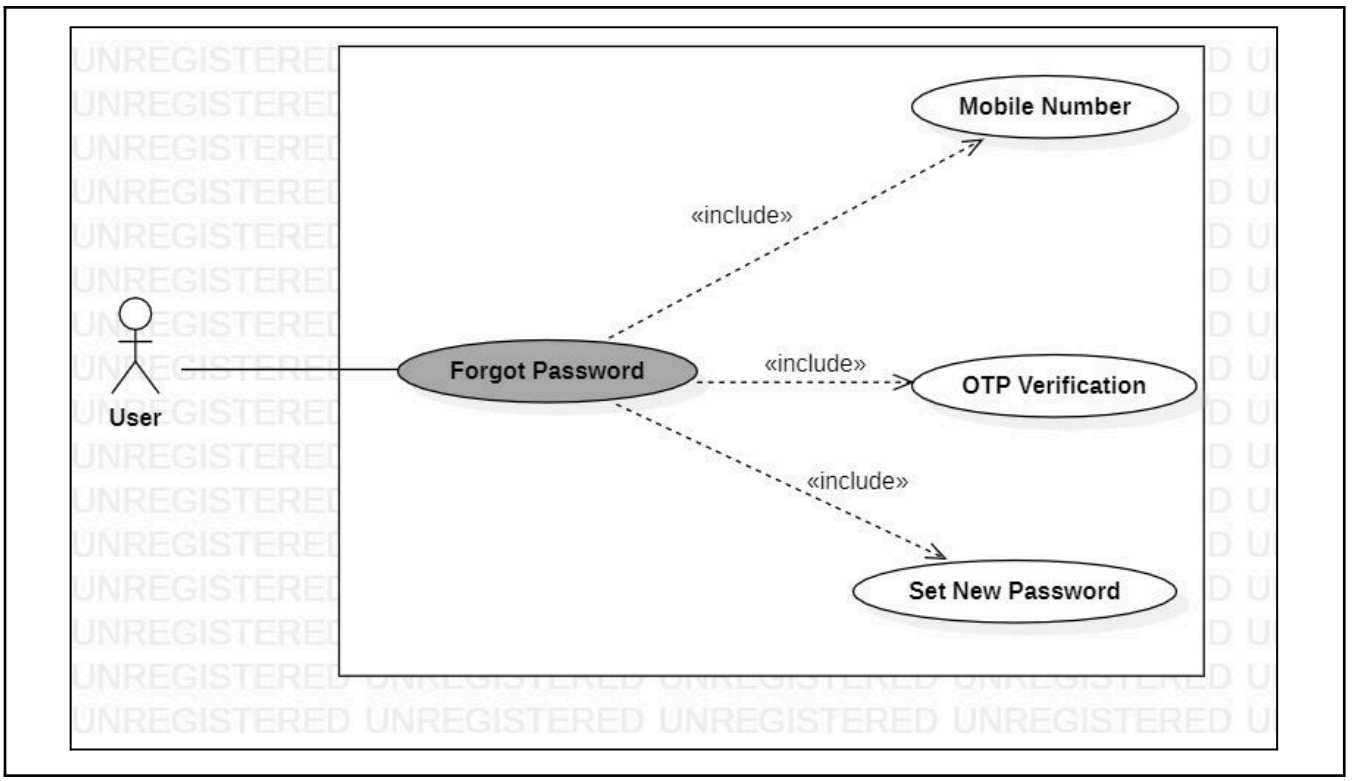
UC3:

Use Case ID	UC3
Use Case Name	Login
Relating Requirements	Mobile Number & Password
Initiating Actor	User
Actor's Goal	To successfully create
Pre-condition	User must be verified
Post-condition	Access the services
Conclusion	User logged in
Flow of Event	1.After the Created Successfully user can log in 2.They can change Password after the logging



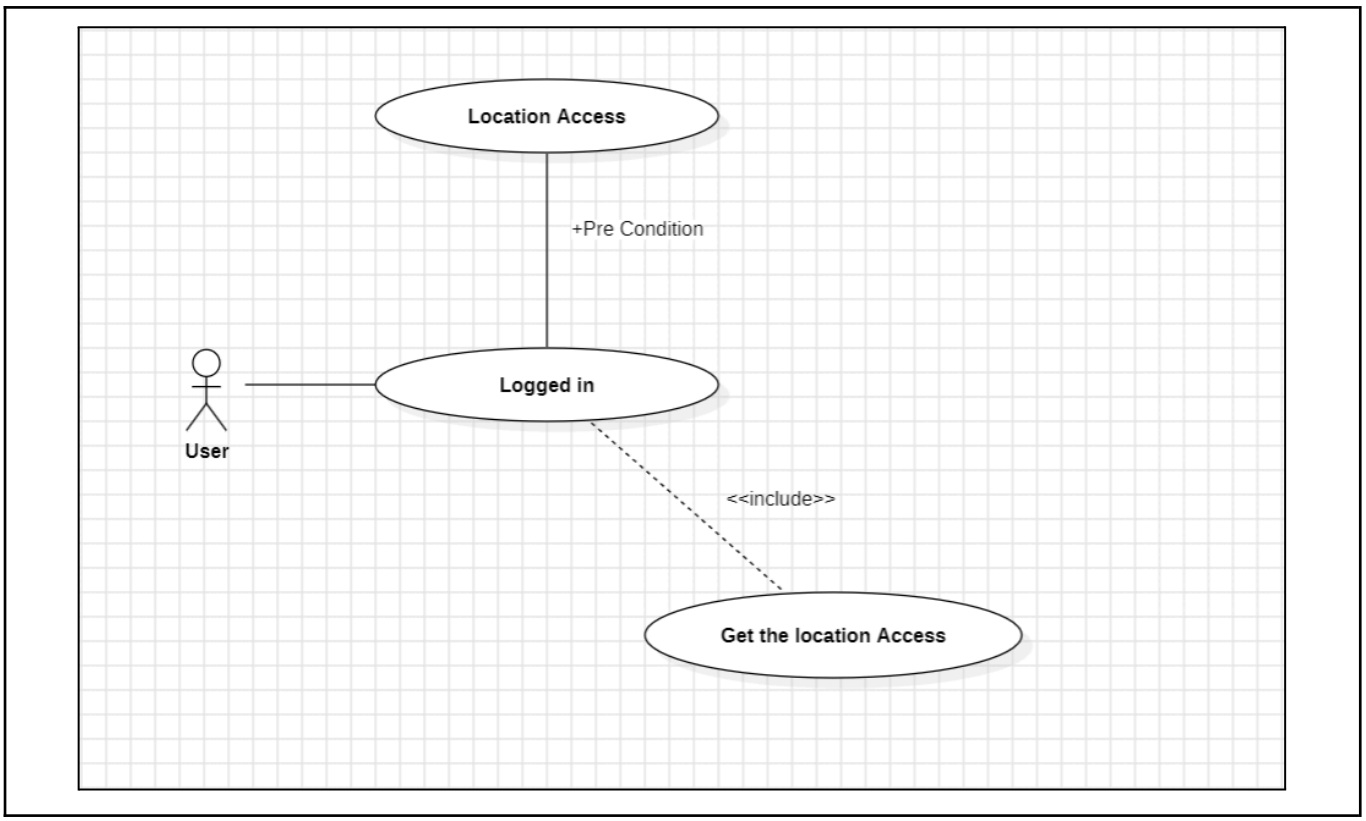
UC4:

Use Case ID	UC1
Use Case Name	Forgot Password
Relating Requirements	None
Initiating Actor	User
Actor's Goal	To successfully change the password
Pre-condition	User must have an account
Post-condition	Register
Conclusion	User change the password
Flow of Event	1. Before the log in if user forgot the password so they can create the new password 2. They need to re-enter the register phone number 3. System ask for OTP 4. user need to enter the OTP and re-try the password



UC1:

Use Case ID	UC5
Use Case Name	Location Access
Relating Requirements	User must be registered
Initiating Actor	User
Actor's Goal	To successfully allow the location access
Pre-condition	User must be logged in
Post-condition	Access the service
Conclusion	Get the nearby locations
Flow of Event	To access the application or services the user needs to give location access.



4.3.2] Application Test Cases

➢ **TC1**

Test-ID	TC1			
Use Case ID	UC1			
Test Case	Register			
Test Scenario	User need to register themself in the application			
Pre Condition	-			
Test Step	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Enter some personal details	Name, Mobile Number, Email-id, Username, Password, Postal Code	"Registered Successfully"	"Registered Successfully"	**Pass**

➢ **TC2**

Test-ID	TC2			
Use Case ID	UC2			
Test Case	Log In			
Test Scenario	For getting the services the user needs to first login themself.			
Pre Condition	Must be registered in the application.			
Test Step	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1) Enter the Username 2) Enter the password	1) Username 2) Password	"Logged in Successfully"	"Logged in Successfully"	**Pass**

➢ **TC3**

Use Case ID	UC3			
Test Case	Forget Password			
Test Scenario	If a user forgot a password, he/she can change it.			
Pre Condition	Register.			
Test Step	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1) Enter Mobile Number 2) Opt for Mobile Number Verification	Mobile Number	"Password Changed"	"Password Updated Successfully"	**Pass**

➢ **TC4**

Test-ID	TC4			
Use Case ID	UC4			
Test Case	Location Access			
Test Scenario	Users need to give his/her location access to the application.			
Pre Condition	Register.			
Test Step	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Click the Access Button	Provide the Location	"Got the location access"	"Location access granted."	**Pass**

4.3.3] System Test Cases (For System)

1. **Test-ID: TC1**

Test Case: Flame Detection

Test Scenario: Search the Service/flame.

Precondition: The flame detection service is deployed and accessible.

Test Steps:

Step 1 - Initiate a search for the flame detection service.

Step 2 - Execute the flame detection service on the microcontroller.

Test Data: Alerting through SMS and Call.

Expected Result: The flame detection service accurately identifies and highlights the presence of flames.

2. **Test-ID: TC2**

Test Case: Smoke Detection

Test Scenario: Search the Service/Smoke.

Precondition: The Smoke detection service is deployed and accessible.

Test Steps:

Step 1 - Initiate a search for the Smoke detection service.

Step 2 - Execute the Smoke detection service on the microcontroller for early detection.

Test Data: Alerting through SMS and Call.

Expected Result: The Smoke detection service accurately identifies and highlights the presence of smokes.

3. **Test-ID: TC3**

Test Case: Humidity and Temperature level Detection

Test Scenario: Search the Service.

Precondition: The Humidity and temperature level Detection (DHT11) service is deployed and accessible.

Test Steps:

Step 1 - Initiate a search for the Humidity level and Temperature level service to know the humidity level and temperature level.

Step 2 - Execute the humidity level and temperature level detection service on the microcontroller and Calibrate the range as per the scenario.

Test Data: Humidity and Temperature level Detected.

Expected Result: The Humidity and Temperature level detection service accurately identifies and sends their level to the desired destination via SMS/IoT Server.

4. **Test-ID: TC4**

Test Case: Water level Detection

Test Scenario: Search the Service.

Precondition: The Water level Detection(Ultrasonic sensor) service is deployed and accessible.

Test Steps:

Step 1 - Initiate a search for the Water level service to know the water level.

Step 2 - Execute the water level detection service on the microcontroller and Calibrate the range as per the scenario.

Test Data:

1. Water level Detected.
2. Water level sent through sms.

Expected Result: The water level detection service accurately identifies and sends water level to the desired destination via SMS.

4.4] Security Issues:-

- **Data security:** The data collected by the fire detection system, such as sensor readings, fire incident reports, and user information, must be protected from unauthorised access or data breaches. This is because misuse or manipulation of this critical information could have serious consequences.
  Solution:

1. *Use strong passwords and encryption to protect data from unauthorised access.*
2. *Implement access controls to limit who can access data.*

3. *Regularly backup data to prevent data loss.*
4. *Monitor data for unauthorised access or changes.*

- **Network security:** The communication channels used by the fire detection system to transmit data must also be secure. This means using encryption and secure protocols to prevent eavesdropping or tampering during data transmission.
  Solution :

1. *Use firewalls and intrusion detection systems to protect networks from unauthorised access.*
2. *Use secure protocols to transmit data.*
3. *Encrypt data in transit and at rest.*
4. *Monitor networks for unauthorised access or changes.*

- **False alarms:** False alarms can erode trust in the fire detection system. Therefore, it is important to consider mitigating factors such as environmental conditions, sensor malfunctions, or intentional interference to reduce the number of false positives.
  Solution:

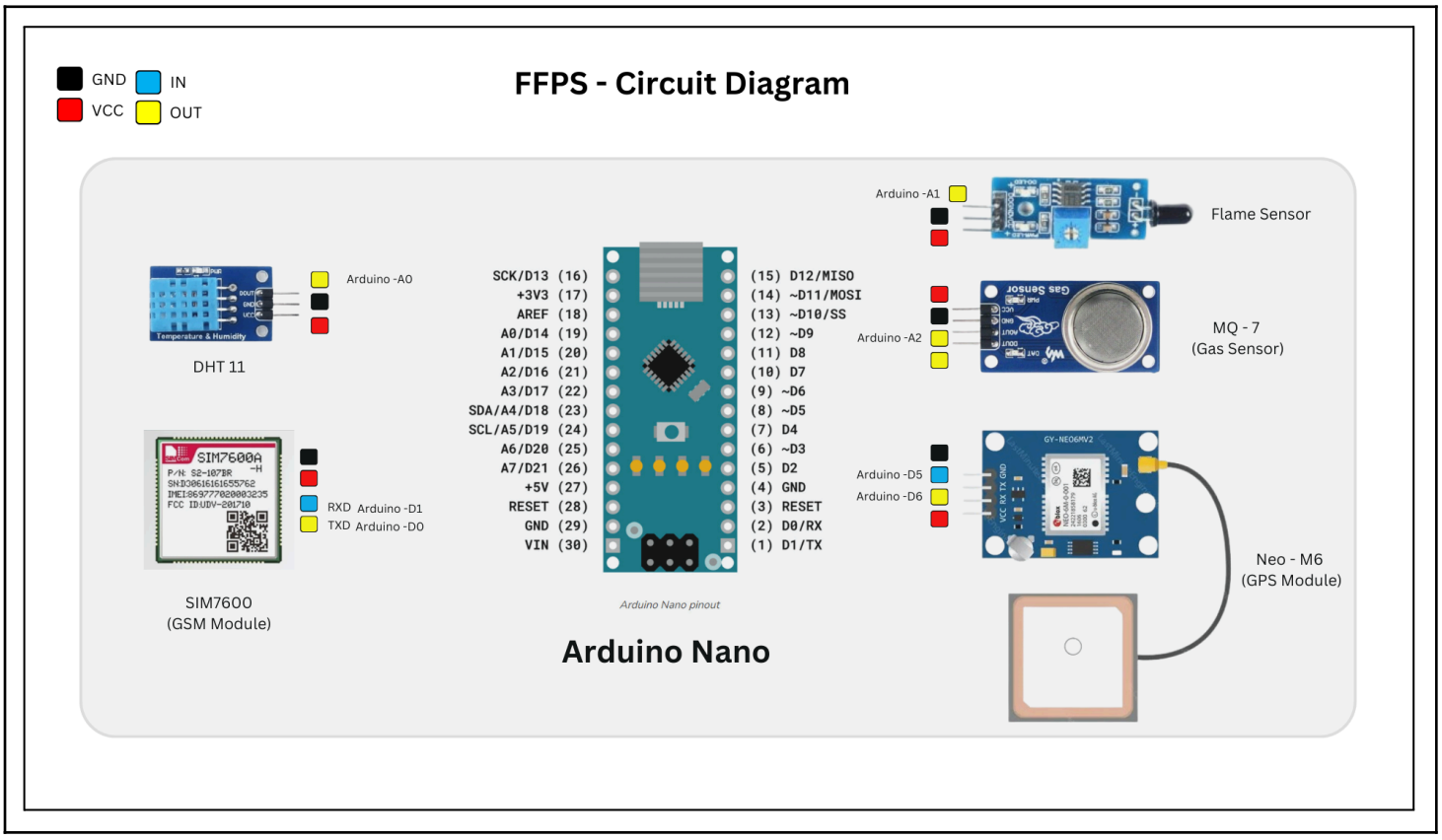
1. *Train users on how to avoid causing false alarms.*
2. *Install detectors in areas where false alarms are less likely to occur.*
3. *Use detectors with features that can reduce false alarms, such as dual sensors or heat detectors.*
4. *Monitor the system for false alarms and take steps to correct the problem.*

- **Software vulnerabilities:** The software used in the fire detection system may have vulnerabilities that can be exploited by attackers. Regular software updates and security testing are essential to patch these vulnerabilities and keep the system secure.
  Solution :

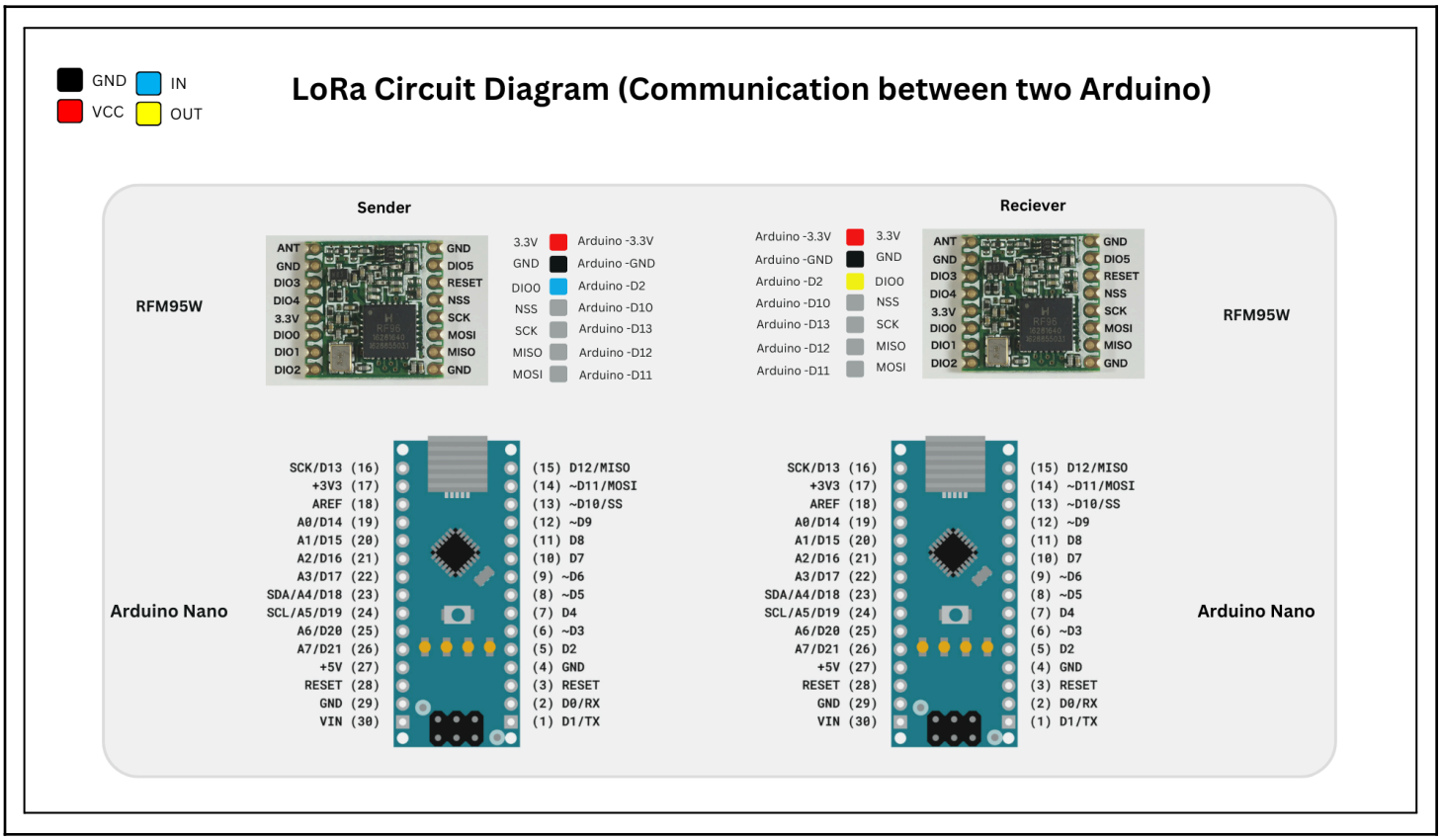
1. *Keep software up to date with the latest security patches.*
2. *Use a secure software development process to reduce the risk of vulnerabilities.*
3. *Test software for security vulnerabilities before deploying it.*
4. *Monitor software for vulnerabilities and take steps to correct them.*

4.6] Circuit Diagrams

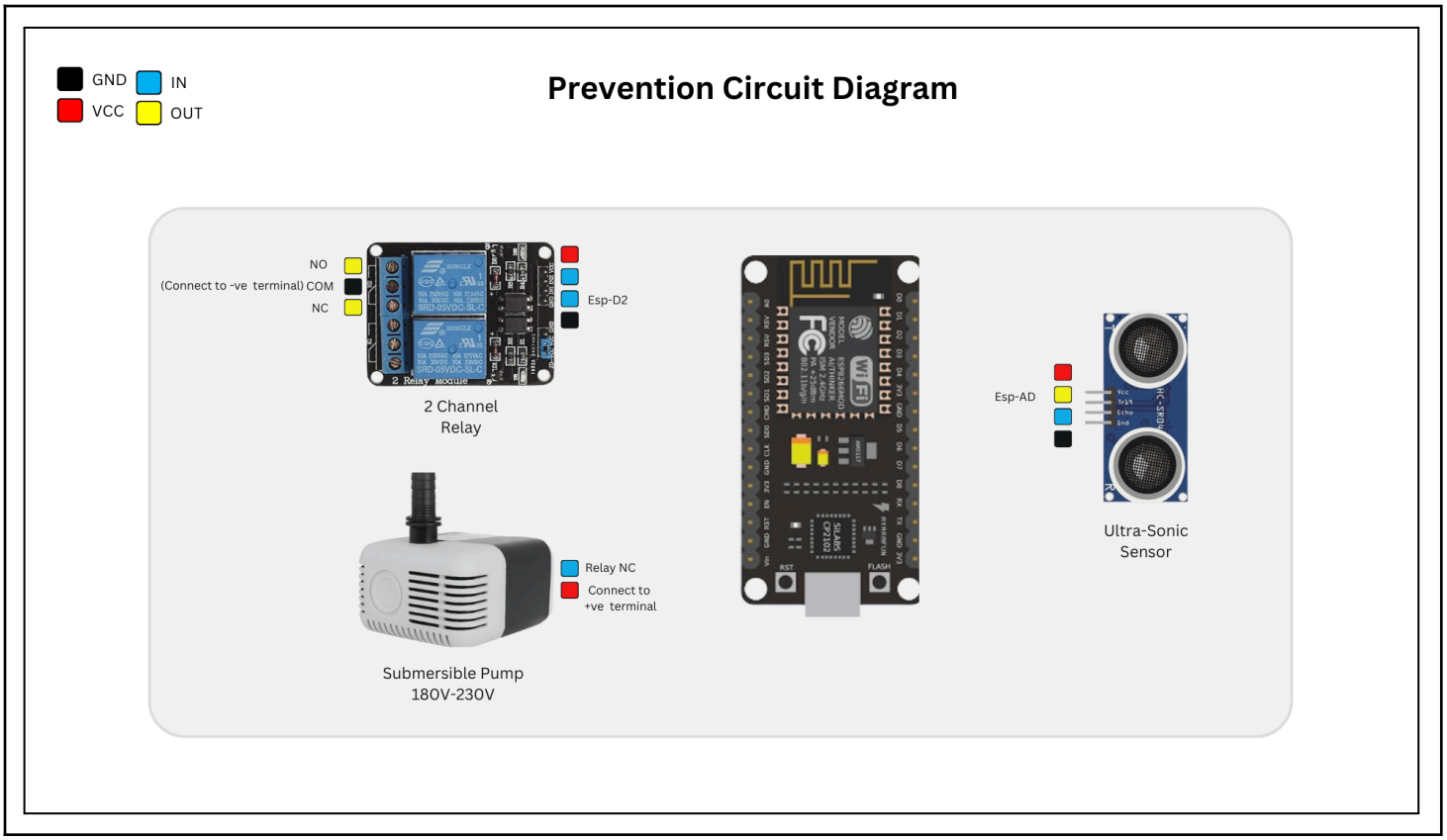
1. FFPS - Circuit Diagram



2. LoRa Circuit Diagram (Communication between two Arduino)



3. Prevention Circuit Diagram



CHAPTER 5 - RESULT AND DISCUSSION

Chapter Outline

5.1] Screenshots

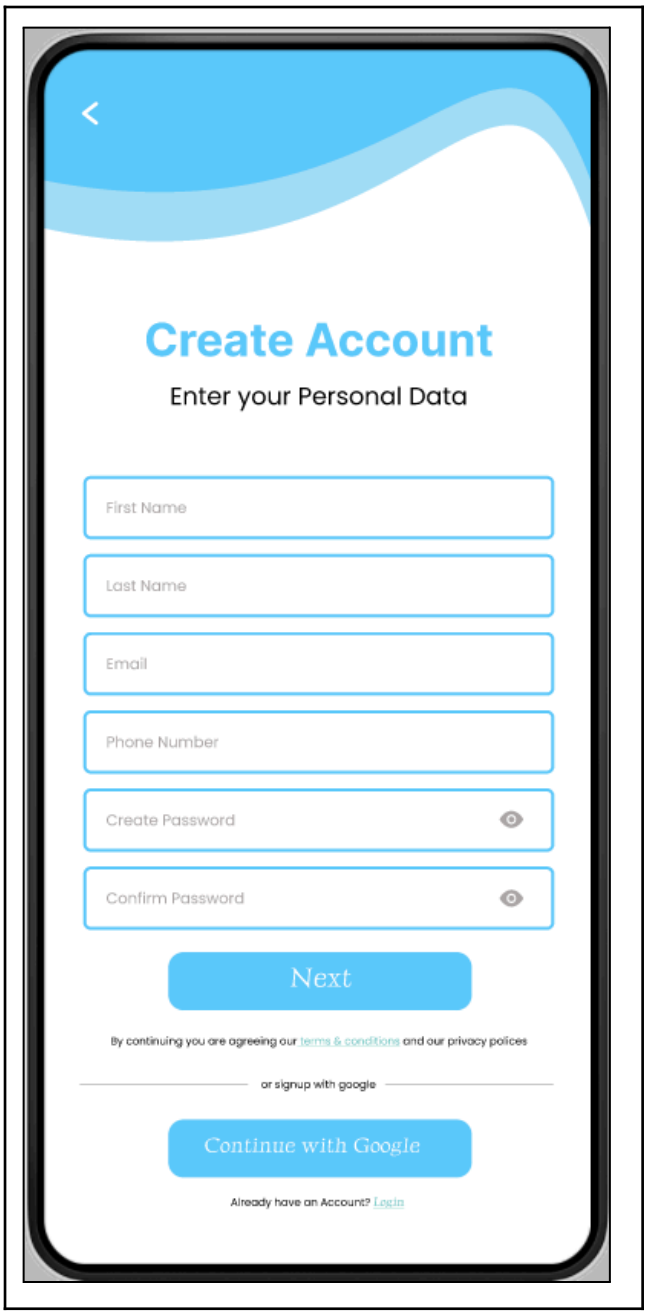
5.2] Source Codes

5.1] Screenshots

Functionality :

1. User need to register themselves in the application for getting services.
2. They need to provide First Name, Last Name, Phone Number, Username, Email id & Password.

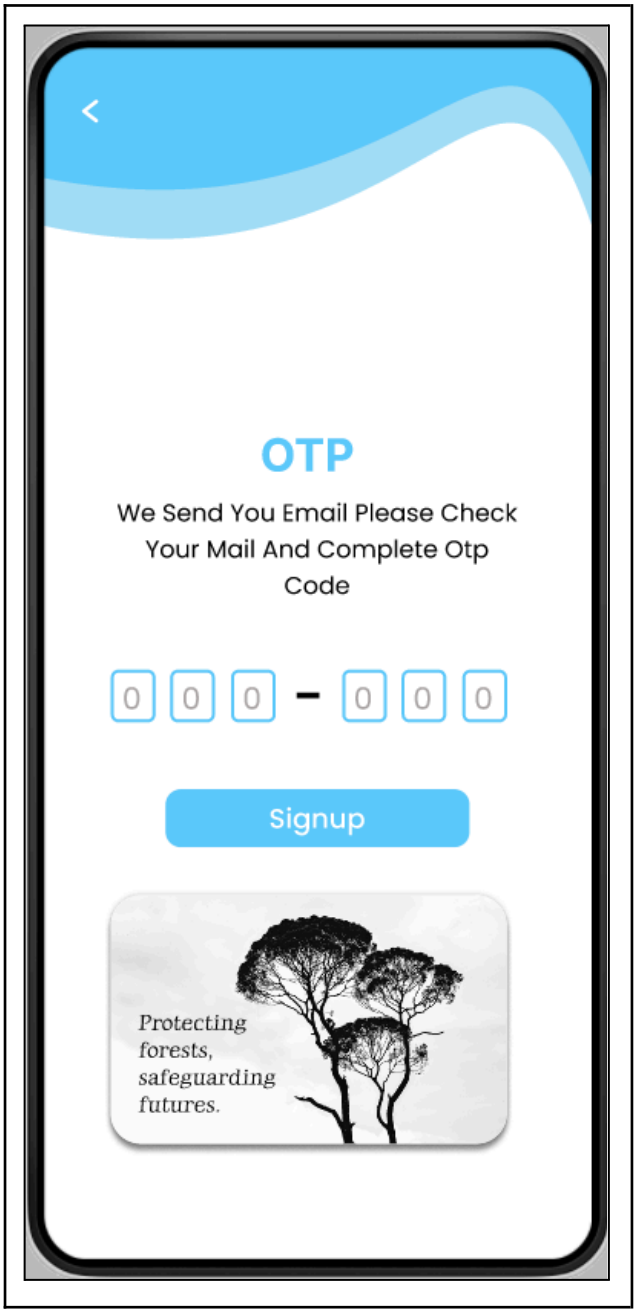
Image (Register Page) :-


Create Account
Enter your Personal Data
First Name
Last Name
Email
Phone Number
Create Password
Confirm Password
Next
By continuing you are agreeing our terms & conditions and our privacy polices
or signup with google
Continue with Google
Already have an Account? Login

Functionality:

1. OTP verification is required for the correct User or not.
2. After completing, the data of the users is stored in the database (firebase), so the next time the user can directly login itself.

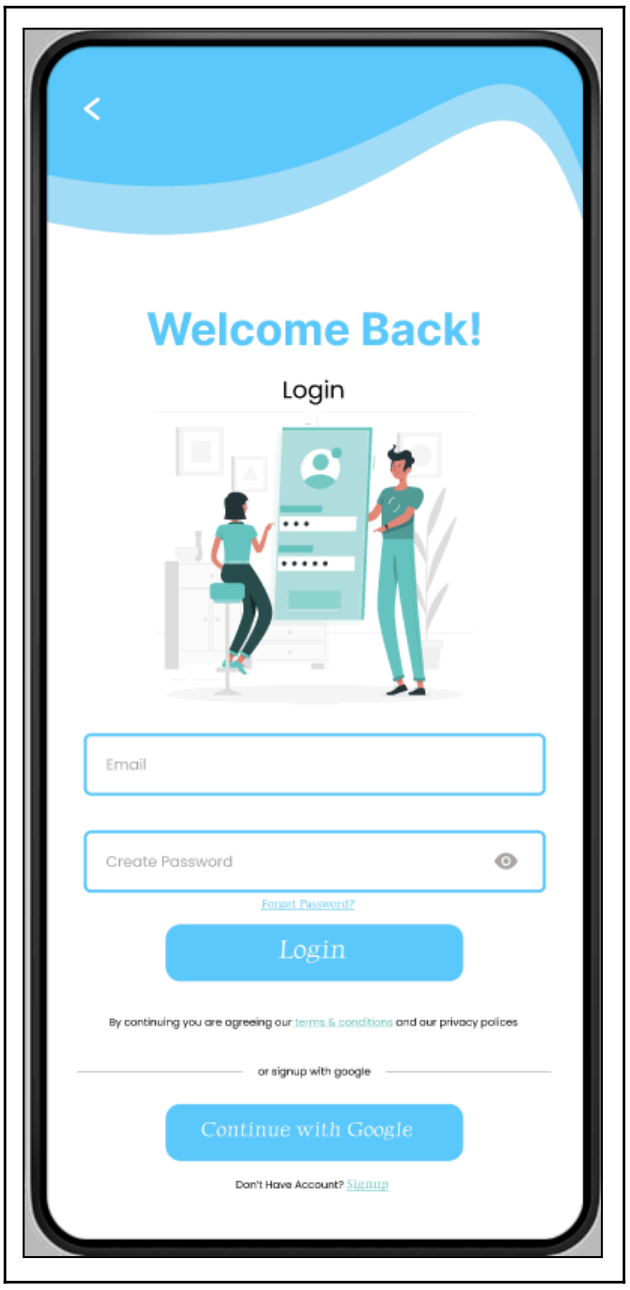
Image (OTP Page) :-



Functionality

1. After the register page, the user can log in.
2. For login user need to enter the username and password.

Image (Login Page) :-



Functionality

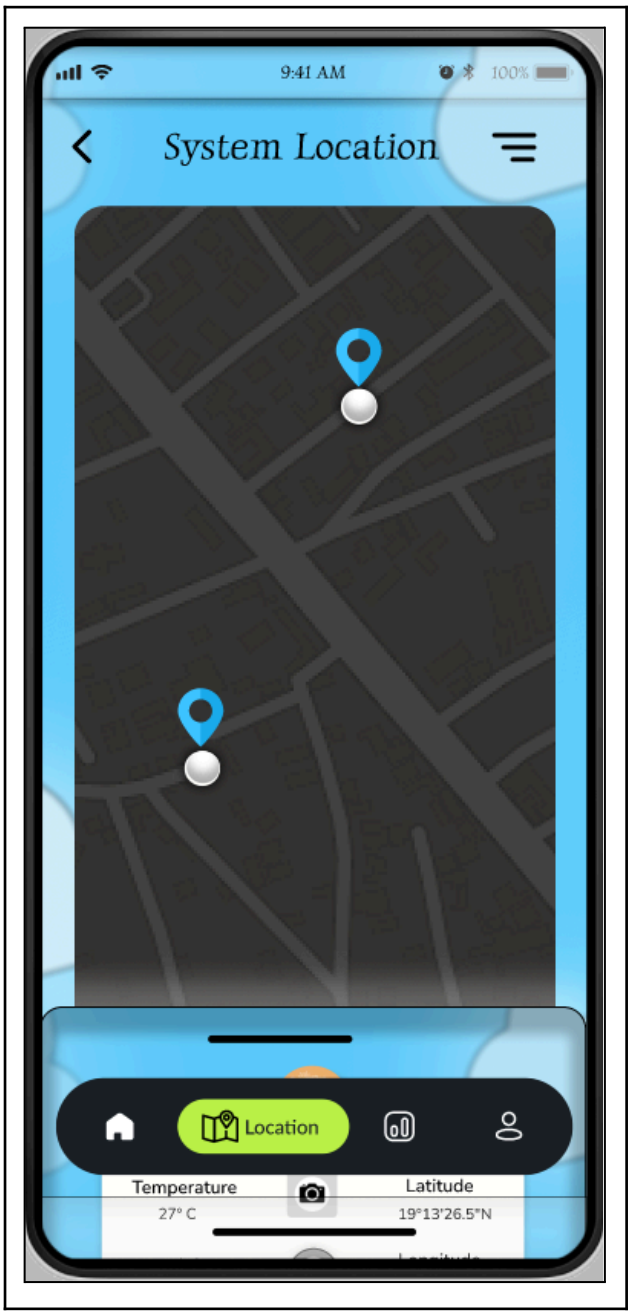
1. Location Awareness: Date, time, and place will automatically update based on the user's location.
2. Today's Weather Highlight: Leverages either an external weather API or device sensors to generate a dynamic report.
3. This report will include:
    A. Today's Highlight: A concise summary of the day's weather conditions (e.g., "Sunny and warm", "Rainy with strong winds").
    B. Temperature: Current temperature reading.
    C. Humidity: Current humidity level.
    D. Precipitation: Chance of precipitation for the day (optional: amount or intensity).
    E. Wind Speed: Current wind speed (optional: direction).
4. Bottom Navbar: Provides easy access to key functionalities through the following icons:
    A. Home: Takes the user back to the homepage.
    B. System Location: Provides information about the locations where the system is implemented.
    C. Analytics: Presumably, this section offers data analysis or visualisations related to the system's performance.
    D. Profile: Allows users to access their personal profile settings.

Image (Home Page) :-

Functionality

1. The Location page will display the various locations where the system is currently implemented.
2. When a user hover over a specific implementation point on the location page, additional details about the system in that location will be revealed.

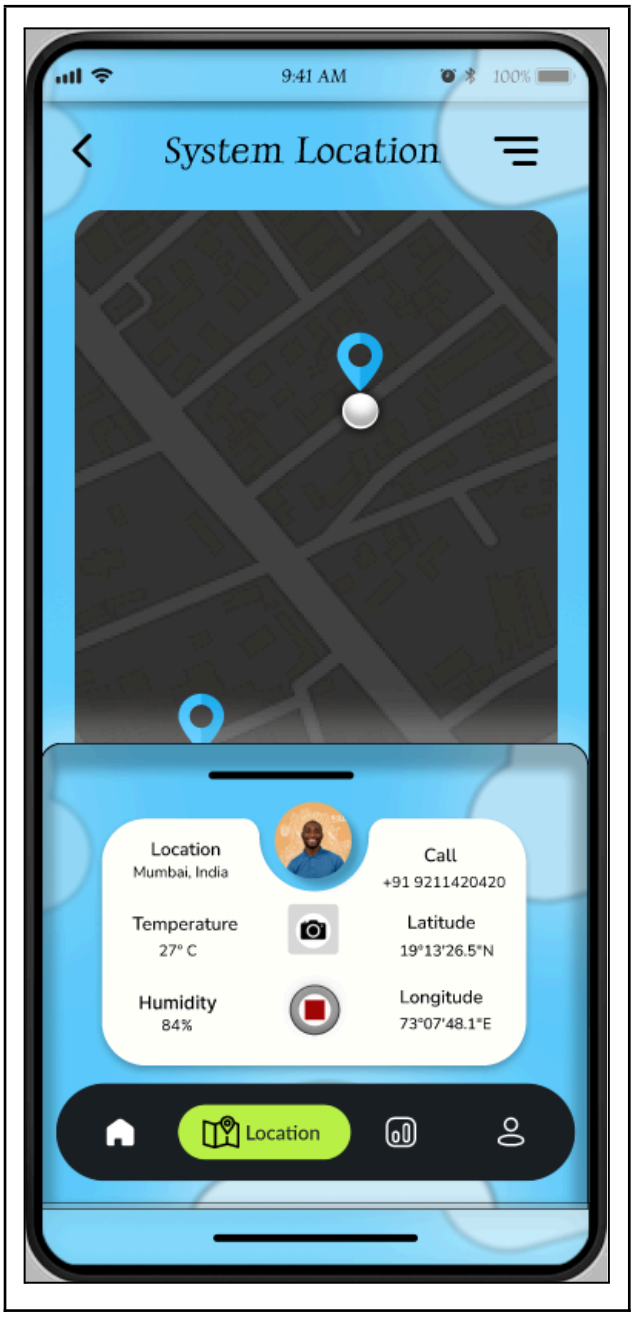
Image (Location Page - 1):-



Functionality ;

1. Geolocation: Displays the latitude and longitude coordinates of the specific system implementation.
2. Location Name: Shows the designated name or description of the location.
3. Environmental Data: Provides current readings for:
    A. Temperature
    B. Humidity
4. Thermal Camera: A camera button allows users to open a live feed from the thermal camera at that location.
5. Alert Status:
    A. Displays the current status of any alerts generated at that location.
    B. Includes a stop button to silence or deactivate the alert for that specific instance.

Image (Location Page - 2):-



Functionality

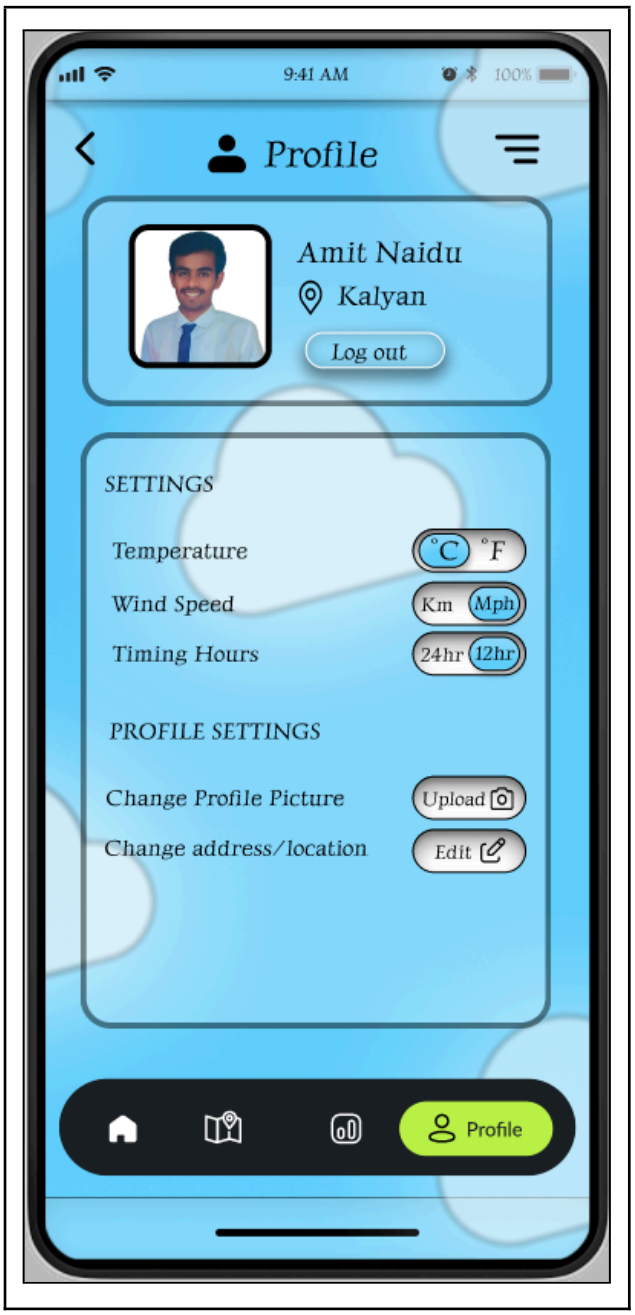
1. Date Selection: A calendar at the top allows users to choose a specific date to view the corresponding analytics data.
2. Performance Visualisation: This section provides insightful visualisations of key metrics through graphs:
    A. Temperature & Humidity: Tracks trends and fluctuations in temperature and humidity over time.
    B. System Performance: Monitors and visually represents the system's overall health and functionality. This could include metrics like uptime, response times, or resource utilisation.
    C. Alert Frequency: Displays the frequency of alerts generated by the system, potentially helping identify areas requiring attention.

Image (Analytics Page) :-

Functionality

1. User Details: This section displays the user's personal information.
2. Settings: This section allows users to modify their account settings and preferences.
3. Log Out: This button provides a clear way for users to end their session and securely log out of the system.

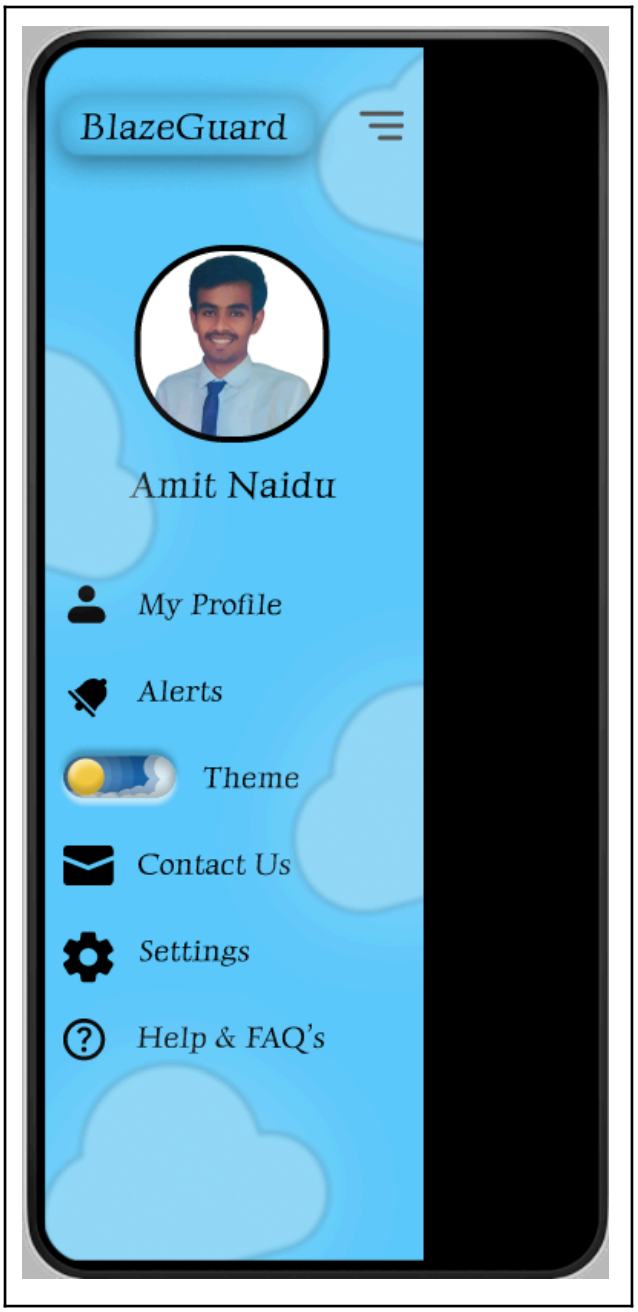
Image (Profile Page) :-



Functionality

1. Provides access to various functionalities through a collapsible menu, typically represented by a hamburger icon (three horizontal lines).
2. Menu options:
    A. My Profile: Takes the user directly to their profile page for managing personal information.
    B. Alerts: Presumably, this section allows users to view and manage any system alerts they have received.
    C. Theme: This option likely gives users control over the app's visual appearance (e.g., light/dark mode).
    D. Contact Us: Provides a way for users to contact the system's administrators or support team.
    E. Settings: This section likely offers users more granular control over preferences related to the system's operation.
    F. Help & FAQs: This section should provide users with easy access to helpful documentation, tutorials, or frequently asked questions (FAQs) to assist them in using the system effectively.

Image (Hamburger Menu):-


BlazeGuard
Amit Naidu
My Profile
Alerts
Theme
Contact Us
Settings
Help & FAQ's

5.2] Source Codes :-

Final Source Code:

```
// THIS EXAMPLE SHOWS HOW VVM501 ESP32 4G LTE MODULE CAN USE TO SEND AND RECEIVE SMS AND CALL
// FOR VVM501 PRODUCT DETAILS VISIT www.vv-mobility.com
#define RXD2 27    // VVM501 MODULE RXD INTERNALLY CONNECTED
#define TXD2 26    // VVM501 MODULE TXD INTERNALLY CONNECTED
#define powerPin 4 // VVM501 MODULE ESP32 PIN D4 CONNECTED TO POWER PIN OF A7670C CHIPSET, INTERNALLY CONNECTED
#define relay 2
#define SerialAT Serial1
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <WiFi.h>
#include <DHT.h> // Include the DHT library
#include <Firebase_ESP_Client.h>

// Firebase configuration
#define FIREBASE_HOST "your_firebase_project_url"
#define FIREBASE_AUTH "your_firebase_auth_token"

// GPS RX to D1 & GPS TX to D2 and Serial Connection
const int RXPin = 4, TXPin = 5;
const uint32_t GPSBaud = 9600;
SoftwareSerial gps_module(RXPin, TXPin);

TinyGPSPlus gps;
#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor

FirebaseData firebaseData;

int rx = -1;
String rxString;
int _timeout;
String _buffer;
String number = "+91XXXXXXXXXX"; // REPLACE WITH YOUR NUMBER
String SMS = "MESSAGE FROM VVM501 ESP32 4G LTE MODULE";

bool relayOn = false;
```

```
const int flameSensorPin = A0;  // Analog pin connected to the flame sensor
const int SmokeSensorPin = A0;  // Analog pin connected to the flame sensor
#define BuzzerPin 1 // Digital pin connected to Buzzer

unsigned long sendDataPrevMillis = 0;
int ldrdata = 0;
float voltage = 0.0;
int pwmValue = 0;
boolean ledStatus = false;

void setup() {
  InitUnit();
}

int result = 0;

void loop() {
  if (Firebase.ready() && (millis() - sendDataPrevMillis > 5000 || sendDataPrevMillis == 0)) {
    sendDataPrevMillis = millis();

    // Read humidity and temperature from DHT sensor
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();

    // Check if any reads failed and exit early (to try again).
    if (isnan(humidity) || isnan(temperature)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }

    // Read latitude and longitude from GPS module
    float latitude = gps.location.lat();
    float longitude = gps.location.lng();

    // Send latitude and longitude to Firebase
    if (Firebase.RTDB.setFloat(&fbdo, "location/latitude", latitude) &&
        Firebase.RTDB.setFloat(&fbdo, "location/longitude", longitude)) {
      Serial.println("Location data sent to Firebase!");
    } else {
      Serial.println("Failed to send location data to Firebase!");
      Serial.println(fbdo.errorReason());
    }

    // Send humidity and temperature to Firebase
    if (Firebase.RTDB.setFloat(&fbdo, "environment/humidity", humidity) &&
        Firebase.RTDB.setFloat(&fbdo, "environment/temperature", temperature)) {
      Serial.println("Humidity and temperature data sent to Firebase!");
```

```
  } else {
    Serial.println("Failed to send humidity and temperature data to Firebase!");
    Serial.println(fbdo.errorReason());
  }

  // Example: Reading data from Firebase
  if (Firebase.RTDB.getInt(&fbdo, "/LED/analog")) {
    if (fbdo.dataType() == "int") {
      pwmValue = fbdo.intData();
      Serial.print("Successful Read from ");
      Serial.print(fbdo.dataPath());
      Serial.print(": ");
      Serial.print(pwmValue);
      Serial.print(" (");
      Serial.print(fbdo.dataType());
      Serial.println(")");
      analogWrite(LED1_PIN, pwmValue);
    }
  } else {
    Serial.print("Failed: ");
    Serial.println(fbdo.errorReason());
  }

  if (Firebase.RTDB.getBool(&fbdo, "/LED/digital")) {
    if (fbdo.dataType() == "boolean") {
      ledStatus = fbdo.boolData();
      Serial.print("Successful Read from ");
      Serial.print(fbdo.dataPath());
      Serial.print(": ");
      Serial.print(ledStatus);
      Serial.print(" (");
      Serial.print(fbdo.dataType());
      Serial.println(")");
      digitalWrite(LED2_PIN, ledStatus);
    }
  } else {
    Serial.print("Failed: ");
    Serial.println(fbdo.errorReason());
  }
}

// Update GPS data
while (Serial1.available() > 0) {
  if (gps.encode(Serial1.read())) {
    break;
  }
}
```

```
}
 int flameSensorValue = analogRead(flameSensorPin);
 int SmokeSensorValue = analogRead(SmokeSensorPin);

 // Adjust the threshold based on your sensor and environment
 int flamethreshold = 500;
 int smokethreshold = 150;

 // Flame detection logic
 if (flameSensorValue < flamethreshold || SmokeSensorValue < smokethreshold) {
  // Flame or smoke detected, take action
  digitalWrite(BuzzerPin, HIGH);
  Serial.println("Fire Detected!");
  SMS = "Fire Detected!";
  result = SendMessage();

  // Read GPS data
  if (gps.encode(gps_module.read())) {
   displayGPSInfo();

   // Send GPS data to Firebase
   sendGPSDataToFirebase();
  }
 } else {
  // No flame or smoke detected
  digitalWrite(BuzzerPin , LOW);
 }

 int command = CheckForValidCallSMS();

 // Handle commands based on SMS or call
 if (command == 1) {
  // Handle valid call
  if (relayOn == false) {
   digitalWrite(relay, HIGH);
   relayOn = true;
   delay(1000);
  } else {
   digitalWrite(relay, LOW);
   relayOn = false;
   delay(5000);
  }
  CutTheCall();
  delay(5000);
  CallNumber();
  delay(15000);
  CutTheCall();
```

```
} else if (command == 2) {
  // Handle invalid call
  CutTheCall();
} else if (command == 3) {
  // Handle ON SMS command
  digitalWrite(relay, HIGH);
  relayOn = true;
  SMS = "UNIT TURNED ON SUCCESSFULLY...!";
  result = SendMessage();
  while (result == 0) {
    SMS = "UNIT TURNED ON SUCCESSFULLY...!";
    result = SendMessage();
  }
} else if (command == 4) {
  // Handle OFF SMS command
  digitalWrite(relay, LOW);
  relayOn = false;
  SMS = "UNIT TURNED OFF SUCCESSFULLY...!";
  result = SendMessage();
  while (result == 0) {
    SMS = "UNIT TURNED OFF SUCCESSFULLY...!";
    result = SendMessage();
  }
} else if (command == 5) {
  // Handle STATUS SMS command
  if (relayOn == false)
    SMS = "UNIT IS OFF...";
  else
    SMS = "UNIT IS ON...";
  result = SendMessage();
  while (result == 0) {
    if (relayOn == false)
      SMS = "UNIT IS OFF...";
    else
      SMS = "UNIT IS ON...";
    result = SendMessage();
  }
} else if (command == 6) {
  // Handle invalid SMS keyword
  SMS = "INVALID KEYWORD...";
  result = SendMessage();
  while (result == 0) {
    SMS = "INVALID KEYWORD...";
    result = SendMessage();
  }
}
delay(1000);
```

```
}

int CheckForValidCallSMS() {
  if (SerialAT.available() > 0) {
    rxString = "";
    rxString = SerialAT.readString();
    if (rxString != "")
      Serial.println(rxString);
    String expectedString = "+CLIP:";
    rx = rxString.indexOf(expectedString);
    if (rx != -1) {
      // Phone is ringing; Check for authorized number
      rx = rxString.indexOf(number);
      if (rx != -1) {
        // Authentic Call
        Serial.println("Authentic Call...");
        return 1;
      } else {
        Serial.println("Un-Authentic Call...");
        return 2;
      }
    } else {
      expectedString = "+CMT:";
      rx = rxString.indexOf(expectedString);
      if (rx != -1) {
        // SMS is received; Check for authorized number
        rx = rxString.indexOf(number);
        if (rx != -1) {
          // Authentic SMS
          Serial.println("SMS From Authentic Number...");
          // Check for ON / OFF Commands
          Serial.println(rxString);
          rx = rxString.indexOf("ON");
          if (rx != -1) {
            Serial.println("TURN ON");
            return 3; //ON COMMAND
          }
          rx = rxString.indexOf("OFF");
          if (rx != -1) {
            Serial.println("TURN OFF");
            return 4; //OFF COMMAND
          }
          rx = rxString.indexOf("STATUS");
          if (rx != -1) {
            Serial.println("GIVE STATUS");
            return 5; //STATUS COMMAND
          }
```

```
        Serial.println("INVALID KEYWORD");
        return 6; //INVALID KEYWORD
       } else {
        Serial.println("SMS From Un-Authentic Number...");
        return 0;
       }
     } else
      return 0;
   }
  }
}

int SendMessage() {
  delay(5000);
  int res = 0;
  char end[2];
  end[0]=0x1a;
  end[1]='\0';
  SerialAT.println("AT+CMGS=?"); //9s
  rxString = SerialAT.readString();
  Serial.print("Got: ");
  Serial.println(rxString);
  rx = rxString.indexOf("OK");
  if (rx != -1) {
    Serial.println ("Sending Message");
    SerialAT.print("AT+CMGS=\"" + number + "\"\r"); //Mobile phone number to send message
    delay(1000);
    Serial.println ("1");
    SerialAT.print(SMS);
    SerialAT.println(end);// ASCII code of CTRL+Z
    Serial.println ("2");
    _buffer = _readSerial();
    SerialAT.println("AT+CMGD=4");    //Delete all read messages
    Serial.println ("3");
    delay(2000);
    Serial.println ("Message Sent");
    res = 1;
  }
  else
    Serial.println ("Can't Send Message");
  return res;
}

void sendGPSDataToFirebase() {
  if (gps.location.isValid()) {
    float latitude = gps.location.lat();
    float longitude = gps.location.lng();
```

```
    // Create JSON object for GPS data
    String json = "{\"latitude\": " + String(latitude, 6) + ", \"longitude\": " + String(longitude, 6) + "}";

    // Push GPS data to Firebase
    if (Firebase.pushString(firebaseData, "/gps_data", json)) {
      Serial.println("GPS data sent to Firebase!");
    } else {
      Serial.println("Failed to send GPS data to Firebase!");
      Serial.println(firebaseData.errorReason());
    }
  }
}

String _readSerial() {
  _timeout = 0;
  while  (!SerialAT.available() && _timeout < 12000  ) {
    delay(13);
    _timeout++;
  }
  if (SerialAT.available()) {
    return SerialAT.readString();
  }
}

void CutTheCall() {
  Serial.println("CUT THE CALL...");
  SerialAT.print(F("AT+CHUP"));
  SerialAT.print(F(";\r\n"));
  _buffer = _readSerial();
  Serial.println(_buffer);
}

void CallNumber() {
  Serial.println("CALLING THE ADMIN...");
  SerialAT.print(F("ATD"));
  SerialAT.print(number);
  SerialAT.print(F(";\r\n"));
  _buffer = _readSerial();
  Serial.println(_buffer);
}


void InitUnit() {
  pinMode(powerPin, OUTPUT);
  digitalWrite(powerPin, LOW);
  pinMode(relay, OUTPUT);
```

```
digitalWrite(relay, LOW);
pinMode(BuzzerPin, OUTPUT);
digitalWrite(BuzzerPin, LOW);
pinMode(LED2_PIN, OUTPUT);
analogWriteRange(255);
analogWriteFreq(freq);
analogWrite(LED1_PIN, 0);

Serial.begin(115200);
delay(100);
// Initialize DHT sensor
dht.begin();

// Initialize Firebase
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

SerialAT.begin(115200, SERIAL_8N1, RXD2, TXD2);
delay(10000);
Serial.println("Modem Reset, please wait");
SerialAT.println("AT+CRESET");
delay(1000);
SerialAT.println("AT+CRESET");
delay(20000);  // WAITING FOR SOME TIME TO CONFIGURE MODEM
SerialAT.flush();
Serial.println("Echo Off");
SerialAT.println("ATE0");   //120s
delay(1000);
SerialAT.println("ATE0");   //120s
rxString = SerialAT.readString();
Serial.print("Got: ");
Serial.println(rxString);
rx = rxString.indexOf("OK");
if (rx != -1)
  Serial.println("Modem Ready");
delay(1000);
Serial.println("SIM card check");
SerialAT.println("AT+CPIN?"); //9s
rxString = SerialAT.readString();
Serial.print("Got: ");
Serial.println(rxString);
rx = rxString.indexOf("+CPIN: READY");
if (rx != -1)
  Serial.println("SIM Card Ready");
delay(1000);
SerialAT.println("AT+CMGF=1");    // Sets the GSM Module in Text Mode
delay(2000);
SerialAT.println("AT+CMGD=4");    // Delete all read messages
```

```
 delay(2000);
 RecieveMessage(); // RECEIVE MESSAGE FROM THE MENTIONED PHONE NUMBER TO SIM
}
```

CHAPTER 6 - CONCLUSION AND FUTURE WORK

Chapter Outline

6.1] Conclusion

6.2] Future Enhancement

6.1] Conclusion :-

The Forest Fire Prevention System stands as a groundbreaking innovation in the ongoing battle against forest fires, harnessing the capabilities of IoT (Internet of Things) technology to revolutionise how wildfires are detected, monitored, and ultimately prevented. By amalgamating cutting-edge sensors, real-time communication infrastructure, and sophisticated data analytics, this system offers a proactive approach to forest fire management, significantly enhancing the ability to safeguard forests, wildlife, and human lives.

At the core of the system lies its ability to swiftly detect potential fire hazards through a network of advanced sensors strategically deployed across forested areas. These sensors continuously monitor environmental conditions such as temperature, humidity, and air quality, detecting any anomalies that may indicate the presence of a wildfire. This early detection capability enables firefighting agencies to initiate rapid response measures, including deploying firefighting teams and resources, well before the fire escalates into a catastrophic event.

The Forest Fire Prevention System also prioritises user accessibility and ease of use, incorporating user-friendly interfaces that allow firefighting personnel and decision-makers to visualise data, track fire incidents, and coordinate response efforts effectively. By providing actionable insights in a comprehensible format, the system empowers stakeholders at all levels to collaborate seamlessly and respond swiftly to emerging fire threats.

In essence, the Forest Fire Prevention System represents a paradigm shift in forest fire management, transcending the limitations of traditional monitoring methods and offering a scalable, adaptable solution to the ever-growing challenge of wildfire prevention. By harnessing the power of IoT technology, data analytics, and real-time communication, this system holds the potential to significantly reduce the impact of wildfires, preserving invaluable natural resources and safeguarding communities against the devastating effects of uncontrolled fires.

6.2] Future Enhancement:-

1. **Enhanced Sensor Technology:** Continued research and development in sensor technology can lead to the creation of more advanced and robust sensors capable of detecting a wider range of environmental parameters associated with fire risk.

2. **Integration of AI and Machine Learning:** Incorporating artificial intelligence and machine learning

algorithms can improve the system's predictive capabilities, enabling it to anticipate fire outbreaks based on historical data and environmental trends.

3. **Scalability and Accessibility:** Expanding the system's coverage to encompass larger forested areas and remote regions will require scalable infrastructure and solutions to ensure seamless data transmission and connectivity.

4. **International Collaboration:** Collaboration with international organisations and stakeholders can facilitate knowledge sharing, data exchange, and the development of standardised protocols for forest fire prevention and response.

5. **Community Engagement:** Engaging local communities in forest fire prevention efforts through education, awareness campaigns, and participatory monitoring initiatives can enhance the system's effectiveness and foster a culture of fire safety.

6. **Climate Change Adaptation:** Considering the impact of climate change on fire behaviour and frequency, future iterations of the Forest Fire Prevention System should incorporate adaptive strategies to mitigate evolving risks and challenges.

7. **Ecosystem Monitoring:** Expanding the system's capabilities to include monitoring of ecosystem health and biodiversity can provide valuable insights into the long-term impact of forest fires and inform conservation efforts.

8. **Policy and Regulation:** Policy support and regulatory frameworks are essential to incentivize the adoption of forest fire prevention technologies and ensure compliance with best practices in fire management and land use planning.

CHAPTER 7 - REFERENCES

1. https://srituhobby.com/how-to-set-up-the-new-blynk-app-step-by-step-nodemcu-esp8266-with-blynk-app/

2. https://srituhobby.com/how-to-make-a-home-automation-system-using-the-nodemcu-esp8266-board-and-the-new-blynk-app/

3. http://arduiniana.org/libraries/tinygpsplus/

4. https://circuitdigest.com/microcontroller-projects/iot-based-forest-fire-detection-using-arduino-and-gsm-module

5. https://eos.com/blog/wildfire-prevention/

6. https://www.bosch.com/stories/early-forest-fire-detection-sensors/https://www.bosch.com/stories/early-forest-fire-detection-sensors/

7. https://www.nature.com/articles/s41598-021-03882-9

8. https://www.arduino.cc/reference/en/libraries/