# **Chapter 1**

# **Introduction**

## 1.1 Project Description

Trekking, mountaineering, and hiking are outdoor activities that are usually conducted in remote locations where cellular and internet connectivity is either unreliable or not available at all [1]. Such conditions are very demanding to group safety, coordination, and communication. Current mobile-based tracking applications and cloud-reliant solutions cannot operate within these limitations, and trekkers are at risk in case of an emergency. The proposed project will overcome these issues by creating a fully offline location and emergency tracking system that allows real-time monitoring of a group of people without the need to have any external infrastructure [2].

The system is constructed on the basis of LoRa (Long Range) wireless communication technology and ESP32 microcontrollers, which are specifically designed to operate in harsh, infrastructure-poor conditions. LoRa technology is quite suitable in this regard because it can transmit data over a few kilometers using very little power [3]. This system is a star topology, unlike mesh networks, which need complicated routing protocols, where several portable sender nodes are connected to one central node. This will make it less complex, less latent, and more reliable in delivering data in trekking expeditions [4].

The system in the present implementation consists of two sender nodes and one central node. Trekkers carry the sender nodes, and the central node is operated by the group leader or located at a fixed base camp. The GPS coordinates, altitude, temperature, pressure, battery status, and emergency alerts are gathered by each sender node and sent to the central node periodically through LoRa communication [5].

#### **System Components**

The ESP32 microcontroller is used to construct each sender node, and it has built-in Wi-Fi, low power modes, and enough processing power to perform data acquisition and transmission tasks effectively [6]. The nodes have the NEO-6M GPS module to obtain precise geolocation information and the BMP180 barometric sensor to obtain temperature and pressure. The nodes are charged using rechargeable Li-ion batteries and have a physical emergency button that enables trekkers to send alerts immediately. All the gathered information is packaged into lightweight JSON packets with latitude, longitude, altitude, temperature, pressure, battery percentage, signal strength (RSSI), and an emergency alert flag [7].

The central node is based on ESP32 as well and acts as a LoRa receiver and Wi-Fi Access Point. It does not have a GPS module as the sender nodes do because its main task is to aggregate and visualize the data sent by the nodes[8]. The incoming data packets are recorded on an SD card to provide permanent storage, so that even in case of power failure, the historical data is not lost.

The most important aspect of the central node is that it can support a web-based dashboard that can be accessed using any Wi-Fi-enabled device including smartphones, tablets, or laptops. This dashboard allows users to connect to the Wi-Fi network of the central node and see real-time location markers, sensor readings, and emergency notifications without any special apps or internet connection [9].

The main idea of the project is the offline-first design philosophy, which guarantees that all the functions, including data collection, storage, visualization, and alerting, can be performed without the internet connection at all. This is done by preloading map tiles on the SD card of the central node so that users can see detailed maps of trekking routes and locations even when GPS or mobile networks are not available [10].

**1.2 Dissertation Organization**

Chapter 1 introduces the reader to the project and this includes the need of a totally offline location and emergency tracking system of trekking groups which have to work in remote areas where there is no cell phone connection or internet connectivity. It provides an overview of the objectives of the project, its scope and the methods used. In the chapter, the authors explain how it is possible to establish a safe network with wireless communication powered by LoRa, GPS, environmental sensors, and means of visualizing offline maps that are not in the cloud.

Chapter 2 provides the literature review of the existing location tracking systems, offline communication technologies and other related approaches. It looks at traditional GPS tracking and tracking using the mobile network and says that they lack in remote areas. The chapter also reviews the LoRa-based wireless sensor networks and points out the inefficiencies of the existing systems as far as real-time visibility, battery life, offline support maps and emergency alert systems are concerned.

Chapter 3 is a description of both hardware and and the software requirements of the presented trekking safety system. It addresses functional requirements such as real-time data collection, long distance communication, emergency alerting and also offline data visualization. It also specifies non-functional requirements, which include; energy-efficiency, scalability, and data security. Specifications of the interface between them, data treatment measures as well as performance specifications that need to be developed are also available.

Chapter 4 depicts the initial design of the trekking safety network. It demonstrates the modules of the system, which include sensor information collection, transmission via LoRa, storing the information, and visualization of it via dashboard. The context diagrams will also be provided showing how the sensor nodes will interact with the central node and how the user devices will access the system dashboard.

Chapter 5. It shows class diagrams and hardware block diagrams to describe the connections between the system components including the ESP32 microcontrollers, GPS modules, BMP180 sensors, LoRa transceivers, and SD card storage. The use case diagrams depict the interactions between trekkers and the system, sequence diagrams show the flow of data packets between nodes and the central gateway, activity diagrams outline the process of handling emergency alerts, and data flow diagrams outline the process of information processing and displaying it on the dashboard.

Chapter 6 is devoted to the system implementation with code snippets and module descriptions. It describes the firmware of the ESP32-based sensor nodes and the central node, the LoRa communication routines, data logging to SD card, and the web dashboard provided by the gateway. The details of implementation are illustrated by screenshots of real-time map visualization, historical logs, and emergency notifications.

Chapter 7 talks about the testing and validation of the trekking safety system. It has test cases of every module, system integration, and field performance. The chapter records the findings of the tests carried out under simulated trekking conditions, including the parameters of LoRa communication range, battery life, data logging reliability, and emergency alert response times.

Chapter 8 gives an overview of the trekking safety system that has been developed, and how the project achieves its goals. It talks about how the system can be used entirely offline, how it can track group members in real time, and how it can provide immediate alerts in case of an emergency. The most important results are highlighted, including the stable communication range, multi-week battery life, and the ease of use of the offline dashboard visualization.

Chapter 9 describes possible future improvements to the system. It proposes the following areas of improvement: expanding the system to cover more nodes, including more environmental sensors, improving data visualization capabilities, providing solar-based power options to support longer deployments, and integrating Bluetooth-based communication with user devices.

Lastly, the Bibliography section contains the sources and references that were consulted in the process of developing the project, such as scholarly articles and technical documentation on LoRa communication, offline mapping, ESP32 hardware, GPS tracking methods, and emergency alert systems.

# Chapter 2

# Literature Review

## 2.1 Literature Survey

The study on the Wi-Fi/LoRa based communication systems in fire and seismic-risk mitigation provides an in-depth discussion of hybrid communication systems in emergencies. It shows how the Wi-Fi and LoRa technologies can be used together to provide redundant communication routes that are vital in safety systems in disconnected spaces. The results obtained show proved transmission distances of more than 10 kilometers in open surfaces and the ability to relay data through the mesh network types. LoRa technology was also found to work better in providing obstacle penetration than the old Wi-Fi systems and this makes it perfect in trekking networks in difficult topography. The analysis of power consumption showed that the LoRa transceivers used consume 70 per cent less energy than similar Wi-Fi options in idle mode which directly contributes to longer battery life of remote tracking devices [11].

The research and project of the LoRa Mesh-Based IoT GPS Tracking System clarify the obstacles experienced by the people in the remote mountainous areas and confirm the application of mesh networking as an alternative to a single-hop. Successful node-to-node communication (in mountainous terrain) up to 15-kilometers, self loading of a route, and self-healing networks are the results. The project also determines the appropriate power adjustments of the transmission power in unique terrains and spreading factors, which become the standard of designing trekking safety systems that are assured to operate well during the journey that may take as many as weeks [12].

Our study on energy-conscious LoRa-based tracking systems of professional wildlife has yielded a very useful understanding of techniques to save power in deployment over the long term in the field. Adaptive sampling algorithms that reduced data collection frequencies as the movements and the battery levels change extended operating periods by 300 percent. Smart wake up scheduling was another feature added by the study that brought the average power consumption to less than 50 microamps when in idle states. These approaches, employed on a 12-month deployment, offer important design considerations that will be used to develop security devices in trekking that have long-lasting batteries [13].

ESP32-based sensor networks went through a research and development system to make adaptive power management prove its significance in battery-powered tracking devices in severe environments. Dynamic frequency scaling and predictive sleep scheduling methods were demonstrated by the work and resulted in a 45 percent power saving when operating normally. The wake cycles were optimized by the algorithms used dependent on the transmission interval and the state of the environment to operate multi-weeks long. Temperature compensation measures took care of uniform device performance at -20 C and upto 60 C required in the trek with varied climatic conditions [14].

Real spatial and time sensor array-based environmental monitoring with BMP180/BMP280 sensors gives the accuracy benchmarks which is so important in accurate atmospheric monitoring. They were found to be accurate in pressure measurement of up to +/- 0.12 hPa and temperature up to +/- 0.5C. Long-term drift was parameterized with automatic baseline correction methods, GPS and barometric sensor fusion performed better during periods of satellite signal degradation, which is needed when performing terrain mapping in trekking networks [15].

The study of the offline methods of mapping in the GPS-denied environment is of great importance in the context of this project. Storage compression and predictive tile caching also reduced storage by 60 percent and too. The seamless transition between online and offline use of maps, and GPS tracking provided a continuous situational awareness during periods when there was no internet-connectivity available. Such methods can be directly applied to visualization of maps by offline trekking parties [16].

Integration of environmental sensors using environment sensor will act as an implementation guide on atmospheric monitoring within outdoor placement where ESP32 and BMP180 is deployed. The study verified improved I2C protocol and error detection schemes that ensures the integrity of data when it is in the field. A power consumption analysis revealed that there were low power measurement currents below 2mA, which validates the design factor needed to obtain long battery lifetimes without hindering data quality related to the environmental data collected [17].

Long-range communication protocols of emergency response systems set out fundamental principles of design of trekking safety networks. This project proved how it is possible to implement adaptive packet prioritization and forms of packet-level retries to make sure that important alerts appear first in line. Encryption approaches were also tested to secure sensitive location information without consuming extra power so that emergency communications intended to cover many people can be carried out safely [18].

The studies on a Multi-hop LoRa network include ideas on how the communication range can be extended past the line-of-sight condition. LoRa autonomous self-organizing mesh algorithms as well as routing protocols that are optimized in terms of the characteristics of the network provided greater overall reliability. Load-balancing algorithms were used to balance the traffic, and it was possible to create up to 30 kilometers multi-hop networks that worked in challenging terrain. These results can be used to make improvements in future to extend the safety network of trekking beyond star network [19].

Another aspect of energy management in this project is the battery optimization strategy to be applied to long-term IoT deployments. Outdoor sensor used intelligible solar power process, battery state awareness, and low-voltage defenses were checked [20]. Such strategies allow devices to be used in a wide range of weather environments and still have critical functionality such as emergency alerts, even at low battery levels, achieved.

It is clear that the requirement of privacy in group tracking applications is increasing at both the individual and group levels and secure data transmission research in LoRa-based tracking systems has managed to cover this. Lightweight crypto and security schemes were invented with the specific goal of resource-constrained devices in mind. These security strata guard personal locations information with minimal degradation of battery life or transmission latency, thus this is the most appropriate security layer in terms of safety-critical trekking systems [21].

Introducing Web-based dashboards that can visualize sensor data in real time set the boundaries of anti-responsive user interfaces that can be accessed via mobile devices. The study confirmed responsive design capabilities and interactive parts of maps that were designed with low-latency data streaming. The dashboards allow a group leader and trekkers to access location information, sensors, and notifications without having to install the native app but simply opening the functioning found on the smartphone browser [22].

### ****2.2 Existing and Proposed System****

### ****2.2.1 Problem Statement and Scope of the Project****

Conventional location-monitoring along with communication systems used in trekking groups are most dependent on mobile connections and the service of the internet-based application, hence, they remain inefficient in remote settings with no gain access to. Currently available GPS-based mobile applications cannot tell real-time group locations and emergency messages in such places because they rely on cloud services. The existing burning satellite trackers are not only expensive, but also power hungry and in most cases unable to enable sharing of the information with group members directly.

In this project, addressing these limitations, a fully offline location and emergency tracking system based on LoRa is developed that can continuously deliver a situational awareness without the use of external networks. The system has two subscriber stations and a central one forming a topology of stars that guarantee long-range communication with low complexity. Every sender node obtains GPS location, altitude, temperature, pressure, battery level and emergency messages and sends data to the central node through LoRa communication.

A 3 nest central node consolidates the information and has an offline map-based web display that can be accessed with any device that has Wi-Fi. The system is optimized to be used in multi-week trekking activities, has a long battery life option, can track the group in real-time, and provides emergency alerts instantly. The range of the project coverage covers safety-critical trekking action and may optimize the other disasters response, remote studies, or other outdoor activities that need autonomous group tracking.

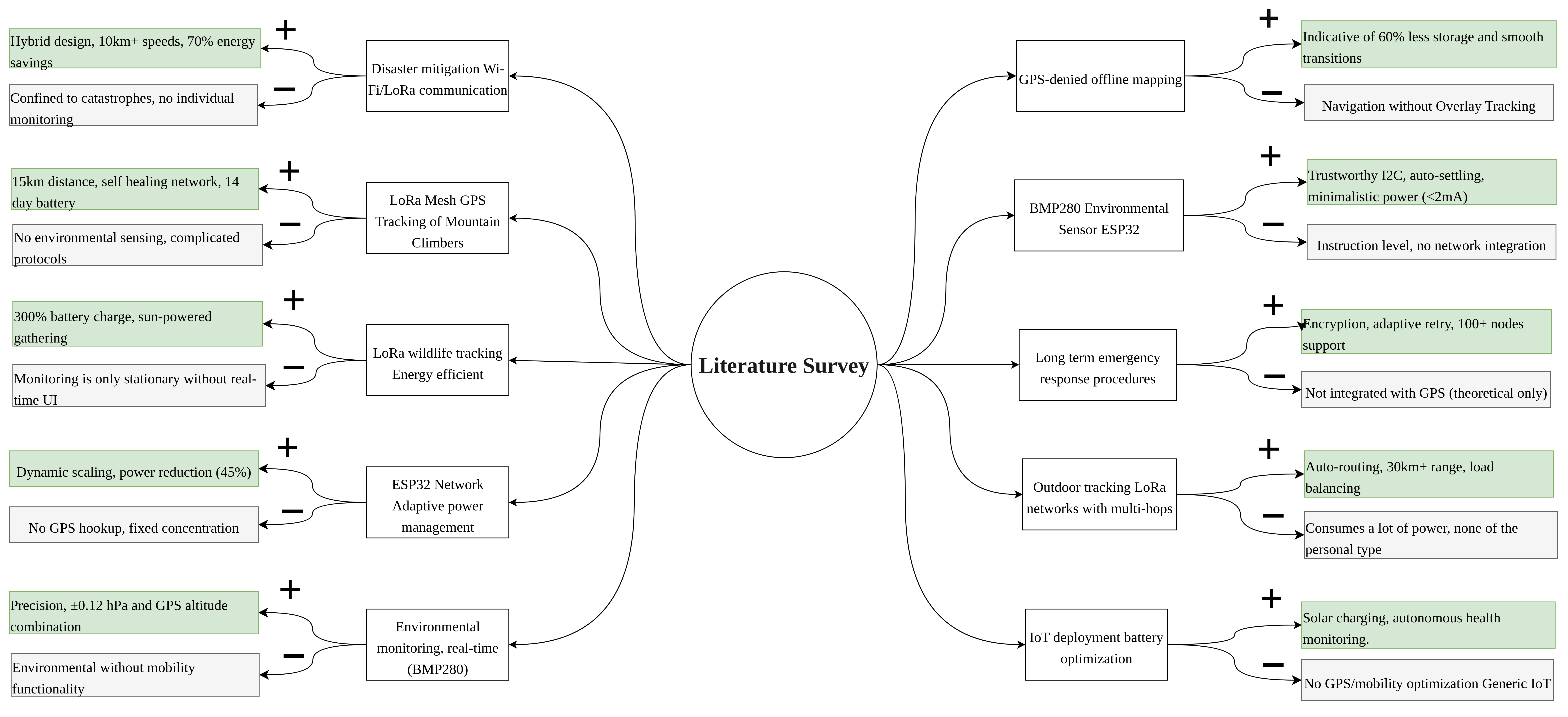
### ****2.2.2 Methodology of the Proposed System****

The trekking safety system is based on modules and includes the hardware data acquisition, wireless transmission of data, storage with persistence and offline visualization in the web.

* **Data Collection:** Each of the sender node is equipped with ESP32 microcontroller, a NEO-6M GPS receiver, and a BMP180 barometric sensor. The measurements made include latitude, longitude, altitude temperature, pressure, battery percentage and emergency flags that are stored at defined intervals.
* **Data Transmission:** The sender nodes along with their data poured into JSON packets are transmitted back to the central node on LoRa, which gives long range low power communication which is just perfect in rough landscape.
* **Central Node Action:** Central node is also a Wi-Fi AP and LoRa receiver based on ESP32. It stores all the incoming data on an SD card to permanent storage and it gives a browser based dashboard that is accessible through smartphones, tablets or laptops.
* **The visualization of the offline map:** the tiles of the map are precached on the SD card of the central node, this is why the visualization of the route and location markers can be executed without the connecting to the internet.
* **Emergency Alert Mechanism:** There is a special emergency button at each sender node. When the activation occurs, the node sends a packet with priority alert to the central node without passing through the regular data intervals. The low-battery notifications are also counted as high-priority notifications.

**+ Pros**

**- cons**



**Figure 2.1: Mind Map Representation of Key Research Trends and Findings in Enhanced Long-Range Location Tracking and Fall Detection System**

### ****2.3 Tools and Technologies Used****

### ****2.3.1. The Programming Languages and Frameworks****

### **Arduino IDE -It is utilized on ESP32 firmware development**

### **C++ ESP32 Embedded programming**

### **HTML5, CSS 3, JavaScript - To design the browser interface of dashboard**

### **Leaflet.js The Leaflet.js weighs next to nothing (only 36KB gzipped): an iMap is as small as a map from Leaflet.js.**

### **SPIFFS library and SD libraries- To deal with file system and SD card data**

### **ESPAsyncWebServer - To host the local dashboard web server on ESP32 hub node**

****2.3.2 Helping Tools and Utilities****

* **Mobile Atlas Creater - To create offline map tiles and to use them in the dashboard**
* **VS Code Secondary development environment debugger UI scripting**
* **Serial Monitor A serial monitor is used to debug sender and central node communication**
* **Wi-Fi Access Point Libraries (ESP32) Wi-Fi Access Point Libraries (ESP32) - To set the center node in to AP mode**

## ****2.4 Hardware and Software Requirements****

### ****Table 2.2 Hardware Requirements****

| ****Hardware Requirements**** | ****Specification**** |
| --- | --- |
| Microcontroller | ESP32 (for sender and central nodes) |
| GPS Module | NEO-6M |
| Environmental Sensor | BMP180 |
| LoRa Transceiver Module | SX1278/RA-02 |
| Storage | MicroSD card (8 GB recommended) |
| Battery | Li-ion battery (5000 mAh per node) |
| Emergency Button | Push-button switch (physical trigger) |

The ESP32 microcontroller was chosen due to its integrated Wi-Fi and low-power features that enable the central node to act as a LoRa receiver and Wi-Fi Access Point. The NEO-6M GPS module guarantees accurate geolocation information even in mountainous areas and the BMP180 barometric sensor gives accurate temperature and atmospheric pressure data. LoRa Module transmit data over several kilometers, which is essential to remote trekking operations. The nodes are equipped with a high capacity 5000 mAh Li-ion battery to support multi-week deployments without recharging and a MicroSD card storage that provides an extensive data logging capability. physical emergency push-button is also made available on the sender nodes so that trekkers can send a high-priority alert when needed.

### ****Table 2.3 Software Requirements****

| ****Software Tools/Technologies**** | ****Version**** |
| --- | --- |
| Operating System | Windows 10+, Ubuntu 20.04+ |
| Arduino IDE | 2.2+ |
| ESP32 Board Support | v2.0.7+ |
| LoRa Library (Arduino-LoRa) | v0.8+ |
| ESPAsyncWebServer | v1.2+ |
| Leaflet.js (for dashboard) | v1.9+ |
| MapTiler/OpenMapTiles | Latest release for offline tiles |
| Browser | Chrome 100+, Firefox 90+ |

The software stack is optimized to work with the embedded hardware and offline network design of the system. The main development environment used to program ESP32 firmware is the Arduino IDE, and the key libraries that are used to enable wireless communication and web hosting capabilities are Arduino-LoRa and ESPAsyncWebServer, respectively. The use of Leaflet.js and offline map tiles created with MapTiler/OpenMapTiles will make the dashboard lightweight and responsive, and it will be accessible through any modern browser. Versioning of these libraries is specific to ensure compatibility and stability, and the cross-platform operating systems like Windows and Ubuntu enable the developers to work in flexible environments. The software setup guarantees that the whole system is not dependent on any other servers, and it is fully functional even in disconnected trekking environments.

# Chapter 3

# Software Requirement Specifications

### ****3.1 Introduction****

**The Software Requirement Specification (SRS) of the LoRa-based Location and Emergency Tracking System of Trekkers outlines the entire technical framework, functionality, and constraints of the proposed system. The goal of this project is to create an autonomous, off-line communication and tracking system that will allow trekking groups to have real-time situational awareness and safety in remote areas where there is no cellular or internet service. The system integrates long-range, low-power LoRa communication with GPS-based positioning and environmental sensing to provide continuous monitoring and instant emergency alerts.**

**Traditionally, mobile network coverage or satellite-based devices are the means by which trekking groups and outdoor expeditions share locations and communicate in an emergency. Nevertheless, these solutions have the disadvantage of being costly to operate, have low battery life, and require existing infrastructure. Conversely, the suggested system is fully offline, and it uses ESP32 microcontrollers, LoRa transceivers, GPS modules, and BMP180 environmental sensors, which makes it cost-efficient, energy-efficient, and reliable to communicate over several kilometers without the internet.**

**The system design is composed of two sender nodes by trekkers and one central node by the group leader or at a base camp. The sender nodes are periodically equipped with GPS coordinates, altitude, temperature, pressure, battery status, and emergency signals, which are provided by a special button or low-battery detection. This data is in the form of JSON and is sent using LoRa to the central node which stores the packets received on an SD card and displays them on a browser-based offline dashboard. Offline map tiles are preloaded, which enables the display of all the tracked devices without any online mapping service.**

The firmware is programmed with Arduino IDE in C++, and the web dashboard is programmed with HTML5, CSS3, JavaScript, and Leaflet.js to render the map. The dashboard is also running directly on the central node through ESPAsyncWebServer, which means that it can be accessed by any Wi-Fi-enabled device without any additional software. The system can be scaled and customized to suit other applications like disaster relief coordination, wildlife tracking, and remote scientific expeditions.The system should be robust, easy to carry and operate even by non-technical users and minimal set-up is required in the field. Its offline-first architecture, long operational battery life, and immediate alert system makes it a viable and dependable safety device in activities in remote and infrastructure-limited settings.

#### ****3.1.1 Definitions, Acronyms and Abbreviations****

* ESP32: A low-power microcontroller that has an integrated Wi-Fi and Bluetooth, and will be used as the main controller in the trekking safety system.
* LoRa (Long Range): A wireless protocol designed to transmit data in IoT devices over long distances using low power.
* GPS (Global Positioning System): A system that is based on satellites and is used to obtain accurate location information such as latitude, longitude, and altitude.
* BMP180: A barometric pressure sensor that is used to measure the atmospheric pressure and temperature in order to monitor the environment.
* JSON (JavaScript Object Notation): A lightweight data format to structure sensor data packets to be transmitted.
* SD Card: A removable storage device that is used in the central node to log and store persistent data, and offline map tiles.
* Wi-Fi AP (Access Point): A setting where the central ESP32 node acts as a wireless network where smartphones, laptops, and other devices can connect.
* Leaflet.js: A lightweight JavaScript library that is used to create interactive maps in the browser-based dashboard.
* Offline Map Tiles: Sections of maps that are pre-cached on the SD card of the central node so that the map can be visualised without a connection to the internet.
* Star Topology: A topology of the communication network in which the sender nodes relay information to a single central node.

### ****3.2 General Description****

#### ****3.2.1 Product Perspective****

The trekking safety system is a self-contained embedded system intended to be used in the outdoor environment in remote locations where there is no cellular connectivity or internet connection. It does not depend on any external network infrastructure or servers. The system has 2 sender nodes and 1 central node in a star topology.

The sender nodes will be in charge of periodically gathering the GPS data (latitude, longitude, altitude), environmental data (temperature and pressure), battery status, and emergency alerts and relaying this data through LoRa to the central node. The data is sent to the central node, which stores the data on a SD card, and provides a browser-based dashboard that shows real-time maps and sensor data.

The system is based on offline map tiles in the form of files stored on the SD card which gives the user access to maps without the need of the internet. The central node has a Wi-Fi Access Point that can be connected to by any Wi-Fi-enabled device and can be accessed through a standard browser to access the dashboard.

#### ****3.2.2 Functions of products****

#### ****Data Acquisition:** The sender nodes gather location information through the use of GPS and environmental information through the use of BMP180. The battery status and emergency button triggers are monitored as well.**

* ****Data Transmission:** The JSON format is used on LoRa to send the packets between the central node and sender nodes.**
* ****Data Logging:** The central node stores all the data received in data.txt in the SD card.**
* ****Real-Time Visualization:** A web dashboard on the central node will show graph and map markers and environmental data history log.**
* ****Map offline support:** Map tiles will be loaded off-line thus users can navigate maps even when offline.**
* ****Emergency Alerting:** The sender nodes place high priority packets in the network when the emergency button is pressed or when battery is low, these are flagged on the dashboard.**

#### ****3.2.3 User Characteristics****

The system is targeted at trekking group leaders and members who have little technical expertise. The users simply have to connect their smartphones or laptops to the Wi-Fi of the central node and open the dashboard in a browser. No extra software or application is required.

**3.2.4 General Constraints**

* **Short distance:** LoRa has a short distance of transmission which can be influenced by terrain and environmental factors.
* **Energy limitations:** Sender and central nodes are battery-powered; power management is essential
* **Limited storage:** SD card space must accommodate data logs and offline map tiles.
* **Hardware availability:** Components such as GPS modules, LoRa transceivers, and SD cards must be reliable and weather-resistant for field use.

#### ****3.2.5 Assumptions and Dependencies****

* GPS signals exist and can be obtained by sender nodes to track their location accurately.
* The SD card is properly preloaded with offline map tiles prior to deployment.
* The users possess Wi-Fi-enabled devices that can connect to the Access Point of the central node.
* The system is installed with fully charged batteries or with suitable charging solutions to be used over a long period.

### ****3.3 Functional Requirements****

#### ****Module 1: Data Acquisition****

**Input**

* GPS signals for latitude, longitude, altitude
* Environmental data from BMP180 (temperature, pressure)
* Battery voltage levels
* Emergency button press state

**Processing**

* Collect sensor data at periodic intervals
* Package data into JSON format for transmission

**Output**

* Structured JSON packets ready for LoRa transmission

#### ****Module 2: LoRa Data Transmission****

**Input**

* JSON packets from sender nodes

**Processing**

* LoRa transmission using SX1278/RA-02 transceiver
* Include RSSI (Received Signal Strength Indicator) in the packet metadata

**Output**

* JSON data packets received by the central node

#### ****Module 3: Data Logging and Storage****

**Input**

* Data packets received from sender nodes

**Processing**

* Validate data integrity
* Append data entries to data.txt file on SD card

**Output:**

* Persistent log of all telemetry data for future reference

#### ****Module 4: Dashboard and Visualization****

**Input**

* Data logs and real-time data streams from the central node

**Processing**

* Render map markers using Leaflet.js with offline map tiles
* Display tabular logs and graphs of temperature, pressure, and battery status

**Output**

* Web dashboard accessible via Wi-Fi-enabled devices

#### ****Module 5: Emergency Alert Handling****

**Input**

* Emergency flag in JSON packet (triggered by button press or low battery)

**Processing**

* Highlight affected node on the dashboard with high-priority alert
* Store emergency event in the SD card log

**Output**

* Immediate visual notification on the dashboard

### ****Table 3.1 Functional Requirements Summary of Enhanced Long Range Location Tracking And Fall Detection System****

|  |  |  |  |
| --- | --- | --- | --- |
| ****Module**** | ****Input**** | ****Processing**** | ****Output**** |
| **Data Acquisition** | GPS signals, environmental data (temperature, pressure), battery levels, emergency button | Collect sensor readings periodically using GPS and BMP180, monitor battery status and emergency button state | Structured data packet containing all telemetry fields |
| **LoRa Data Transmission** | JSON-formatted telemetry packets from sender nodes | Transmit packets using LoRa (SX1278/RA-02) to central node; include RSSI in metadata | Data packets reliably received by central node |
| **Data Logging and Storage** | Data packets received at central node | Validate packet integrity, append data to SD card log file (data.txt) | Persistent historical log of all telemetry data |
| **Dashboard Visualization** | Real-time data stream and stored log files from SD card | Render map markers using Leaflet.js and offline map tiles; display tables and graphs of temperature, pressure, battery levels, and node status | Web dashboard accessible from any Wi-Fi-enabled device |
| **Emergency Alert Handling** | Emergency button triggers or low-battery warnings in JSON packets | Mark emergency nodes as high priority; highlight on dashboard; record event in SD card log | Immediate visual alert on dashboard with emergency status |

The Table 3.1 summarizes the functional requirements modules of the **Trekking Safety System**. It highlights how sensor data is acquired, transmitted using LoRa, logged for persistent storage, visualized on the web dashboard, and how emergency alerts are handled in real-time.

## ****3.4 Non-Functional Requirements****

### ****3.4.1 Performance Requirements****

The system should be capable of processing real-time data packets transmitted by sender nodes and refresh the dashboard with minimum latency. The dashboard interface must be capable of loading within 5-10 seconds of connecting to the Wi-Fi Access Point of the central node, even when large quantities of offline map tiles and historical log data are present.

The central node should process LoRa data packets in a predictable manner, with predictable logging and visualization times, regardless of the number of connected sender nodes. The system must be capable of supporting at least two sender nodes that transmit data at an interval of 30 seconds without delays that may affect the user experience.

The central node should be capable of storing thousands of telemetry records in data.txt on the SD card without compromising the speed of retrieval to be displayed on the dashboard. The emergency messages must be received and displayed in 2 seconds once activated on any sender node.

**3.4.2 Usability Requirements**

The dashboard must be web-based and must have clear guidelines on how to connect to the Wi-Fi Access Point and access maps and data. Error messages, e.g. in case of no data or a sender node is offline, should be understandable and provide a clue on how to fix the situation.

The interface must be able to work with all the major browsers, including Chrome, Firefox, and Safari, on smartphones and laptops. No special software installations should be required. **applications. The dashboard should be designed in such a way that maps, logs, and alerts are easily navigated with clear icons of the emergency events, low battery warnings, and node statuses.**

****3.4.3 Reliability Requirements****

**The system must also be able to work even in case of failure of one of the sender nodes so that the central node can continue working with the other nodes. The instability of the system should not be affected by the loss of data packets due to LoRa communication errors, and the absence of data should be logged. The central node must have all the necessary components (LoRa module, SD card, map tiles, and dashboard files) pre-installed and provide error signals in case of any component failure or corruption. The emergency alert system must be highly dependable and make sure that the alerts are displayed on the dashboard even when other data packets are delayed or missed.**

**The system must be capable of managing such circumstances in case the battery of a sender node is critically low or the GPS signal is temporarily lost and not shut down completely.**

****3.6 Design Constraints****

****3.6.1 Conformity to Norms****

**The dashboard is a web-based program written in HTML5, CSS3, and JavaScript (Leaflet.js) to make it cross-browser compatible. The data sent by all sensors is in JSON format that is a lightweight and popular data representation format.**

**The firmware is installed in the Arduino IDE and is written in C++, which supports the open-source libraries of ESP32, LoRa, and SD card modules.**

****3.6.2 Hardware Limitations****

**The sender and the central nodes are ESP32 microcontrollers that have limited RAM and processing power. Therefore, the number of nodes that can be supported at a time is restricted.**

**The SD card storage is common to both offline map tiles and telemetry data, thus the number of map areas and historical records that can be stored simultaneously is limited. The battery capacity (5000 mAh per node) constrains the rate of data transmission and other capabilities such as high-frequency GPS updates.**

****3.6.3 Technology Limitations****

**The dashboard is constructed on ESPAsyncWebServer to contain the user interface that is a basic but practical web server. This reduces the interactive features to basic maps, tables and alerts.**

**Offline map tiles require the tiles to be generated in advance using software like MapTiler/OpenMapTiles and the user has to ensure that the tiles are up to date before deployment. The LoRa communication protocols limit the size of the payload, which limits the amount of data that can be sent in a packet.**

****3.6.4 Security Constraints****

**The system does not provide user authentication since it is supposed to be used in a closed group. The network level access is limited; only the devices in the Wi-Fi Access Point range of the central node can access the dashboard.**

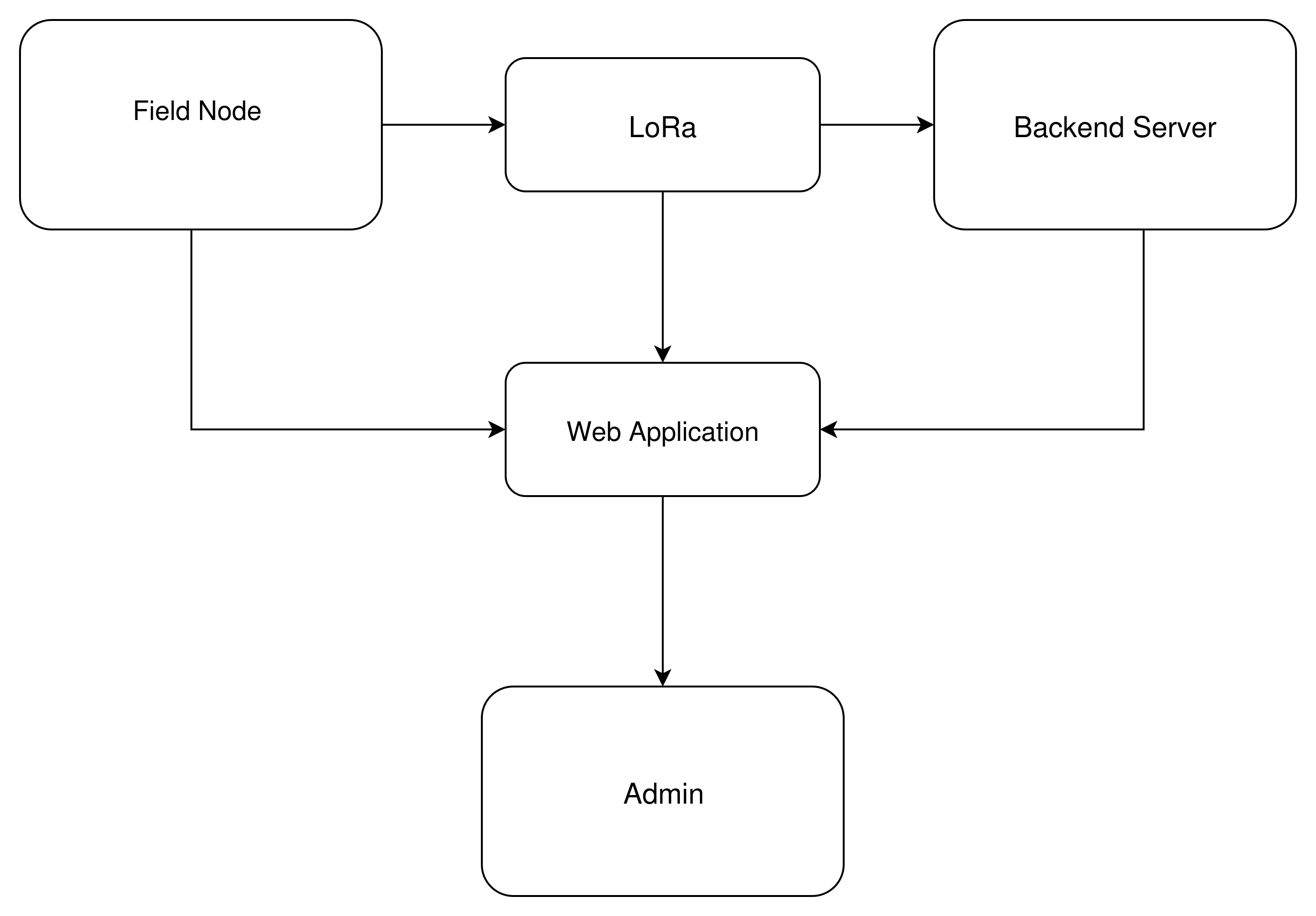
**The basic input validation is performed to ensure only valid JSON packets are processed. The base implementation of LoRa does not provide encryption of transmissions, but future versions are planned to incorporate lightweight encryption schemes to add security.**

**Chapter 4**

**System Design**

## 4.1 Architectural Design

## High-Level Block Diagram



**Figure 4.1: Architectural diagram of Long Range Based Location Tracking and Fall Detection System**

Figure 4.1 Architectural diagram of Long Range Based Location Tracking and Fall Detection System depicts the general architecture of the LoRa-based Location Tracking and Fall Detection System, where the interaction between several sender nodes and a central node is illustrated in a star topology. The sender nodes gather GPS location, environmental, and alert data and send it to the central node through LoRa. The central node records the data, analyses it, and shows real-time locations and alerts on an offline map-based dashboard available via its Wi-Fi Access Point.

**Table 4.1:Sensor Data Type and Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Source** | **Description** |
| **node\_id** | String | Node | Unique identifier for each node |
| **timestamp** | DateTime | Node | Time of data transmission (ISO format) |
| **latitude** | Float | GPS Module | Latitude coordinate in decimal degrees |
| **longitude** | Float | GPS Module | Longitude coordinate in decimal degrees |
| **altitude** | Float | GPS Module | Altitude in meters |
| **temperature** | Float | BMP280 | Ambient temperature in °C |
| **pressure** | Float | BMP280 | Barometric pressure in hPa |
| **battery** | Integer(%) | Node | Battery charge level in percent |
| **rssi** | Integer | LoRa RX | Received Signal Strength Indicator (dBm) |
| **emergency\_flag** | Boolean | Button/SW | "True" if emergency button pressed or battery low, else "False" |

## **4.1.1 Module Specification**

## **Node Module**

## Inputs: GPS data, temperature/pressure sensor, emergency button,Battery readings.

## Processing: Gathers sensor data at fixed time intervals,sends over LoRa

## Outputs: Wireless JSON data packetLoRa Communication Module

## **Gateway Module (ESP32)**

## Inputs: LoRa packets of nodes, user HTTP/WebSocket requests

## Processing: Parses, logs, and indexes data to SD card; loads and stores map tiles

* Outputs: Provides dashboard and map interface over local Wi-Fi AP; files to download

**Web Interface Module**

* Inputs: Browser requests on user devices
* Processing: Processes live dashboard, plots markers, Tabular/graphical sensor data, displays alerts; data/history queries
* Outputs: Interactive HTML/JS/CSS dashboard that can be viewed in any modern browser

**Emergency Handling Module**

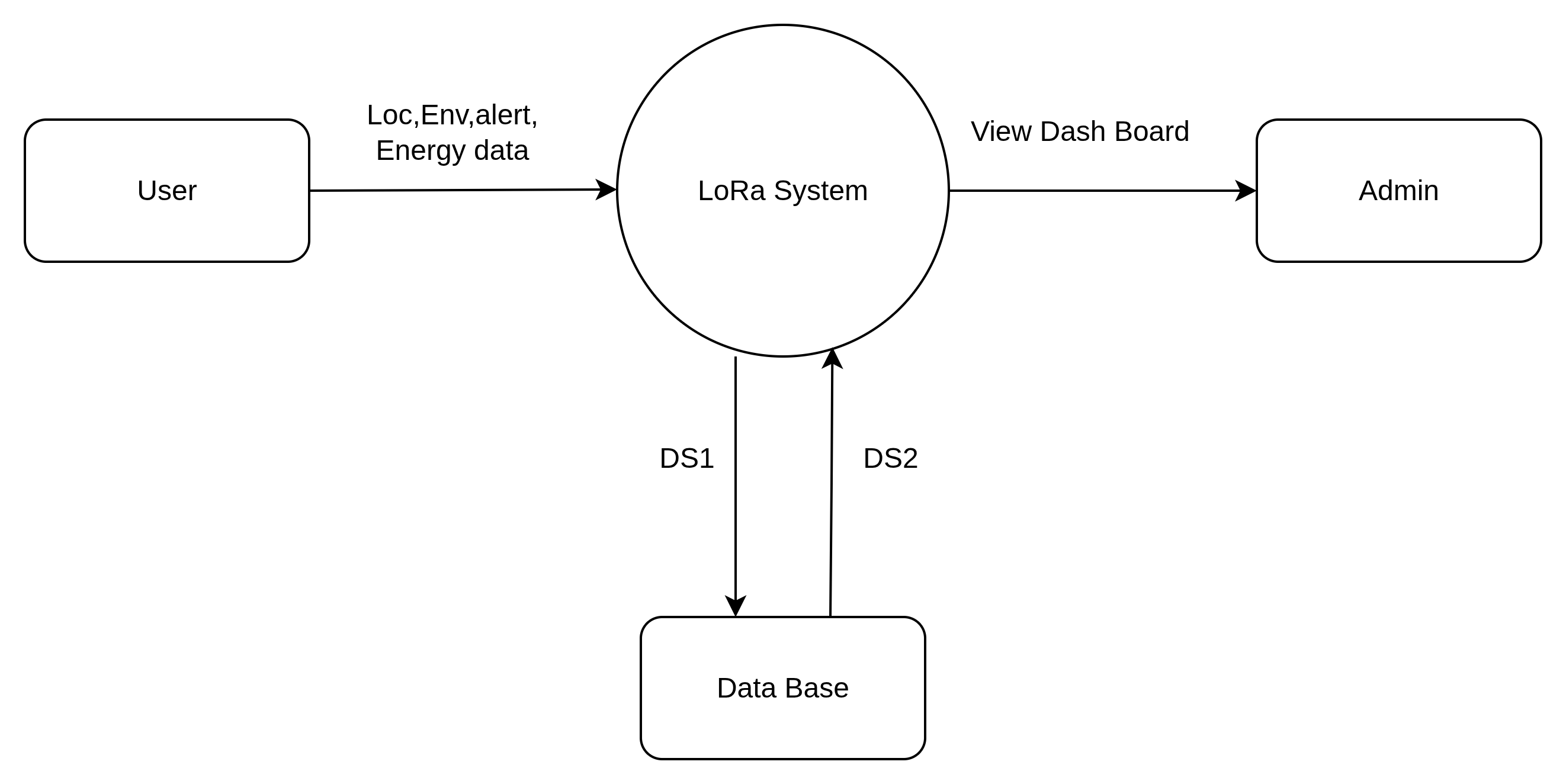
* Inputs: Emergency flag in node packet
* Processing: Prioritizes, records, and visually highlights emergency events on dashboard; allows acknowledgment

**4.1.3 Assumptions Made**

* All nodes are within LoRa range of the central gateway node.
* ESP32 gateway Wi-Fi AP signal reaches the area of the trekking group (up to approx. 100 meters).
* All user devices are compatible with modern web browsers and Wi-Fi connection.
* Map tile data is pre-loaded on SD card and does not need to be fetched online.
* Emergency button and associated logic are hardware-debounced and reliability tested.
* Battery life is adequate to cover anticipated trip time; users are educated to charge when necessary.
* Timestamping of data packets is done consistently to enable proper temporal logging and visualization

## **4.2 Context Diagram**

The context diagram below provides a high-level, process-centric view of the system, depicting all primary external entities, input/output flows, and the system boundary.



**Figure 4.2:Context diagram of Enhanced Long-Range Location Tracking and Fall Detection System**

****DS1:**Longitude,Latitude,Temperature,Pressure,Altitude,Battery data, Alert flag**

****DS2:**Longitude,Latitude,Temperature,Pressure,Altitude,Battery data, Alert flag**

**Figure 4.2 shows the context diagram of the Enhanced Long-Range Location Tracking and Fall Detection System, which shows how the system interacts with external entities. The sender nodes are data sources that transmit GPS coordinates, environmental readings, and fall/emergency alerts to the central node through LoRa communication. This information is processed and stored in the central node and can be accessed by users via an offline web dashboard on its Wi-Fi Access Point.**

# **Chapter 5**

# **Detailed Design**

## 5.1 Overview of System Design

The Offline Location Tracking System proposed based on LoRa is object-oriented because it is modular, extensible, and maintainable. The feature additions, code reuse, and future scalability are efficient because the behavior of sensor nodes, gateway functions, and user interactions can be encapsulated using object orientation. The design is organized in three phases: object modeling, dynamic modeling, and functional modeling to fully explain the core architecture of the system.

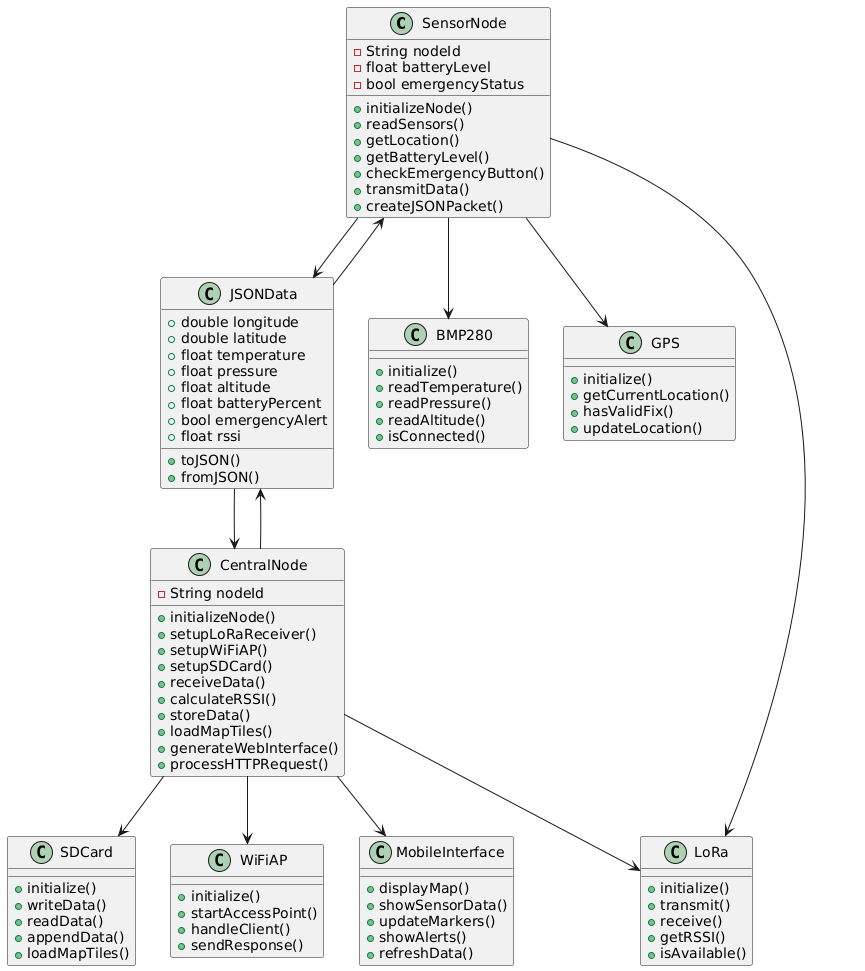
**5.2 Object Modeling**

**Brief Description**

Object modeling determines the static structure of the system by determining the main objects, their attributes, and the relations between them. The object modeling in the proposed LoRa-based Offline Location Tracking System reflects the real-world objects and interactions in an object-oriented paradigm, which greatly increases modularity, code reuse, and future scalability. The system has a number of core classes, each of which is a key element in the tracking and monitoring process.

The Node object is a single wearable device that is connected to a trekker or personnel. The nodes have sensors (GPS, BMP180), LoRa transceiver, and ESP32 microcontroller. It possesses such properties as nodeID, location, temperature, humidity, batteryStatus, and signalStrength. Its techniques involve gathering sensor data, transmitting data to the gateway, and initiating an emergency alert. The Gateway class is the main LoRa receiver node that controls the data gathering of all deployed nodes. It has properties like gatewayID, receivedPackets, connectedNodes, and dataBuffer, and methods to receive, parse, store data, and broadcast alerts in case of emergencies.It has methods to validate and format data to be stored. The EmergencyAlert class is employed in critical situations to handle alerts created by the nodes, e.g. a fall or manual .

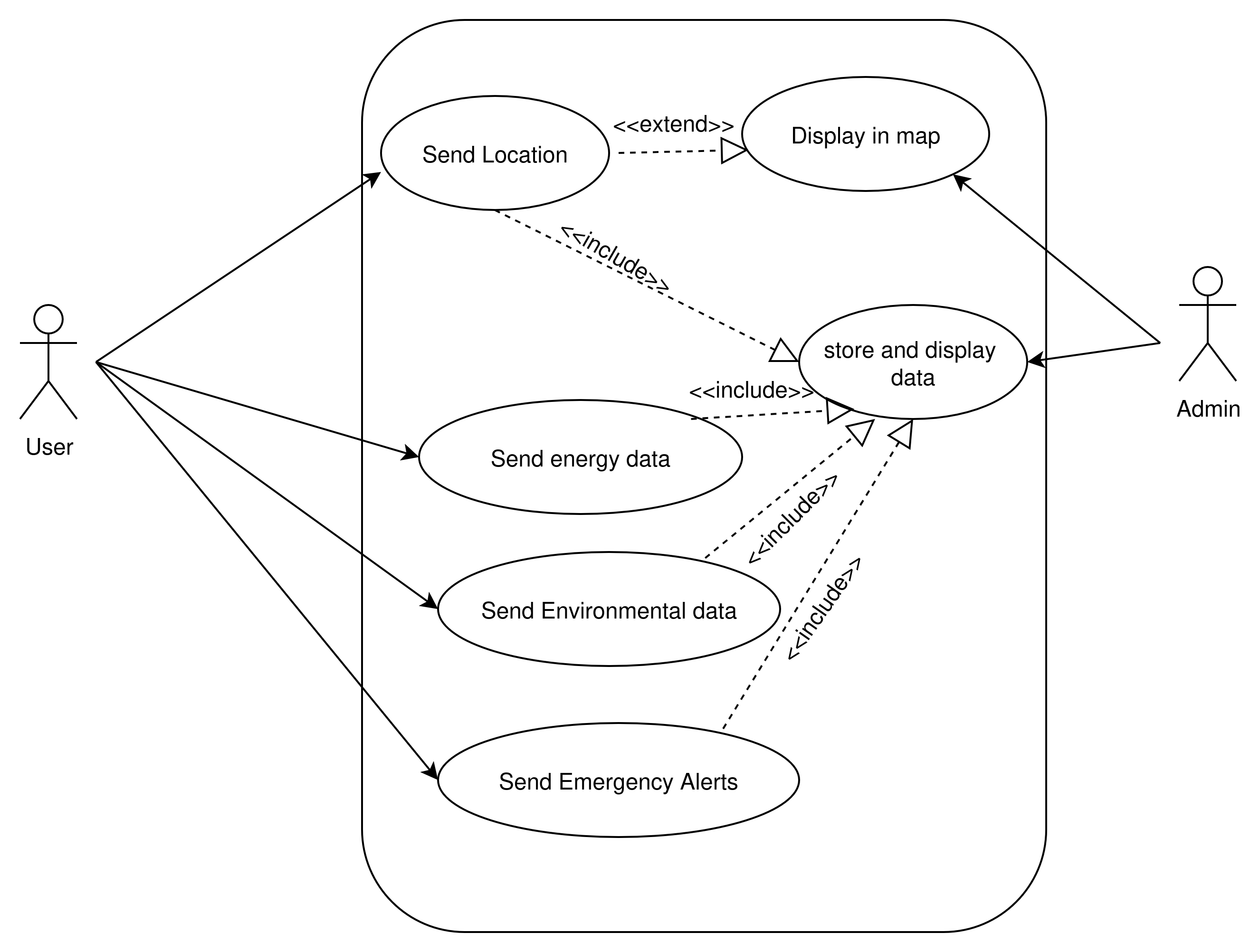
## **5.2.1 Class Diagram**

**Figure 5.1: Class Diagram of Enhanced Long-Range Location Tracking and Fall Detection System**

**Figure 5.1** shows the class diagram of the **Enhanced Long-Range Location Tracking and Fall Detection System**, outlining the main system components, their attributes, methods, and relationships. It represents classes such as **SenderNode**, **CentralNode**, **LoRaModule**, **GPSModule**, **SensorModule**, **DataLogger**, and **WebDashboard**, along with their interactions. The diagram illustrates how these classes collaborate to collect sensor data, transmit it via LoRa, store it on an SD card, and present it on the offline dashboard.

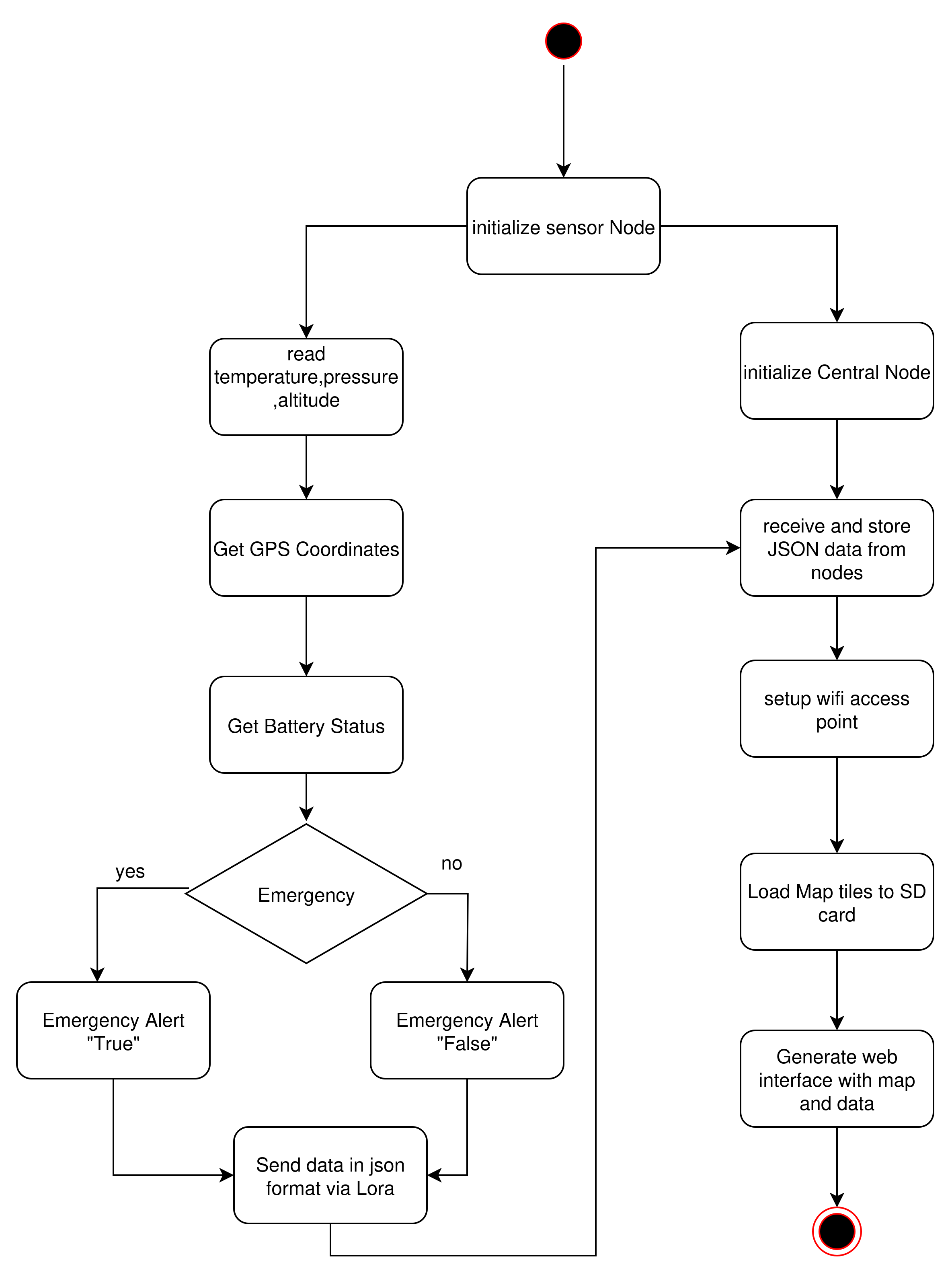
## **5.3 Dynamic Modeling**

## 5.3.1 Use Case Diagram

**Figure 5.2: Use Case diagram****of Enhanced Long-Range Location Tracking and Fall Detection System**

**Figure 5.2** illustrates the use case diagram of the **Enhanced Long-Range Location Tracking and Fall Detection System**, showing the interactions between system actors and its core functionalities. Trekkers (sender node users) can send location, environmental data, and emergency alerts, while the group leader or base camp operator (central node user) can view real-time locations, access historical logs, and monitor alerts via the offline dashboard. The diagram also depicts system operations such as data collection, LoRa transmission, SD card logging, map visualization, and emergency notification handling.

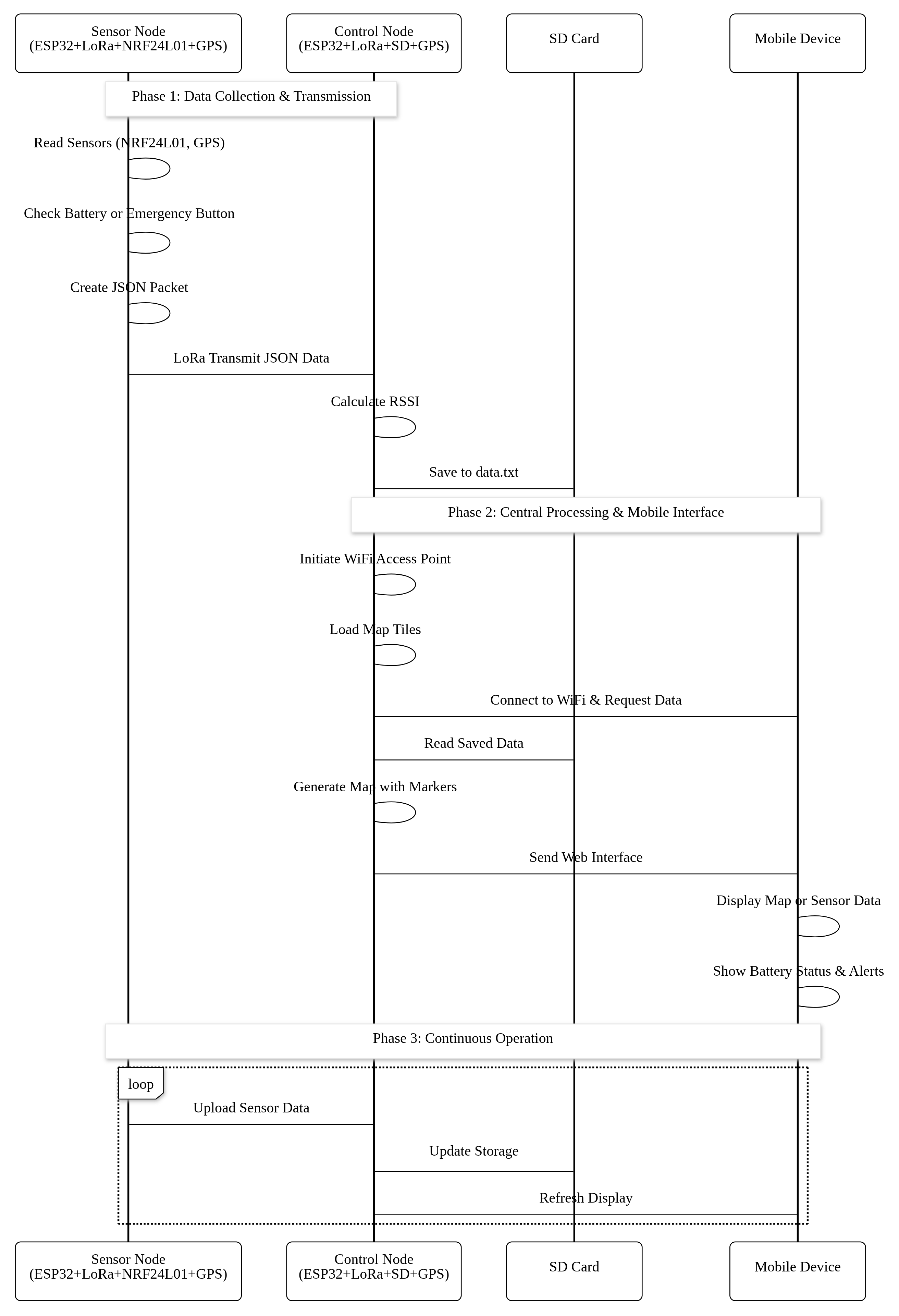
## **5.3.2 Activity Diagram**

****

**Figure 5.3: Activity Diagram of Enhanced Long-Range Location Tracking and Fall Detection System**

**Figure 5.3** presents the activity diagram of the **Enhanced Long-Range Location Tracking and Fall Detection System**, detailing the sequential flow of operations within the system. The process begins with sender nodes acquiring GPS and sensor data, followed by formatting the data into JSON and transmitting it via LoRa.

## **5.3.3 Sequence Diagram**



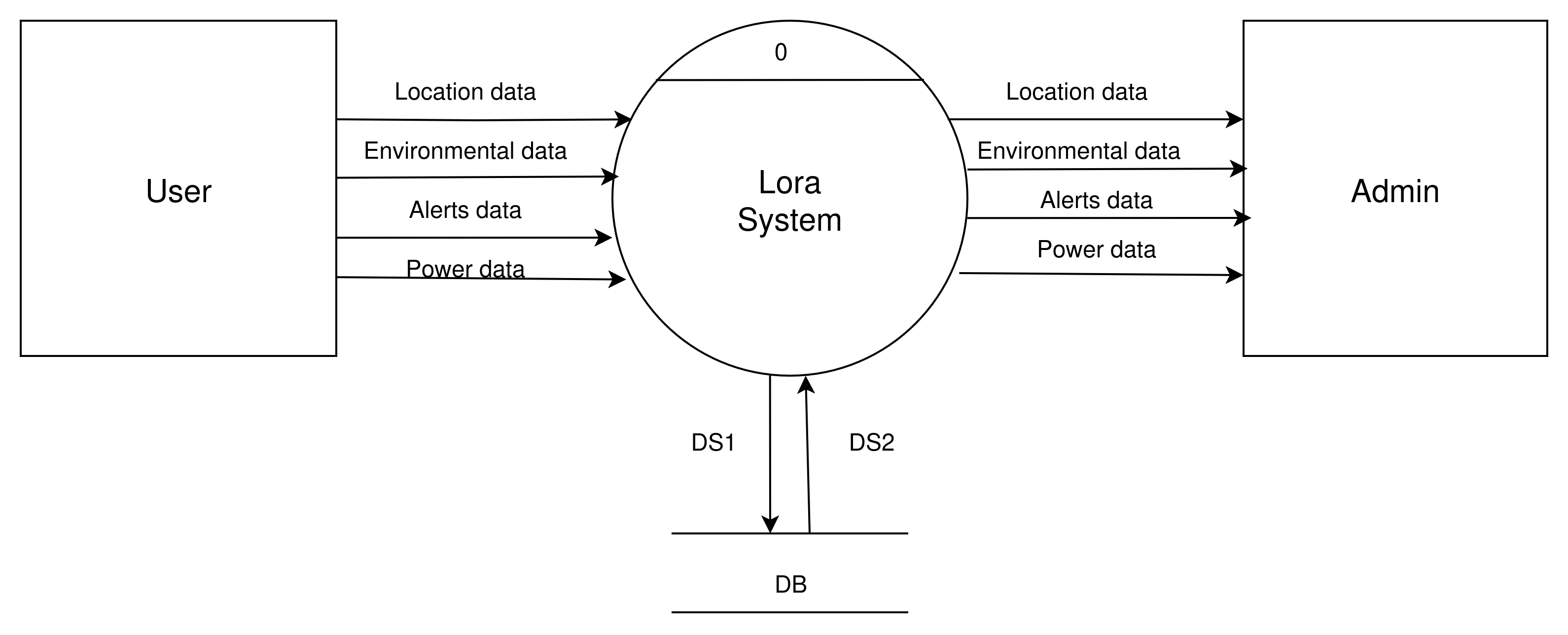
****Figure 5.4: Sequence Diagram of Enhanced Long-Range Location Tracking and Fall Detection System****

**Figure 5.4** illustrates the sequence diagram of the **Enhanced Long-Range Location Tracking and Fall Detection System**, showing the chronological interaction between system components. The sender node collects GPS and sensor data, formats it into JSON, and transmits it via LoRa to the

central node. The central node receives the data, logs it onto the SD card, updates the offline dashboard, and, if an emergency flag is detected, triggers an alert display.

## **5.4 Functional Modeling**

## Data Flow Diagram ****Level-0****

****

**Figure 5.5: Data Flow Diagram of Enhanced Long-Range Location Tracking and Fall Detection System Level-0**

****DS1:**Longitude,Latitude,Temperature,Pressure,Altitude,Battery data, Alert flag**

****DS2:**Longitude,Latitude,Temperature,Pressure,Altitude,Battery data, Alert flag**

The Level-0 Data Flow Diagram (DFD) of the **Enhanced Long-Range Location Tracking and Fall Detection System**, providing a high-level view of the system’s data processes and interactions.

**Data Flow Diagram Level-1**

**

****Figure 5.6:** Data Flow Diagram of Enhanced Long-Range Location Tracking and Fall Detection System Level-1**

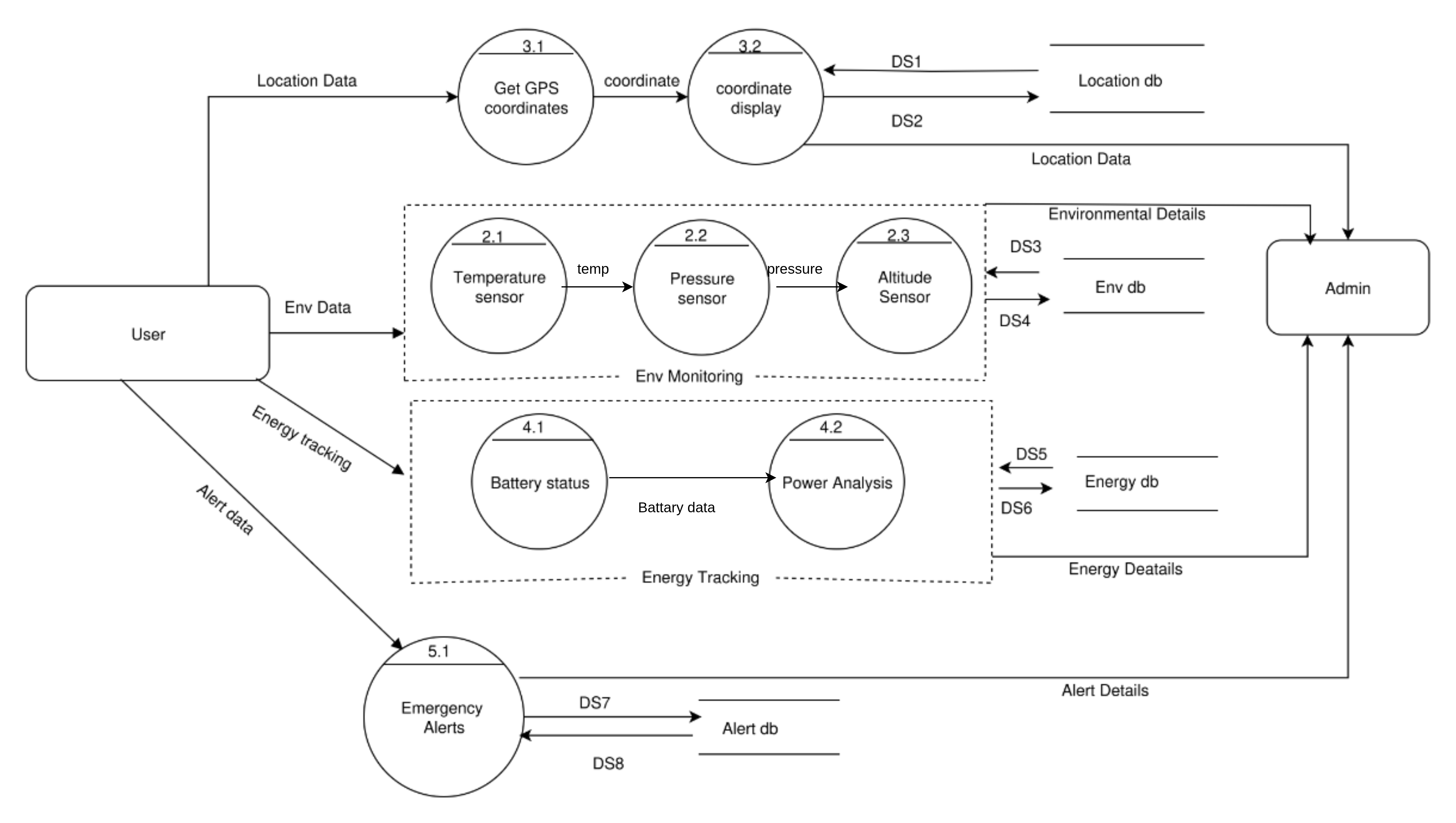
****DS1:**Longitude,Latitude,Temperature,Pressure,Altitude,Battery data, Alert flag**

****DS2:**Longitude,Latitude,Temperature,Pressure,Altitude,Battery data, Alert flag**

**Figure 5.6 presents the Level-1 Data Flow Diagram (DFD) of the Enhanced Long-Range Location Tracking and Fall Detection System, breaking down the main processes into detailed sub-processes.**

**At the sender node, GPS and environmental sensors collect data, which is formatted into JSON and transmitted via LoRa.**

**Data Flow Diagram Level-2**

**

**Figure 5.6: Data Flow Diagram of Enhanced Long-Range Location Tracking and Fall Detection System Level-2**

**Figure 5.6** illustrates the Level-2 Data Flow Diagram (DFD) of the **Enhanced Long-Range Location Tracking and Fall Detection System**, providing a detailed breakdown of the data handling within individual modules. In the sender node, the GPS module supplies location coordinates, the BMP180 sensor provides temperature and pressure readings, and the emergency button triggers an alert flag. These inputs are combined into a JSON packet and transmitted via the LoRa module. At the central node, the LoRa receiver decodes the packet, the data validation process checks for completeness and accuracy, and the SD card logger stores it in data.txt.

## 5.4.1 Assumptions Made

* Every sensor node generates a node\_id.
* Time synchronization (timestamp) is done using the GPS time of the node
* Emergency events are clearly recorded and take precedence in displays.
* Data files are saved locally on the SD card in simple tabular (CSV/JSON) formats to be robust and compatible.
* Hardware limitations allow the ESP32 gateway to support 8-10 simultaneous Wi-Fi user sessions.

## ****5.5 Detailed Design****

****Design Decisions****

* **Data Structures: Sensor data is packaged as JSON objects to communicate and as rows (CSV/JSON) to store persistently. The corresponding C++ structs or classes represent each entity (node, alert, user session) and make it more readable.**
* **Modularity: The functional decomposition into classes (Node, Gateway, SensorData, EmergencyAlert, UserSession, DataStorage) provides separation of concerns, which makes testing and future upgrades easier.**
* **Efficiency: Minimal dynamic memory usage; rings/circular buffers of recent data; pre-indexed SD card access to allow fast lookups.**

****5.5.1 Logic Design****

**Node Data Acquisition Mod**

**Begin**

**Read all sensor values (GPS, temperature, pressure, battery)**

**When emergency button pressed or battery < threshold, set alert flag**

**Wrap all data in JSON**

**Send JSON packet over LoRa**

**End**

**Data Reception Module**

Begin

Always listen to LoRa packets

When packet is received, parse JSON

Log to SD card (append)

Refresh shared memory/datastructure of dashboard

In case of alert\_flag, update alert status

End

**Web Dashboard Module**

• Begin

When a user connects, authenticate session when necessary

Serve main HTML dashboard and map tiles

On data/API request, send most recent and historical telemetry

On alert acknowledgment, update state and refresh UI

• End

## ****Chapter 6****

## ****Implementation****

This chapter gives the full implementation details of the LoRa-based Location and Emergency Tracking System. It discusses the code organization, logic, and practical implementation of each of the core modules: sensor data acquisition, LoRa transmission, SD card logging, offline dashboard hosting, and emergency alert handling.

**Code Snippets 6.1**

**Node Code**

Manages the acquisition of GPS and BMP180 sensor data, emergency button detection, JSON packet formatting, and LoRa transmission to the central node.



**Figure 6.1:Sender Node code Snippet of Enhanced Long-Range Location Tracking and Fall Detection System**

**Central Node Code**

Receives LoRa packets, validates them, stores them in data.txt on the SD card, updates latest.json to display in real-time dashboard, and serves the offline map-based dashboard via ESPAsyncWebServer in AP mode.



**Figure 6.2:Receiver Node code Snippet of Enhanced Long-Range Location Tracking and Fall Detection System**

**6.2 Implementation**

The project was developed in a number of core modules that interact to offer offline location tracking and emergency alerting.

* Sensor Data Acquisition: The sender node reads GPS coordinates (latitude, longitude, altitude) of the NEO-6M GPS module and environmental data (temperature, pressure) of the BMP180 sensor. It also checks battery voltage and the state of the emergency button.
* LoRa Data Transmission: The sensor data is encoded as a JSON object and transmitted using the SX1278/RA-02 LoRa transceiver. Signal quality monitoring is included in the metadata as RSSI (Received Signal Strength Indicator).
* Data Logging on Central Node: All the packets received by the central node are stored in data.txt on the SD card as a historical record. At the same time, it also updates latest.json to show the latest data in real-time on the dashboard.
* Offline Map and Dashboard Hosting: The SD card stores preloaded offline map tiles. The ESP32 central node runs a browser-based dashboard using HTML, CSS, and JavaScript with Leaflet.js to render the map, so any Wi-Fi-enabled device can connect and see live data without an internet connection.
* Emergency Alert Handling: In case of pressing the emergency button or the battery level drops below a specified level, a high-priority alert is added to the JSON packet. This alert is shown on the dashboard with a visual indication and is logged to a file to be reviewed later.
* Testing and Validation: The system was tested with two sender nodes and one central node over distances greater than 1 km in open terrain, confirming both communication reliability and dashboard update speed.

### ****Implementation Screenshots****

****Sender Node Hardware Configuration****

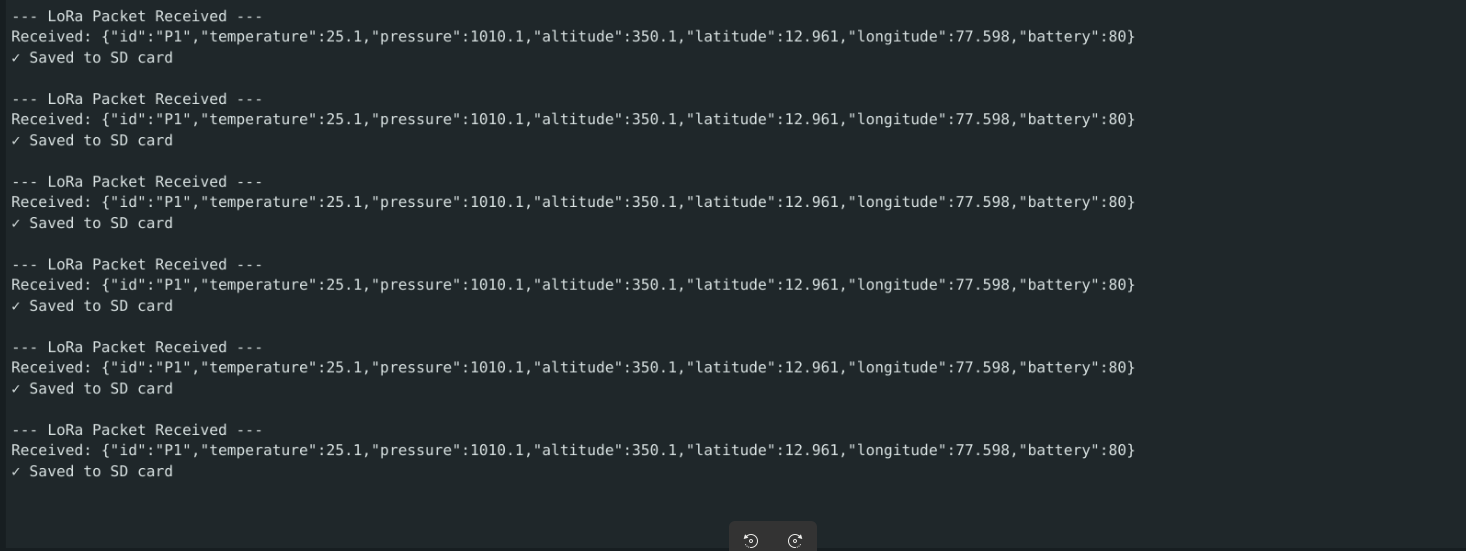
**Displays the completed sender node with ESP32, GPS module, BMP180 sensor, LoRa transceiver, emergency button, and battery pack on a small PCB.**

****Central Node Dashboard Interface (Figure 6.2)****

**Shows the real-time dashboard hosted by the central node, with map markers indicating the location of each sender node, temperature, pressure, battery percentage, and emergency alerts.**

****Data Logging on SD Card****

**Shows the data.txt file on the SD card of the central node, which contains JSON entries of each data packet received by the sender nodes, with timestamps to track history.**



**Figure 6.3:Implementation received code of Enhanced Long-Range Location Tracking and Fall Detection System**

## ****Chapter 7****

## ****Software Testing****

Software development life cycle is a process that requires testing to guarantee quality, reliability, and functionality of the application. The primary goal of software testing in the project is to detect faults, ensure that the system works according to the requirements, and that it behaves as expected in various conditions of operation.

**Test Cases 7.1**

**7.1.1 Testing methodologies**

The testing processes in this project identify the way in which the application was tested to ensure that it is functional, performs well, and is reliable. The methods of testing used are as follows:

**7.1.2 Unit Testing**

Unit testing was performed to ensure that the individual modules of the LoRa-based trekking safety system work correctly and then they were combined into the overall system. This will make sure that every part does what it is supposed to do.

The modules that were unit tested in the Trekking Safety System are:

**LoRa Receiver Module**

* Objective: Proper reception of JSON packets by the sender nodes.
* Test: Run test packets between a sender node and ensure that the central node correctly receives and interprets them.
* Outcome: All the packets were received and parsed properly.

**GPS Data Acquisition Module**

* Objective: To ensure proper retrieval of latitude, longitude and altitude values of the GPS module.
* Test: Put the sender node at known coordinates and check GPS readings.
* Output: GPS coordinates were within the acceptable error margin of the expected location

**BMP180 Sensor Module**

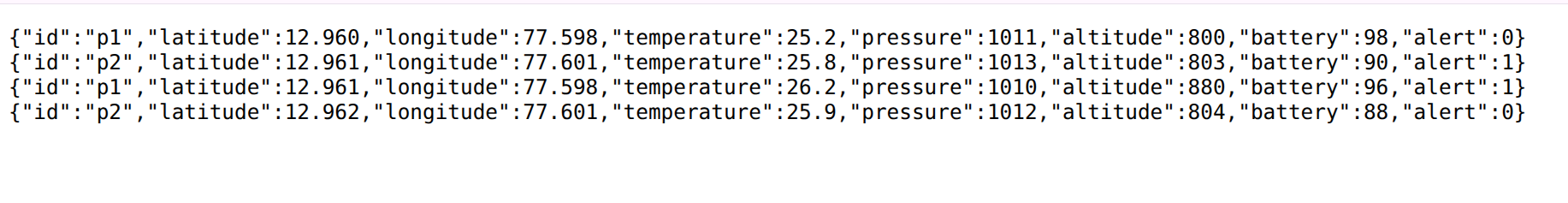
* Objective: Check the correctness of temperature and pressure measurement.
* Test: A calibrated weather station versus BMP180.
* And the result is:Accurate wether data

**Module: Emergency Alert Button Module**

* Objective: Check that high-priority packets are alerted properly.
* Test: Hold down the emergency button and see whether the alert flag has made its way to the LoRa packet.
* Output: The alert packets are sent and correctly show on dashboard.

**SD Card Logging Module**

* Purpose: The aim is to maintain all the incoming data continually in data.txt.
* Test: Retrieve data sent by the sender nodes, save them to SD card and check bundle contents.
* Outcome: Accurate data in terms of information and timestamps of activity.



**Figure 7.1:SD Card Logging of Enhanced Long-Range Location Tracking and Fall Detection System**

**7.1.4 Integration Testing**

System testing was done to ensure that all the modules work as a single cohesive unit. This involves verification of the transmission of data between the sender nodes, the central node, the SD card and the web dashboard.

**LoRa Data Receiving and Recording**

* Purpose: Make sure that received packets are properly logged onto SD card.
* Test: Send several packets using sender node, look into data.txt and see whether there is correct records.
* Output: All the packets received were recorded without corruption.

**GPS + Visualization of Offline Map**

* Purpose: Make sure that the offline map tiles have GPS data properly displayed.
* Test Move sender node to new locations, refresh dashboard, check to see proper positions are set on the markers.
* Output: Up to date map markers in real time.

**BMP180 Sensor Module**

* Hypothesis: It is to scrutinize the accuracy of temperature and pressure.
* Weather Station Calibration test BMP180

**Panic Button Module**

* Instead, high priority packets alert.
* Press and hold the emergency button and monitor the ground station to see whether there is an alert flag received in the LoRa packet or not.
* Output: The alert packets are being sent out and they are properly displayed on the dashboard.

**SD Card Logging Module**

* Purpose: It is to append all the incoming data on a continual basis to data.txt.
* read: Read informations that are sent by the sending nodes, store them in SD card and check what the informations inside the bundles .
* Results: Valuable data in respect of information and timing of activity.

### ****Test Cases****

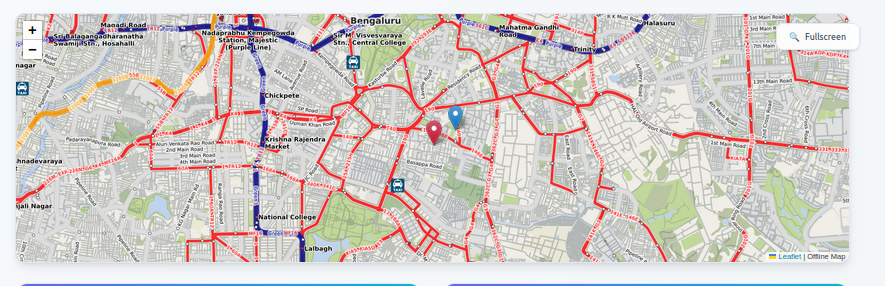
#### ****Table 7.1: Test Cases – Data Transmission and Logging****

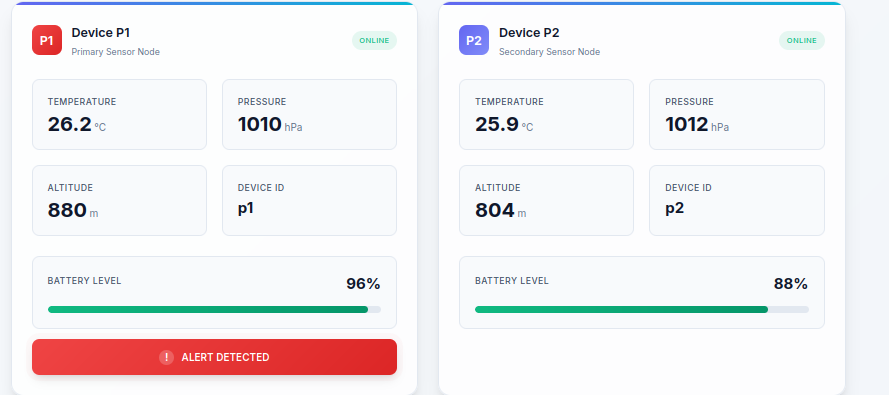
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Input** | **Expected Result** | **Actual Result** | **Status** |
| TC1 | Receive and store valid LoRa packet | JSON packet from sender node | Packet stored in data.txt and displayed on dashboard | Successful | Pass |
| TC2 | Handle corrupted LoRa packet | Packet with missing JSON fields | Ignore packet, display error in debug log | Error not shown in logs | Fail |
| TC3 | Store GPS data accurately | Known location coordinates | GPS matches within ±5 meters | Correct location logged | Pass |
| TC4 | Display latest environmental readings | Temperature = 25°C, Pressure = 1013 hPa | Dashboard updates within 5 seconds | Data updated instantly | Pass |
| TC5 | Handle emergency button press | Button pressed on sender node | Dashboard shows emergency alert within 2 seconds | Alert displayed | Pass |

#### ****Table 7.2: Test Cases – Dashboard and Map Visualization****

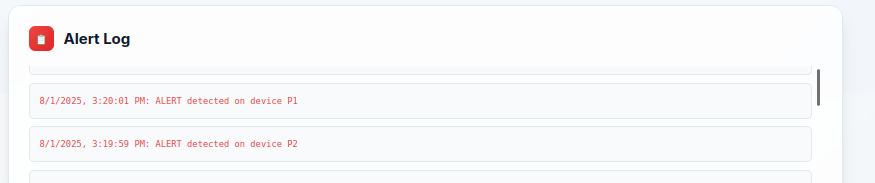
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Input** | **Expected Result** | **Actual Result** | **Status** |
| TC6 | Display node on offline map | GPS location from sender node | Map marker updates to correct position | Correctly displayed | Pass |
| TC7 | Display multiple sender nodes | Data from two sender nodes | Both markers visible on map | Both visible | Pass |
| TC8 | Handle lost GPS signal | No GPS data in packet | Node remains at last known position with warning indicator | Location Pointing Error | Fail |
| TC9 | Display historical data from SD card | data.txt log file | Dashboard table matches SD card entries | Matched | Pass |
| TC10 | Load dashboard quickly after connection | Access via Wi-Fi AP | Dashboard loads within 10 seconds | Loaded in 7 seconds | Pass |

**7.2 Validations**

**Figure 7.2:Offline Map of Enhanced Long-Range Location Tracking and Fall Detection System**



**Figure 7.3:Nodes data of Enhanced Long-Range Location Tracking and Fall Detection System**

**Figure 7.4:Alert Log of Enhanced Long-Range Location Tracking and Fall Detection System**

## ****Chapter 8****

## ****Conclusion****

In this project there has been successful development of an offline emergency tracking system (and location tracking system) of self sufficient trekking groups in remote areas. The suggested system takes advantage of LoRa mode of communication with ESP32 microcontrollers to allow high-range, low-power data transfer and a sense of safety and situational awareness in the cell and internet-free areas.

The system development and implementation was based on a tried and tested methodology that included; problem analysis, literature review, system requirements specification, system architecture design, implementation and testing. it has two sender nodes and one central node that work in a star topology. The nodes deploy GPS coordinates, altitude, temperature, pressure, battery level and emergency messages to the main node through the use of LoRa. The central node stores every data it receives on an SD card and has an offline browser-based dashboard where trekkers can see their positions on saved map tiles without being connected to the internet.

Its key modules will be sensor data acquisition, LoRa transmission, SD card logging, offline map rendering using Leaflet.js and the emergency alert handling mechanism. The emergency device will see to it that the emergency issues are communicated with highest priority with visual color indicators on the dashboard bridging seconds to emergency. This is also accompanied by optimizing battery lifetime by data scheduling efficiency and low power modes, so that the system can run weeks in the field.

Testing also proved the system to be within its requirements regarding accuracy of performance, because real-time location tracking, satisfying environmental monitoring and emergency alert response speed were achieved.

In conclusion, I would recommend this project in the following way wearing a LoRa connected personal safety device, running an embedded sensor system that integrates BLE, and offline mapping to form the basis of a safe expedition in remote areas. In the future, it is possible to increase the number of nodes, add more environmental sensors, equip the device with solar charging to extend the use time, and equip the dashboard with advanced data analytics.

## ****Chapter 9****

## ****Future Enhancements****

Despite the fact the present system has demonstrated steady performance in offline tracking of location and alerting of a trekking group of people in case of emergencies, an array of improvements can be integrated to enhance its performance, effectiveness and scalability:

* Scalability of Nodes: Extend the system to support additional sender nodes thus allowing more trekking groups or multi-team expeditions to be tracked at any given time without affecting the performance.
* Mesh Networking Capability: Extend LoRa mesh networking, which allows the nodes to pass data to other nodes, and enables a longer range of communication than the current star topology (limited to the direct line-of-sight range).
* Solar Charging Integration: Introduce solar-powered charging systems to enable the sender and central nodes to work months without manual recharge, extend deployment time off-grid.
* Extra Sensor Connection: Connect and use more sensors like humidity, air quality (PM2.5), or movement sensors in order to gain a more in-depth environmental monitoring and increased situational awareness.
* Mobile application solutions: Implement a unique mobile application on Google and Apple platforms with a highly functional and more user-interactive interface with emergency notification via push notifications.
* GPS-Free Positioning: Currently pose RSSI-based triangulation even when the GPS is out of range, e.g. when a path is covered with dense trees or in bottom of valleys.
* The implementation of all these improvements will go a long way in enhancing the efficiency, coverage and the level of user experience in the trekking safety system, thus making it more effective in expeditions and rescue missions and even monitoring applications that take place in remote areas.