SRS Documentation

MAKERERE UNIVERSITY

Smart Water Level Monitoring System for Tanks/Drums

Prepared by

GROUP-3

NO	NAME	STUDENT NUMBER	REG. NO
1	KIGOZI ALLAN	2400725792	24/U/25792/PS
2	KEITH PAUL KATO	2400726593	24/U/26593/EVE
3	ECONGU PAUL	2400722570	24/U/22570
4	BWIRE RODNEY	2400714889	24/U/14889/PS
5	CHELIMO EMMA	2400714938	24/U/14938/PSA

Mentor: Dr. Paddy

Course: CSC 1304 Practical Skills Development

Date: 26TH July 2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 Introduction

The Smart Water Level Monitoring System is an IoT-based solution designed to monitor and report the water levels in tanks or drums. The system automates the process of checking water availability and helps users avoid water shortages by sending timely alerts. This report presents the full technical breakdown of the project, including its goals, system requirements, implementation design, and testing procedures.
This section outlines the document's purpose, intended audience, scope of the product, and the standards followed throughout its creation. It also includes definitions of important acronyms and references to key materials used.

1.1 Document Purpose

The purpose of this document is to specify the software requirements for the Smart Water Level Monitoring System developed by us Group 3. It provides a comprehensive description of the system's functionality, interfaces, and expected behaviour. The document serves as a communication bridge between the developers and stakeholders, including our supervisor, Dr Paddy and mentors, ensuring that all design and performance expectations are clearly defined by the guide lines of the Software Requirements Specification.

1.2 Product Scope

This product is intended to continuously measure and monitor the level of water in storage containers using an ultrasonic sensor and to notify the user via SMS when water is low, medium and at high level. This keeps the users always informed about their water usages. It also calculates weekly usage and estimates how many days of water are remaining based on recent consumption. The data is logged to the ThingSpeak cloud platform and displayed locally on an LCD. This information can also be visualized on our website. This system benefits users by preventing water outages, enabling water conservation, and offering accessible real-time monitoring.

1.3 Intended Audience and Document Overview

The primary audience includes the course facilitator, Dr. Paddy, client users like household or small-scale tank users, and fellow developers.

- ***Introduction** That provides an overview of the project, its purpose, scope, and target audience.*
- ***Overall Description** details the product's perspective, functions, user characteristics, and general constraints.*
- ***Specific Requirements** elaborates on functional requirements, performance requirements, design constraints, and quality attributes.*
- ***System Architecture and Hardware Design** presents the overall system architecture, component selection, and detailed pinout configurations.*
- ***Software Design and Implementation** Which will describe the firmware logic, data flow, and communication protocols.*

- ***Testing and Validation*** *Which outlines the testing methodologies we used and results.*
- ***Limitations and Future Work*** *That discusses current limitations and potential enhancements.*
- ***Conclusion*** *Which summarizes our project's achievements.*
- ***Appendices*** *This includes our project work plan and team member contributions.*

1.4 Definitions, Acronyms and Abbreviations

- ***Arduino IDE:*** *Integrated Development Environment for Arduino*
- ***API:*** *Application Programming Interface*
- ***FR:*** *Functional Requirement*
- ***GSM:*** *Global System for Mobile Communications*
- ***HC-SR04:*** *Ultrasonic Sensor*
- ***IEEE:*** *Institute of Electrical and Electronic Engineers*
- ***I2C:*** *Inter-Integrated Circuit*
- ***IoT:*** *Internet of Things*
- ***LCD:*** *Liquid Crystal Display*
- ***MCU:*** *Microcontroller Unit*
- ***PR:*** *Performance Requirements*
- ***SRS:*** *Software Requirements Specification*
- ***SMS:*** *Short Message Service*
- ***S.W.L.M.S:*** *Smart Water Level Monitoring System*
- ***ThingSpeak:*** *An open-source IoT analytics platform service*

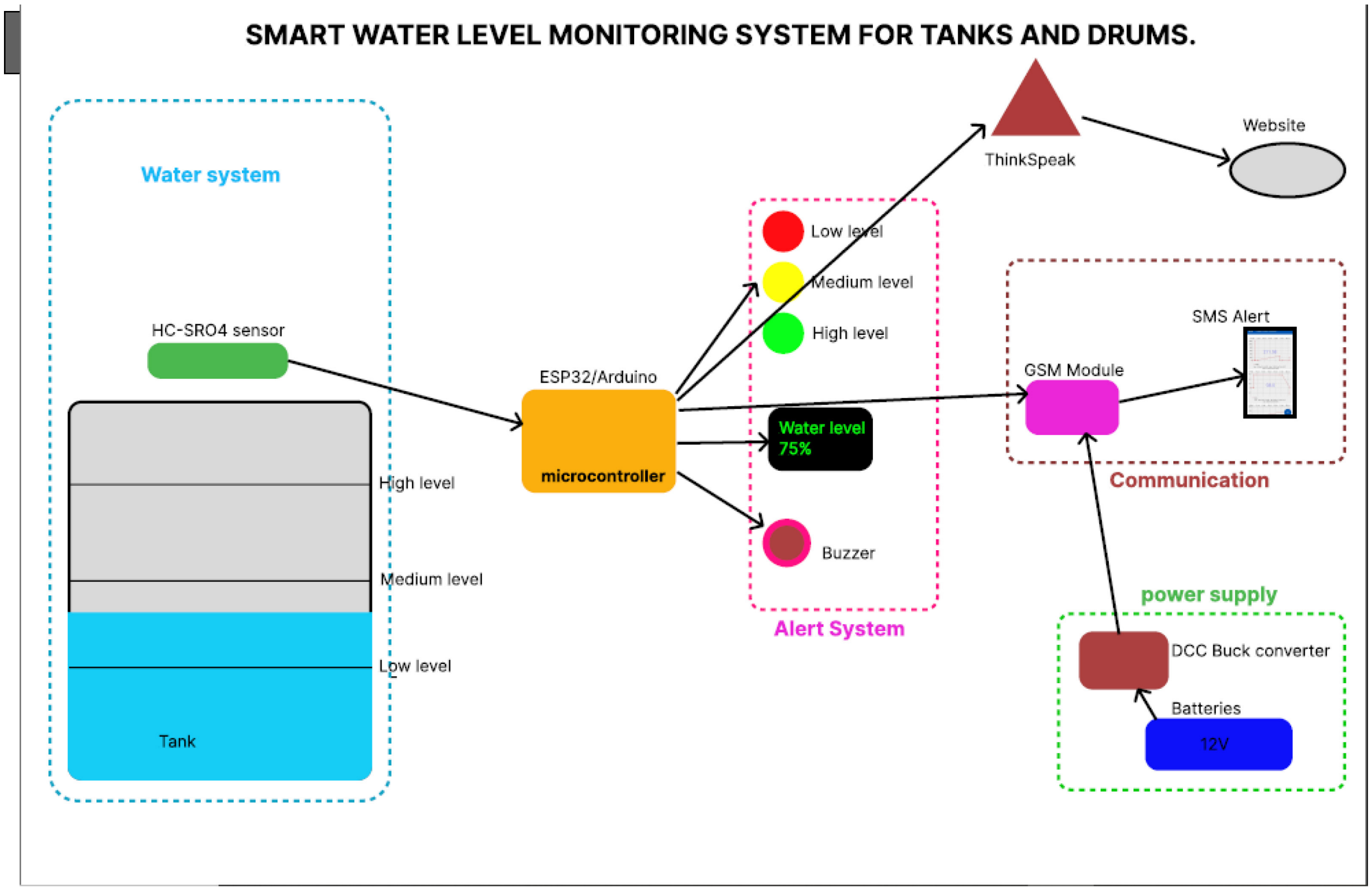
1.5 Document Conventions

This document uses Arial font, size 12, and follows IEEE formatting for technical reports. Major sections are bolded and numbered. Acronyms are spelled out at first use, and metric units are used throughout the document. Functional and non-functional requirements are categorized and described in paragraph form.

1.6 References and Acknowledgments

- ***IEEE*** *http://www.ieee.org/documents/ieeecitationref.pdf*
- ***Arduino Reference Documentation****: https://www.arduino.cc/reference/en/*
- ***ThingSpeak API Documentation****: https://thingspeak.com/docs*

We would like to acknowledge Dr. Paddy for his mentorship, and all team members for their active contributions and collaboration.



2.2 Product Functionality

The key functions of the system include measuring water levels using an HC-SRO4 sensor, converting the measurement into percentage and volume in litres, displaying this on an LCD screen, storing it in ThingSpeak, and sending SMS alerts by utilizing the at commands via GSM. It also calculates average consumption and estimates how many days the remaining water will last based on usage trends. This can also be visualised in our website. These functions operate autonomously, requiring minimal user input.

2.3 Users and Characteristics

The system is primarily designed for domestic users, institutions, and small businesses relying on tanks or drums. Most users are expected to have limited technical knowledge, so the system prioritizes simplicity and reliability. These domestic users use the system daily or weekly with high security level so as to have access to water usage data and sms alerts.
Administrators or technical personnel may interact with the ThingSpeak dashboard for detailed analysis and maintenance. These use the system occasionally for troubleshooting and maintenance. They require access to the code and hardware for calibration and updates.

2.4 Operating Environment

The system is intended to operate in residential or institutional environments with access to power and mobile network coverage. It is housed in a weather-resistant enclosure to protect the electronics. The ESP32 operates on 3.3V logic and requires a stable 5V power supply. The system uses the Arduino development environment and communicates via GSM (SIM800L) that also needs 5V which we get from batteries and Wi-Fi for ThingSpeak updates. The LCD will be in-house for visual display of the data.

2.5 Design and Implementation Constraints

- ❖ *The GSM module requires strong mobile signal strength to function reliably.*
- ❖ *ThingSpeak has limited API call frequency (once every 15 seconds on free tier).*
- ❖ *Limited power supply may affect long-term deployments in rural areas.*
- ❖ *The system must be programmed in C++ using Arduino libraries.*
- ❖ *Ultrasonic sensors have a limited range (~400 cm) and may be affected by wind or water turbulence.*

2.6 User Documentation

Though this report does not include a full user manual, we propose delivering a quick start guide and troubleshooting checklist. The LCD interface is intuitive and does not require prior training. For more detailed information, the SMS format and usage summaries will be documented for end users. In addition, brief Quick Start Guides in print or PDF will help non-technical users get started quickly. For developers or future maintainers, Technical Reference Documentation including code structure, libraries used, and API integrations will be prepared. Documentation will be delivered in PDF format and, where possible, embedded online like in a GitHub README. Visual aids such as diagrams, annotated screenshots, and simple flowcharts will be included to assist users at different levels of expertise.

2.7 Assumptions and Dependencies

- *A reliable GSM mobile signal is available where the tank is located.*
- *The tank is stationary and does not change height over time.*
- *Users have basic literacy to interpret SMS messages.*
- *Wi-Fi is available for cloud uploads in deployments that include ThingSpeak.*
- *Power outages are infrequent or the system is supported with a battery backup.*

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The system features a user-friendly LCD interface that continuously displays the water level in percentage format. This ensures that users can quickly interpret tank status at a glance. Additionally, users receive periodic SMS messages containing weekly usage summaries and the estimated number of days remaining based on consumption trends. For advanced users, a web-based dashboard via ThingSpeak presents historical data through graphical visualizations.

3.1.2 Hardware Interfaces.

Ultrasonic Sensor (HC-SR04)

Physical Interface: Connected to the ESP32 via digital I/O pins (TRIG and ECHO).

Logical Interface: The software triggers the sensor by sending a pulse via the TRIG pin and measures the time it takes to receive a response via the ECHO pin. This time is converted into distance (water level).

Libraries Used: No special library is required; interaction is handled using basic Arduino digitalWrite () and pulseIn () functions.

LCD Display (I2C 16x2 LCD)

Physical Interface: Communicates with the ESP32 through I2C protocol using the SDA and SCL pins.

Logical Interface: The software sends formatted text messages to be displayed (e.g., current water level, status).

Libraries Used: LiquidCrystal_I2C.h is used for controlling the LCD with simple commands such as lcd.print () and lcd. setCursor ().

GSM Module (SIM800L)

Physical Interface: Communicates via UART (TX and RX pins) with the ESP32.

Logical Interface: The ESP32 sends AT commands to the GSM module to send SMS alerts and monitor network status.

Libraries Used: Basic SoftwareSerial and HardwareSerial is used in conjunction with Serial.print() and Serial.read() to communicate using AT command sets.

Wi-Fi (Built-in ESP32)

Physical Interface: No external connection; the ESP32's built-in Wi-Fi module is used.

Logical Interface: Used to send data to ThingSpeak over the internet for remote monitoring and reporting.

Libraries Used: WiFi.h for connecting to the network and ThingSpeak.h for pushing data to the cloud.

Power Supply

Physical Interface: The system is powered via a regulated 5V DC source (e.g., battery or adapter).

Logical Interface: The software may monitor voltage levels to detect low-power states (optional feature).
Libraries Used: If voltage monitoring is implemented, the analogRead () function will be used.

3.1.3 Software Interfaces

The software components include the ThingSpeak API for cloud data storage and visualization, as well as the Arduino IDE for firmware development. The project utilizes several open-source libraries that support Wi-Fi communication, GSM messaging, and sensor data processing. These components are integrated through standard APIs and follow the modular programming approach to ensure code reusability and maintainability.

3.1.4 Communications Interfaces

The Smart Water Level Monitoring System relies on two primary communication functions: cloud-based data transmission via Wi-Fi and SMS notifications via GSM. The system uses the built-in Wi-Fi capability of the ESP32 to send sensor data to ThingSpeak, a cloud-based IoT platform. This communication is carried out using HTTP protocol via the ThingSpeak.h library, with data formatted as key-value pairs in standard HTTP requests. This enables remote monitoring of water levels and usage trends from any web browser.
For local alerts, the system uses the GSM (SIM800L) module to send SMS messages to a predefined mobile number. These messages include water level warnings, estimated remaining days of water, and weekly usage summaries. SMS communication is carried out through AT command sets over UART, without reliance on the internet.
While encryption is not applied to SMS messages, the Wi-Fi communication can optionally support encrypted HTTP (HTTPS) for secure data transfer to the cloud. No email or FTP functionalities are required. Communication is designed to be asynchronous, with timed updates to ThingSpeak and event-triggered SMS alerts, ensuring the system remains responsive and efficient without the need for continuous connectivity.

3.2 Functional Requirements

Water Level Monitoring Functions
These functions are responsible for gathering and interpreting water level data from the tank.

- ***Trigger Ultrasonic Sensor;***
  The system shall activate the ultrasonic sensor at defined intervals (e.g., every 15 seconds) to measure the distance between the sensor and the water surface.
- ***Calculate Water Level;***
  The system shall convert the distance measured by the ultrasonic sensor into a percentage representation of the water level based on the known tank height.
- ***Determine Water Volume;***
  The system shall calculate the approximate volume of water (in litres) remaining in the tank using the level percentage and the total tank capacity.

Display and Local Feedback Functions
These functions provide local feedback to the user via an LCD screen.

- ***Display Real-Time Data on LCD;***
  The system shall continuously update the LCD to show current water level percentage and remaining water volume.
- ***Display Status Messages;***
  The system shall show predefined warning messages (e.g., "LOW WATER LEVEL!") when the level drops below a threshold.

Cloud Data Upload Functions (ThingSpeak)
These functions are responsible for sending data to a cloud server for remote monitoring and analytics.

- ***Connect to Wi-Fi Network;***
  The system shall automatically connect to a configured Wi-Fi network using stored credentials.
- ***Send Data to ThingSpeak;***
  The system shall periodically (e.g., every 15 minutes) send current water level data, usage stats, and timestamps to a ThingSpeak channel using HTTP requests.
- ***Log Weekly Usage;***
  The system shall track daily readings and compute total weekly water usage, then upload that value to a dedicated ThingSpeak field.

SMS Alert and Notification Functions
These functions ensure users receive real-time notifications when attention is needed.

- ***GSM Network Initialization;***
  The system shall initialize the GSM module and check for network availability during startup.
- ***Send Low-Level, medium-level and high-level Alert;***
  The system shall automatically send an SMS to the registered phone number when the water level falls below a defined threshold, at the minimum and maximum thresholds.
- ***Send Weekly Usage Summary;***
  The system shall send an SMS every 7 days containing the total amount of water consumed during the week and an estimate of how many days the remaining water may last at the current usage rate.

System Configuration and Maintenance Functions
These functions support system updates, configuration, and maintenance.

- ***Manual Threshold Configuration*** *(Optional / Future Expansion)*
  The system shall allow the threshold level for SMS alerts to be reconfigured via serial interface or Bluetooth (if added).
- ***Serial Debug Output***
  The system shall output debug messages via the serial monitor for use by developers or technicians during setup and troubleshooting.
- ***Error Detection and Reporting***
  The system shall detect sensor failures or connectivity issues and report them via LCD or serial messages.

Power and Reset Management

- ***Boot and Initialization***
  Upon powering on, the system shall initialize all components: LCD, ultrasonic sensor, GSM module, and Wi-Fi module.

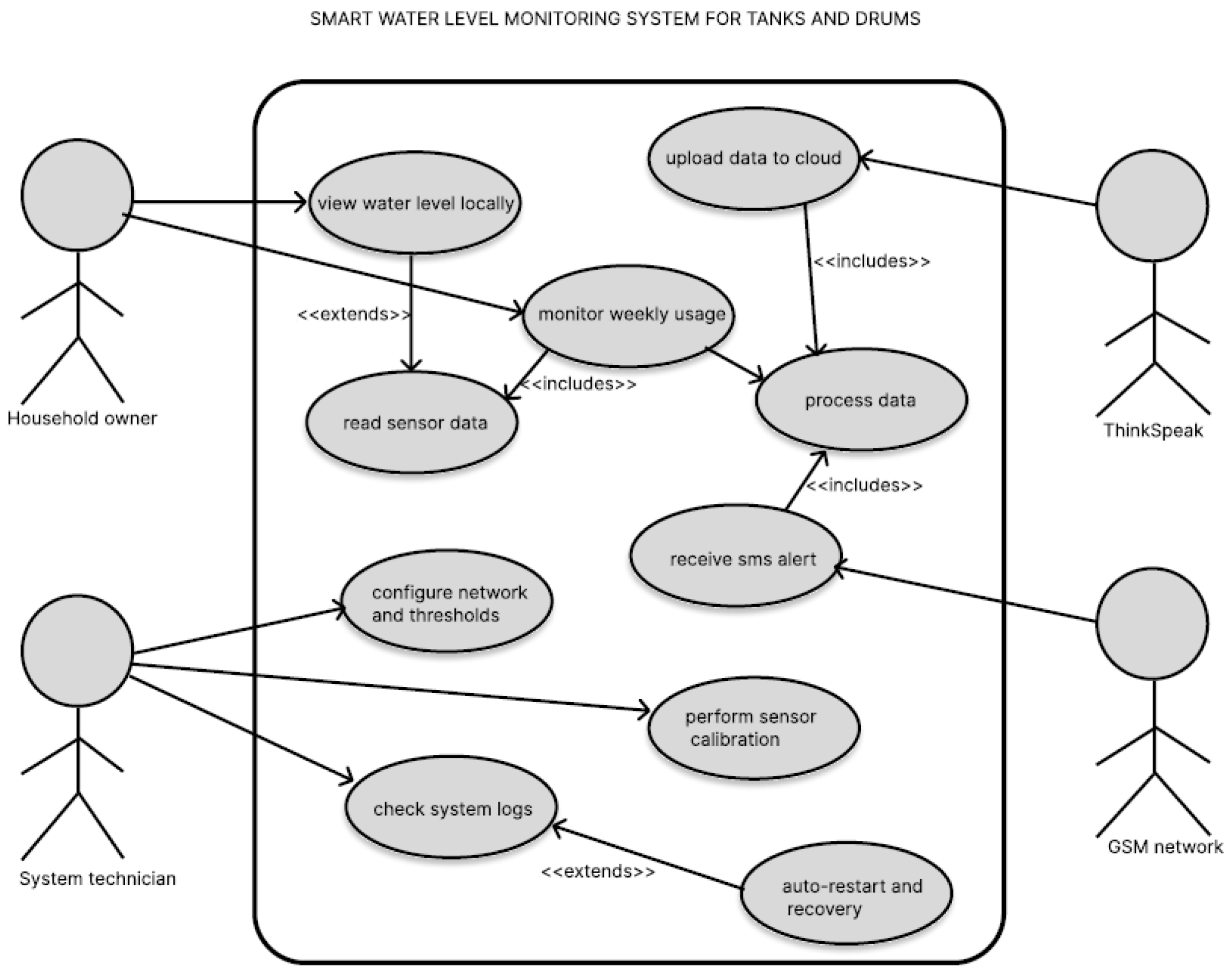
- ***Auto-Restart on Failure***
  The system shall restart its operations if it detects a failure in the sensor, GSM, or Wi-Fi modules to ensure continuous monitoring.

3.3 Behaviour Requirements

3.3.1 Use Case View

The system includes two primary actors: the User and the Admin. The User interacts with the system mainly through SMS alerts and the LCD display, receiving real-time water level updates and consumption reports. The admin, typically the system installer or technician, accesses the ThingSpeak dashboard to analyse trends and configure advanced settings. The major use cases include monitoring water levels, sending alerts, displaying data locally, and logging historical data online. This setup ensures both accessibility and detailed data analysis.

Figure 2: Use case Diagram.



4 Other Non-functional Requirements

4.1 Performance Requirements

.

Sensor Reading Frequency

- ***Requirement**: The HC-SRO4 sensor must read and process the water level at least once every **15 seconds**.*
- ***Rationale**: Frequent updates are necessary to provide near real-time level monitoring and to detect rapid water consumption or refilling events.*

LCD Update Response Time

- ***Requirement**: The LCD display must update the water level reading within 1 second after a new sensor measurement is taken.*
- ***Rationale**: Users rely on immediate visual feedback from the display, especially when checking tank status manually.*

SMS Alert Delivery Time

- ***Requirement**: When the water level drops below the critical threshold, an SMS alert must be generated and sent within 10 seconds.*
- ***Rationale**: Timely notifications allow users to respond quickly to low-water situations and avoid complete depletion.*

ThingSpeak Upload Interval

- ***Requirement**: The system must upload data to the ThingSpeak platform every 15 minutes without missing more than 1 update per day.*
- ***Rationale**: Regular data uploads ensure continuity in remote monitoring and support weekly trend analysis without significant gaps.*

System Boot and Initialization Time

- ***Requirement**: On power-up or reset, the system must complete initialization of all hardware modules (LCD, HC-SRO4 sensor, SIM800L GSM) and begin operation within 5 seconds.*
- ***Rationale**: Fast startup ensures minimal downtime in case of power interruptions or resets, maintaining the system's reliability.*

4.2 Safety and Security Requirements

Electrical Safety for Outdoor Installations

- ***Requirement**: All components, especially the ESP32, GSM module, and ultrasonic sensor, must be enclosed in waterproof and dust-resistant casing when installed in environments exposed to water or outdoor elements.*
- ***Rationale**: Prevents short-circuits, electrical damage, or system failure due to weather exposure.*

Safe Voltage Handling

- ***Requirement**: The system must operate on a safe low-voltage DC supply (≤ 5V) with overcurrent protection (e.g., fuse or voltage regulator).*
- ***Rationale**: Reduces the risk of electric shock or component burnout in case of power surges or connection errors.*

Alert Redundancy and Warning Labels

- ***Requirement****: The system shall issue both LCD warnings and SMS alerts when water levels are critically low. Physical labels should warn users about system dependence for water awareness.*
- ***Rationale****: Informs the user even if one method of alert fails, and prevents over-reliance on the system without visual backup or manual checks.*

Security Requirements

Although the system operates primarily offline, it handles sensitive data such as phone numbers, Wi-Fi credentials, and water usage history. The client expects basic security to prevent unauthorized access, especially via Wi-Fi or during SMS transmission.

- ***Expected Level of Security***
  The system must provide moderate security sufficient for a non-commercial, private household application. Focus is on data confidentiality, device access control, and safe transmission of critical alerts.

Key Security Requirements

1. ***Wi-Fi Credentials Protection***
   - *Wi-Fi SSID and password must be stored securely within code or external EEPROM and not exposed through serial monitor after deployment.*
2. ***SMS Alert Restrictions***
   - *SMS alerts must be sent only to authorized phone numbers preconfigured in the code. No open input for phone number configuration via SMS or console.*
3. ***ThingSpeak Channel Privacy***
   - *Data uploaded to ThingSpeak must be sent to a private channel using a secure API key to restrict unauthorized access.*
4. ***Authentication for Firmware Update (Optional Future Upgrade)***
   - *Future versions should support firmware update authentication (e.g., via physical button press or password input) to prevent tampering.*
5. ***Limited Serial Debug Access***
   - *Serial debug output should be disabled or password-protected after deployment to prevent exposure of sensitive information during system operation.*

4.3 Software Quality Attributes

Reliability

The system is built to recover from temporary network or power outages by retrying data uploads and storing missed messages in a queue. This ensures continuity of service.

Portability

Our software is written in Arduino-compatible C++ and can be deployed across various ESP32 platforms, making it flexible for different hardware setups.

Maintainability

To enhance long-term support, the code is divided into logical modules such as sensor_read (), send_sms (), and upload data ().

Usability

The user interface is designed with simplicity in mind. SMS messages use plain language, while the LCD displays clear and concise tank status updates. This makes the system usable by individuals with minimal technical background.

5 Other Requirements

The system stores data on ThingSpeak for 30 days, constrained by the free tier's storage limit. Currently, all user-facing outputs such as SMS and the web dashboard are in English. This version of the system complies with local regulations under the Uganda Communications Commission, particularly concerning the use of GSM communication modules for public deployment.

Appendix B - Group Log

- *01/07/2025: Group brainstorming and project proposal*
- *04/07/2025: Hardware components identified and ordered*
- *08/07/2025: Initial coding and sensor calibration*
- *12/07/2025: GSM module tested for SMS functionality*
- *16/07/2025: ThingSpeak integration complete*
- *20/07/2025: Final testing and debugging*
- *23/07/2025: Draft report writing started*
- *25/07/2025: Final report compilation and formatting*