West Texas A&M University

College of Engineering

Kirk Blood

Hector Ceballos

Dale Andre Descallar

Preston DeShazo

William Kolath

CS4390 and CS4391 Senior Capstone Project

TerraTek

Supervised by:

Dr. Mohammad Faridul H. Siddiqui. Ph.D.

Dr. Joshua Partheepan. ph.D.

Fall 2024 and Spring 2025

Department of Computer Science

Abstract

This project presents the development of TerraTek, an IoT-based monitoring system designed to assist a customer in tracking environmental data. The system utilizes Arduino-controlled sensor clusters to gather data on key parameters such as rainfall, water pH, water level, and so on, which are then transmitted to a centralized server. A responsive web interface allows users to visualize data trends and generate reports to aid decision-making processes. Through data analysis, the project demonstrates improved statistics by utilizing historical and real-time data. Future work will involve extending the system's capabilities by providing accessible ways to understand and interpret the collected data.

Plagiarism Declaration

With the exception of any statement to the contrary, all the material presented in this report is the result of my own efforts. In addition, no parts of this report are copied from other sources. I understand that any evidence of plagiarism and/or the use of unacknowledged third party materials will be dealt with as a serious matter.

Signed

William Kolath

Contents

List of Figures

List of Tables

Chapter 1

Software Requirement Specification

1.1 Introduction

1.1.1 Purpose

The purpose of this document is to provide a detailed description of the TerraTek system. It will explain the intended scope, purposes and features, constraints, and interactions between its systems. This document is intended for all stakeholders and developers of the system.

1.1.2 Project Scope

The intended scope of TerraTek is to provide a web interface for the user to observe telemetry data. Implementing clusters of sensors that will collect and aggregate data to a central database which can be parsed and graphed through a web interface. This will be implemented on small scale with the ability to expand to scale. The project scope is focused on rain water harvesting (RWH) systems, although the technology is capable of expanding to other applications.

1.2 Overall Description

1.2.1 Product Perspective

TerraTek will encapsulate a property consisting of rainwater harvesting tanks, garden areas, and a playa. There are four tanks in total with three of them holding rainwater and the fourth containing gray water. Each tank will contain a cluster of sensors with an additional cluster located at the playa. The cluster in the playa will provide weather data in addition to playa conditions. This is a small scale project to serve as a foundation for larger scale implementation.

1.2.2 Product Features

TerraTek will include the five features listed below:

Feature 1: Tank Monitoring

Feature 2: Weather Monitoring

Feature 3: Playa Monitoring

Feature 4: Remote access to data

Feature 5: Data visualizations

1.2.3 User Classes and Characteristics

There are two main classes of users for our project.

- Owner of RWH system
- Other users

The owner of the RWH is primarily focused on being able to understand the current and historical status of the RWH system. The weather conditions will be used to make informed decisions regarding garden planting and watering. They will also use the playa monitoring functionality to assess the heath of the playa ecosystem. They also must be able to make configuration changes to the TerraTek system to suit their changing needs.

Other users will be primarily focused on learning about how a RWH system works as well as seeing different weather data from the area visualized.

1.2.4 Operating Environment

The server will use Ubuntu Server Version 24.04.1 LTS, the most current version as of writing. The server will utilize MySQL for database operations. The sensors will be programmed with Arduino to communicate with a Raspberry Pi over LoRaWAN, which will aggregate and send all the data to the server.

1.2.5 Design and Implementation Constraints

A significant portion of the system will be deployed outdoors. Some portion of the system will deployed in water tanks, either submerged or directly above water. Water and weather proofing will be a significant constraint we must work around in this project. Another constraint is longevity. The system needs to be designed to operate for long periods of time without replacing hardware components or performing maintenance on the software.

1.2.6 User Documentation

We will create a user manual for the TerraTek system. There will also be a small online tutorial for users of the website, explaining what the system is monitoring and how to operate the website.

1.2.7 Assumptions and Dependencies

We are making the following assumptions:

- All sensors will be programmable on Arduino
- Each cluster contains only a singular sensor of its type

We are dependent on the following systems:

- Linux Server
- Home WiFi of the RWH owner
- Pre existing RWH system

1.3 Semester 1 System Features

Feature 1: Tank Monitoring

Each of the RWH tanks on the property needs to be monitored using various sensors. This is a very high priority feature.

REQ-1.1: The system must monitor tank level.

REQ-1.2: The system must monitor tank water PH.

REQ-1.3: The system must monitor tank TDC.

REQ-1.4: The system must monitor tank temperature.

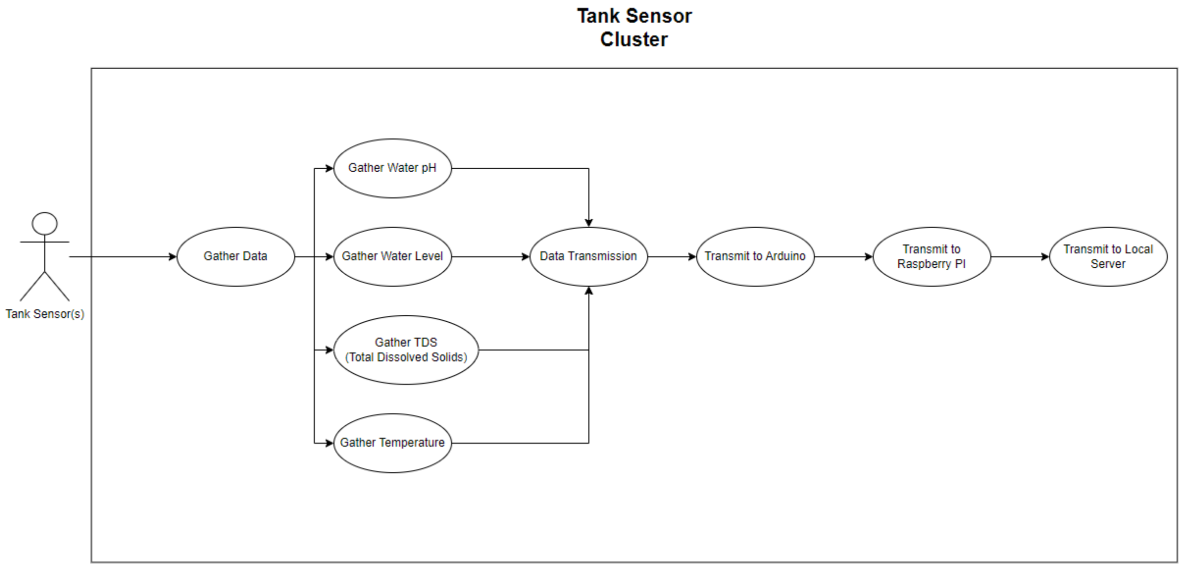

Figure 1.1: Use Case Diagram for Tank Sensors

Feature 2: Playa Monitoring

The playa will have sensors installed to monitor the health of the soil and current weather conditions. The weather monitoring functionality will allow the owner of the garden to make informed decisions regarding watering and planting. This is a high priority feature.

REQ-2.1: The system should monitor soil moisture in multiple locations of the playa.

REQ-2.2: The system must measure rainfall in the playa.

REQ-2.3: The system should measure atmospheric temperature at multiple locations in the playa.

REQ-2.4: The system should measure atmospheric pressure at multiple locations in the playa.

REQ-2.5: The system should measure wind speed in the playa. REQ-2.6: The system should measure wind direction in the playa.

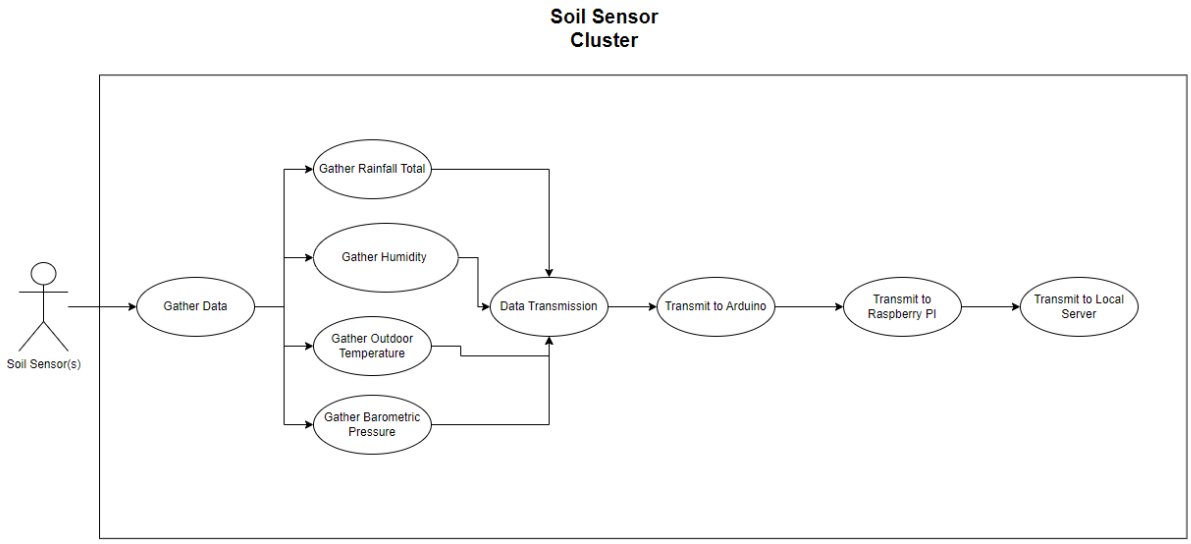

Figure 1.2: Use Case Diagram for Soil Sensors

Feature 3: Remote access to data

Users of the system must be able to access all of the gathered data from anywhere in the world. This feature enables the owner of the RWH system to see its current status even if they are not home. It also allows users interested in leaning about RWH systems to see one in action remotely. This is a very high priority feature.

REQ-3.1: The user must be able to access all current data using a web interface.

REQ-3.2: The user must be able to access historical data using a web interface.

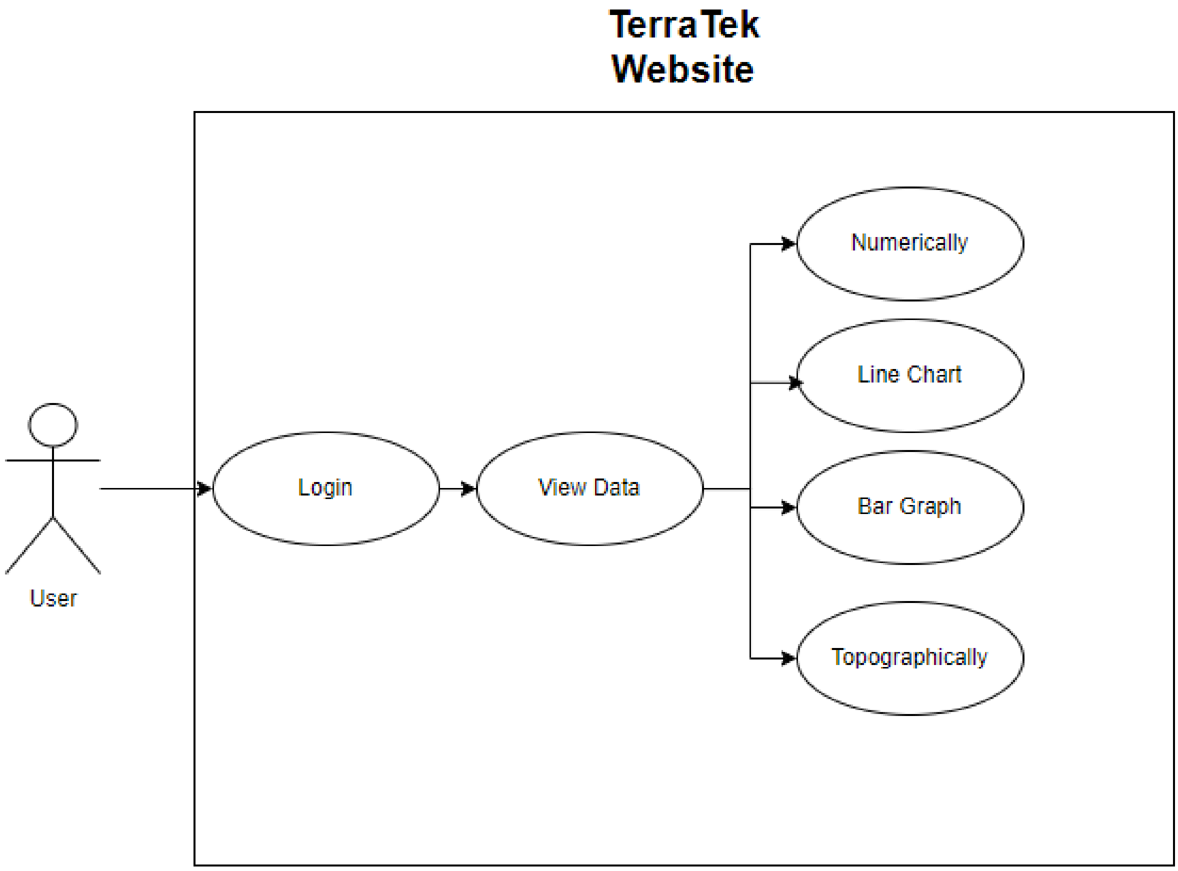

Figure 1.3: Use Case Diagram for Website

1.4 External Interface Requirements

1.4.1 User Interfaces

Users will interface with the system using a web browser to access the website. The system will support Firefox, Safari, and Chromium based browsers.

1.4.2 Software Interfaces

There will be many software interfaces. The Sensors use many different predefined interfacing protocols such as I2C. The sensor clusters will communicate to the central hub using LoRaWAN. The central hub will then send the aggregated sensor data to the server using a standard internet protocols.

1.5 Other Nonfunctional Requirements

1.5.1 Performance Requirements

REQ-6.4: The system should be built with no dependency on subscription based or free tier cloud services.

1.5.2 Software Quality Attributes

REQ-7.1: The website must be easy to use.

REQ-7.2: The website must remain accessible even if portions of the sensor network fail.

REQ-7.3: The system must be secure

1.6 Semester 2 System Features

Feature 1: Home Page

REQ-1.1: The user must be able to select a drop-down menu to move from page to page.

REQ-1.2: The user must be able to view the water level of the Main tanks Via a custom animation.

REQ-1.3: The user must be able to select between three types of graph to view the water level.

REQ-1.4: The user must be able to view current and predicted weather.

REQ-1.5: The user must be able to view the purpose and about section for background on the project.

REQ-1.6: The user must be able click on a help menu if they are having trouble with navigation.

REQ-1.7: The user must be able to change the orientation of the web page and still have it look presentable.

Feature 2: Tank Pages

REQ-2.1: The user must be able to view the freshwater tank and greywater tank pages properly

REQ-2.2: The user must be able to view the current water level, water pH, water TDC, water temperature

REQ-2.3: The user must be able to view the historical readings of the water level for the month

REQ-2.4: The user should be able to seamlessly navigate to other pages using the dropdown menu options

Feature 3: Weather Pages

REQ-3.1: The user must be able view current weather data

REQ-3.2: The user must be able to see temperature, humidity, illuminance, dew point, wind speed, and wind direction.

REQ-3.3: The user must be able to see separately soil moisture data for the Playa.

REQ-3.4: The user must be able see camera shot showing playa under playa data section.

REQ-3.5: The user must be able see current weather data from a nearby weather station.

REQ-3.6: The user must be able see weather forecast from nearby weather station.

REQ-3.7: The user must be able to choose between hourly or 7 day forecasting.

REQ-3.8: The user should be able to seamlessly navigate to other pages using the dropdown menu options.

Feature 4: Reports Page

REQ-4.1: The user must be able to select any the cluster and sensor

REQ-4.2: The user must be able to select a custom time range

REQ-4.3: The user must be able to select up to 3 unique sensors

REQ-4.4: The user must be able to chose which chart type they would like to generate

REQ-4.5: The user must be able to download the generated chart as either a PNG or a JPEG

REQ-4.6: The use must be able to download the raw data used to generate the chart as either a csv file or a JSON file

REQ-4.7: The user should be able to specify how frequent the data point on the chart appear

REQ-4.8: The user should be able to specify the aggregation method used to reduce the number of data points

REQ-4.9: The user must be able to switch between metric and imperial units.

REQ-4.10: The user should be able to customize the colors use in the charts

REQ-4.11: The page must include an easy to understand help page

REQ-4.12: The user must be able to generate line charts

REQ-4.13: The user must be able to generate scatter plots

REQ-4.14: The user must be able to generate bar charts

REQ-4.15: The user must be able to generate heat maps

REQ-4.16: The user must be able to generate histogram

REQ-4.17: The user must be able to generate box plots

REQ-4.17: The Page must including at a glance statistics about the selected data, including min, max, and mean

Feature 5: API

REQ-5.1: The user must be able query the database for data

REQ-5.2: The user must be able to specify which cluster and/or sensors it will query from

REQ-5.3: The user must be able to receive parsed or raw data from their query

REQ-5.4: The user must be able to specify a given timeframe of weeks/days/hours

REQ-5.5: The user must be able to specify how the data is calculated

REQ-5.6: The user must be able to mix and match these queries to suit their needs

Feature 6: System Health Page

REQ-6.1: The user must be able to see the health of all sensors.

REQ-6.2: The user must be able to respond to any error sent by the receivers.

REQ-6.3: The user must be able to get an alert if any sensor sends an error

REQ-6.4: The user must be able to see the location of the effected sensor.

REQ-6.5: The user must be see transmission history for each node.

Feature 7: Remote camera

Users of the system could be able to show the users live views of the tanks and surrounding property. This enables users interested in RWH systems to actually see one in action. This is a low priority feature.

REQ-7.1: The user could be able to view live images of the tanks

REQ-7.2: The user could be able to view live images of the property

1.7 Nonfunctional Requirements

1.7.1 Performance Requirements

REQ-8.1: The website must load the homepage in less than 2 seconds

REQ-8.2: The server must be able to handle at least 200 users accessing the website at the same time.

REQ-8.3: The system should be able to create historical reports in less than 5 seconds

REQ-8.4: The system should be built with no dependency on subscription based or free tier cloud services.

1.7.2 Software Quality Attributes

REQ-9.1: The website must be easy to use.

REQ-9.2: The website must remain accessible even if portions of the sensor network fail.

REQ-9m ,.3: The system must be secure

Chapter 2

Software Design

2.1 Research

2.1.1 Arduino Sensors

DFRobot Gravity Analog Electrical Conductivity Meter (DFR0300)

The DFRobot Gravity Analog Electrical Conductivity Meter is designed to measure water's electrical conductivity, providing insight into the dissolved salts and overall water quality. This sensor is particularly useful in applications such as hydroponics and aquarium management. Using the DFRobot Gravity library, developers can easily calibrate the sensor and collect data efficiently, enhancing the user's ability to monitor water quality in real time [19].

Weather Meter Kit (SEN-15901)

The Weather Meter Kit is a comprehensive package that includes sensors to monitor weather conditions, such as wind speed, wind direction, and rainfall. These measurements are essential for setting up weather stations, where real-time data can be analyzed to predict weather trends. Integration with this kit varies depending on the specific sensors, as they may require unique libraries or standard analog and I2C communication protocols to process data accurately [24].

ME007YS Waterproof Ultrasonic Distance Sensor (SEN0312)

The ME007YS is a robust ultrasonic sensor designed for distance measurement in outdoor and industrial environments, even underwater. This sensor operates by emitting ultrasonic waves and capturing reflections. The NewPing library simplifies handling ultrasonic pulses and interpreting distance measurements, making this sensor reliable for applications in challenging environments [2].

Weather-proof Ultrasonic Sensor (SEN0208)

Similar to the ME007YS, the SEN0208 sensor provides reliable distance measurements in outdoor settings and is weatherproof. Utilizing the NewPing library, this sensor is capable of accurate data acquisition in harsh weather, making it suitable for long-term outdoor installations [25].

Waterproof DS18B20 Temperature Sensor Kit (KIT0021)

The DS18B20 is a popular waterproof temperature sensor known for its durability in harsh conditions. Using the OneWire protocol, multiple sensors can connect to a single data pin, a unique feature beneficial for applications requiring distributed temperature monitoring. The DallasTemperature library enables easy data acquisition from this sensor, allowing for seamless integration in temperature-critical monitoring setups [3].

Arduino MKR Environmental Shield (ASX00029)

The Arduino MKR Environmental Shield includes sensors for temperature, humidity, barometric pressure, and light intensity. This shield is particularly suitable for comprehensive environmental monitoring applications. The Arduino-MKRENV library simplifies data handling across multiple sensors, streamlining the process of integrating environmental metrics into projects [22].

Gravity Non-contact Liquid Level Sensor (SEN0204)

This non-contact sensor is ideal for environments where the liquid should not be touched directly, such as in sterile or hazardous conditions. The DFRobot-Sensor library offers straightforward methods for obtaining liquid level data, allowing for quick and safe implementation in sensitive monitoring applications [18].

MODBUS-RTU RS485 4-in-1 Soil Sensor (SEN0604)

This Soil Sensor Provides 4 different data outputs; Soil Moisture, Temperature, PH, and EC. This sensor is also built to last buried in the soil it is measuring, allowing it to be non-invasive and low maintenance. The ArduinoModbus Library allows for straightforward data gathering from the sensor to allow for quick setup and easy deployment. [4]

2.1.2 Website Framework

React.JS

ReactJS is a JavaScript library developed by Facebook for building user interfaces, primarily focused on component-based development, DOM(Document Object Model) manipulation, and various other functionalities [9]. It allows developers to create reusable UI components that handle the view layer of web applications efficiently. React updates and renders only the components that change, which improves performance and user experience. It also features a virtual DOM, which speeds up updates by minimizing direct manipulation of the real DOM.

NextJS

Nextjs is a React-based framework for building web applications [9]. It adds powerful features such as server-side rendering (SSR), static site generation (SSG), and dynamic routing, which enhance performance and SEO(Search Engine Optimization). With Nextjs, developers can render pages on the server before sending them to the client, making the initial load faster and more SEO-friendly. It also integrates well with APIs and data fetching techniques for modern web applications.

RESTFUL API

RESTful APIs, an acronym for Representational State Transfer, represent a design philosophy for creating lightweight, scalable, and easily maintainable web services. Emerging from Roy Fielding's 2000 dissertation [5], REST leverages standard web protocols like HTTP to create resource-based architectures.

Key characteristics of RESTful APIs include:

- **Statelessness**: Each client request must contain all the necessary information, ensuring the server doesn't maintain client state, which enhances scalability.
- **Uniform Interface**: REST enforces a consistent, intuitive interface for resources, typically represented by URLs. Common HTTP methods (GET, POST, PUT, DELETE) align with CRUD operations (Create, Read, Update, Delete).
- **Resource Representation**: Resources are represented in various formats like JSON, XML, or HTML, with clients capable of specifying their preferred format.
- **HATEOAS**: Hypermedia As The Engine Of Application State allows clients to dynamically discover available actions based on their current state.

REST is simpler and more suitable for web and mobile environments [13]. RESTful APIs are widely adopted in industries like social media, e-commerce, and IoT due to their scalability and compatibility with modern development paradigms.

For developers, designing a robust RESTful API involves adhering to best practices, such as versioning endpoints, providing intuitive error messages, and ensuring comprehensive documentation. Such practices maximize reusability, developer productivity, and user satisfaction, making RESTful APIs a cornerstone of modern software development

2.1.3 LoRaWAN

LoRa

LoRa, which stands for long range, is a wireless transmission protocol designed for long range, low power, applications. LoRa is capable of transmitting up to 5 kilometers in urban environments and up to 10 kilometers in rural environments with line of sight between devices [1]. The reason that LoRa is capable of transmitting such long distances is that it uses a radio modulation technology known as Chirp Spread Spectrum or (CSS) [14]. CSS works by sending series of chirps that encode data. Other radio protocols such as amplitude modulation (AM) and frequency modulation (FM) are highly susceptible to noise and obstructions and require high powered transmission devices to achieve the distances that they do. CSS, on the other hand, is very resilient to noise and obstructions, which makes it an ideal choice for urban environments [26]. The other thing that makes LoRa stand out is its incredibly low power usage. Unlike other radio protocols, it takes very little power to transmit and receive LoRa. This enables the use of batteries and solar in remote end-nodes [20], which makes LoRa a perfect choice for rural and farming applications. This is also what makes LoRa a great choice for our project. LoRa operates on license free sub-gigahertz bands, in the United States a common frequency choice is 915 MHz [17]. This means that there fewer costs and complexity that would be associated with operating on other bands. The major drawback to LoRa is data transmission rates, you can expect to see transmission speeds ranging from 250bit/s to 11kbit/s, however, for end nodes collecting data every couple of minutes this does not pose a significant issue [10].

LoRaWAN

LoRaWAN acts as a media access control (MAC) layer for the end nodes on the LoRa network [12]. LoRaWAN is a star network topology with many end nodes connecting to a gateway. Each gateway can support hundreds of devices, depending on how often they transmit data [20]. The gateway is responsible for collecting the packets sent by the end nodes and forwarding them to a network server.

LoRaWAN network server

A network runs software that manages one or more gateways. The network server typically stores the packets it receives from the various end nodes in a database and allows users to integrate their projects with the server. This entire architecture is a star of stars model and allows one network to cover a vast area with many gateways [26]. There are two main categories of networks servers, public and private. The main difference between the two categories is the entity operating the server and how access to the network is configured [6].

Public network servers (such as The Things Network) allow any end node that has been registered with the network to connect to any gateway on the network and have their packets forwarded to the users application. Public servers typically offer a tiered subscription model, many offer a free tier with support for a limited number ov devices and features [15]. It is worth noting that any end node can connect to any gateway, however, only end nodes that have been configured with the same network server will have their packets routed correctly.[12] All communication is encrypted and only the person who registered the end node is able to read the data, even if they do not own the gateway that the end node is connected to [12].

The other type of network servers are private network servers. These servers are hosted by the network owner either locally or on a web hosting platform and allow for more secure and private handling of data [6]. For many beginners, using a public network server is very convenient, all that they need to do is configure their gateway to forward packets to a public server like The Things Network, and they are ready to start connecting their end nodes. It is even possible that someone else nearby has a gateway that can be used instead of acquiring their own gateway. Setting up a private network server is a more complex undertaking and requires setting up a server to run the software that will control the gateways [6]. For this project we will be creating our own private network server to host our application.

ChirpStack

For people wanting to configure their own private network servers, there are several open-source projects that package all the network server components into one place. One of these projects is ChirpStack [23]. ChirpStack even has a dedicated Raspberry Pi OS that is ready to be configured as a network server. ChirpStack also includes a web interface for easy configuration and end node management. Additionally, ChirpStack provides a python package that enables easy integration using the gRPC API [7]. For these reasons, we believe that ChirpStack will work well for our project.

Gateways

As mentioned above, the gateway is responsible for collecting packets from end nodes and forwarding them via the internet to the network server. This typically occurs using UDP or a similar protocol. There are many gateways available with many different applications. One interesting option that we are currently investigating for our project is creating our own gateway using a Raspberry Pi and a gateway hat. These gateway hats, such as Elecrow's LR1302 HAT, connect directly to the Raspberry Pi and allow for a lot of configuration and flexibility [27]. The same Raspberry Pi running the gateway can also run the network server, this is a very compact and elegant solution that offers a lot of flexibility. The main concern is range, this is something that we need to test to determine if it is a viable solution for our project.

End Nodes

There are numerous end nodes that work using LoRa. The one that we have chosen for our project is the Arduino MKR WAN 1310. This board comes with the advantage of having all the support, documentation, and libraries that Arduino provides for all of its products. This board provides an easy way to integrate all the required sensors and connect to a gateway using LoRa [11].

2.1.4 PCB Design

For this project we opted to use a custom PCB to allow us to simplify the final form factor of the end nodes. The PCB design software used was KiCad. KiCad is an excellent option because it is open source, relatively easy to use, and supports many different types of hardware. [8] There are many considerations when designing a PCB such as materials, trace thickness and routing, vias, silkscreens, etc. [21]

2.1.5 Database

A database solution was required as for the implementation of the TSS would require a way to both store and organize the data collected from all sources.

SQL

SQL was decided as the database solution for this specific implementation. It was chosen as all members were familiar with the language as well as being perfectly suited for processing and storing text outputs collected from sensor data. How SQL expresses data is through mathematical relations [16]. The data collected from the raspberry pi will be expressed as the following relation:

ER Diagram

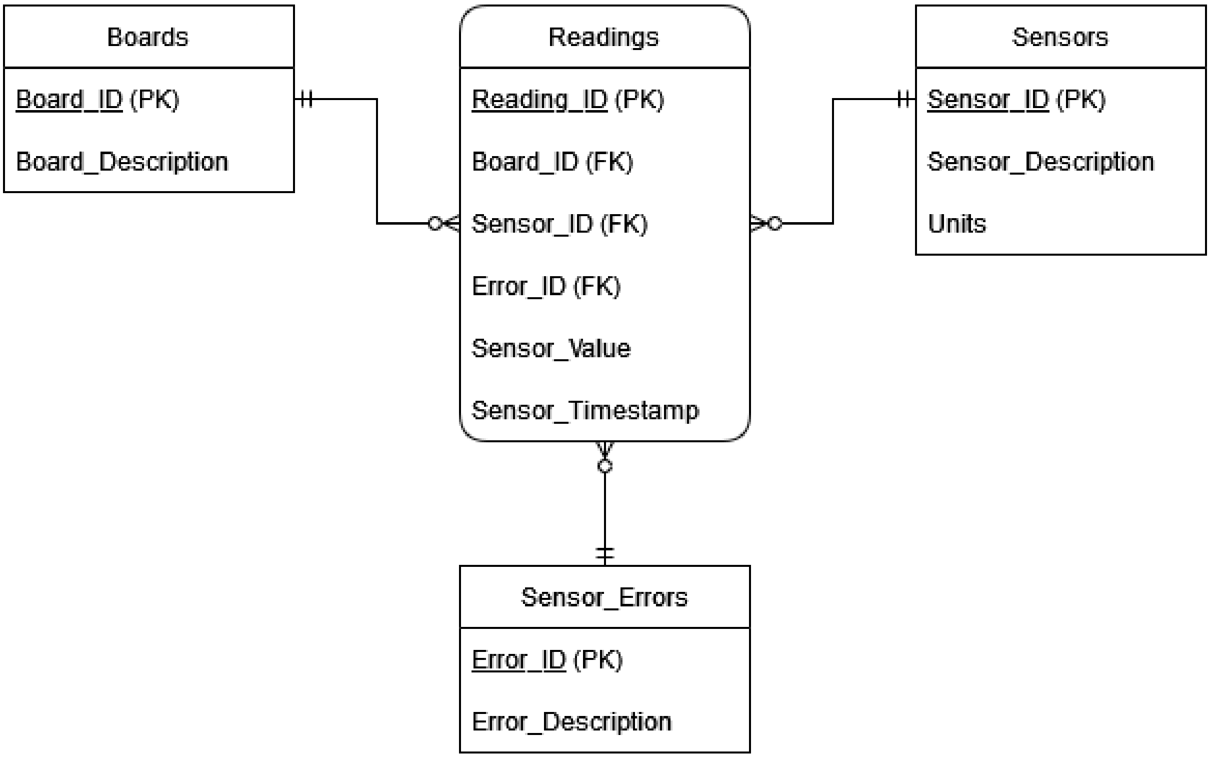

Figure 2.1: ER Diagram

This will serve as our schema for the database. Boards, Sensors and Readings serving as the entities present within. The Sensors table will represent individual sensor readings and the kind of sensor that is reading the data. The Boards table will represent a Cluster of sensors as well as that Cluster's location. Finally the Readings table will take the Board_ID and Sensor_ID from the Boards and Sensors tables in order to express the readings from any given sensor, which Cluster that sensor belongs too, and the time that the sensor has taken that specific reading.

2.1.6 Deployment Location Mapping

Figure 2.2: Water Tank Map

Water Tanks

There are 4 water tank locations as shown above in the photo. The two southern-most are fed by the house, fresh water from the roof catching rain, and gray water from the house appliances after use. The fresh water from the house is used to feed all of the houses water use, and the gray water is used to water plants in the area surrounding the house. The 2 northern tanks are fed by either side of the garage with fresh rainwater, and this water is used mostly to water the gardens they are adjacent to.

Garden Plots

There are many garden plots that we could attempt to gather data on, but the overhead of running so many sensors may prove to be extensive. To mitigate this the project will likely only be working on a select few of the garden plots to gather data. There are 4 raised garden boxes on the northern side of the garage, these 4 are used to grow annual vegetables.

2.2 Semester One Diagrams

2.2.1 Use Case Diagram

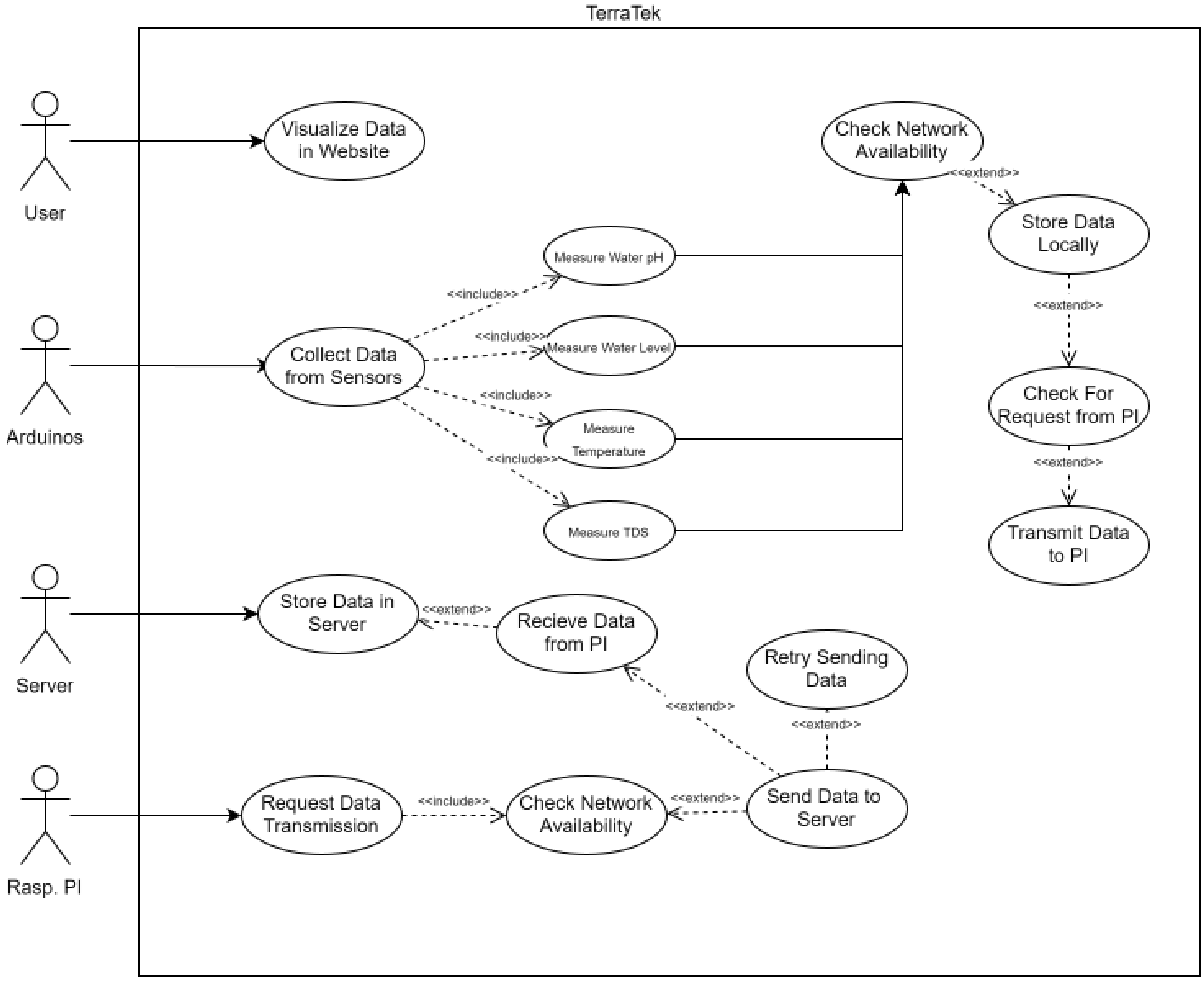

Figure 2.3: Use Case Diagram

2.2.2 Flow Diagram

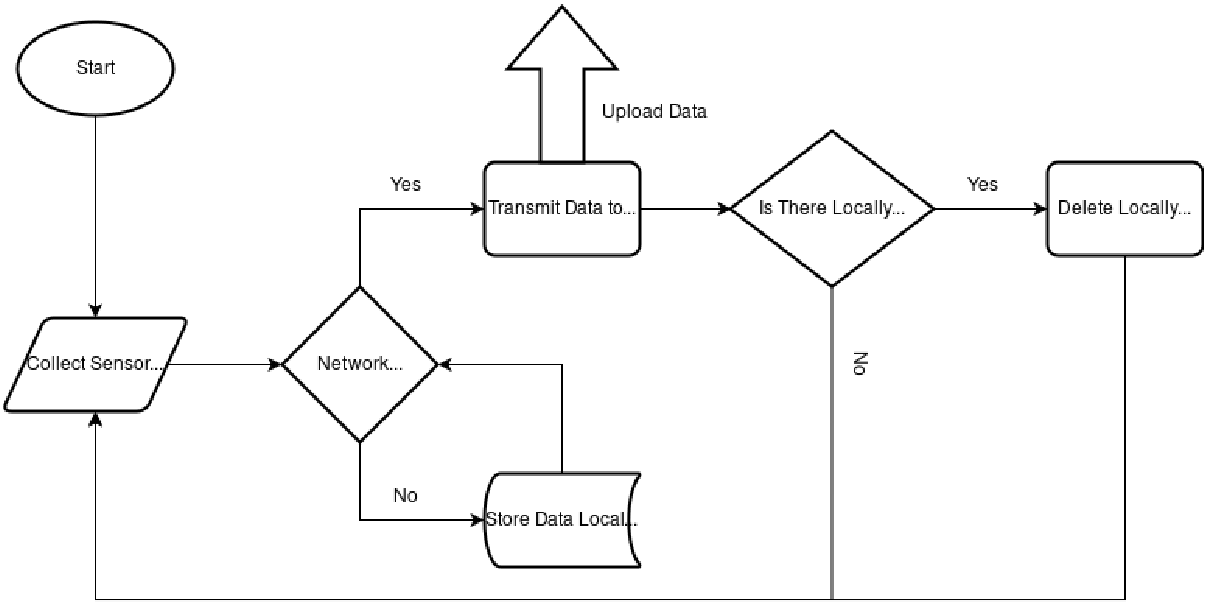

Figure 2.4: Flow Diagram

2.2.3 ER Diagram

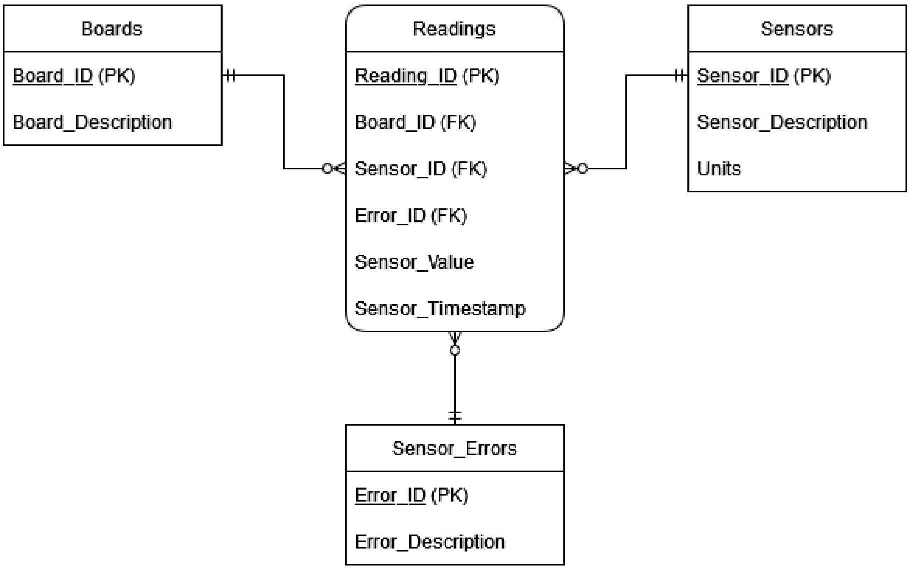

Figure 2.5: ER Diagram

2.2.4 Class Diagram

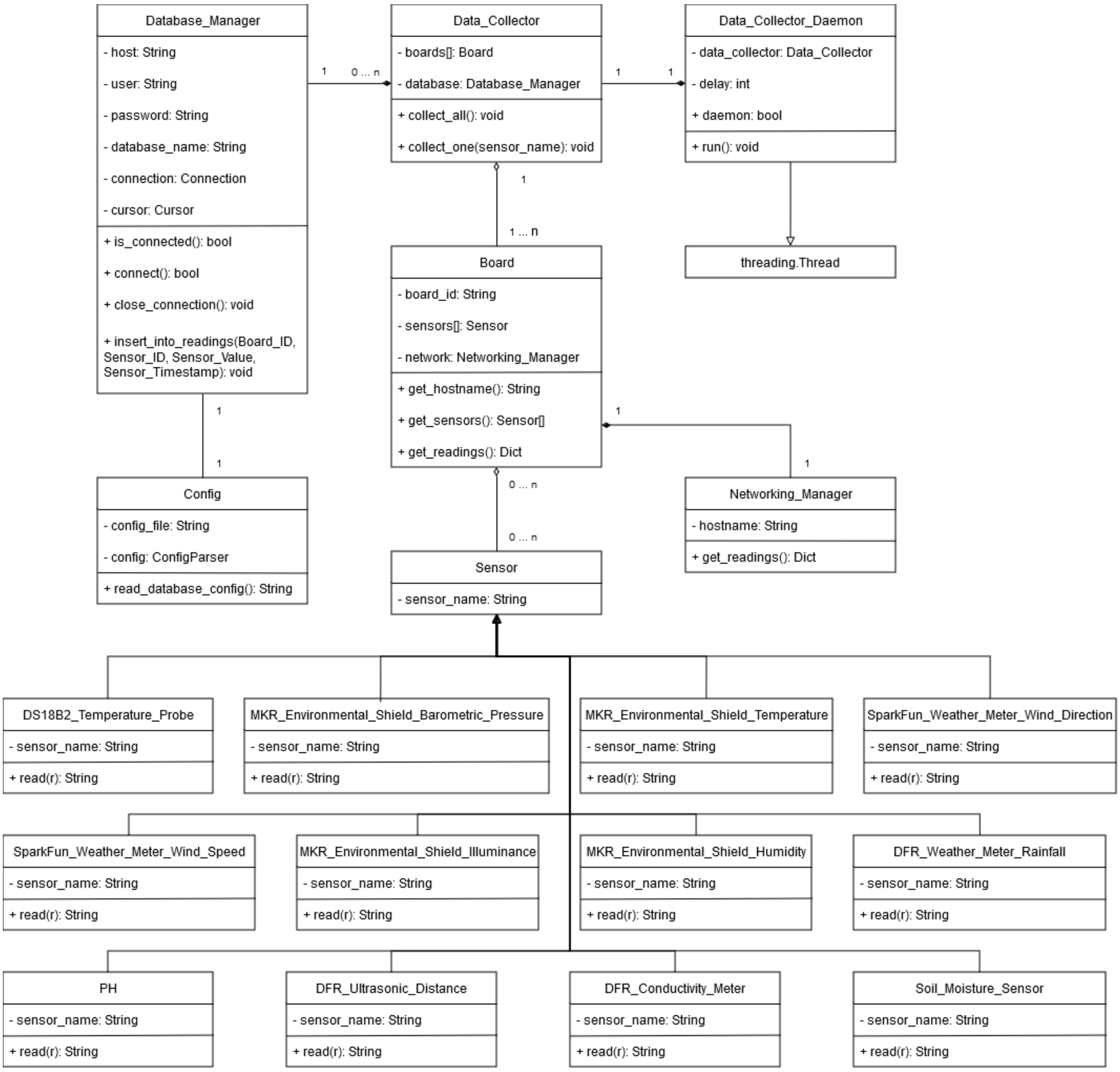

Figure 2.6: Class Diagram (Raspberry Pi WiFi)

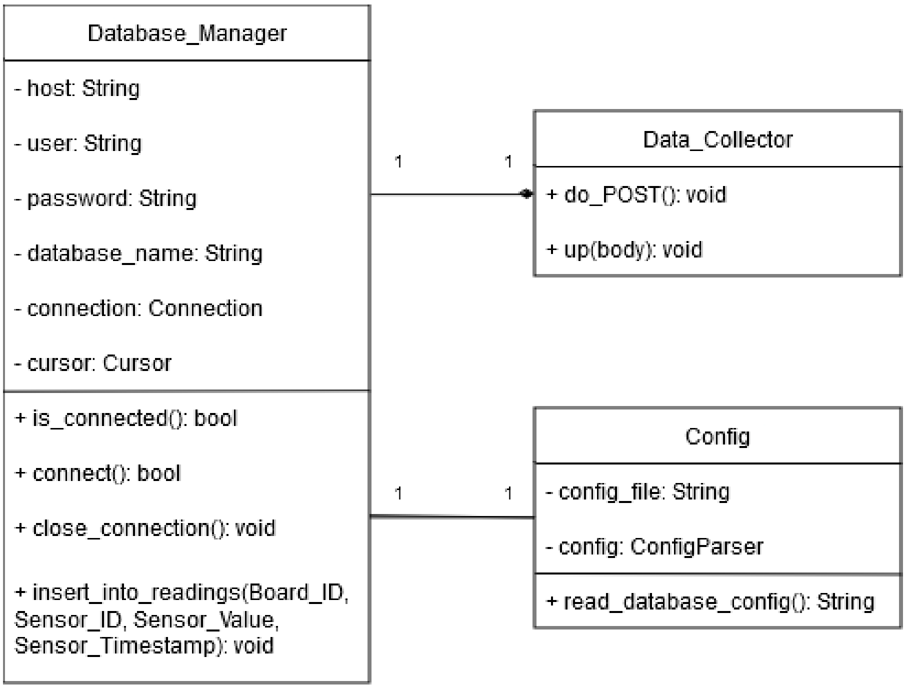

Figure 2.7: Class Diagram (Raspberry Pi LoRaWAN)

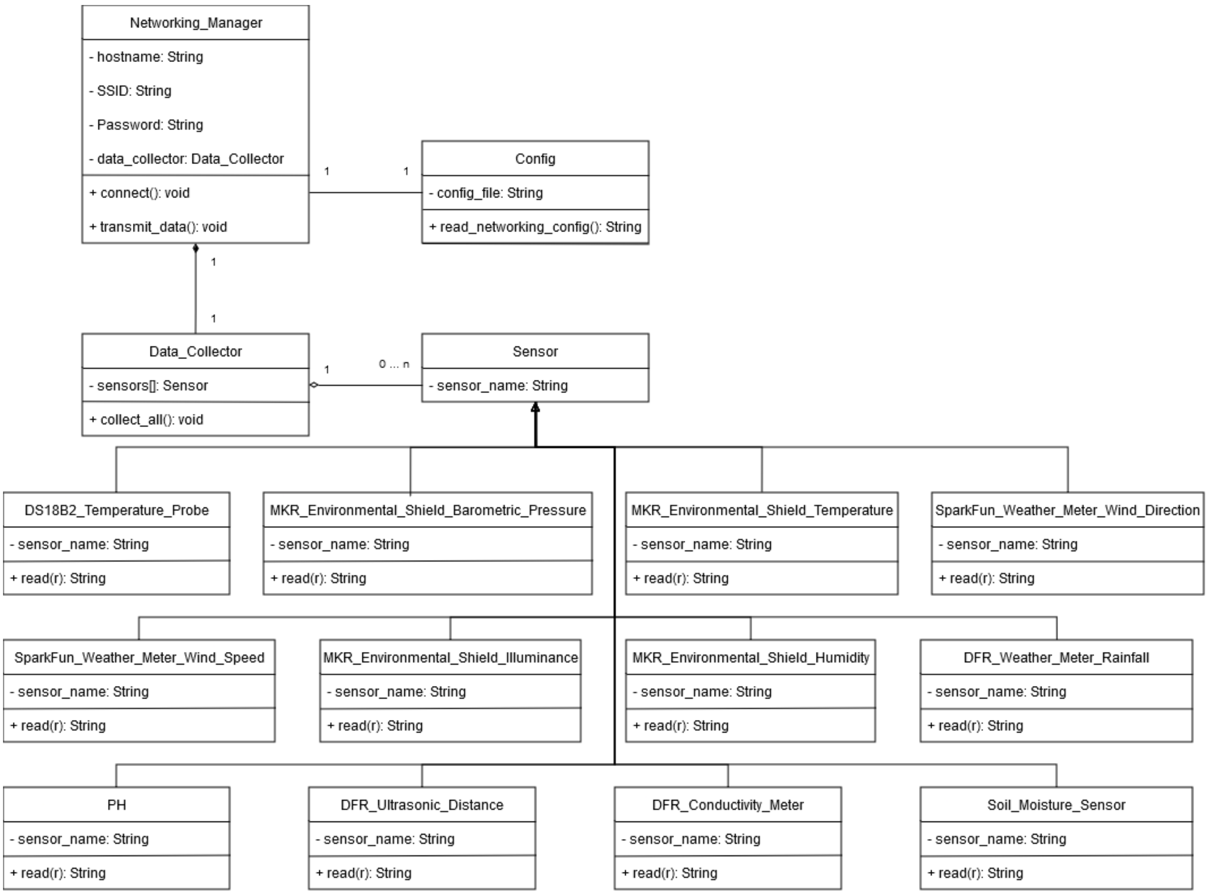

Figure 2.8: Class Diagram (Arduino)

2.2.5 Object Diagram

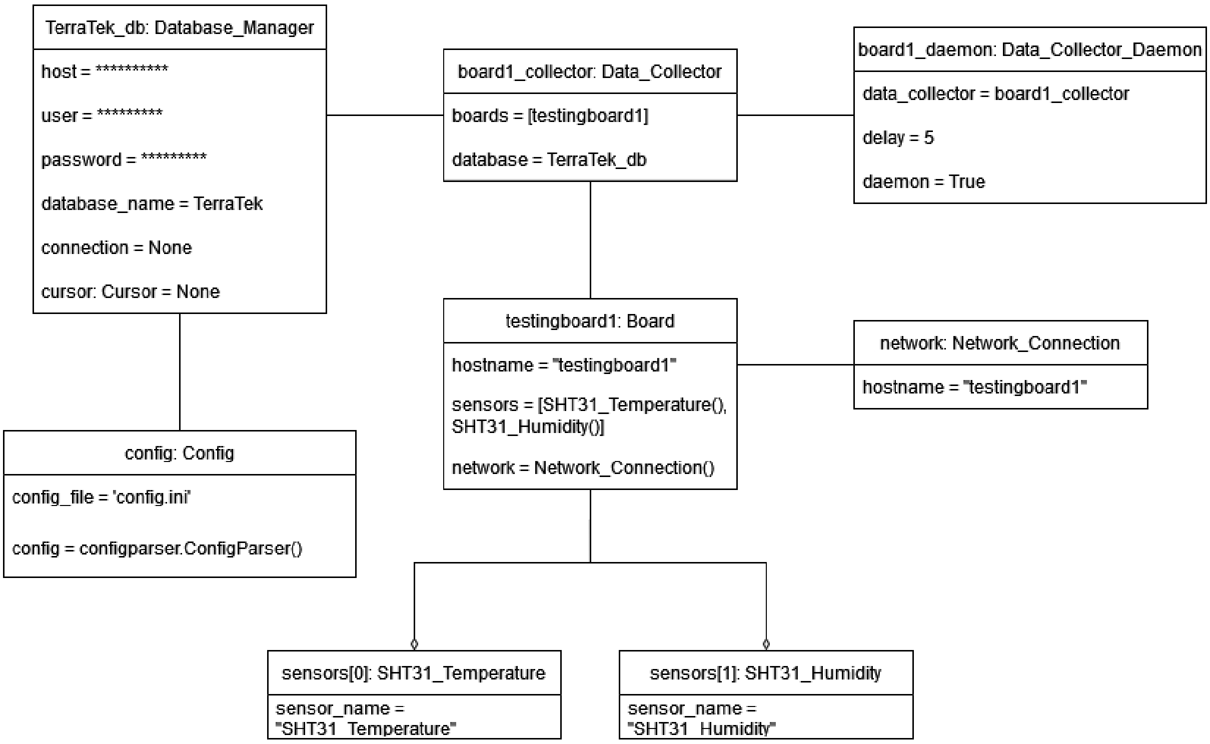

Figure 2.9: Object Diagram

2.2.6 Sequence Diagram

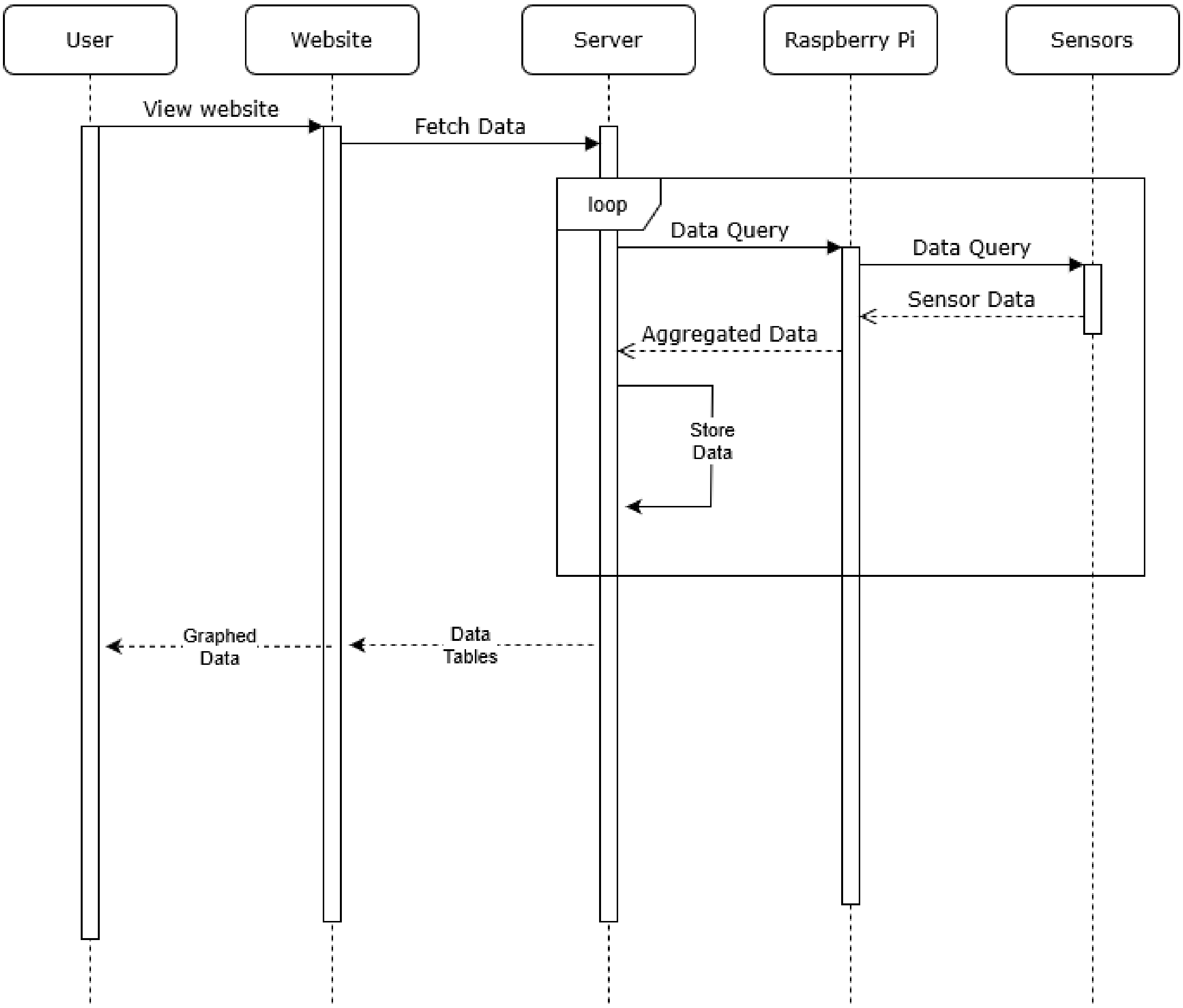

Figure 2.10: Sequence Diagram

2.2.7 DFD Level 0 Diagram

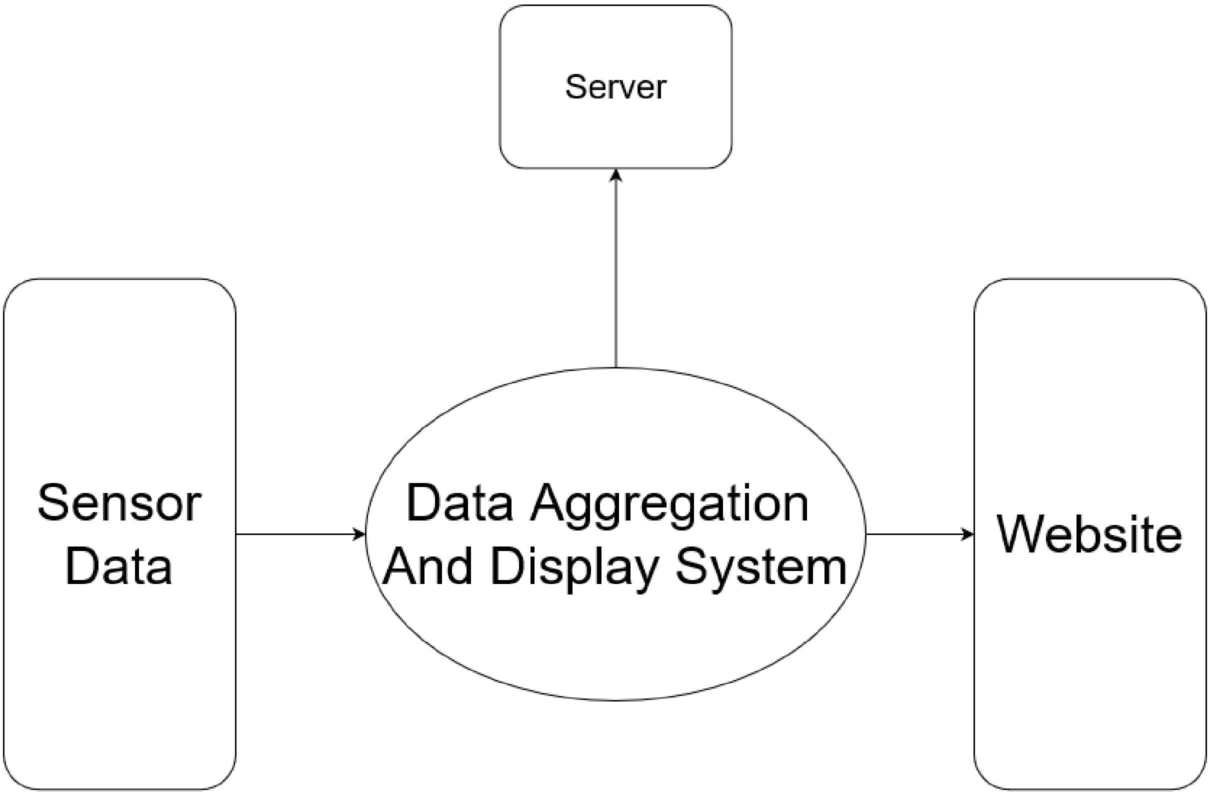

Figure 2.11: Data Flow Diagram Level 0

2.2.8 DFD Level 1 Diagram

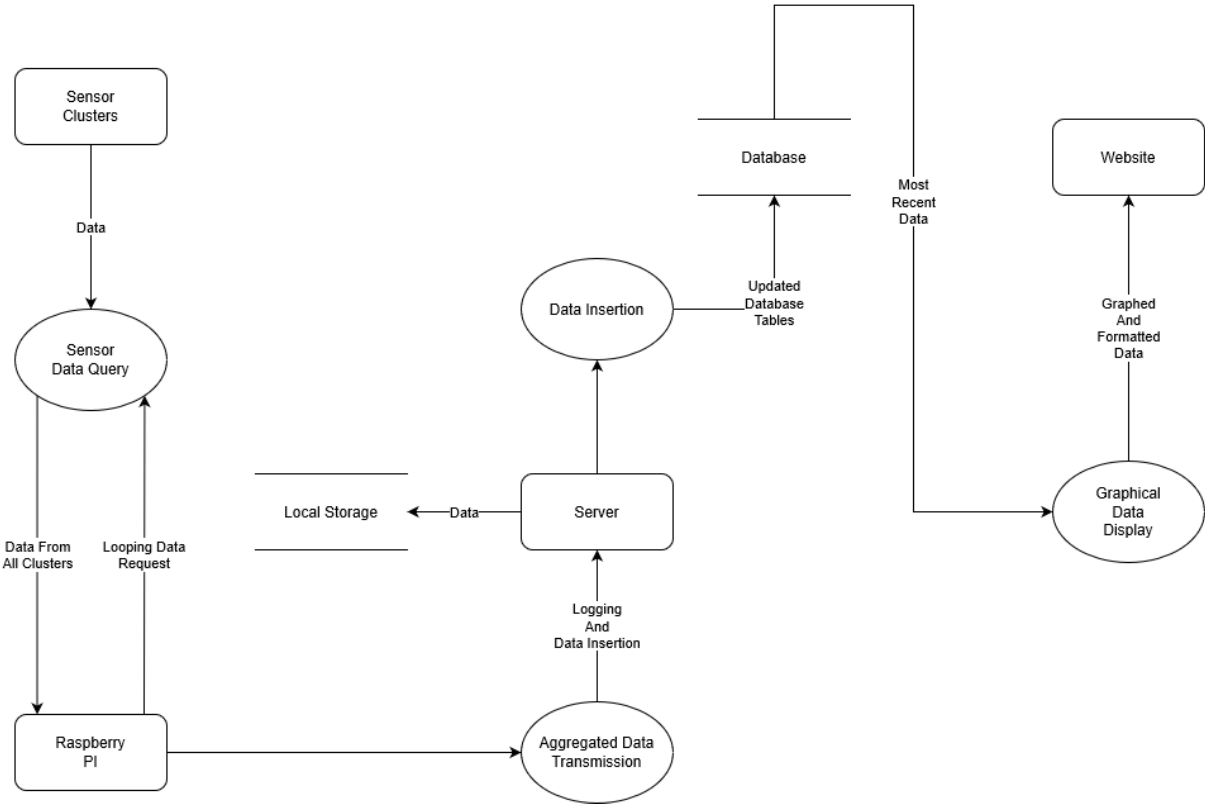

Figure 2.12: Data Flow Diagram Level 1

2.2.9 Raspberry Pi State Machine Diagram

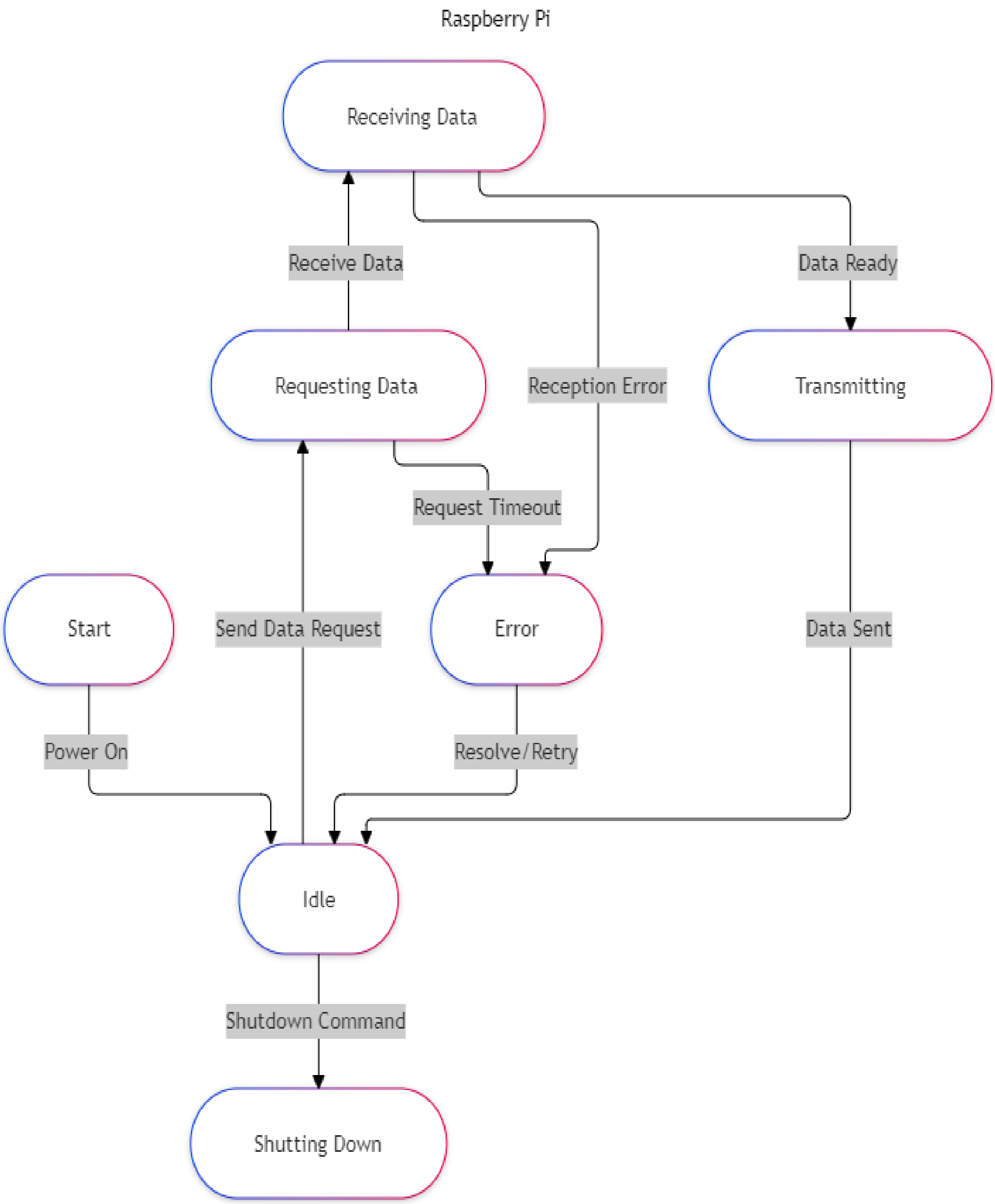

Figure 2.13: State Machine For Raspberry Pi

2.2.10 Sensor Cluster State Machine Diagram

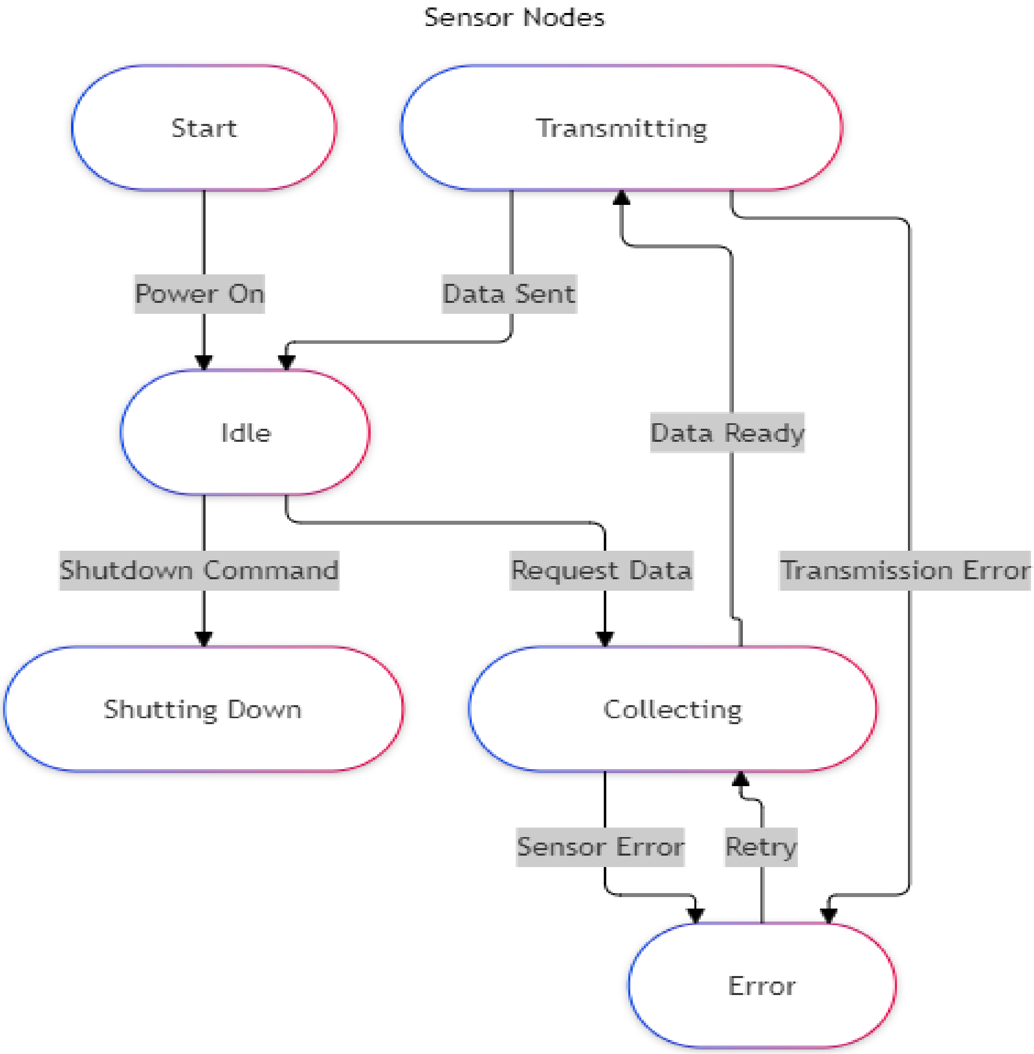

Figure 2.14: State Machine For Sensor Clusters

2.2.11 Server State Machine Diagram

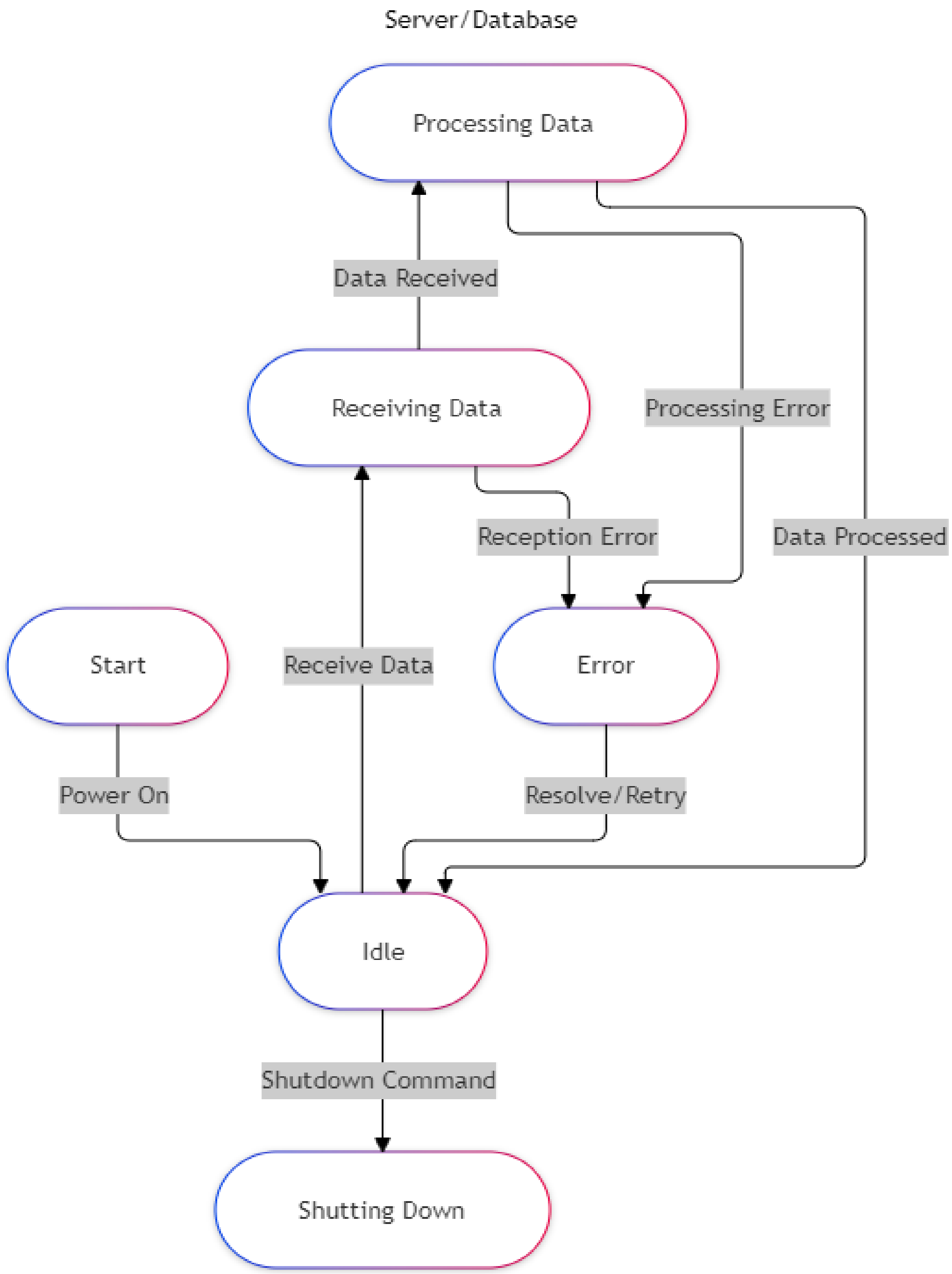

Figure 2.15: State Machine For Server

Website State Machine Diagram

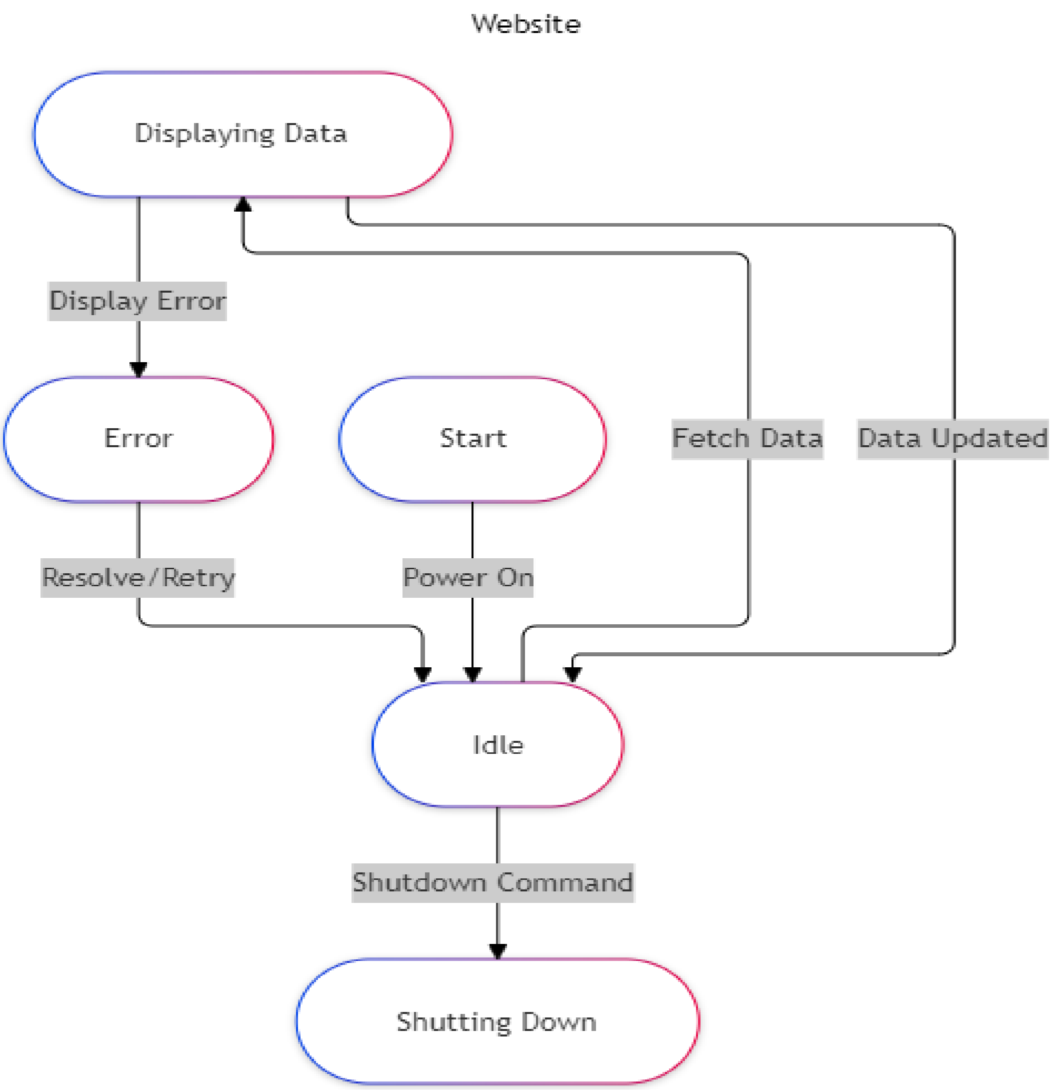

Figure 2.16: State Machine For Website

2.2.12 Component Diagram

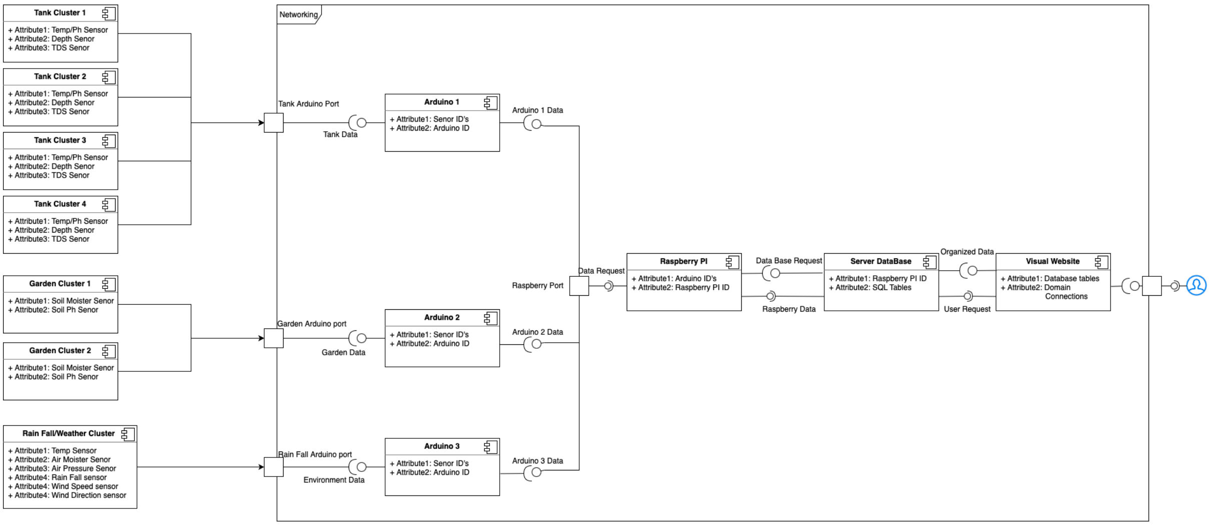

Figure 2.17: Component Diagram

2.2.13 Schematic

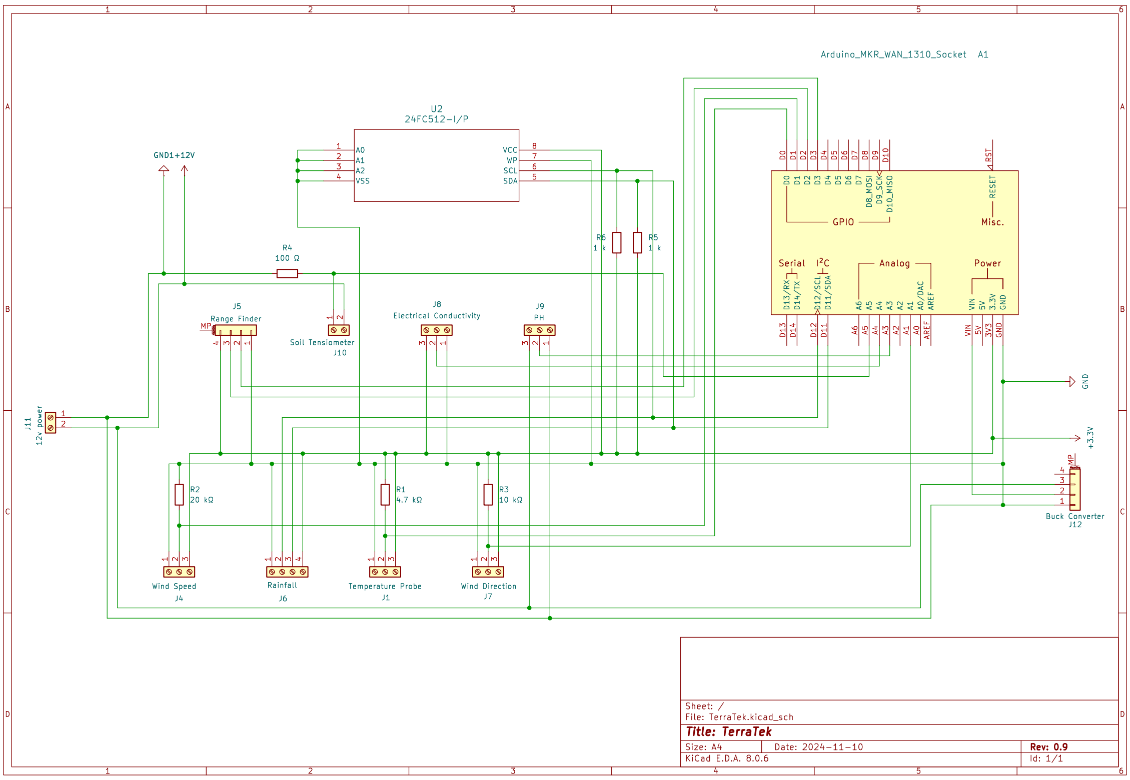

Figure 2.18: Schematic

2.3 Semester Two Diagrams

2.3.1 Website Use Case Diagram

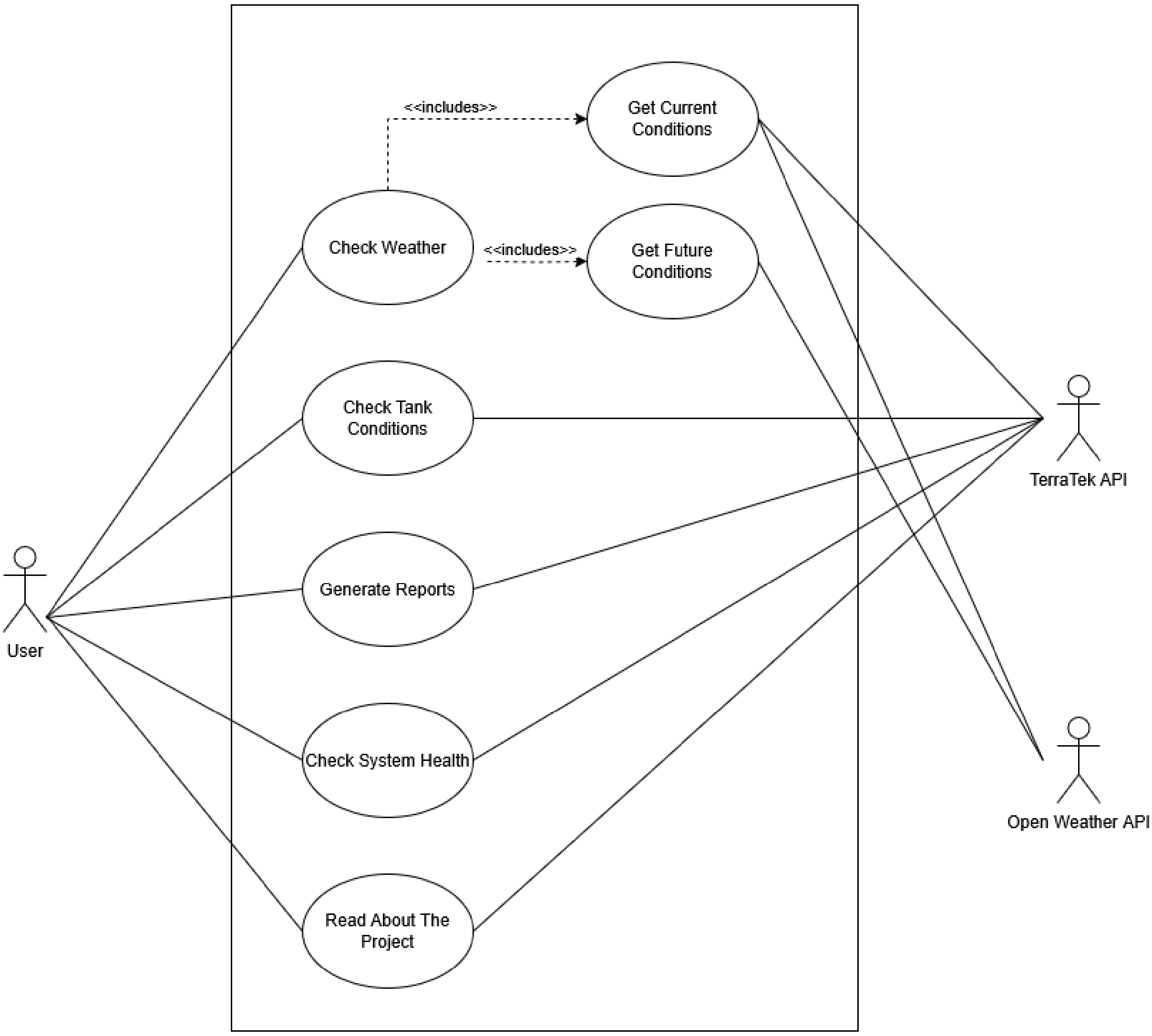

Figure 2.19: Use Case Diagram

2.3.2 Home Page Weather Activity Diagram

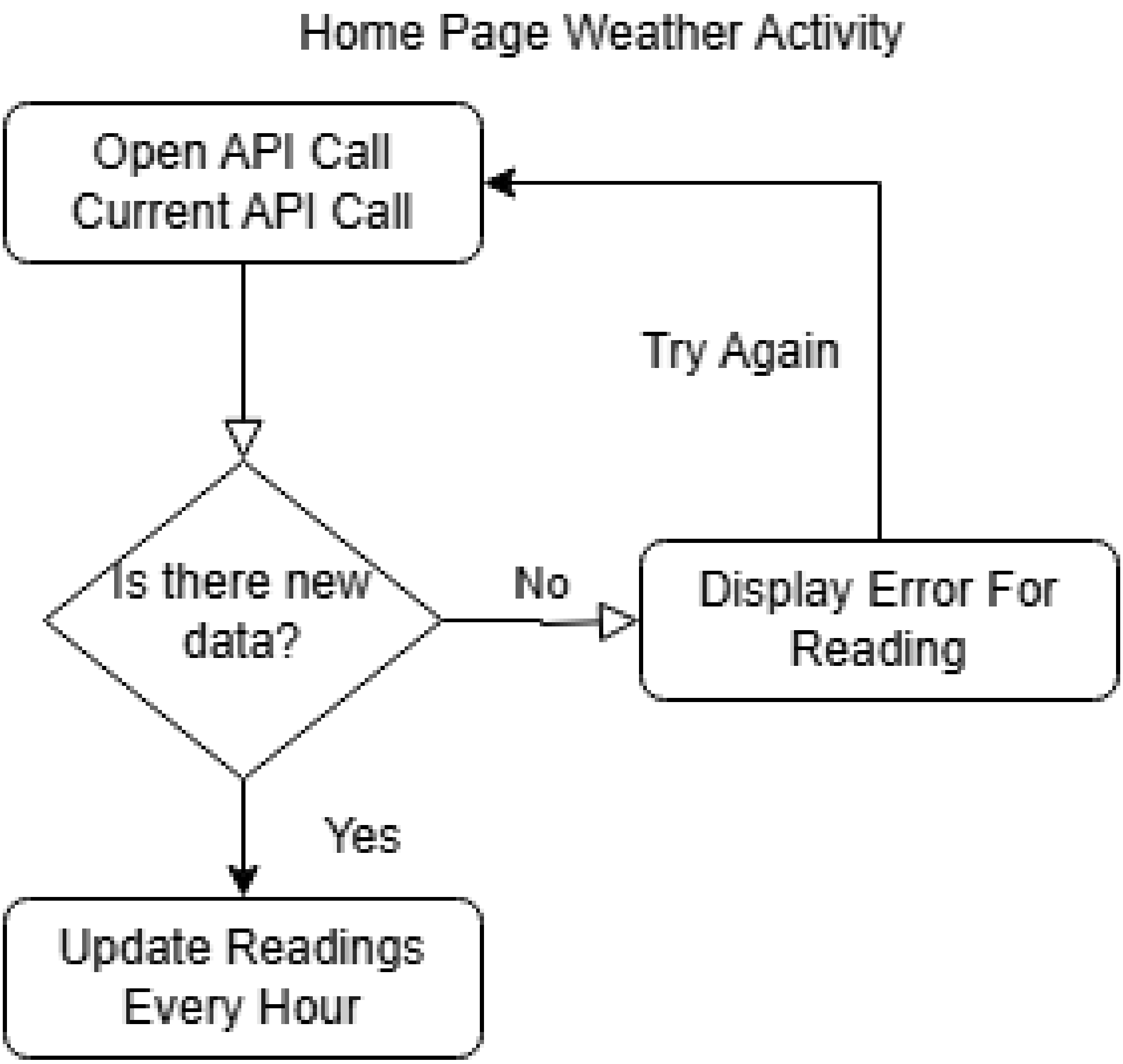

Figure 2.20: Home Page Weather Activity Diagram

2.3.3 Home Page

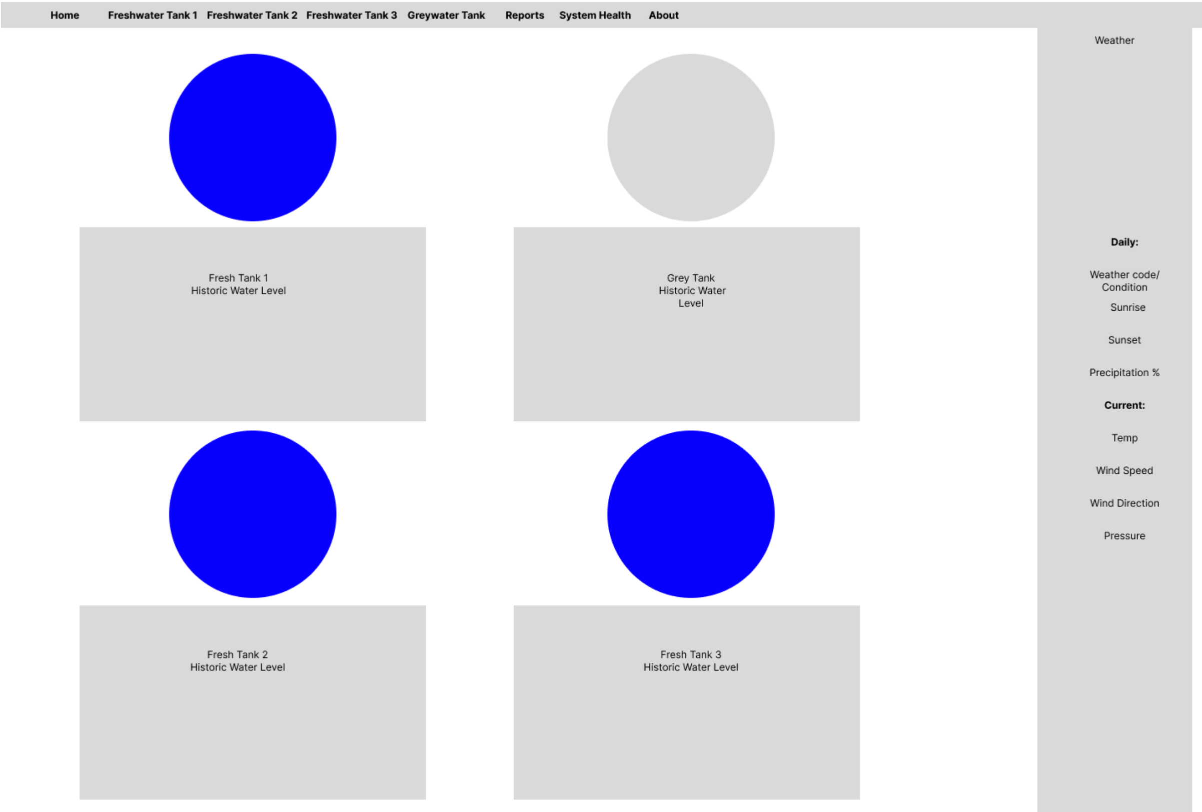

Figure 2.21: Home Page

2.3.4 Home Page Water Level Flow Diagram

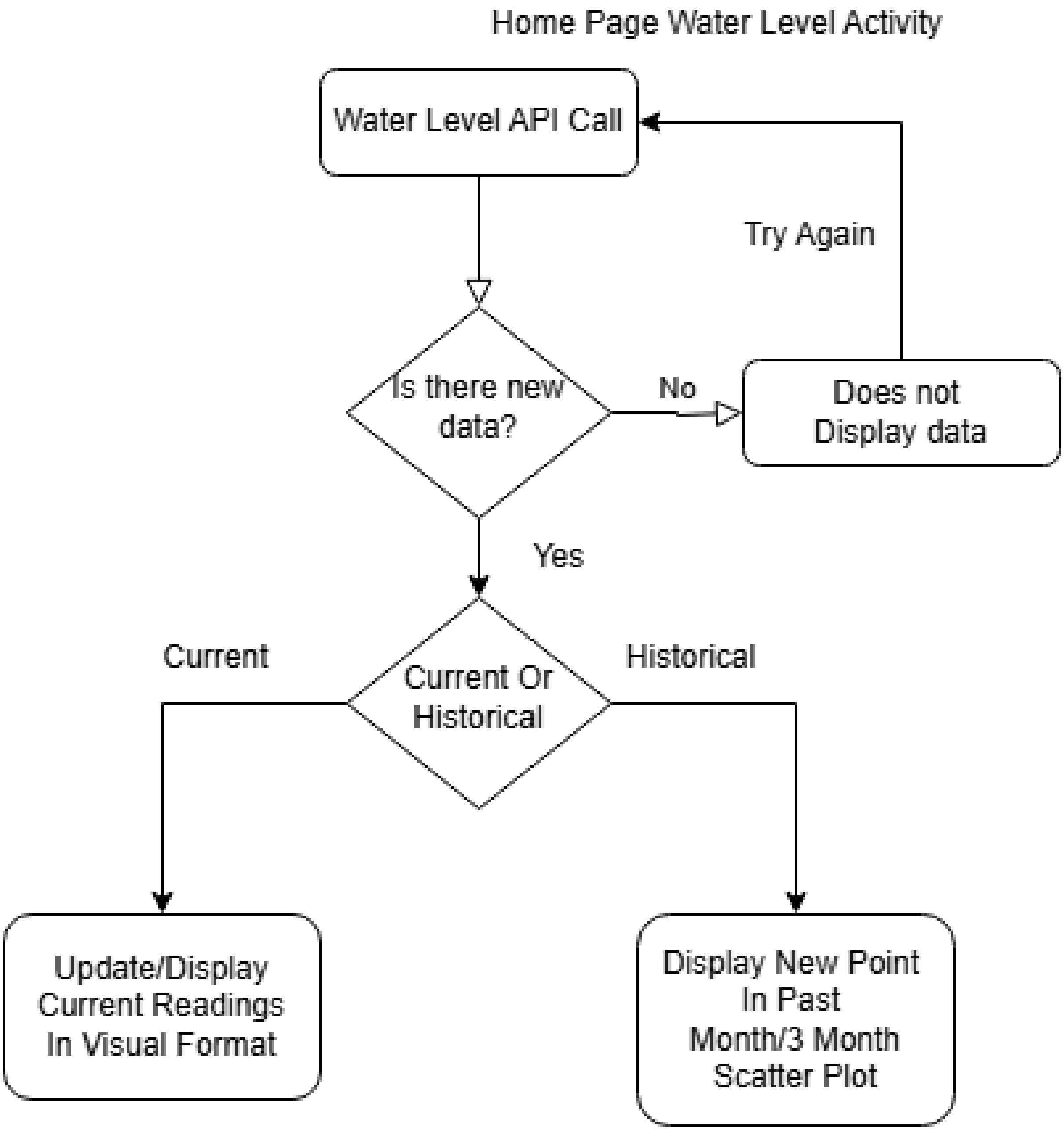

Figure 2.22: Home Page Water Level Flow Diagram

2.3.5 Flow Chart for Tank Pages

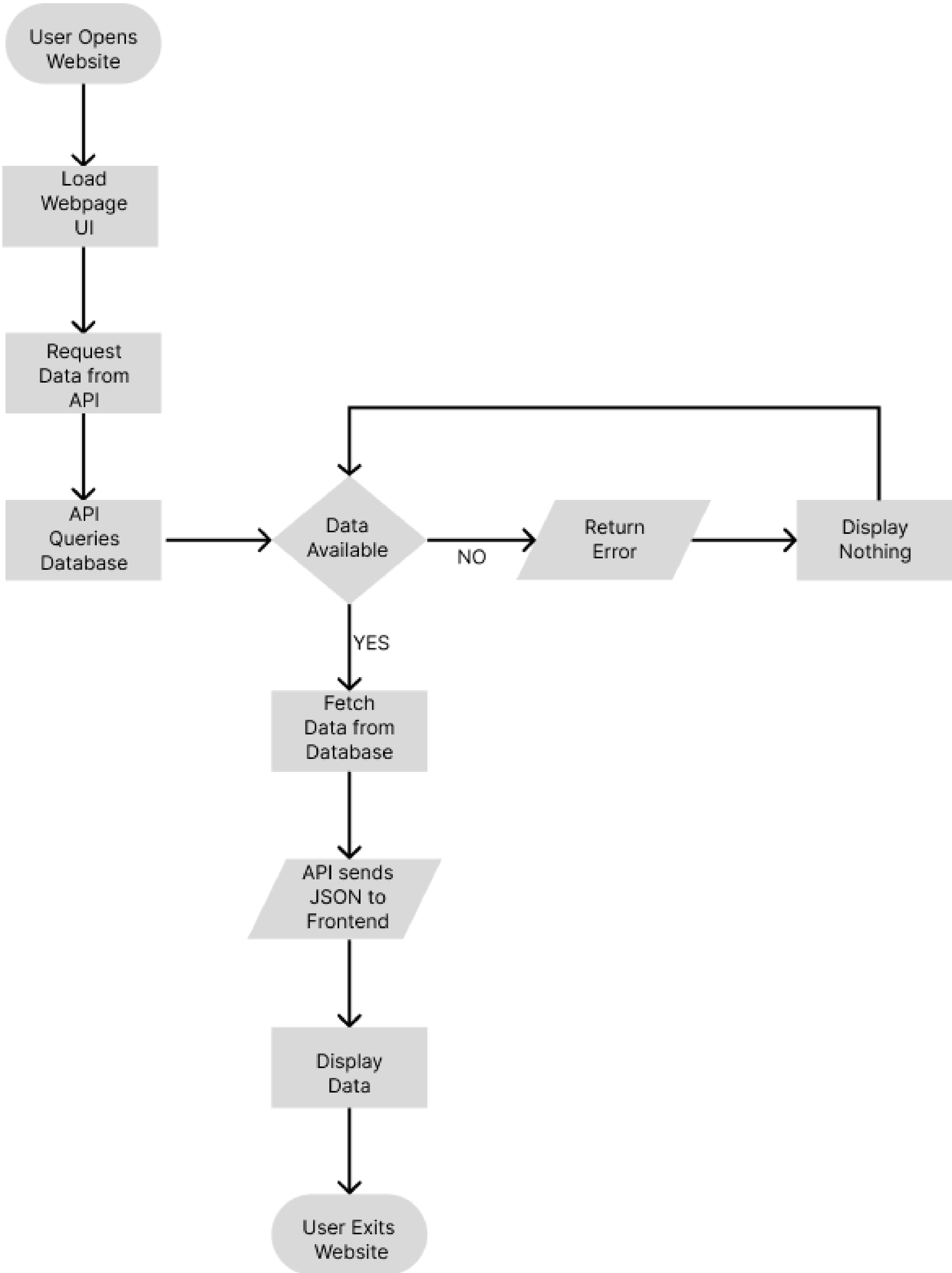

Figure 2.23: Flow Diagram For Tank Pages

2.3.6 Freshwater Tank 1 Page

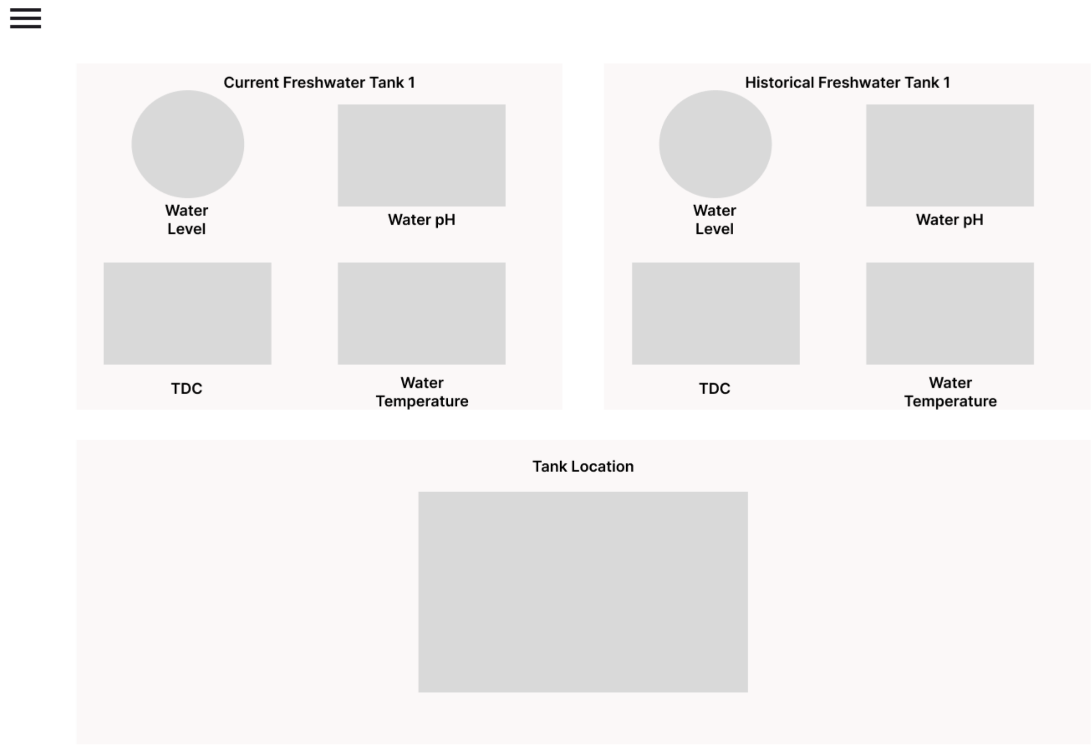

Figure 2.24: Freshwater Tank 1 Wireframe

2.3.7 Freshwater Tank 2 Page

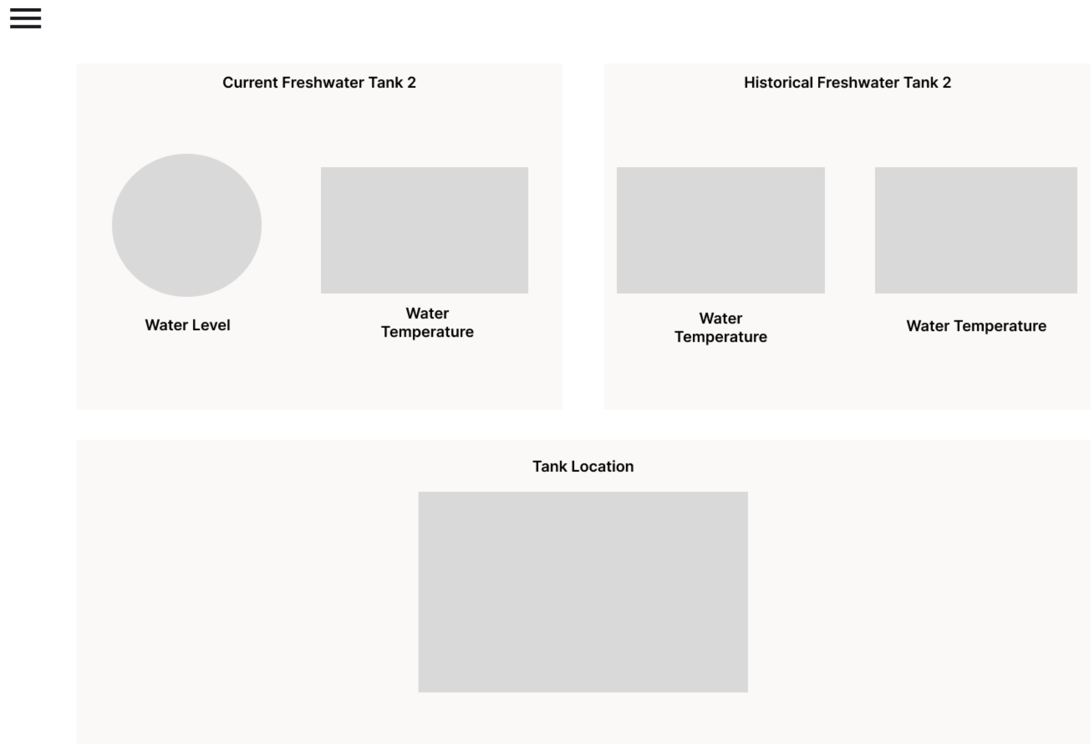

Figure 2.25: Freshwater Tank 2 Wireframe

2.3.8 Freshwater Tank 3 Page

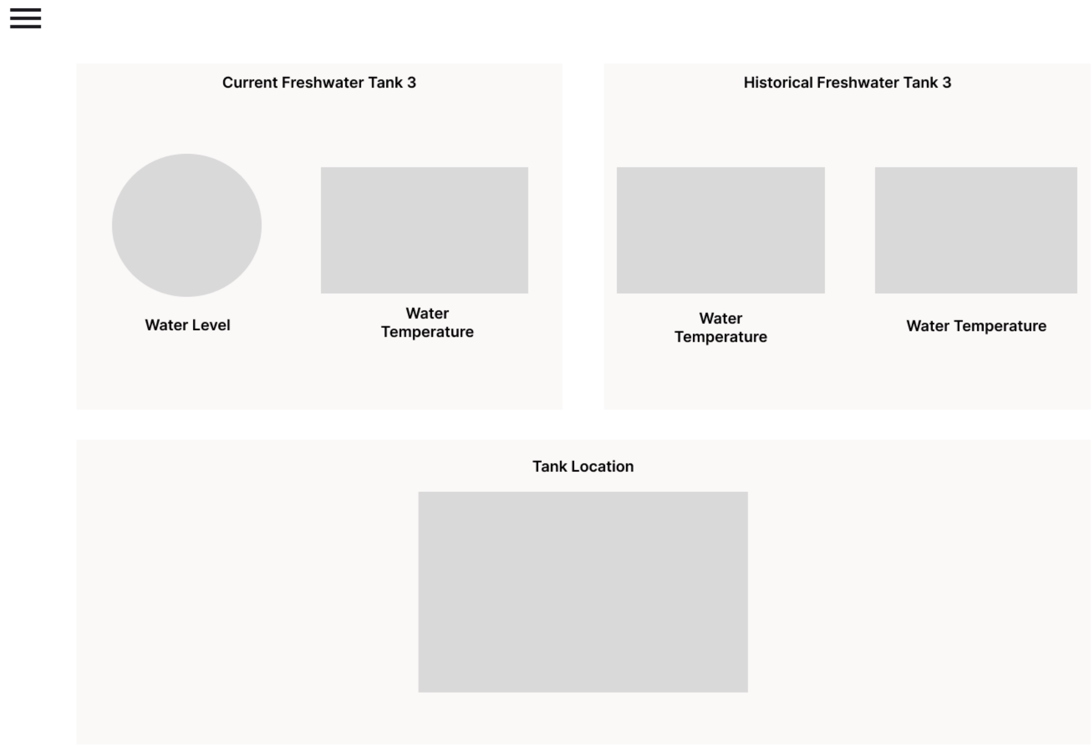

Figure 2.26: Freshwater Tank 3 Wireframe

2.3.9 Greywater Tank Page

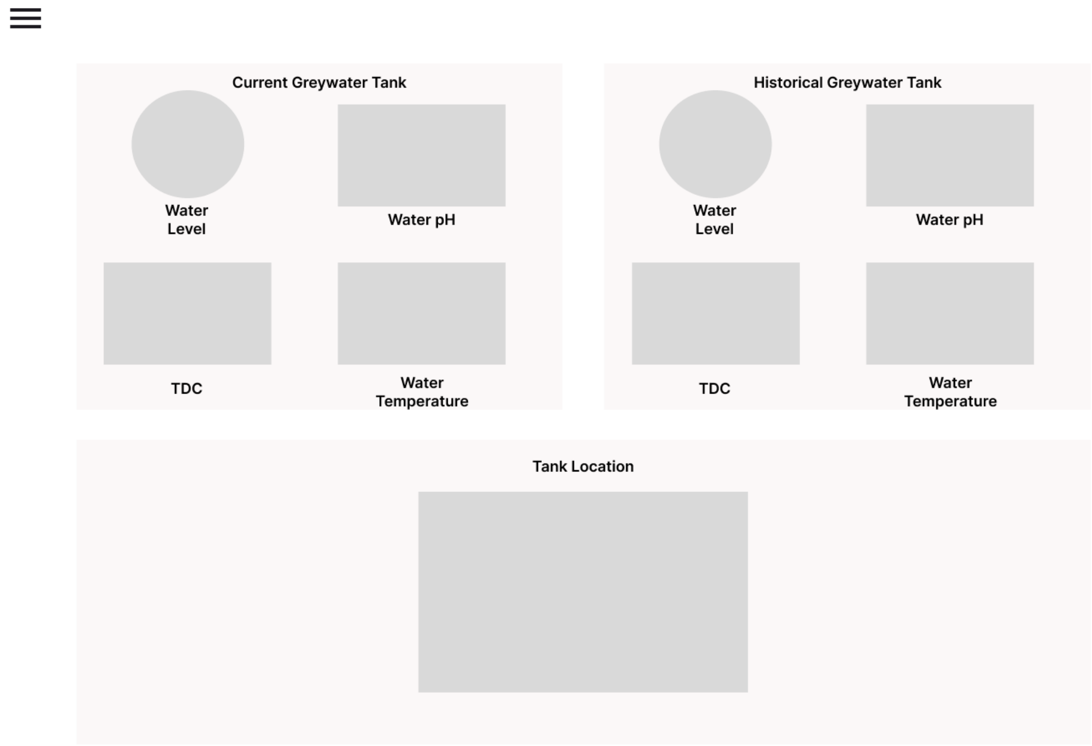

Figure 2.27: Greywater Tank Wireframe

2.3.10 Weather Conditions Page

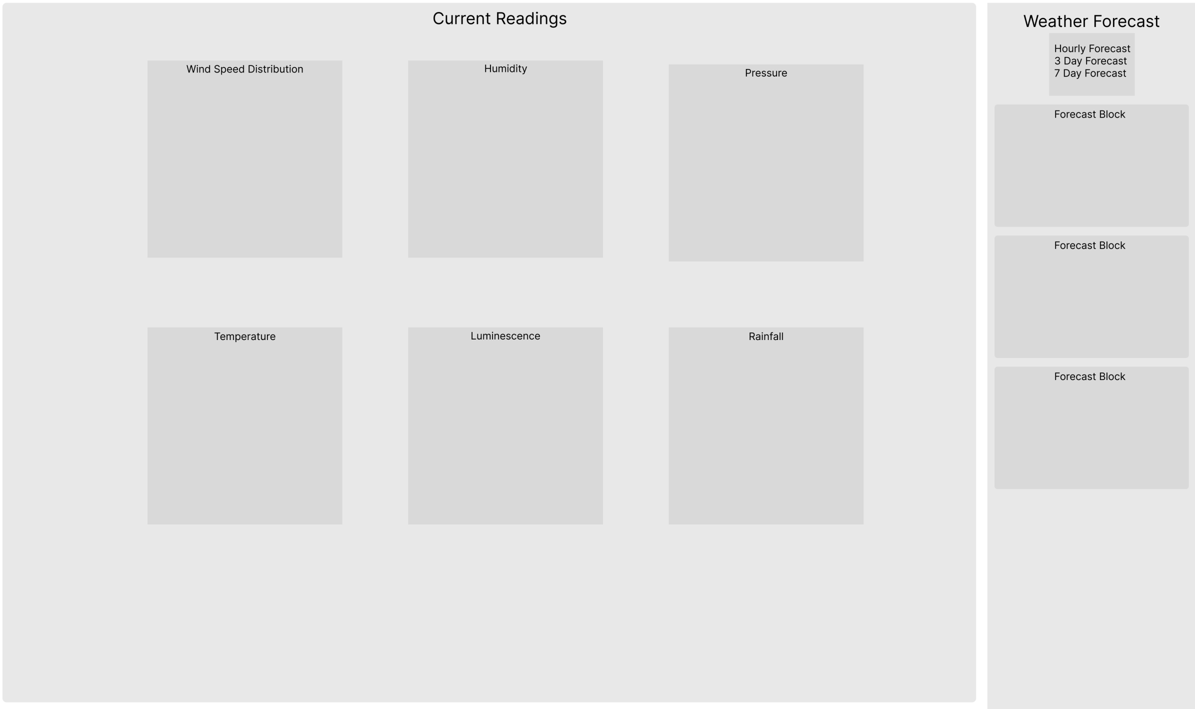

Figure 2.28: Weather Condition Page Wire Frame

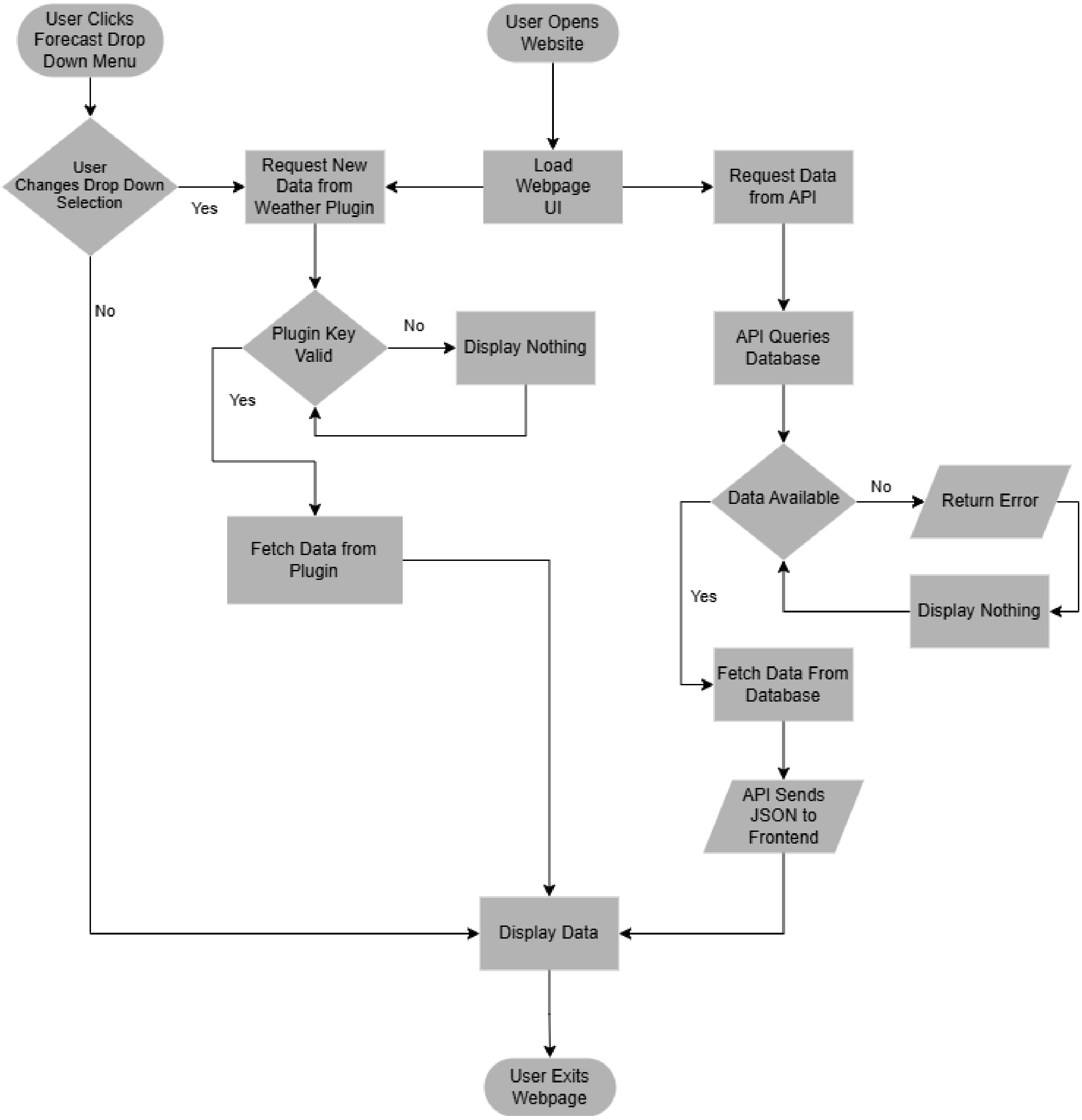

Figure 2.29: Weather Condition Page Flow Chart

2.3.11 Reports Page Flow Chart

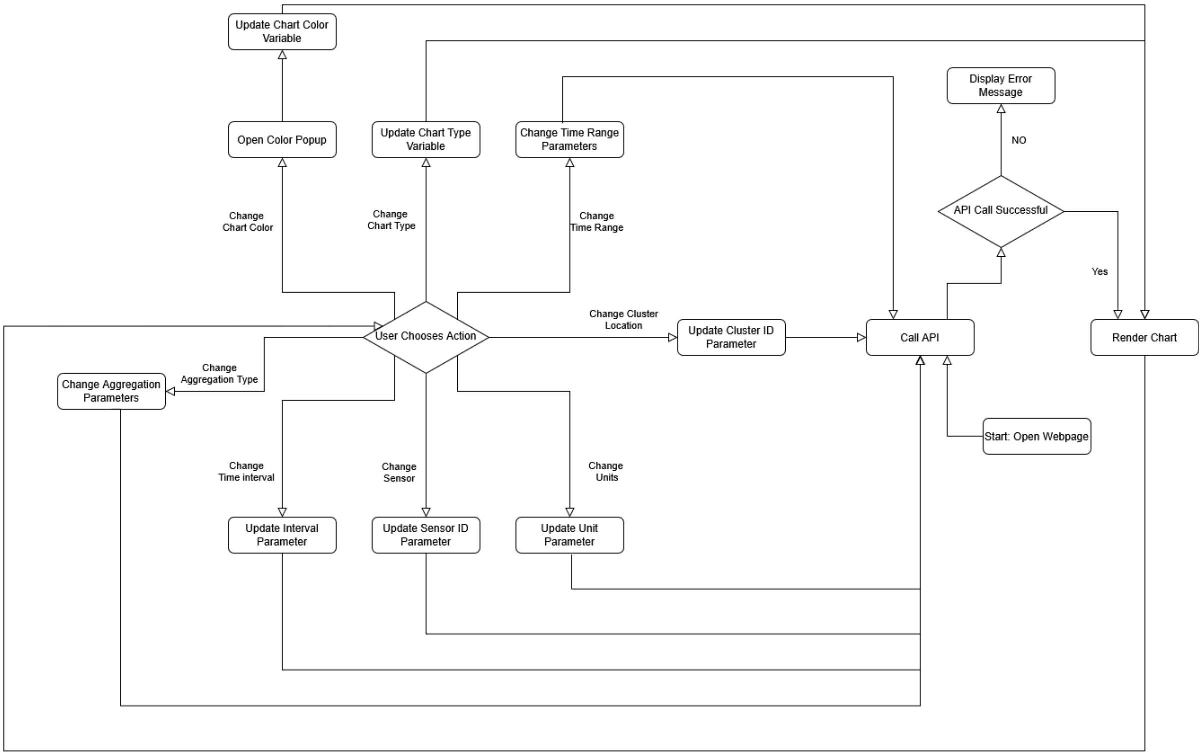

Figure 2.30: Reports Page Flow Chart

2.3.12 Reports Page

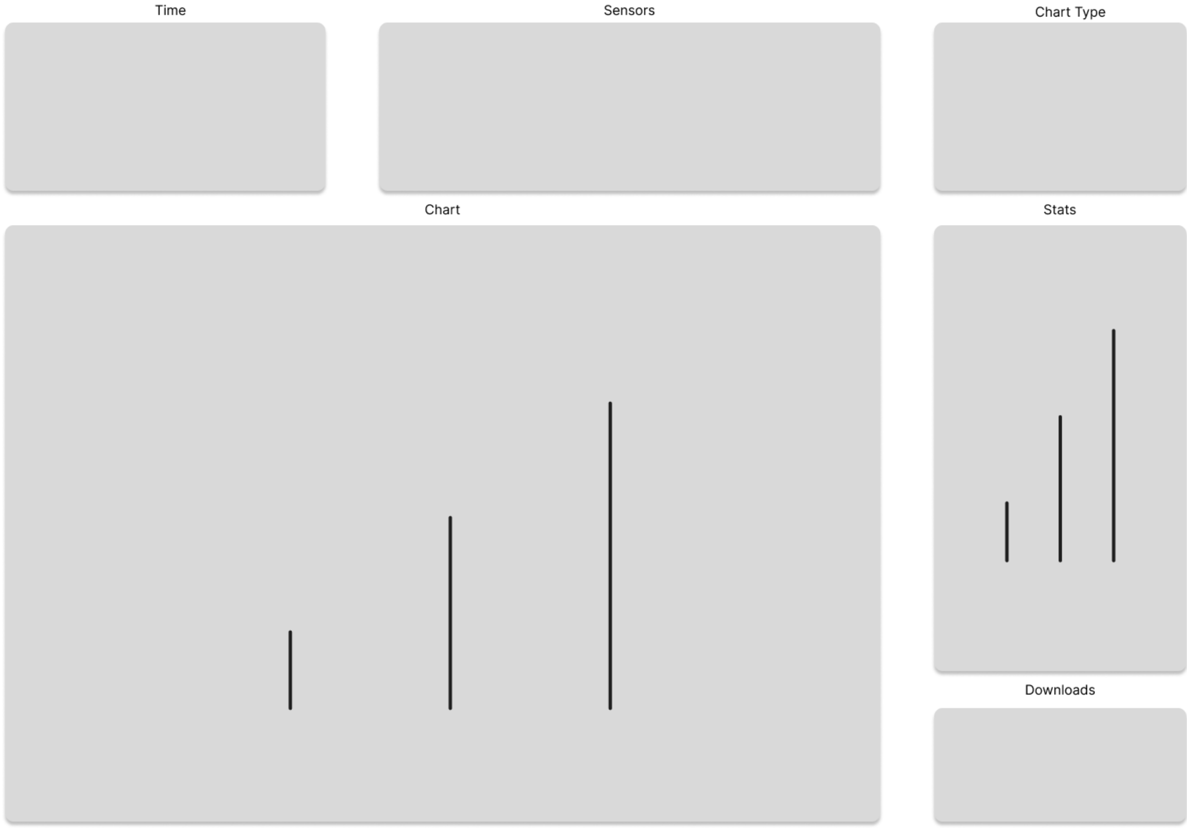

Figure 2.31: Reports Page

2.3.13 System Health Page Flow Chart

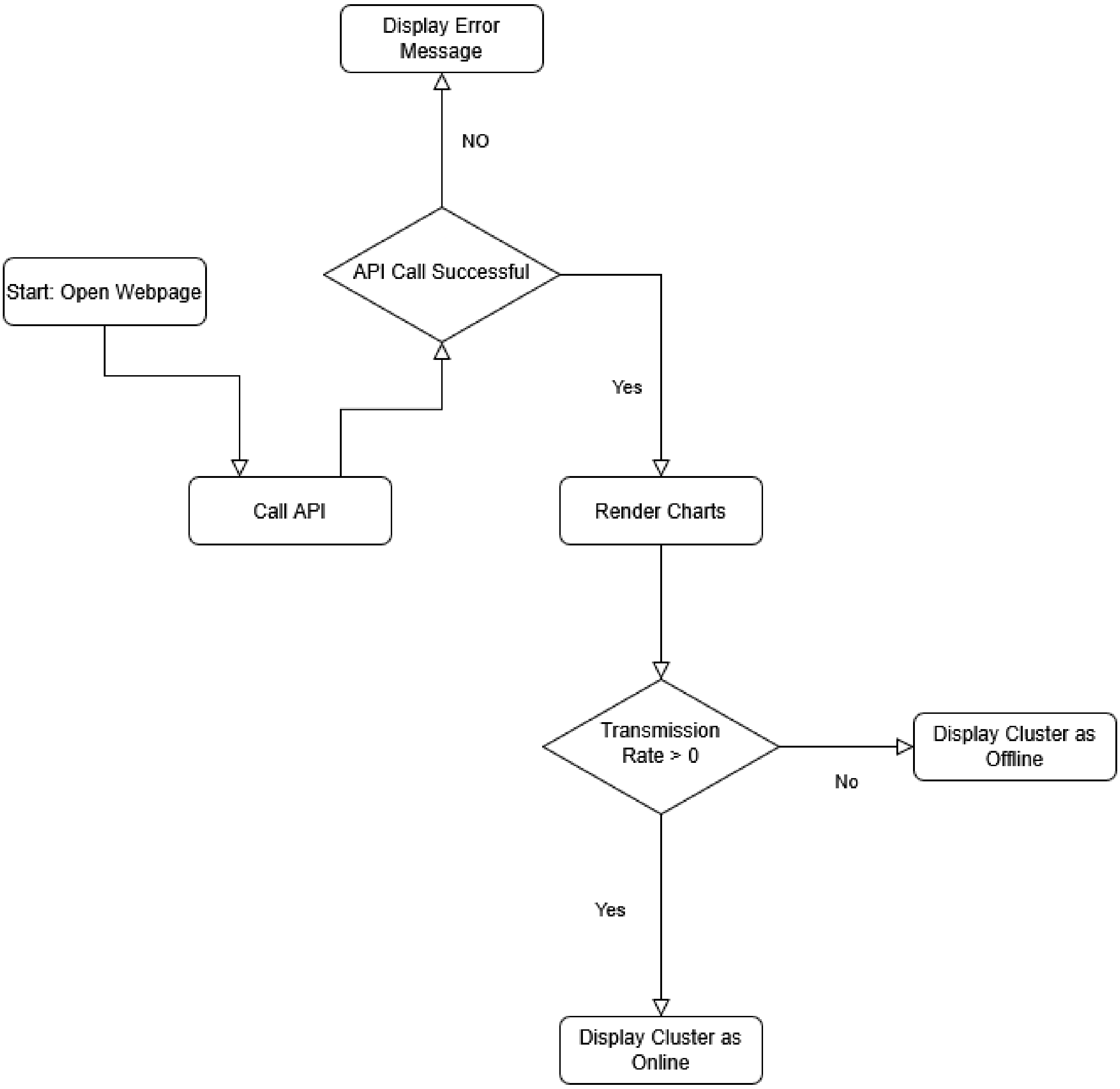

Figure 2.32: System Health Page Flow Chart

2.3.14 System Health Page

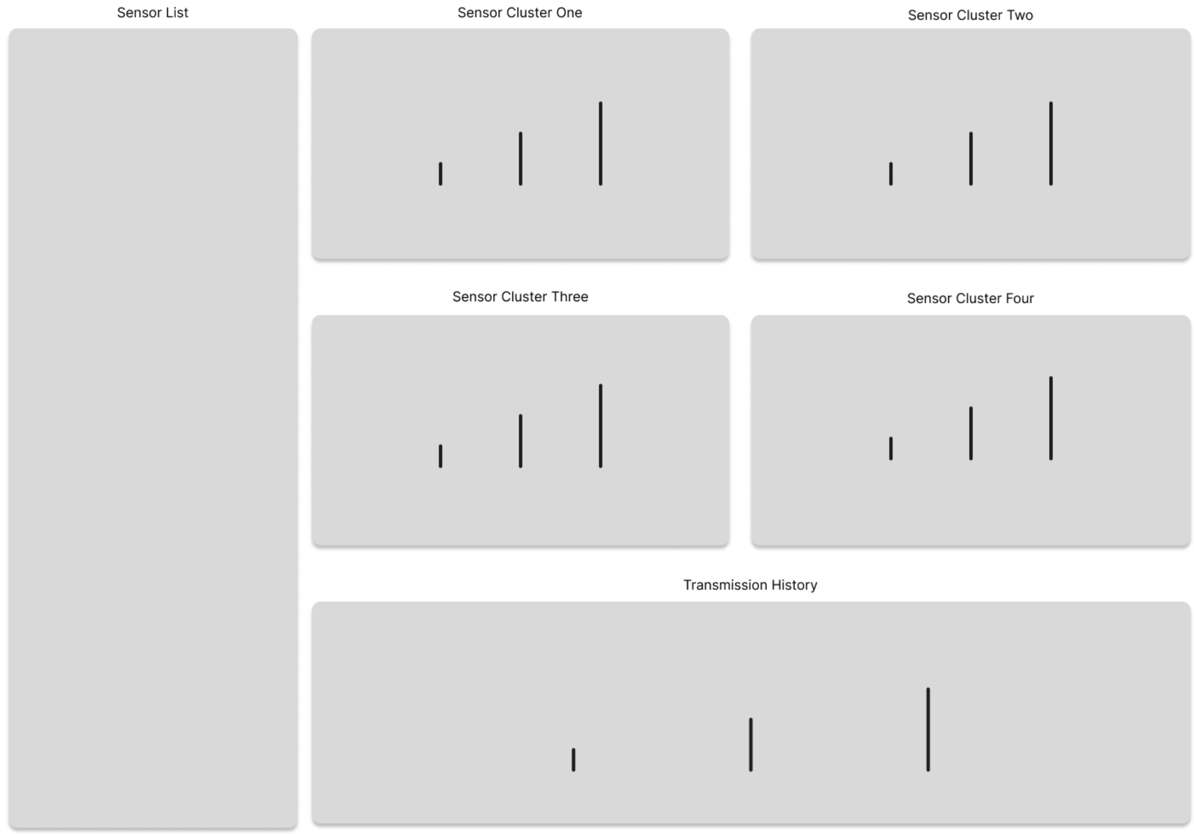

Figure 2.33: System Health Page

2.3.15 Updated Component Diagram

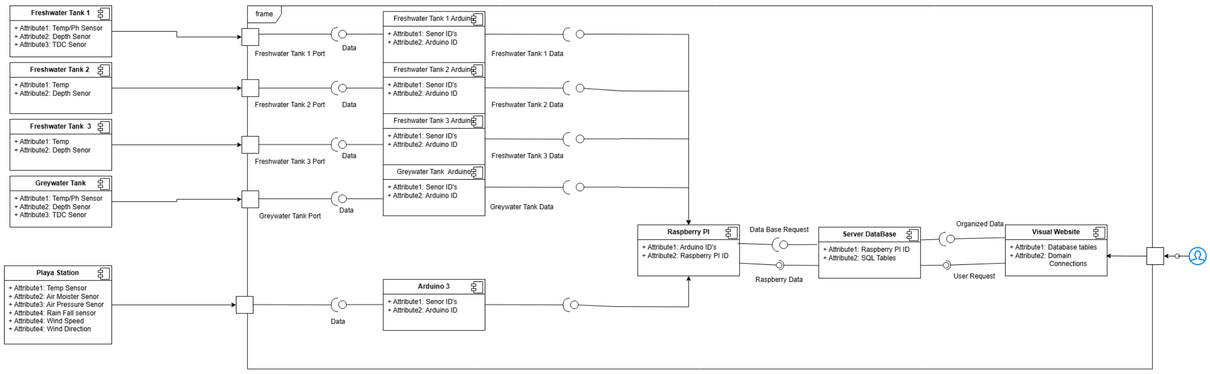

Figure 2.34: Updated Component Diagram

2.3.16 Open Source Weather API Flow Diagram

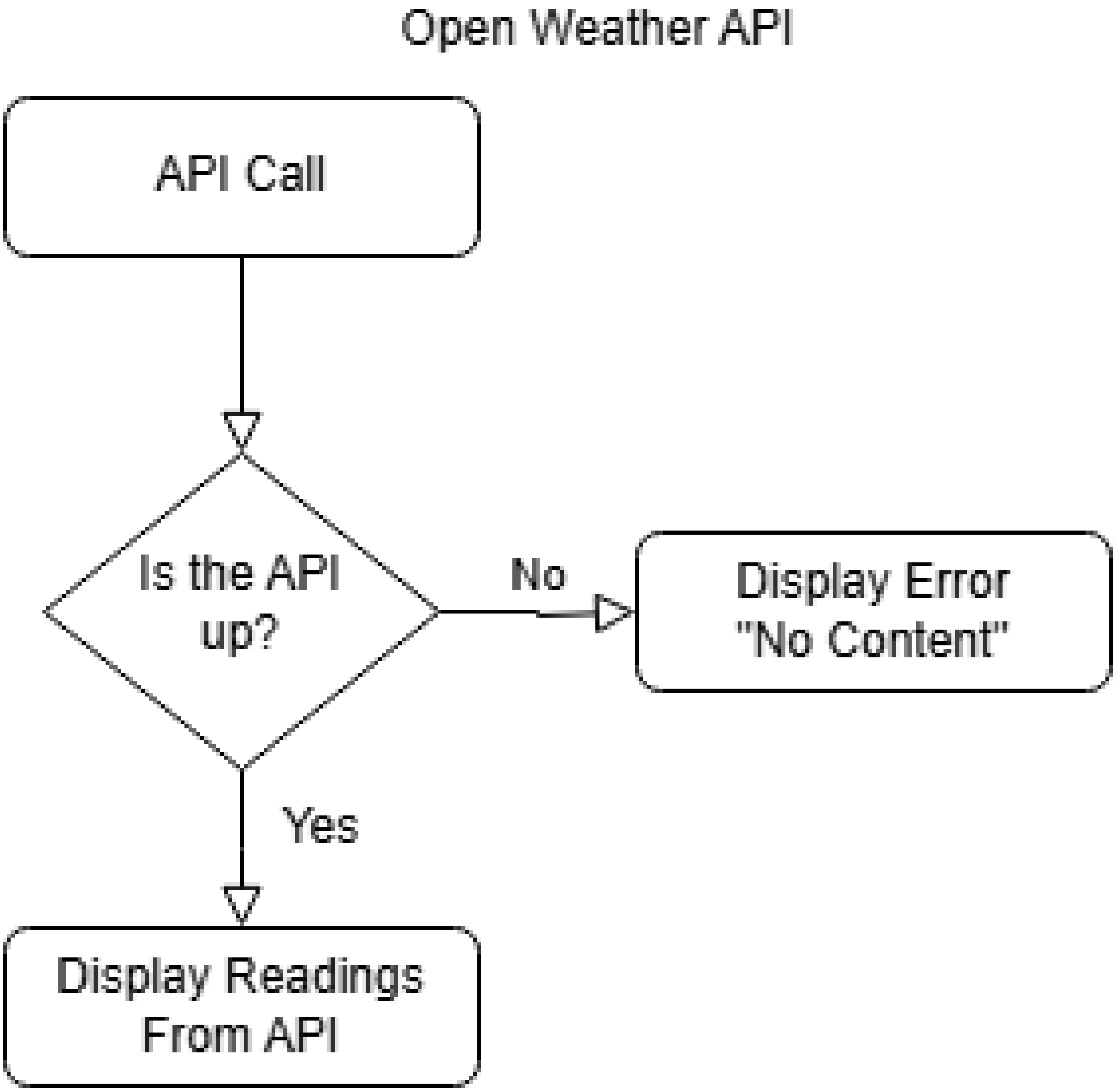

Figure 2.35: Open Source Weather API Flow Diagram

2.3.17 API Flow Diagram

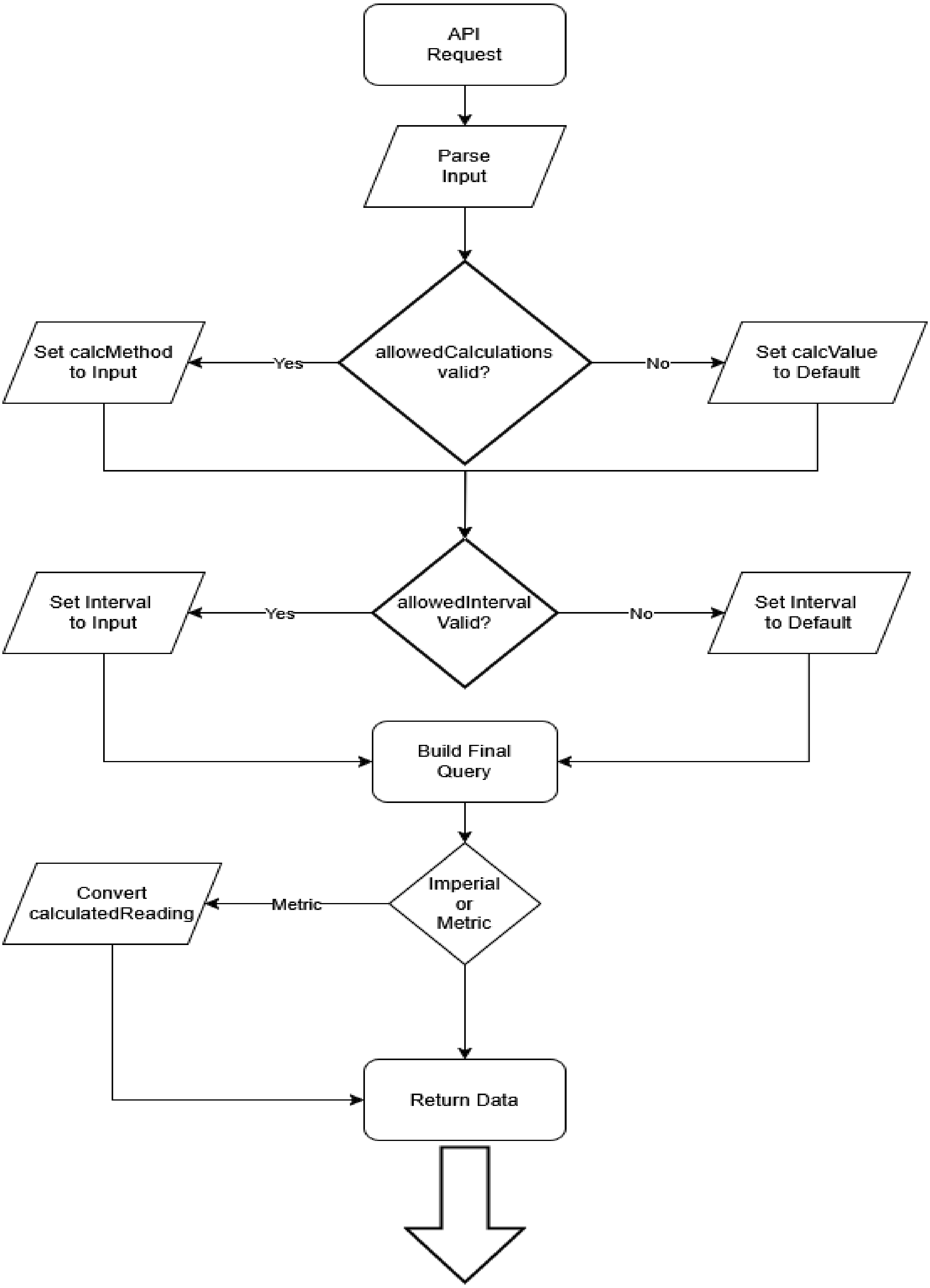

Figure 2.36: API Flow Diagram

2.3.18 Schematic Version 2

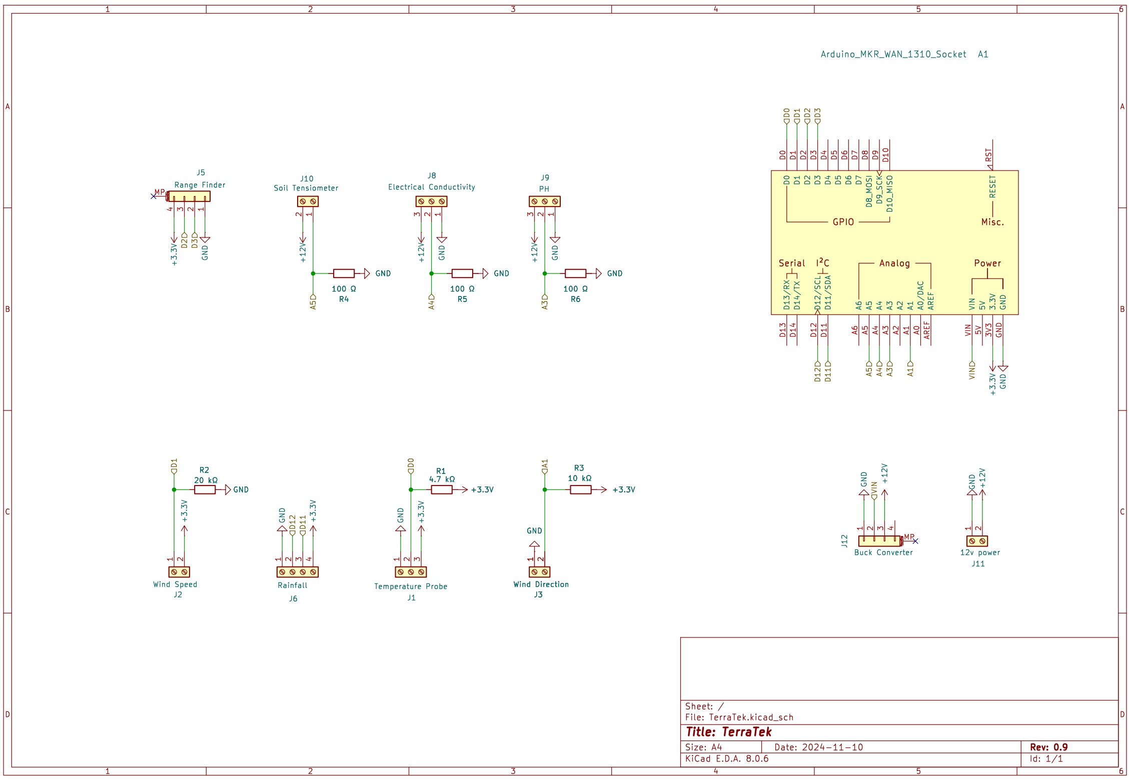

Figure 2.37: Schematic Version 2

Chapter 3

Software Implementation and Testing

3.1 Test Cases

3.2 Semester 1 Test Cases:

3.2.1 Test Case 1: Arduino Properly Transmits Data to LoRaWAN Gateway

Purpose:	
The purpose of this test case is to ensure that the Arduino can properly transmit data over the LoRaWAN connection to the gateway.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
William Kolath 12/04/2024	In order for this test to be performed, the gateway must be running. The gateway must be configured to receive on the US915_1 band. The gateway must be registered with the local Chirp Stack application server.
Notes:	
Test Case PASS	

Table 3.2: Test Case 1 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Run the 'FirstConfiguration.ino' example sketch on the Arduino	Arduino outputs device EUI to serial monitor	Arduino outputs device EUI to serial monitor	N/A	PASS
2	On the Chirp Stack web interface create a new device named "test" using the MKRWAN 1310 template.	New device named "test" has been created.	New device "test" displayed on TerraTek application page.	N/A	PASS
3	On the device configuration page enter the device EUI from step one.	The "test" device EUI matches the output on the serial monitor.	Device EUI correctly inputted.	N/A	PASS
4	On the device configuration page enter "0000000000000000" for the APP EUI and generate a random number for the APP KEY.	The "test" device has "0000000000000000" for the APP EUI and a random hex number for the APP KEY.	APP EUI entered and APP EUI generated correctly.	N/A	PASS
5	On the serial monitor enter 1 to enter OTAA Join.	After entering 1, the Arduino will ask for the APP EUI.	Arduino requests the APP EUI.	N/A	PASS
6	On the serial monitor enter "0000000000000000" for the APP EUI.	After entering the APP EUI, the Arduino will ask for the APP KEY.	Arduino requests the APP KEY.	N/A	PASS
7	On the serial monitor enter the APP KEY generated in step 4.	The Arduino will join the LoRaWAN network and print "Message sent correctly!" to indicate that the device setup was successful.	After a short delay the message "Message sent correctly!" is printed to the serial monitor.	N/A	PASS

Table 3.4: Arduino Properly Transmits Data to LoRaWAN Gateway

3.2.2 Test Case 2: Raspberry Pi is Able to Request Data from LoRaWAN Application Server

Purpose:	
The purpose of this test is to ensure that the Raspberry Pi hub is able to properly request data from the LoRaWAN Application server.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
William Kolath 12/04/2024	Chirp Stack must be installed and running on the gateway. At least one device must be connected and configures to transmit data in order to test data transfer.
Notes:	
Test Case PASS	

Table 3.6: Test Case 2 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	In the ChipStack web interface configure the http integration to send binary data to http://localhost:8090	configuration successful	successful configuration	N/A	PASS
2	Run the "ChipStack-DataCollector.py" python script on the Raspberry Pi.	The most recent data sent to the LoRaWAN gateway will be printed to the terminal periodically.	The most recent data sent to the LoRaWAN gateway will be printed to the terminal periodically.	N/A	PASS

Table 3.8: Raspberry Pi is Able to Request Data from LoRaWAN Application Server

3.2.3 Test Case 3: Raspberry Pi Properly Sends Data to Database

Purpose:	
The purpose of this test is to ensure that the Raspberry Pi hub is able to insert data in to the database.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
William Kolath 12/04/2024	The server must be running and port 3306 must be forwarded. The database must be running and the tables must match the ER Diagram.
Notes:	
Test Case PASS	

Table 3.9: Test Case 3 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Vali- dated	Pass /Fail
1	Enter the database IP address, username, password, and database name in the ”config.ini” file on the Rasspberry Pi.	The ”config.ini” properly reflects database configuration.	Data entered correctly in ”config.ini”	N/A	PASS
2	Run the ”Data$_C$*ollector.py”pythonscript*	A message will be displayed on the terminal periodically indicating that data has been inserted in the database.	Terminal Messages appear every 20 Seconds indicating that data has been inserted in database.	N/A	PASS
3	On MySQL workbench, run ”DisplayTables.sql”	The output shows the data that the raspberry pi inserted in step 2.	Readings table updated with new data from the end node every 20 seconds.	REQ-3.1	PASS

Table 3.11: Raspberry Pi Properly Sends Data to Database

3.2.4 Test Case 4: Arduino Handles Invalid Sensor Values from Ultrasonic Sensor

Purpose:	
The purpose of this test is to ensure that the Arduino knows how to handle the invalid results from the ultrasonic sensor ranges	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector Ceballos 12/05/2024	The sensor must be properly wired up to the ENV SHIELD which is on the MKR WAN board. The probe must be free of any obstacles to get proper readings.
Notes:	

Table 3.12: Test Case 4 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Sensor must be powered up and running.	The lights on the boards should be on	Lights Function when power is added.	REQ-1.1	Pass
2	As sensor reads data and outputs data, check if values are properly being read.	Values ranging from 21 to 600 should be outputted	Values are corrctly read.	REQ-1.1	Pass
3	If range reads at a constant 600 or below 21, no matter what , there is an error	If true, then proceed to error handling	Error message is printed and sent to database.	REQ-1.1	Pass
4	Output error handler for the ultrasonic sensor.	Output error codes for the ultrasonic sensor	Error message is printed and sent to database.	REQ-1.1	Pass

Table 3.14: Arduino Handles Invalid Sensor Values from Ultrasonic Sensor

3.2.5 Test Case 5: Arduino Handles Invalid Sensor Values from Temperature Probe

Purpose:	
The purpose of this test is to ensure that the Arduino knows how to handle the invalid results from the temperature probe sensor	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector Ceballos 12/05/2024	The sensor must be properly wired up to the ENV SHIELD which is on the MKR WAN board. The probe must be free of any obstacles to get proper readings.
Notes:	

Table 3.15: Test Case 5 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Sensor must be powered up and running	The lights on the boards should be on	Board powers on	YES	PASS
2	As sensor reads data and outputs data, check if values are properly being read	Values ranging from -125 to 196 should be outputted	Values are properly being read	REQ 1.4	PASS
3	If temperatures reading -125 and 196, there is an error in reading the temperature	If true, then proceed to error handling	Read -125 and went to error handling	REQ 1.4	PASS
4	Output error handler for miscalculated temperatures	Output error codes for the temperature probe	Error is outputted	REQ 1.4	PASS

Table 3.17: Arduino Handles Invalid Sensor Values from Temperature Probe

3.2.6 Test Case 6: Arduino Handles Invalid Sensor Values from Electrical Conductivity Sensor

Purpose:	
The purpose of this test case is to verify that the Arduino can identify and handle invalid or extreme values from the Electrical Conductivity Sensor	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Kirk Blood 12/05/2024	The Sensor must be properly wired up to the ENV Shield and PCB. The PCB must have an external EEProm chip installed. Sensor must be submerged in the target liquid.
Notes:	

Table 3.18: Test Case 6 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Disconnect the Electrical Conductivity sensor or send out of range data to simulate an error.	Arduino recognizes invalid data and outputs an error message on the monitor	Couldn't get to work with final board layout.	REQ-1.3	Fail
2	Verify Arduino catches the invalid data and outputs the proper error code.	The Arduino outputs a invalid data error code and blocks the transmission of bad data.	Couldn't get to work with final board.	REQ-1.3	Fail
3	Reconnect the sensor and check calibration	Sensor Powers on and either logs proper data or needs calibration	Couldn't get to work with final board layout.	REQ-1.3	Fail
4	Calibrate the sensor if needed, using proper calibration solutions.	Sensor transmits proper data	Couldn't get to work with final board.	REQ-1.3	Fail

Table 3.20: EC Sensor Test Case

3.2.7 Test Case 7: Arduino Handles Invalid Sensor Values from Ph Sensor

Purpose:	
The purpose of this test case is to verify that the Arduino can identify and handle invalid or extreme values from the PH Sensor	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Kirk Blood 12/05/2024	The Sensor must be properly wired up to the ENV Shield and PCB. The PCB must have an external EEProm chip installed. Sensor must be submerged in the target liquid.
Notes:	

Table 3.21: Test Case 7 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Disconnect the PH sensor or send out of range data to simulate an error.	Arduino recognizes invalid data and outputs an error message on the monitor	Proper recognition of error and prints in serial monitor.	REQ-1.2	Fail
2	Verify Arduino catches the invalid data and outputs the proper error code.	The Arduino outputs a invalid data error code and sends to database with error code attached.	Proper Error Handling and transmission of data with error code.	REQ-1.2	Fail
3	Reconnect the sensor and check calibration	Sensor Powers on and either logs proper data or needs calibration	Calibration was saved successfully.	REQ-1.2	Pass
4	Calibrate the sensor if needed, using proper calibration solutions.	Sensor transmits proper data	Successful Calibration.	REQ-1.2	Pass

Table 3.23: PH Sensor Test Case

3.2.8 Test Case 8: Arduino Handles Invalid Sensor Values from Rainfall Sensor

Purpose:	
The purpose of this test case is to verify that the Arduino can identify and handle invalid or extreme values from the rainfall sensor.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Preston DeShazo 12/05/2024	The rainfall sensor must be connected to the Arduino, and the Arduino should be programmed to read values from this sensor.
Notes:	

Table 3.24: Test Case 8 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Disconnect the rainfall sensor or send an out-of-range voltage to simulate an error.	Arduino recognizes invalid data and outputs an error message on the serial monitor.	Error messgae was printed out with the rainfall disconnected after initalizing.	REQ-7.2	Pass
2	Observe if the Arduino flags the error in the system, showing incorrect data with error.	Arduino sends the error, and data is sent to the LoRaWAN gateway with error code.	Readings sent with error message and stored correclty	REQ-7.2	Pass
3	Reconnect the rainfall sensor with valid input values and verify normal operation.	Arduino resumes normal data reading and transmits correct data.	Functions normal after connection is true.	REQ-2.2	Pass

Table 3.26: Arduino Handles Invalid Sensor Values from Rainfall Sensor

3.2.9 Test Case 9: Arduino Handles Invalid Sensor Values from Wind Direction Sensor

Purpose:	
The purpose of this test case is to verify that the Arduino can identify and handle invalid or extreme values from the Wind Direction sensor.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Preston DeShazo 12/05/2024	The wind direction sensor must be connected to the Arduino, and the Arduino should be configured to read values from this sensor.
Notes:	

Table 3.27: Test Case 9 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Disconnect the wind direction sensor or provide an out-of-range signal.	Arduino detects invalid data and outputs an error message on the serial monitor.	No way of showing if data is being recorded accurately.	REQ-2.6	Fail
2	Observe if the Arduino flags the error in the system, showing incorrect data with error.	Arduino sends the error, and data is sent to the LoRaWAN gateway with error code.	No way of showing if data is being recorded accurately.	REQ-2.6	Fail
3	Reconnect the sensor and provide valid data to check system recovery.	Arduino resumes normal data reading and transmits correct data.	Send Results normally.	REQ-2.6	Pass

Table 3.29: Arduino Handles Invalid Sensor Values from Wind Direction Sensor

3.2.10 Test Case 10: Arduino Handles Invalid Sensor Values from Wind Speed Sensor

Purpose:	
The purpose of this test case is to verify that the Arduino can identify and handle invalid or extreme values from the Wind Speed sensor.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Preston DeShazo 12/05/2024	The wind speed sensor must be connected to the Arduino, and the Arduino should be set up to read and interpret values from this sensor.
Notes:	

Table 3.30: Test Case 10 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Disconnect the wind speed sensor or simulate an error by sending out-of-range values.	Arduino recognizes the error and displays a message on the serial monitor.	No way of showing if data is being recorded accurately.	REQ-2.5	Fail
2	Observe if the Arduino flags the error in the system, showing incorrect data with error.	Arduino sends the error, and data is sent to the LoRaWAN gateway with error code.	No way of showing if data is being recorded accurately.	REQ-2.5	Fail
3	Reconnect the wind speed sensor with valid data and check system restoration.	Arduino resumes correct data logging and transmission with valid sensor input.	Send Results normally.	REQ-2.5	Pass

Table 3.32: Arduino Handles Invalid Sensor Values from Wind Speed Sensor

3.2.11 Test Case 11: Arduino Handles Invalid Sensor Values from Atmospheric Humidity Sensor

Purpose:	
The purpose of this test case is to verify that the Arduino can identify and handle invalid/valid values from the Humidity sensor.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector Ceballos 12/05/2024	The sensor must be properly wired up to the ENV SHIELD which is on the MKR WAN board. The probe must be free of any obstacles to get proper readings.
Notes:	

Table 3.33: Test Case 11 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Sensor must be powered up and running	The lights on the boards should be on	Sensor powered on	N/A	PASS
2	As sensor reads data and outputs data, check if values are properly being read	Values ranging from 0 to 100 should be outputted	Values show data properly	N/A	PASS
4	Output error handler for the humidity sensor	Output error codes for the humidity sensor	Error is outputted	N/A	PASS

Table 3.35: Arduino Handles Invalid Sensor Values from Atmospheric Humidity Sensor

3.2.12 Test Case 12: Arduino Handles Invalid Sensor Values from Barometric Pressure Sensor

Purpose:	
The purpose of this test case is to verify that the Arduino can identify and handle invalid/valid values from the barometric sensor.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector Ceballos 12/05/2024	The sensor must be properly wired up to the ENV SHIELD which is on the MKR WAN board. The probe must be free of any obstacles to get proper readings.
Notes:	

Table 3.36: Test Case 12 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Sensor must be powered up and running	The lights on the boards should be on	Board powered on	REQ 2.4	PASS
2	As sensor reads data and outputs data, check if values are properly being read	Values ranging from 0kPA - 260kPA should be outputted	Board reads values properly	REQ 2.4	PASS
4	Output error handler for the humidity sensor	Output error codes for the humidity sensor	Error code is outputted	REQ 2.4	PASS

Table 3.38: Arduino Handles Invalid Sensor Values from Atmospheric Humidity Sensor

3.2.13 Test Case 13: Data from sensor failure transmitted to Database

Purpose:	
This test case is to verify the behavior of the database if any sensor stops functioning in the event of a hardware failure	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Dale Andre Descallar 12/04/2024	Raspberry Pi is receiving data from all sensors and is transmitting data to the database
Notes:	
Partial FAIL	

Table 3.40: Test Case 13 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Disconnect Rainfall Sensor	Error Code Returned to Database	Error Code sent to database and a value of -127 Sent to Database	REQ-7.2	PASS
2	Disconnect wind speed Sensor	Error Code Returned to Database	No Error Code, 0 Returned to Readings	Fail	Fail
3	Disconnect Wind Direction Sensor	Error Code Returned to Database	No Error Code, last known direction sent to database	Fail	Fail
5	Disconnect Shield	Error Code Returned to Database	Arduino stops transmitting data to database	Fail	Fail
7	Disconnect Temperature Sensor	Error Code Returned to Database	No Error Code, -127 Sent to Database	REQ-7.2	PASS
9	Disconnect Ultrasonic Sensor	Error Code Returned to Database	No Error Code, 0 Returned to Readings	REQ-7.2	PASS

Table 3.42: Data from sensor failure transmitted to Database

3.2.14 Test Case 14: No data is transmitted to Database

Purpose:	
This test case is to validate the behavior of the database in the event that it stops receiving data from raspberry pi in the event of hardware failure from the pi itself.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Dale Andre Descallar 12/04/2024	Raspberry Pi is receiving data from all sensors and is transmitting data to the database
Notes:	
Test Case PASS	

Table 3.44: Test Case 14 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Power off Raspberry Pi	Database will stop storing data from that moment	Database stops receiving data	REQ-7.2	pass
2	Wait and power on raspberry pi	Database will resume data insertion at the moment the raspberry pi resumes communication	Database receives data from most recent reading	REQ-7.2	pass

Table 3.46: No data is transmitted to Database

3.2.15 Test Case 15: RESTFUL API should be running constantly to show current data

Purpose:	
The purpose of this test case is to verify that the backend RESTFUL API is running and still able to fetch JSON data from the database.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector Ceballos 12/05/2024	The API should be set up locally on the server
Notes:	

Table 3.47: Test Case 7 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Search https://terratekrwh.com	User is greeted by Hello World text	User is greeted by Hello World text	YES	PASS
2	Search https://terratekrwh.com /boards-data	User should be shown JSON of all boards being used	JSON for boards is shown	YES	PASS
3	Search https://terratekrwh.com /sensors-data	User should be shown JSON of all sensors being used	JSON for sensors is shown	YES	PASS
4	Search https://terratekrwh.com /readings-data	User should be shown JSON data of all readings from sensors	JSON for readings is shown	YES	PASS
5	User will visit site later and type the same URL's	User should be shown data after any amount of time	JSON stops showing less than 24 hours after last visit	NO	FAIL
6	Multiple users should visit site	Site should be rendered as usual			

Table 3.49: API is/is not running constantly

3.3 Semester 2 Test Cases:

3.3.1 Test Case 1: System Health Page Displays Online/Offline Status Correctly:

Purpose:	
The purpose of this test is to determine if the System Health page properly displays the status of sensor clusters.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
William Kolath / 05/01/2025	Server is powered on and conected to the internet. LoRaWAN network is functioning correctly.
Notes:	
Test Case: PASS	

Table 3.51: Test Case 1 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Vali- dated	Pass /Fail
1	Power on Arduino.	Adruino power LED turns on.	Adruino power LED turns on.	N/A	PASS
2	Wait between 5 to 10 minutes for Arduino to connect.	Adruino function LED flashes 3 times rapidly every aproximetly 15 sec indicating that data is being sent.	Adruino function LED flashes 3 times rapidly every aproximetly 15 sec indicating that data is being sent.	N/A	PASS
3	Check the system health page	The appropriate cluster should have an ”ON- LINE” status.	The appropriate cluster should have an ”ONLINE” status.	REQ-6.5:	PASS
4	Power off the Arduino.	Adruino power LED turns off.	Adruino power LED turns off.	N/A	PASS
5	Wait a minimum of 5 minutes for the status to update.	Nothing.	Nothing.	N/A	PASS
6	Check the system health page	The appropriate cluster should have an ”OF- FLINE” status.	The appropriate cluster should have an ”OFFLINE” status.	REQ-6.5:	PASS

Table 3.53

3.3.2 Test Case 2: System Health Page Displays Sensor Outages:

Purpose:	
The purpose of this test is to determine if the System Health page accurately displays sensor outages.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
William Kolath / 05/01/2025	Server is powered on and connected to the internet. LoRaWAN network is functioning correctly. At least one sensor cluster powered on and coneted to the network.
Notes:	
Test Case: PASS	

Table 3.55: Test Case 2 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Check the sensor list with everything powered on and functioning correctly	The sensor list should display all connected senors and indicated that they are functioning normally.	The sensor list should display all connected senors and indicated that they are functioning normally.	REQ-6.1	PASS
2	Power off the Arduino and wait at least 5 minutes.	The Arduino powers off.	The Arduino powers off.	N/A	PASS
3	Check the sensor list.	The sensor list indicates that all of the connected sensors are offline.	The sensor list indicates that all of the connected sensors are offline.	REQ-6.1	PASS
4	Power on the Arduino and wait at least 5 minutes.	The Arduino powers on and connects to the LoRaWAN network.	The Arduino powers on and connects to the LoRaWAN network.	N/A	PASS
5	Check the sensor list.	The sensor list indicates that all of the connected sensors are functioning normal.	The sensor list indicates that all of the connected sensors are functioning normal.	REQ-6.1	PASS
6	Disconnect the temperature probe.	The Arduino transmits error codes for the disconnected sensor.	The Arduino transmits error codes for the disconnected sensor.	N/A	PASS
7	wait at least 5 minutes and check the sensor list.	The sensor list indicates that all of the connected sensors are functioning normally except for the disconnected sensor.	The sensor list indicates that all of the connected sensors are functioning normally except for the disconnected sensor.	N/A	PASS
8	Repeat steps 6 and 7 for each of the remaining sensors.	The sensor list accurately indicates that the sensors are offline as they are disconnected.	The sensor list accurately indicates that the sensors are offline as they are disconnected.	REQ-6.1	PASS

Table 3.57

3.3.3 Test Case 3: Multiple Widgets Overlap on Smaller Screens

Purpose:	
The purpose of this test case is to determine wether or not the homepage correctly shows on mobile.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Preston DeShazo 4/10/2025	The web page must be correct on a pc web browser before testing on mobile.
Notes:	
Test Case PASS	

Table 3.59: Test Case 3 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Open the Home Page on various small screen widths.	All widgets (weather, tank status, charts, dropdowns) stack vertically or collapse into an accordion/flex-column layout.	All screen sizes work correctly, Except Ipad.	REQ-1.7	Pass
2	Observe how the layout adjusts — check for: -Weather data card(s) -Tank visual indicators or animations -Historical Graphs.	No widget overlaps another.	No Widgets overlap each other when on a smaller screen	REQ-1.1	Pass
3	Test both portrait and landscape orientations.	Layout auto adjust for the different orientation.	Both Layouts work, but the words in the weather get crammed together	REQ-1.7	Fail
4	Interact with each UI element and scroll the full page.	Animated elements scale properly and stay centered.	All work as intended on a smaller screen	N/A	Pass
5	Rotate the mobile device mid-session and observe layout adaptation.	Correctly adjusted for the sudden change in layout orientation.	Most elements adjust correctly, the weather still shows data but gets crammed together.	REQ-1.7	Fail

Table 3.61

3.3.4 Test Case 4: Cluster Boxes Withstands Outdoor Temperature Fluctuations

Purpose:	
The porpuse of this test case is to determine wether or not the Arduino and sensors can withsatand Temp changes.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Preston DeShazo 4/10/2025	The boxes must be installed at location and have a way to monitor the outside and inside temps of the boxes.
Notes:	
Test Case PASS	

Table 3.63: Test Case 4 Temperature Fluctuations

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Install the Box with all senors in the box with Temperature probes.	Box is installed and working.	Box is installed and transmitting data.	N/A	Pass
2	Power on the PCB and run the full sensor polling and data transmission.	The PCB remains powered and sending data.	The PCB has remained powered and transmitting data.	N/A	Pass
3	Record sensor output and transmission integrity. Monitor at different Temperature.	The output doesn't change for below freezing and above 90 degrees outside.	The box has stayed powered during freezing temperatures and above 140 degrees inside the box.	N/A	Pass

Table 3.65

3.3.5 Test Case 5: Reports Page Time Section

<table>
<tr><td colspan="2">Purpose:</td></tr>
<tr><td colspan="2">The purpose of this test is to ensure that the Reports Page Time Section functions correctly.</td></tr>
<tr><td colspan="2">Test Run Information:</td></tr>
<tr><td>Tester Name / Date:</td><td>Prerequisites / Required Configuration:</td></tr>
<tr><td>William Kolath / 4/10/25</td><td>Server is powered on and connected to the internet. LoRaWAN network is functioning correctly. At least one sensor cluster powered on and connected to the network.</td></tr>
<tr><td colspan="2">Notes:</td></tr>
<tr><td colspan="2">Test Case: PASS</td></tr>
</table>

Table 3.67: Test Case 5 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Open terratekrwh.com/reports	The reports page opens with a default plot displaying hourly data	The reports page opens with a default plot displaying hourly data	N/A	PASS
2	Select time interval: Daily	The data points are displayed daily	The data points are displayed daily	REQ-4.7	PASS
3	Select time interval: All	All data points are displayed	All data points are displayed	REQ-4.7	PASS
4	Hover over the most recent data point.	A popup appears displaying the time and reading.	A popup appears displaying the time and reading.	N/A	PASS
5	Check if the time displayed is correctly.	Time is correct with the time being displayed in central time zone with AM/PM.	Time is correct with the time being displayed in central time zone with AM/PM.	N/A	PASS
6	Select each of the quick time selection buttons.	The chart is updated to display the past day, past week, and past month.	The chart is updated to display the past day, past week, and past month.	REQ-4.2	PASS
7	Select custom start and end dates and times with the date selector.	The chart is updated to display the correct time range.	The chart is updated to display the correct time range.	REQ-4.2	PASS
8	Attempt to select a end time that is earlier than the start time.	The date selector refuses to select invalid ranges.	The chart is updated to display the correct time range.	REQ-4.2	PASS
9	Switch the units to metric	All of the units throughout the page are displayed in the appropriate metric units	All of the units throughout the page are displayed in the appropriate metric units	REQ-4.9:	PASS
10	Switch the units back to imperial	All of the units throughout the page are displayed in the appropriate imperial units	All of the units throughout the page are displayed in the appropriate imperial units	REQ-4.9:	PASS

Table 3.69

3.3.6 Test Case 6: Reports Page Downloads Section

Purpose:	
The purpose of the test is to ensure that the download functionality of the reports page works as expected.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
William Kolath / 05/01/2025	Server is powered on and connected to the internet. LoRaWAN network is functioning correctly. At least one sensor cluster powered on and connected to the network.
Notes:	
Test Case: PASS	

Table 3.71: Test Case 6 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Open terratekrwh.com/reports	The reports page opens with a default plot displaying hourly data	The reports page opens with a default plot displaying hourly data	N/A	PASS
2	Click Download PNG	A PNG of the default chart is downloaded.	A PNG of the default chart is downloaded.	REQ-4.5	PASS
3	Change download format to JPEG and click Click Download JPEG	A JPEG of the default chart is downloaded.	A JPEG of the default chart is downloaded.	REQ-4.5	PASS
4	Change download format to CSV and click Click Download CSV	A CSV of the default chart is downloaded.	A CSV of the default chart is downloaded.	REQ-4.6	PASS
5	Change download format to JSON and click Click Download JSON	A JSON of the default chart is downloaded.	A JSON of the default chart is downloaded.	REQ-4.6	PASS
6	Select multiple clusters and sensors.	Multiple data plots are displayed on the chart.	Multiple data plots are displayed on the chart.	REQ-4.3	PASS
7	Click Download PNG	A PNG of the chart is downloaded including all of the selected sensors.	A PNG of the default chart is downloaded including all of the selected sensors.	REQ-4.5	PASS
8	Change download format to JPEG and click Click Download JPEG	A JPEG of the chart is downloaded including all of the selected sensors.	A JPEG of the default chart is downloaded including all of the selected sensors.	REQ-4.5	PASS
9	Change download format to CSV and click Click Download CSV	A CSV of the chart is downloaded including all of the selected sensors.	A CSV of the default chart is downloaded that includes only the first sensor.	REQ-4.6	FAIL
10	Change download format to JSON and click Click Download JSON	A JSON of the chart is downloaded including all of the selected sensors.	A JSON of the default chart is downloaded that includes only the first sensor.	REQ-4.6	FAIL

Table 3.73

3.3.7 Test Case 7: Reports Chart Type Section

Purpose:	
The purpose of the test is to ensure that the chart type functionality of the reports page works as expected.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
William Kolath / 05/01/2025	Server is powered on and connected to the internet. LoRaWAN network is functioning correctly. At least one sensor cluster powered on and connected to the network.
Notes:	
Test Case: PASS	

Table 3.75: Test Case 7 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Open terratekrwh.com/reports	The reports page opens with a default plot displaying Line chart	The reports page opens with a default plot displaying Line chart	REQ-4.12	PASS
2	Click Scatter Plot Button	The chart changes to display a scatter plot.	The chart changes to display a scatter plot.	REQ-4.13	PASS
3	Click Bar Chart Button	The chart changes to display a bar chart.	The chart changes to display a bar chart.	REQ-4.14	PASS
4	Click Heat Map Button	The chart changes to display a heat map.	The chart changes to display a heat map.	REQ-4.15	PASS
5	Click Histogram Button	The chart changes to display a histogram.	The chart changes to display a histogram.	REQ-4.16	PASS
6	Click Violin Chart Button	The chart changes to display a violin chart.	The chart changes to display a violin chart.	REQ-4.17	PASS

Table 3.77

3.3.8 Test Case 8: Reports Sensor Section

<table>
<tr><td colspan="2" align="center">Purpose:</td></tr>
<tr><td colspan="2">The purpose of the test is to ensure that the sensor selection functionality of the reports page works as expected.</td></tr>
<tr><td colspan="2" align="center">Test Run Information:</td></tr>
<tr><td>Tester Name / Date:</td><td>Prerequisites / Required Configuration:</td></tr>
<tr><td>William Kolath / 05/01/2025</td><td>Server is powered on and connected to the internet. LoRaWAN network is functioning correctly. At least one sensor cluster powered on and connected to the network.</td></tr>
<tr><td colspan="2" align="center">Notes:</td></tr>
<tr><td colspan="2" align="center">Test Case: PASS</td></tr>
</table>

Table 3.79: Test Case 8 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Open terratekrwh.com/reports	The reports page opens with a default plot.	The reports page opens with a default plot.	N/A	PASS
2	Click Cluster Location button for Sensor 1	A drop down of all the available locations is displayed.	A drop down of all the available locations is displayed.	N/A	PASS
3	Select a different location	The chart updates to display data from the selected location.	The chart updates to display data from the selected location.	REQ-4.1	PASS
4	Click Sensor button for sensor 1	A drop down of all the available sensors is displayed.	A drop down of all the available sensors is displayed.	N/A	PASS
5	Select a different sensor	The chart updates to display data from the selected sensor.	The chart updates to display data from the selected sensor.	REQ-4.1	PASS
6	Click Aggregation Type button for sensor 1	A drop down of all the available aggregation types is displayed.	A drop down of all the available aggregation types is displayed.	N/A	PASS
7	Select a different aggregation type	The chart updates to display data that is aggregated by the selected method.	The chart updates to display data that is aggregated by the selected method.	REQ-4.1	PASS
8	Click Color bottom	A popup of all the available colors is displayed.	A popup of all the available colors is displayed.	N/A	PASS
9	Select a different color.	The color of the displayed chart is updated.	The color of the displayed chart is updated.	REQ-4.1	PASS
10	Repeat steps 2 through 9 for sensor 2 and sensor 3.	Each step produces the expected result and the chart properly displays up to 3 different datasets, one for each sensor selected.	Each step produces the expected result and the chart properly displays up to 3 different datasets, one for each sensor selected.	REQ-4.3	PASS
11	Click the Reset/Clear button for each of the sensors	Sensor 1 is reset to defaults and sensors 2 and 3 are removed for the chart.	Sensor 1 is reset to defaults and sensors 2 and 3 are removed for the chart.	N/A	PASS

Table 3.81

3.3.9 Test Case 9: API data fetching

Purpose: validate the correctness of data queried	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Dale Andre Descallar	
Notes:	
Test Case PARTIAL PASS	

Table 3.83: Test Case 9 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	query for data with timeframe as 0	return data from current hour	returned the past 24 hours	REQ 2.2	fail
2	query for data with timeframe as 1	return data from hour 0 to 1	returned average from 1st hour and second hour	REQ 2.2	fail
3	create two functionally identical queries, returning the past 72 hours	return data is identical	returned data is identical	REQ 2.2	pass

Table 3.85

3.3.10 Test Case 10: API invalid input handling

Purpose: validate wrong how the API handles invalid inputs	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Dale Andre Descallar	
Notes:	
Test Case PASS	

Table 3.87: Test Case 10 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	input query without start or end date	uses default timeframe (past 24 hours)	returned the past 24 hours	REQ 3.2	Pass
2	input query without sensor	return error	return error	RE1 2.3	pass
3	input query without board	return error	return error	RE1 2.3	pass
4	input query without timeframe	defaults timeframe to 24 hours	returned the past 24 hours	RE1 2.3	pass
5	input query without calculation method	defaults to average	returns average	RE1 2.3	pass
6	input query without time interval	defaults interval to hourly	returns hourly results	RE1 2.3	pass
7	input query without unit conversion	defaults to imperial	returned imperial units	RE1 2.3	pass
8	input query where start date and end date are a year before data was collected	return nothing	returned nothing	RE1 2.3	pass
9	input query where start and end date are a year after the most recent data point	return nothing	returned nothing	RE1 2.3	pass

Table 3.89

3.3.11 Test Case 11: Real-Time Sensor Display (Freshwater Tank 1)

Purpose: Ensure real-time sensor data is fetched and displayed accurately for Fresh Tank 1	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector G. Ceballos	All sensors must be connected and transmitting data via LoRaWAN. Frontend must be online.
Notes:	
Test Case FAIL	

Table 3.91: Test Case 11 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Navigate to “Freshwater Tank 1” tab	Page displays real-time data widgets for water level, and temperature	page loads everything except water level and pH	REQ-2.1	Fail
2	Observe pH sensor value	pH value should display quickly	”Loading pH...” remains indefinitely	N/A	Fail
3	Wait for sensor refresh	All sensors update with new data	Water level stays at 0, pH does not show, others update correctly	N/A	Fail

Table 3.93

3.3.12 Test Case 12: Real-Time Sensor Display (Freshwater Tank 2 Page)

Purpose: Ensure real-time sensor data is fetched and displayed for the Freshwater tanks 2 and 3	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector G. Ceballos	All sensors must be connected and transmitting data via LoRaWAN. Frontend must be online.
Notes:	
Test Case PASS	

Table 3.95: Test Case 12 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Navigate to “Freshwater Tank 2” tab	Page displays real-time data widgets for water level, and temperature	page loads everythinge	N/A	Pass
2	Observe the sensor values	values should display quickly	Values display slowly	N/A	Pass
3	Wait for sensor refresh	All sensors update with new data	Everything loads with new data	N/A	pass

Table 3.97

3.3.13 Test Case 13: Real-Time Sensor Display (Fresh-water Tank 3 Page)

Purpose: Ensure real-time sensor data is fetched and displayed accurately for the Freshwater Tank 3	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector G. Ceballos	All sensors must be connected and transmitting data via LoRaWAN. Frontend must be online.
Notes:	
Test Case PASS	

Table 3.99: Test Case 13 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Navigate to "Freshwater Tank 2" tab	Page displays real-time data widgets for water level, and temperature	page loads water level, and temperature	N/A	Pass
2	Observe the sensor values	values should display quickly	Values display relatively quick	N/A	Pass
3	Wait for sensor refresh	All sensors update with new data	Everything shows properly	N/A	Pass

Table 3.101

3.3.14 Test Case 14: Real-Time Sensor Display (Grey-water Tank Page)

Purpose: Ensure real-time sensor data is fetched and displayed accurately for the Greywater Tank	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector G. Ceballos	All sensors must be connected and transmitting data via LoRaWAN. Frontend must be online.
Notes:	
Test Case PARTIAL PASS	

Table 3.103: Test Case 14 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Navigate to "Greywater Tank" tab	Page displays real-time data widgets for water level, TDC, pH, and temperature	page loads everything except for pH	REQ-2.1	Partial Pass
2	Observe pH sensor value	pH value should display quickly	"Loading pH..." remains indefinitely	N/A	Fail
3	Wait for sensor refresh	All sensors update with new data	pH shows nothing, others show properly	N/A	Partial Pass

Table 3.105

3.3.15 Test Case 15: Historical Data Charts Freshwater Tank 1 and Grey-water Tank pages

Purpose: Verify historical sensor values for graphs display time-series data correctly.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector G. Ceballos	Historical sensor data is stored in backend/database.
Notes:	
Test Case PARTIAL PASS	

Table 3.107: Test Case 15 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Load Freshwater 1 Tank and Grey-water Tank pages	Historical graphs for pH, water level, water pH, and TDC are visible	page loads everything except for pH	REQ-2.3	Partial PASS
2	Observe pH sensor value	pH value should display quickly	"Loading pH..." remains indefinitely	N/A	Fail
3	Wait for sensor refresh	All sensors update with new data	pH is not visible, others update correctly	N/A	Partial PASS

Table 3.109

3.3.16 Test Case 16: Historical Data Charts Freshwater Tank 2 and 3 pages

Purpose: Verify historical sensor values for graphs display time-series data correctly.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Hector G. Ceballos	Historical sensor data is stored in backend/-database.
Notes:	
Test Case PASS	

Table 3.111: Test Case 16 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Vali-dated	Pass /Fail
1	Load Freshwater 2 Tank and Freshwater 3 Tank pages	Historical graphs water level and water temper-ature are visible	page loads every-thing	N/A	Pass
2	Wait for sensor refresh	All sensors update with new data	Everything shows properly	N/A	Pass

Table 3.113

3.3.17 Test Case 17: System Behavior with Intermittent Network Connection

Purpose:	
To verify that sensor data is correctly buffered or recovered when the network experiences brief disconnections.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Preston DeShazo 4/28/2025	Simulated network interruptions must be possi-ble (e.g., via firewall or router rules).
Notes:	
Test Case Pass	

Table 3.115: Test Case 17 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Validated	Pass /Fail
1	Start sensor transmission with a stable network	Data is sent to the server normally	The connection is stable to start	REQ-5.1, REQ-5.4	Pass
2	Disconnect the network for 2 minutes	Sensor cluster attempts to reconnect	The sensor cluster reconnects with out issue	REQ-6.5	Pass
3	Connect the network for 2 minutes	Data is sent to the website like before disconnection	The data stream is back to normal	REQ-5.1, REQ-6.5	Pass
4	Check backend database or logs	Data gap is minimal or clearly documented	The Scatter plot will show the disconnection in the data	REQ-4.2, REQ-5.6	Pass

Table 3.117

3.3.18 Test Case 18: Unit Conversion

Purpose:	
Verify that switching between imperial and metric units correctly updates all displayed values across the interface.	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Preston DeShazo / 04/24/2025	There must be data to be shown and converted on the website.
Notes:	
Test Case Pass	

Table 3.119: Test Case 18 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Vali-dated	Pass /Fail
1	Go to reports page	Values are shown in default unit (e.g., metric)	The units are shown in there default untis	REQ-4.9	Pass
2	Switch unit toggle from metric to imperial	All values convert (e.g., °C to °F, cm to inches)	On the reports page, the unit conversion works as intended	REQ-4.9	Pass
3	Switch back to metric	Values revert correctly	The conversion back works aswell	REQ-4.9	Pass

Table 3.121: Test Case 18 Steps

3.3.19 Test Case 19: Real-Time Sensor Display (Weather Page)

Purpose: Ensure real-time sensor data is fetched and displayed accurately	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Kirk A. Blood	All sensors must be connected and transmitting data via LoRaWAN. Frontend must be online.
Notes:	
Test Case Partial Pass	

Table 3.123: Test Case 19 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Vali-dated	Pass /Fail
1	Navigate to "Weather" tab	Page displays real-time data for Wind speed, Wind direction,Humidity,Pressure,Temperature,Luminescence,and Rainfall,and Temperature	All data Displays properly except for rainfall	U6.2	Partial Pass
2	Observe Wind Rose and Data Widgets	Wind Rose and Widgets should show active data smoothly	Wind Rose and all widgets display properly and quickly	U6.2	Pass
3	Wait for sensor refresh	All sensors update with new data	All Sensor update properly, except rainfall	U6.2	Partial Pass

Table 3.125

3.3.20 Test Case 20: Weather Page weather forecast API

Purpose: Ensure real-time sensor data is fetched and displayed accurately	
Test Run Information:	
Tester Name / Date:	Prerequisites / Required Configuration:
Kirk A. Blood	Frontend must be online.
Notes:	
Test Case Pass	

Table 3.127: Test Case 20 Description

TEST SCRIPT STEPS/RESULTS					
Step	Test Step/Input	Expected Results	Actual Results	Req Vali-dated	Pass /Fail
1	Navigate to “Weather” tab	Page displays forecast data based on default drop down selection	The Weather Page properly displays the default 3 day forecast	U6.4	Pass
2	Change Drop Down Selection	Forecast data changes based on selection of 3-day or hourly	Changing the drop down option changes the display quickly and smoothly, showing the correct data	U6.4	Pass
3	Drop Down Options display properly	All Drop Down Options display properly	Both drop down options display properly when changing the forecast selection	U6.4	Pass

Table 3.129

References

[1] *About Lorawan.* Lora Alliance. URL: https://lora-alliance.org/about-lorawan/ (visited on 10/25/2024).

[2] Random Nerd Tutorials Arduino. *ME007YS Waterproof Ultrasonic Distance Sensor (SEN0312).* 2024. URL: https://www.arduino.cc/en/Guide/ME007YS.

[3] Arduino Chip Wired. *Waterproof DS18B20 Temperature Sensor Kit (KIT0021).* 2024. URL: https://www.arduino.cc/en/Guide/DS18B20.

[4] DFRobot. *MODBUS-RTU RS485 4-in-1 Soil Sensor (SEN0604).* 2024. URL: https://wiki.dfrobot.com/RS485_Soil_Sensor_Temperature_Humidity_EC_PH_SKU_SEN0604#target_1.

[5] Roy T Fielding, Richard N Taylor, Justin R Erenkrantz, Michael M Gorlick, Jim Whitehead, Rohit Khare, and Peyman Oreizy. "Reflections on the REST architectural style and" principled design of the modern web architecture"(impact paper award)". In: *Proceedings of the 2017 11th joint meeting on foundations of software engineering.* 2017, pp. 4–14.

[6] Radek Fujdiak, Konstantin Mikhaylov, Jan Pospisil, Ales Povalac, and Jiri Misurec. "Insights into the Issue of Deploying a Private LoRaWAN". In: *Sensors* 22.5 (Mar. 2022), pp. 2042–2042. DOI: https://doi.org/10.3390/s22052042. URL: https://www.mdpi.com/1424-8220/22/5/2042.

[7] Python Package Index. *ChirpStack API.* 2024. URL: https://pypi.org/project/chirpstack-api/ (visited on 10/25/2024).

[8] Kicad.org. *started in kicad — 8.0 — english — documentation — kicad*$_2$024. 2024. URL: https://docs.kicad.org/8.0/en/getting_started_in_kicad/getting_started_in_kicad.html.

[9] Mochammad Fariz Syah Lazuardy and Dyah Anggraini. "Modern front end web architectures with react. js and next. js". In: *Research Journal of Advanced Engineering and Science* 7.1 (2022), pp. 132–141.

[10] *Limitations*. The Things Network. URL: https://www.thethingsnetwork.org/docs/lorawan/limitations/ (visited on 10/25/2024).

[11] *LoRaWAN*. Arduino. 2024. URL: https://docs.arduino.cc/arduino-cloud/hardware/lora/ (visited on 10/25/2024).

[12] Alireza Maleki, Ha H. Nguyen, Ebrahim Bedeer, and Robert Barton. "A Tutorial on Chirp Spread Spectrum Modulation for LoRaWAN: Basics and Key Advances". In: *IEEE Open Journal of the Communications Society* 5 (2024), pp. 4578–4612. DOI: 10.1109/OJCOMS.2024.3433502.

[13] Sanjay Patni. *Pro RESTful APIs*. Springer, 2017.

[14] Biswajit Paul. "An Overview of LoRaWAN". In: *WSEAS TRANSACTIONS ON COMMUNICATIONS* 19 (Jan. 2021), pp. 231–239. DOI: https://doi.org/10.37394/23204.2020.19.27. URL: https://wseas.com/journals/articles.php?id=1302.

[15] *Pricing Plans*. The Things Network. URL: https://www.thethingsindustries.com/stack/plans/#:~:text=%24230%2Fmonth&text=Our%20powerful%20LoRaWAN%20Network%20Server,renowned%20companies%20of%20all%20sizes. (visited on 10/25/2024).

[16] Shamkant B. Navathe Ramez Elmasri. *Fundamentals of Database Systems*. Pearson Education, 2015.

[17] *Regional Parameters*. The Things Network. URL: https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/ (visited on 10/25/2024).

[18] Juan De Dios Santos. *Gravity Non-contact Liquid Level Sensor (SEN0204)*. 2024. URL: https://www.dfrobot.com/wiki/index.php/Gravity:_Non-contact_Liquid_Level_Sensor_SKU:_SEN0204.

[19] Juan De Dios Santos and Chip Wired. *DFRobot Gravity Analog Electrical Conductivity Meter (DFR0300)*. 2024. URL: https://wiki.dfrobot.com/Gravity__Analog_Electrical_Conductivity_Meter_V2_SKU__DFR0300.

[20] Semtech. *LoRa® and LoRaWAN®*. 2024. URL: https://www.semtech.com/uploads/technology/LoRa/lora-and-lorawan.pdf (visited on 10/25/2024).

[21] Sparkfun.com. *basics - sparkfun learn*$_2$022. 2022. URL: https://learn.sparkfun.com/tutorials/pcb-basics/all.

[22] Adafruit Learning System. *Arduino MKR Environmental Shield (ASX00029)*. 2024. URL: https://learn.adafruit.com/arduino-mkr-environmental-shield.

[23] *The ChirpStack project*. ChirpStack. URL: https://www.chirpstack.io/docs/ (visited on 10/25/2024).

[24] Random Nerd Tutorials. *Weather Meter Kit (SEN-15901)*. 2024. URL: https://randomnerdtutorials.com/weather-meter-kit/.

[25] Random Nerd Tutorials. *Weather-proof Ultrasonic Sensor (SEN0208)*. 2024. URL: https://randomnerdtutorials.com/weather-proof-ultrasonic-sensor/.

[26] *What are LoRa and LoRaWAN?* The Things Network. URL: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/ (visited on 10/25/2024).

[27] Elecrow Wiki. *LR1302 LoRaWAN HAT for RPI*$_P$*RD*. URL: https://www.elecrow.com/wiki/lr1302-lorawan-hat-for-rpi-prd.html (visited on 10/25/2024).

Appendix A

Backlog

A.1 User Stories for Semester 1

S. No.	Epic / User Stories	Effort	Priority	Sprint	Owner
E1	**Research and Requirements**	**8**	**Very High**		
U1.1	Research and List Required Sensors For Data Gathering	2	5	2	Kirk
U1.2	Research Arduino and Required Libraries	2	5	4	Preston, Kirk
U1.3	Research Networking and Connectivity	4	4	4	William, Andre
U1.4	Research Databases and Required Hardware For Server	5	4	3	Preston
U1.5	Research Cameras and Raspberry PI for Image and Video Collection	3	1	TDB	TDB
U1.6	Casing landscape For Final/Future Deployment	5	1	3	Kirk
U1.7	Research Server Requirements	3	4	2	Preston
U1.8	User Manual	3	4	6-7	Dale
U1.8	Research Similar Papers	3	4	6-7	Hector
E2	**Hardware Functionality**	**8**	**High**		

U2.1	Arduino enclosure	3	3	6	Preston
U2.2	Arduino Power	3	3	6	Kirk
U2.3	Raspberry Pi Network Hub Enclosure and Power	2	3	TDB	TDB
U2.4	Raspberry Pi Camera Enclosure and Power	2	1	2	William
U2.5	Rain Gauge Functionality	3	4	5	Preston, William
U2.6	Wind Direction Functionality	3	2	5	Preston
U2.7	Wind Speed Functionality	3	2	5	Preston
U2.8	Tank Level Functionality	3	2	5	Hector, Andre, William
U2.9	Water Quality Sensors Functionality (TDS/pH)	3	4	5-6	Kirk
U2.10	MKR Shield	3	4	5	Hector
U2.11	LoRaWAN Gateway	3	4	5	William
U2.12	Raspberry Pi Camera Functionality	5	1	2	William
U2.13	PCB Design	8	1	6	William
E3	**Server Setup**	**3**	**High**		
U3.1	Gather and Assemble Server Hardware	3	3	2	Preston
U3.2	Implement and Deploy Sever onto Network	5	4	2	Preston
U3.3	Design a simple Script for Website	2	2	2	Hector
U3.4	Deploy a Simple Website For Data Visualization	5	2	TDB	TDB
E4	**Networking**	**5**	**Mid**		
U4.1	Hardware Data Transmission to Server	5	3	7	William
U4.2	Server to Internet Communication	3	2	6	William
U4.3	Data Transmission Over Distance on Different Networks to The Server	8	5	7	William
U4.4	Peer to Peer Communication Between Sensors and Data Transmission Device	5	2	2	William
U4.5	Cybersecurity for Security From the Unwanted Individuals	8	1	TDB	TDB

U4.6	Arduino LoRaWAN functionality and finalized code	5	8	7	Preston, William
U4.7	Raspberry Pi LoRaWAN functionality and finalized code	5	8	7	William
E5	**Website Functionality**	**2**	**Very Low**		
U5.1	Receive Data From Database	4	5	4	Hector
U5.2	Displaying Received Data	2	2	6	Hector
U5.2	Website wireframe	2	2	7	Hector
E6	**Database Design**	**8**	**Mid**		
U6.1	Plan Database Structure	5	5	2	Andre
U6.2	Design and Write Database Schema	5	5	3	Andre, Hector, William
U6.2	Error handling and final structure	5	5	7	Andre, William

A.2 User Stories for Semester 2

S. No.	Epic / User Stories	Effort	Priority	Sprint	Owner
E1	**Hardware Installation**	**8**	**8**	**0**	**All**
U1.1	Select solar Power source	3	8	1	Preston
U1.2	Design and manufacture mounting hardware	5	8	1	William
U1.3	Finalize EC and PH sensors	8	8	1	Kirk
U1.4	Visit install site an finalize details of the installation	2	8	0	All
U1.5	Install sensor clusters at each of the required locations	5	8	2	All
U1.6	Install server and configure necessary networking	2	8	2	All
U1.7	Install LoRaWAN gateway	2	8	2	All

U1.8	Test server and networking	3	8	2	All
U1.9	Test each sensor cluster	5	8	2	All
U1.10	Test LoRaWAN connection	2	8	2	All
U1.11	Troubleshoot PH and EC	8	5	5	Kirk
U1.12	Troubleshoot depth sensors	8	5	5	All
U1.13	Final installation trip	5	8	6	All
E2	**API Development**	**5**	**8**		**Andre**
U2.1	API route mapping and documentation	3	8	0, 1, 2, 3	Andre
U2.2	Final API functional	3	8	1, 2, 3	Andre
U2.3	Error handling implementation	5	5	3	Andre
U2.4	API endpoint testing	3	5	3	Andre
E3	**Homepage**	**3**	**5**		**Preston**
U3.1	Wire frame and page prototype	5	5	1	Preston
U3.2	The homepage shows summarized data for water tanks and weather conditions.	8	8	2,3	Preston
U3.3	Include a navigation menu for accessing detailed pages	3	8	2,3	Preston
U3.4	Add History Of Project and Propose	5	5	3	Preston
U3.5	Properly style UI Components on website	3	3	3	Preston
E4	**Reports Page**	**8**	**3**		**William**
U4.1	Wire frame and page prototype	8	2	1	William
U4.2	Allow users to generate custom reports for historical sensor data	8	5	2	William
U4.3	Allow users to select line, box, scatter, histogram, or heatmap to vew data	8	5	2	William
U4.4	Allow users to select custom time range	8	5	3	William
U4.5	Generated reports can be downloaded in PNG, JPEG, JSON, or CSV format	3	8	1	William
U4.6	Allow selection for multiple different sensors	3	3	2	William

U4.7	Properly style UI Components on website	3	3	2	William
U4.8	Create mobile version of the reports page	3	3	3	William
E5	**Tank Pages**	**5**	**8**		**Hector**
U5.1	Wire frame and page prototype	2	5	0	Hector
U5.2	Provide detailed telemetry data for individual water tanks	8	8	3	Hector
U5.3	Displays real-time graphs for tank levels, pH, TDS, and temperature	5	8	3	Hector
U5.4	Users can filter historical data by date range	3	3	3	Hector
U5.5	Make Graphs reflect the condition of the sensor (Red for bad, Yellow for medium, And Green For Good)	3	3	3	Hector
U5.6	Add Level indicator for Tank Level	3	8	1, 2	Hector
U5.7	Properly style UI Components on website	3	3	1, 2	Hector
E6	**Weather Condition Page**	**5**	**5**		**Kirk**
U6.1	Wire frame and page prototype	3	3	1	Kirk
U6.2	Show real-time weather data and historical trends for decision-making	5	5	3	Kirk
U6.3	Add weather Extension for Future Weather Conditions	5	3	3	Preston
U6.5	Properly style UI Components on website	2	5	3	Kirk
u6.4	External Weather API	2	5	2	Preston
E7	**System Health Page**	**5**	**2**		**William**
U7.1	Wire frame and page prototype	2	3	2	William, Preston
U7.2	Show which endpoints are online	5	2	3	William
U7.3	Show transmission history for connected endpoints	8	2	2	William
U7.4	Show any sensor outages	5	2	3	William
E8	**Research and publish paper**	**5**	**2**		**All**

U8.1	Methods	3	8	5-7	William, Preston
U8.2	Related works	3	8	5-7	Hector
U8.3	Findings and Conclusions	3	8	5-7	Kirk, Andre
U8.4	Select conference/journal to publish paper	3	8	5-7	All
U8.5	Create rough draft of paper	8	8	6-7	All
U8.6	Create final draft of paper	8	8	7	All
U8.7	Submit paper	5	8	7	All
U8.8	Create user manual	5	8	8	All