Raspberry Pi Weather Station

Software Requirements Specification

Version: 2

10/4/2018

Joynal Abedin, Justin Roy, Leng Lee, Alumgur Uddin, Naim Cekaj

Revision History

Date	Description	Author	Comments
10-6-2018	Version 1	Raspberry Pi Group Fall 2018	First Version
10-8-2018	Version 1.1	Raspberry Pi Group Fall 2018	Additional references.
10-8-2018	Version 2	Raspberry Pi Group Fall 2018	Changes based on professor/TA feedback
12-5-2018	Version 2.1	Raspberry Pi Group Fall 2018	Changes made to Security and Communication protocols
12-5-2018	Version 2.2	Raspberry Pi Group Fall 2018	Changes made to

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Justin Roy	Team Leader	
	Joynal Abedin	Team Member	
	Leng Lee	Team Member	
	Naim Cekaj	Team Member	

	Alumgur Uddin	Team Member	
	Ryan Wood	Client	
	Dali Ismail	Teachers Assistant	

Table of Contents

Table of Figures

1. Introduction

1.1 Purpose

The purpose of the software requirements specification document is to provide an overview of the requirements in order to complete the Weather Station project. The Weather Station project will be a continuation from last semester and will have new features added to it. This document contains detailed information and explanations of features that are going to be implemented. It also contains the layouts of the system and constraints the applications has.

1.2 Scope

The Raspberry Pi Weather Station Network is a portable weather station system that acquires weather data from multiple sources located in remote areas. This will be completed by utilizing LoRa WAN technology to establish wireless communication between Raspberry Pi devices to gather weather data within a specific radius. The weather data will be stored into a database located in a virtual private cloud to ensure multiple levels of security.

MQTT will be used to establish a secure communication channel between the Raspberry Pi weather stations. Before the data is transferred to another device the message will be encrypted and the receiving pi will use a secret key to ensure its integrity. While at rest, the stored data will also stay encrypted. Prevention techniques for physical and network intrusions will be put in place to enhance overall security of the weather station system.

With each weather station being placed in remote locations, power consumption must be minimized to establish longevity of the system. Each Raspberry Pi will be optimized to use the least power consumption possible to perform the weather station needs.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
LPWAN	Low power wide area network is a type of

	wireless telecommunication network designed to allow long range communication
LoRa WAN	A media access control (MAC) protocol for wide area networks. It is designed to allow low-powered devices to communicate with Internet-connected applications over long range wireless connections
LoRa Hat	An expansion module for LoRa WAN and GPS for use with a Raspberry Pi
MQTT	Message Queuing Telemetry Transport is an ISO standard publish-subscribe based on messaging protocol. It is designed to be used in remote locations or where network bandwidth is limited
Mosquitto	An open source message broker that implements the MQTT protocol
Broker	A broker is an IP that a publisher will send data to queue while waiting for a subscriber to retrieve
Subscriber	A subscriber is a device that will reach out to a broker to retrieve any data queued. Subscriber sends no data to the broker, but just checks it for queued data
Publisher	A publisher is a device that sends data to a broker, where it will be retrieved by a subscriber
Raspberry Pi	An affordable computer chip that is made up of one serial board
API	Application Programming Interface, we use this to grab information from another website and set up URLs the Raspberry Pi can use to send data
RDS	Relational Database Service
AWS RDS	Relational Database Service allows you to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing

	time-consuming database administration tasks
VPC	Amazon Virtual Private Cloud lets you launch AWS resources, such as Amazon RDS DB instances, into a virtual private cloud
Security Group	provides security at the protocol and port access level. Each security group works much like a virtual firewall which contains a set of rules that filter traffic coming into and out of an instance
Internet Gateway	Allow resources in a VPC to communicate with the internet by creating and attaching an internet gateway to the VPC
MySQL	An open-source relational database management system
HTTP	HyperText Transfer Protocol. Generally used to send data across the web
Server	Program that waits for a message to either push or pull data. The server determines if the request should be allowed access to its data, and responds if the request is allowed
Client	Program that sends a message to a server. There can be multiple clients sending messages to the server at the same time.
Encryption	The process of converting information/data into a code to prevent unauthorized access
Decryption	The process of transforming data that has been rendered unreadable through encryption back to its unencrypted form
Hashing	It is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string.
Symmetric Encryption	A form of computerized cryptography using a singular encryption key to disguise a message
Fernet	An implementation of symmetric

	authenticated cryptography. Ferent guarantees that a message with its encryption cannot be manipulated or read without a key
Public Key	It is a large numerical value that can be used to encrypt data which is intended for a particular recipient, so that the encrypted data can be deciphered only by using a second key that is known only to the recipient
Salt	Random data that is used as an additional input to a one-way function that hashes data
Twilio	A web service that provides API calls to allow a program to send SMS text messages.
Barometer	Sensor used to measure atmospheric pressure. Units are measured in pounds per square inch (psi)
Thermometer	Sensor used to measure temperature. Units are measured in Fahrenheit or Celsius.
Hygrometer	Sensor used to measure humidity. This value is measured in a percentage that denotes the amount of water the air can still absorb.
Accelerometer	Sensor used to measure acceleration. This value is measured in meters per second squared.
Gyroscope	Sensor used to determine the orientation of the device. This value is expressed as a degree.
React	React is a frontend Javascript framework that allows small parts of each web page to be built using components. These components make it easier to split up a web page and keep each individual piece bug free.
Python	An interpreted high-level backend programming language.
Node	Node is a backend runtime environment, so that Javascript can be executed on the server

	side. This means that while the Node server is either retrieving or sending data, it will be able to continue working instead of being blocked as that interaction is happening.

1.4 References

This subsection should:

[1]"FAQs." Rotate Display 90º? - Raspberry Pi Forums, www.raspberrypi.org/documentation/faqs/.

[2] MQTT.org. (2018). Retrieved October 4, 2018, from https://mqtt.org/

[3] Welcome to pyca/cryptography¶. (2018). Retrieved October 4, 2018, from https://cryptography.io/en/latest/

[4] What is LPWAN (low-power wide area network)? - Definition from WhatIs.com. (2018). Retrieved October 4, 2018, from https://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network

[5] Jackson, B., Atiyeh, B., Kodali, J., & Malarkey, T. (2018). Weather Station Software Requirements Specifications (pp. 1-40, Tech. No. 1). Detroit, Michigan [MI].

1.5 Overview

This document will now go over the general description of the weather station system which will consist of the product perspective, product functions, user characteristics, general constraints, assumptions and dependencies. The following section (three) will then establish the specific requirements with external interfaces such as user, hardware, software, and communications. The remainder of the document will elaborate on function and non-functional requirements, design restraints, logical database requirements along with any additional requirements for the weather station system. The document will be concluded with analysis models such as a data flow diagram.

2. General Description

2.1 Product Perspective

The Raspberry Pi Weather Station Network is intended to build on the previous weather station version. Through the use of LoRa WAN technology, multiple raspberry pi weather stations will communicate to a central pi sending weather data that will be sent to a database located on a virtual private cloud. MQTT implementation will establish a secure communication channel between weather stations. Data encryption during data at transit and data at rest will be used to ensure integrity of the data. Optimizing power consumption of each weather station to establish longevity of the system.

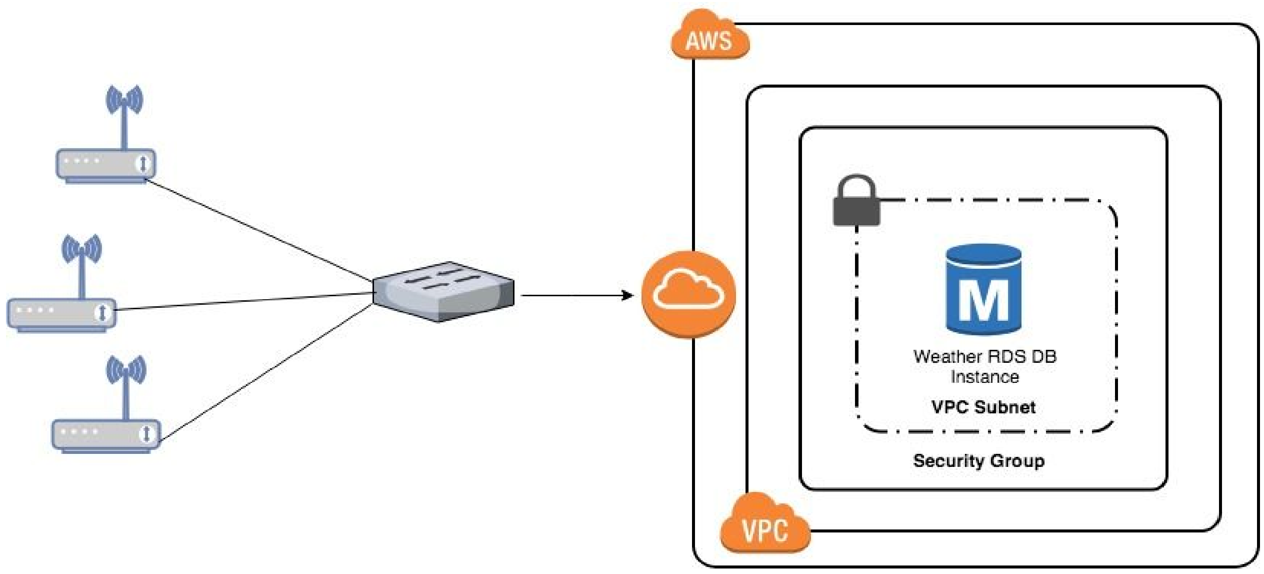


Figure 2.1.1: Diagram of Raspberry Pi Weather Station Network

2.2 Product Functions

Encryption and Decryption

While transferring data from one device to another, the data would be secured by encrypting before the actual transfer. Once the data has been successfully transfered from a servant station to the master station, it would be decrypted by the master. Once decrypted it will make a post request to our server, where it will then be parsed and inserted into our database accordingly.

Data will also be encrypted at rest on all servant stations by using RSA 128. RSA uses an private/public key scheme, which allows a user to pass their public key around to be used for encryption. Any data that is encrypted using the public key will only be decrypted using the private which is generated with the public key when generating keys. Each encryption will be using PKCS1_OAEP, which is a padding scheme used to pad each encryption to a size of 128 bytes.

As each station's generates a weather data file, immediately it will be passed to our encryption algorithm to encrypt the data with RSA 128. This way, any files generated will safely rest on the servant stations because they will not have access to the private key.

Auto-Start Python Scripts on Bootup
On initial boot of the system (Raspberry Pi) will start collecting data from the sensors attached to it. Once the information has been collected, it will then be broadcasted to a listener (master) via LoRa. LoRa will search a given directory for any new files that is being generated by our station.

On the master station, our posting script will also auto-start on bootup. The script will search inside of directory "EncryptedFilesRecieved" for any data within it. If master finds any data within the directory, it will then decrypt the data using our private key, then make a post request to our server with the decrypted data.

LoRa Wan Network
A connection between three Raspberry Pi devices will be established using LoRa WAN technology. With the LoRa/GPS Long Range Transceiver HAT, the range of communication for the three pi are amplified. Once connection has been established, each hat will be used to send data from servants to master. Master device will be the only device that will have a internet connection, which will allow it to send data directly into the database.

Health Status
A simple dashboard that can display battery level, RAM stats, CPU Utilization. The system will display the data on the front-end and will be accessible by a user admin.

Email Alerts
Currently, the software sends alerts to users when an alarm is triggered. Additional functionality will include sending SMS messages to users and additional alarm triggers.

2.3 User Characteristics

A random user can deploy multiple Raspberry Pi units in a remote area. The user will have access to two accounts.

Normal user will be able to view the weather information such as temperature, humidity, pressure and so on. The normal user will also be able to access the dashboard that shows the health check of all the systems that is being used to deliver weather data.

Super user will have all the functionality as the normal user, however, they can also assign/revoke user privileges. Super user will also be able to remote login to the master Raspberry Pi and reboot the device in the event of an emergency.

2.4 General Constraints

With LoRaWAN technology being used as the main source of communication between the weather stations the use of the LoRa/GPS Long Range Transceiver HAT will be necessary. A potential constraint is due to the terrain and environment, if the HAT was to break, communication to that specific weatherstation would be lost.

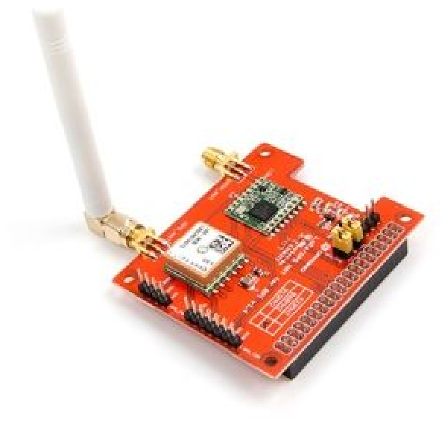


Figure 2.4.1: LoRa/GPS Long Range Transceiver HAT

With the master weather station having access to an internet source it will have the capability to allow remote access. The constraint is that once the rest of the weather stations are deployed there will be no remote access to them.

Thirdly, testing the network at a high scale is not possible due to the lack of weather stations available. Having only three Raspberry Pi weather stations it will not be possible to stress test our communication system to replicate a real world experience.

2.5 Assumptions and Dependencies

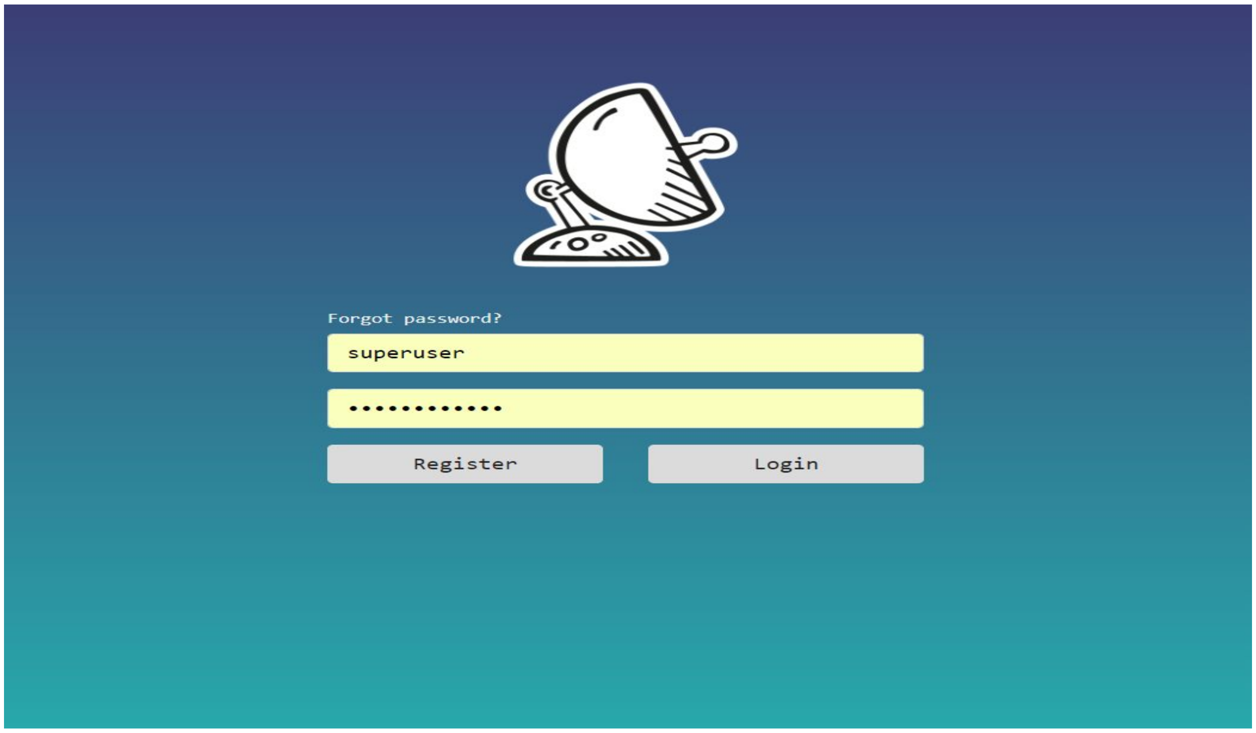
It is assumed that the master unit will always have a stable internet connection to establish communication with the other weather stations. It is also assumed that when the weather stations are deployed the communication between them will be near instantaneous and successful. Lastly, it is assumed that the Raspberry Pi Weather Stations will be pre configured with all libraries and dependencies needed.

3. Specific Requirements

3.1 External Interface Requirements

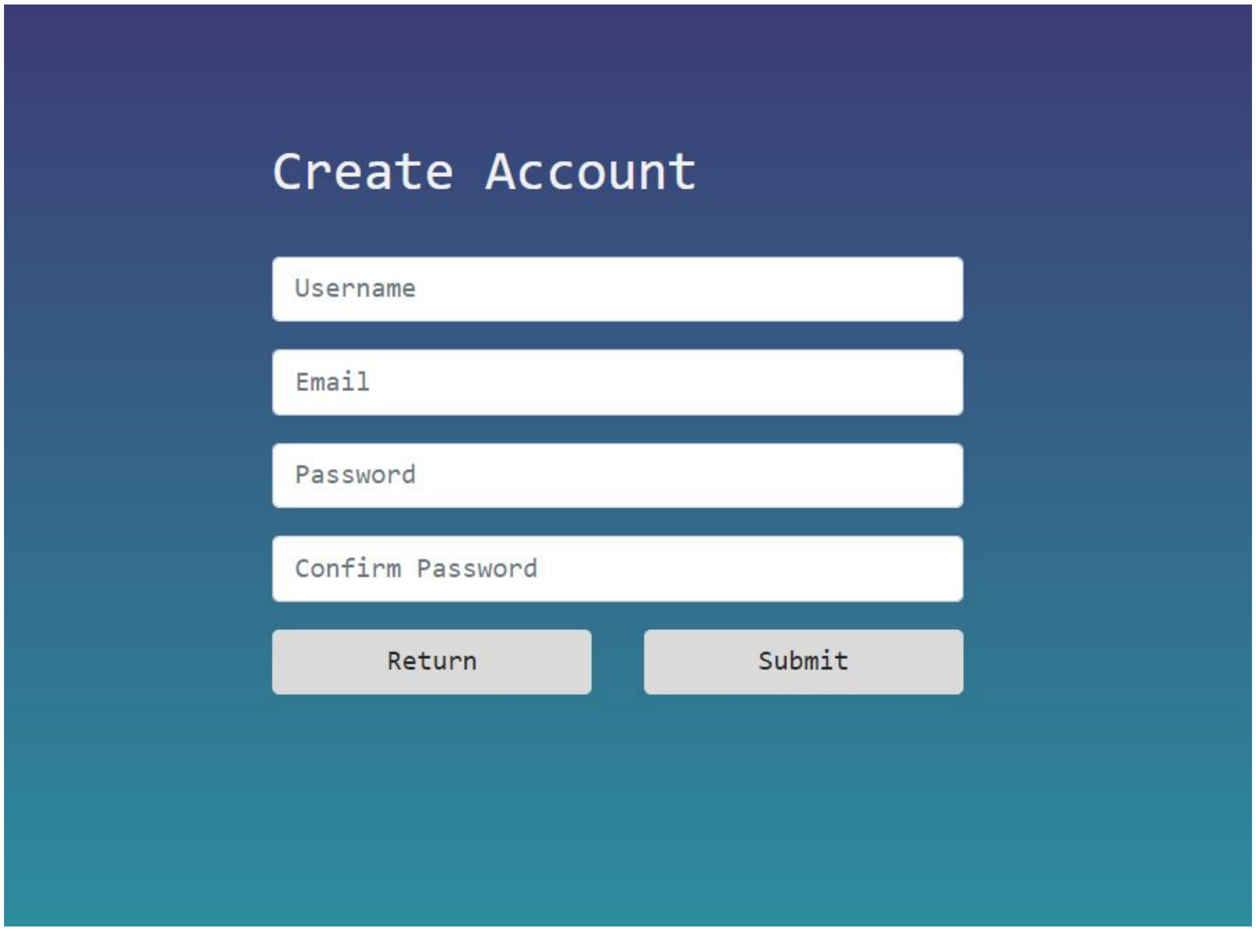
3.1.1 User Interfaces

Since this is an extension of the previous semester, the user interface created by the previous semester shall be utilized for this project as well. The user will begin the web application by having the option of either logging in or registering a new account. The page will have the following view with two fields for username and password along with two buttons for logging in or registering a new account. In the case where the user forgets their password then the user may click on the link to gain access to their account.



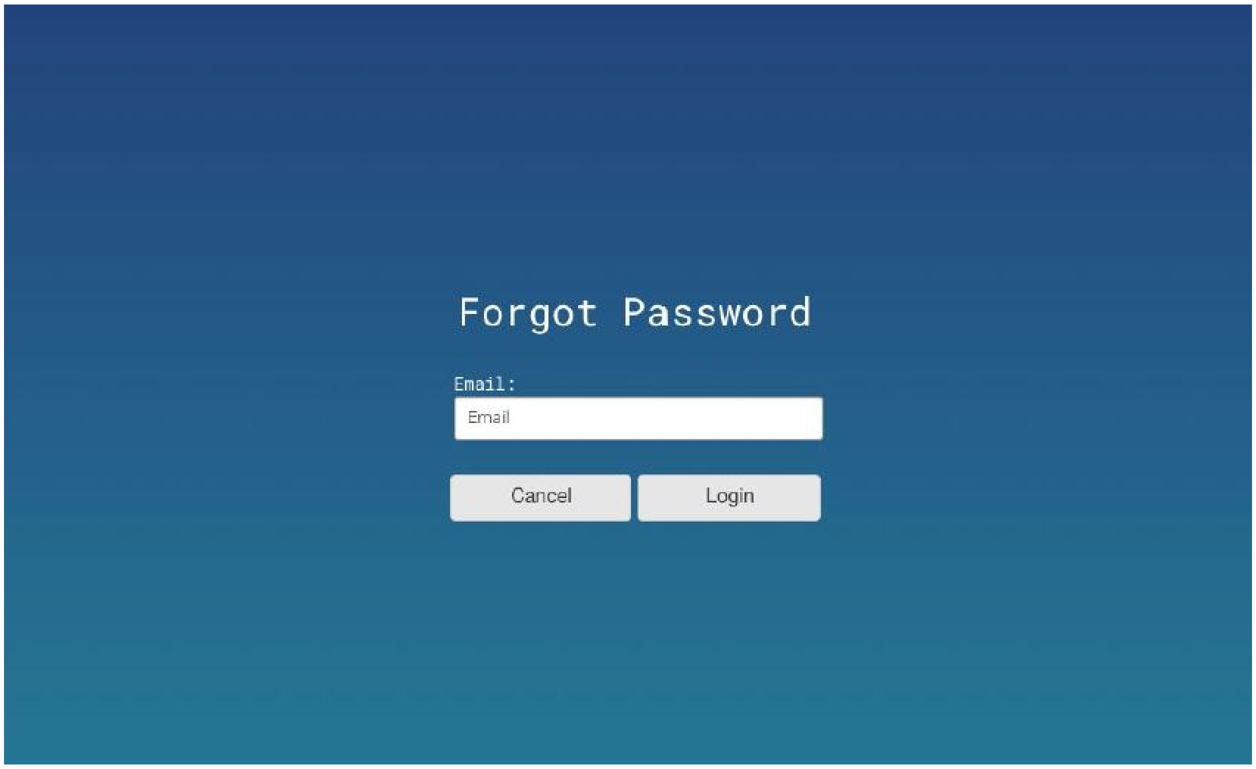
(Jackson, Atiyeh, Kodali & Marlarkey, 2018) **Figure 3.1.1-1: Login View**

If the user wishes to create a new account they would click on the "Register" button located on the Login View, refer to figure 3.1.1-1. The user will then be redirected to the register page where they will enter their username, email, and password. Once the user completes registering a new account they will then be redirected to the login page.

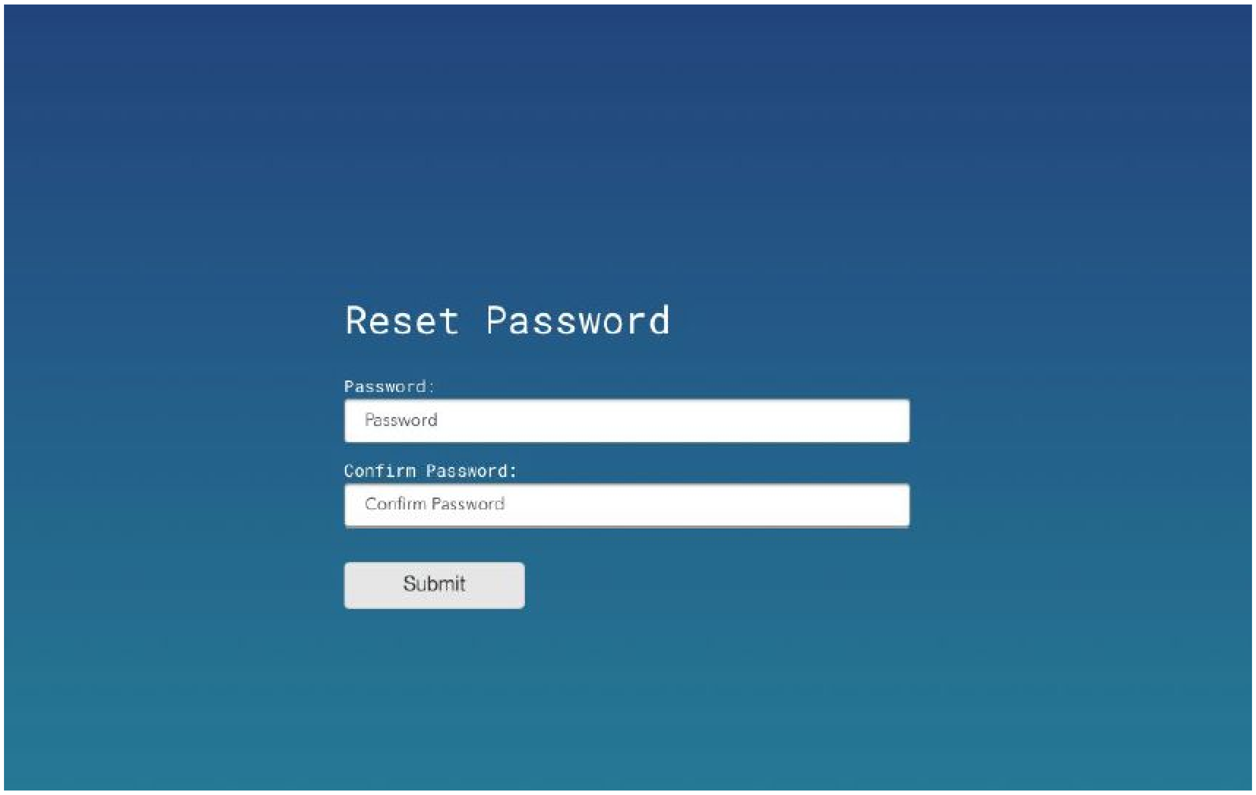


(Jackson et al.,2018) Figure 3.1.1-2: Create Account View

In the case of a user forgetting their password, the user may click on the “Forgot password?” link that will redirect the user to the forgot password view. Here the user enters their email that is linked to their account. The user will then receive an email with a link to changing their password credentials for future logins.

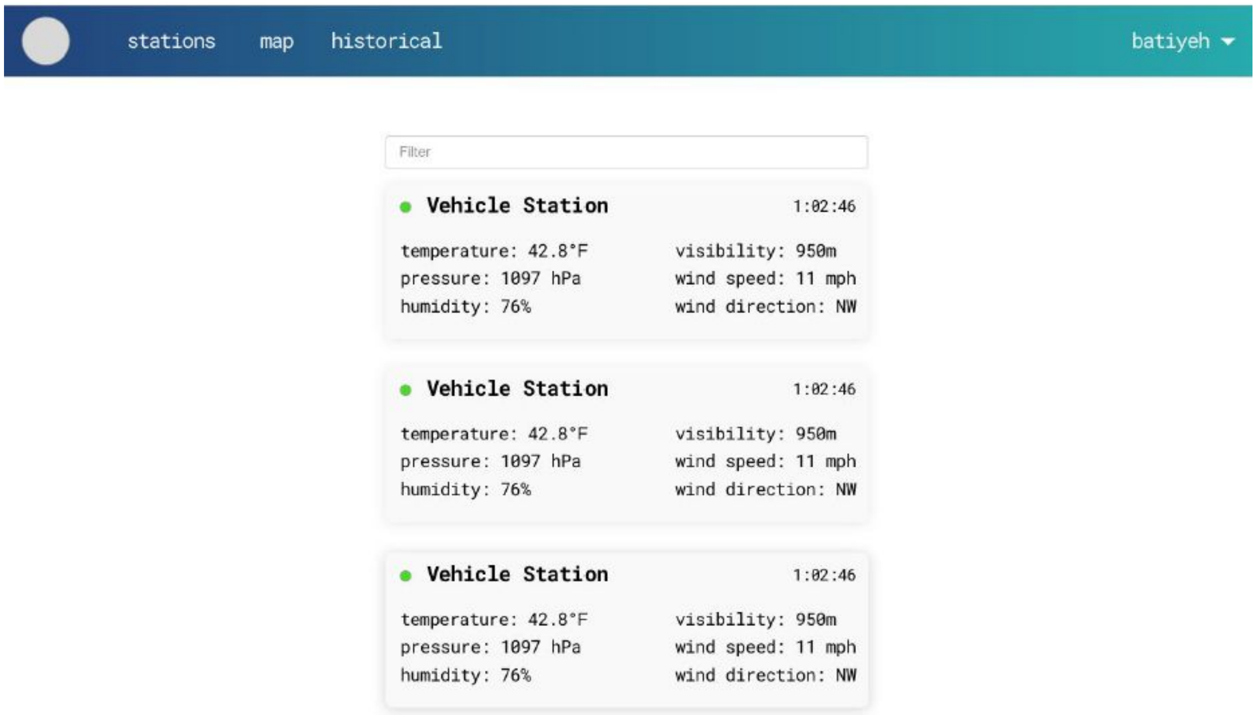


(Jackson et al.,2018) Figure 3.1.1-3: Forgot Password View



(Jackson et al.,2018) Figure 3.1.1-4: Reset Password View

Once the user successfully logins into the web application, they will be able to view the connected weather stations. This page displays all currently connected weather stations in a scrollable list. This list includes a filter feature to enable the user to filter the stations by name.



● Vehicle Station 1:02:46

temperature: 42.8°F visibility: 950m
pressure: 1097 hPa wind speed: 11 mph
humidity: 76% wind direction: NW

● Vehicle Station 1:02:46

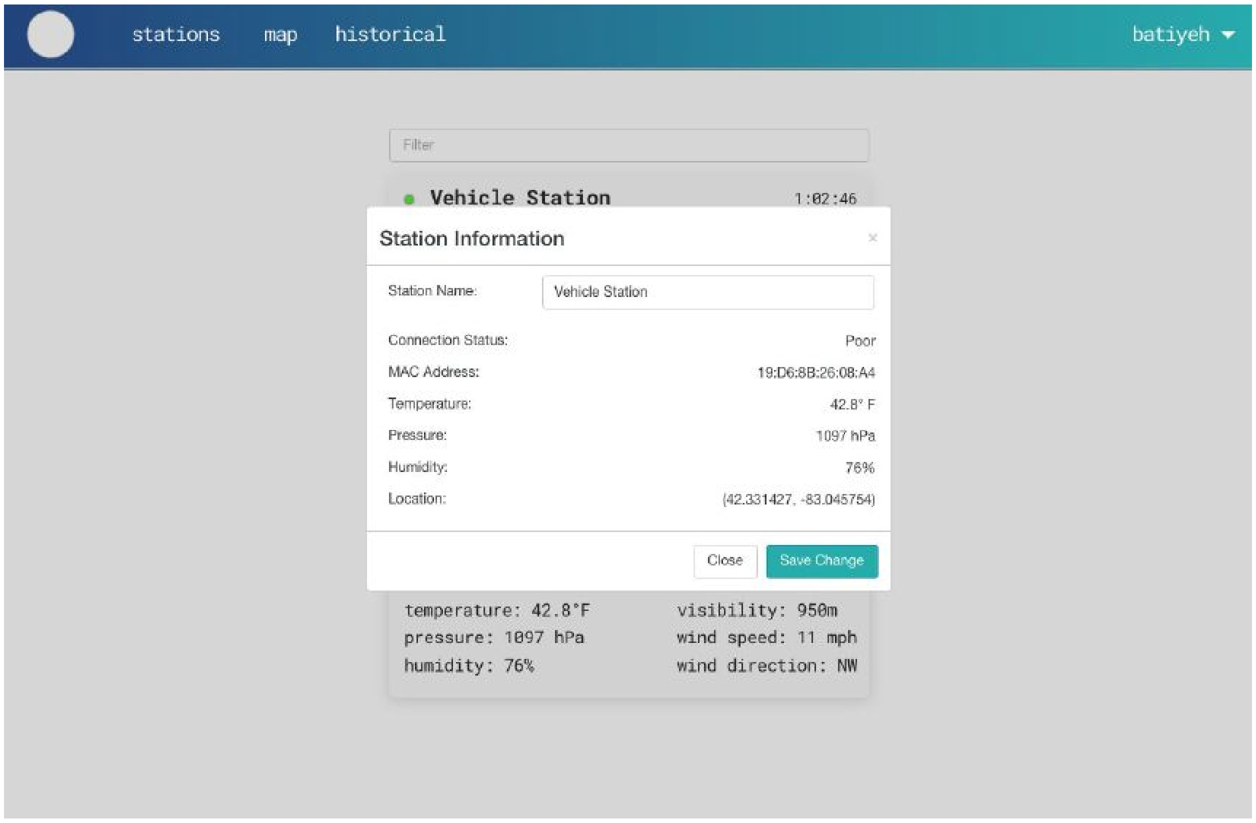
temperature: 42.8°F visibility: 950m
pressure: 1097 hPa wind speed: 11 mph
humidity: 76% wind direction: NW

● Vehicle Station 1:02:46

temperature: 42.8°F visibility: 950m
pressure: 1097 hPa wind speed: 11 mph
humidity: 76% wind direction: NW

(Jackson et al.,2018) Figure 3.1.1-5: Connected Stations View

While on this view the user can click onto each weather station card to view additional information. Additional information such as MAC address, latitude, longitude, health status and an input box for the station name will be available in this view.



● Vehicle Station 1:02:46

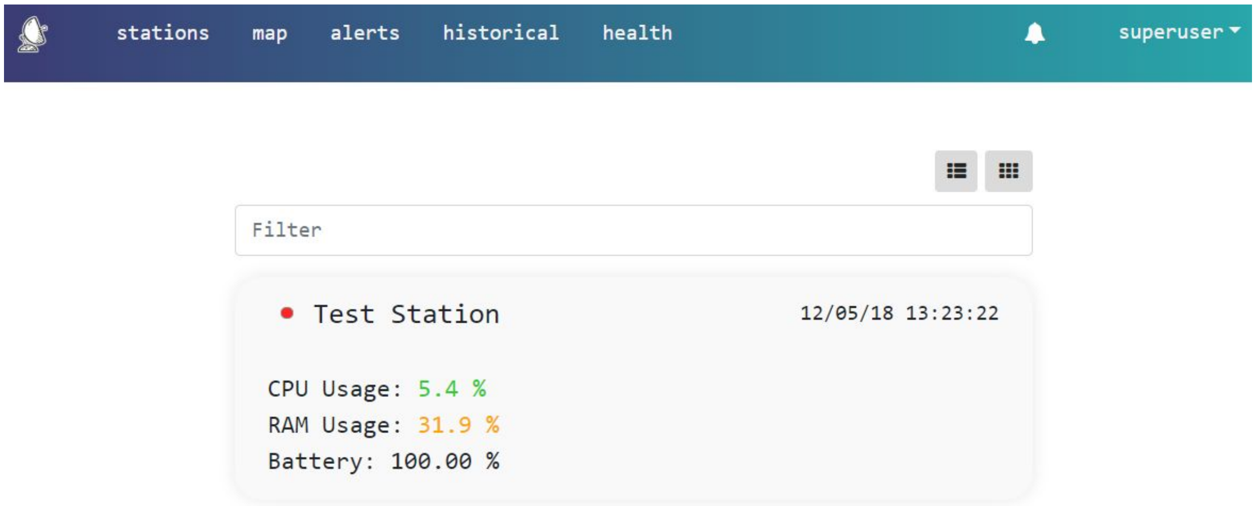
Station Information

Station Name:	Vehicle Station
Connection Status:	Poor
MAC Address:	19:D6:8B:26:08:A4
Temperature:	42.8° F
Pressure:	1097 hPa
Humidity:	76%
Location:	(42.331427, -83.045754)

```
temperature: 42.8°F        visibility: 950m
pressure: 1097 hPa         wind speed: 11 mph
humidity: 76%              wind direction: NW
```

(Jackson et al.,2018) Figure 3.1.1-6: Connected Station Additional Information View

The user can navigate to the station health page by clicking on the health tab. The health page displays the weather stations CPU and RAM usage and battery percentage. This page displays all currently connected weather stations in a scrollable list. This list includes a filter feature to enable the user to filter the stations by name.

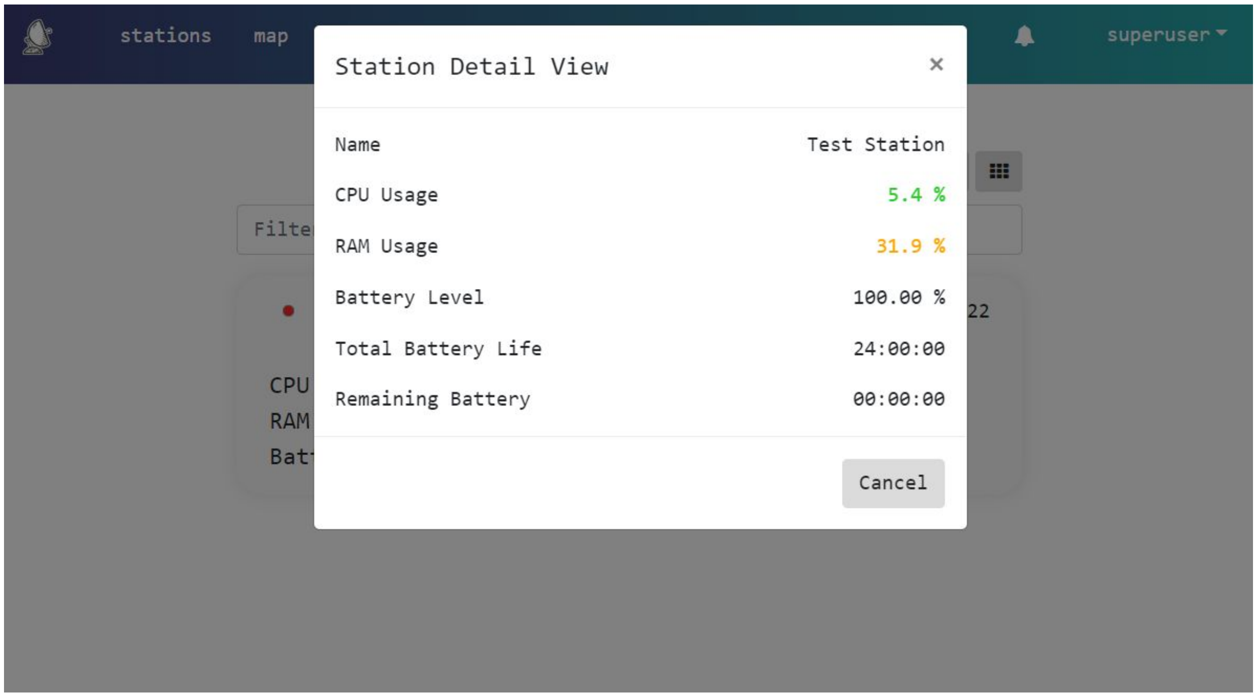


● Test Station 12/05/18 13:23:22

CPU Usage: 5.4 %
RAM Usage: 31.9 %
Battery: 100.00 %

Figure 3.1.1-7: Station Health View

While on this view the user can click onto each weather station card to view additional information. Additional information such as CPU usage, RAM usage, battery percentage, expected total life of the battery, and remaining battery life.

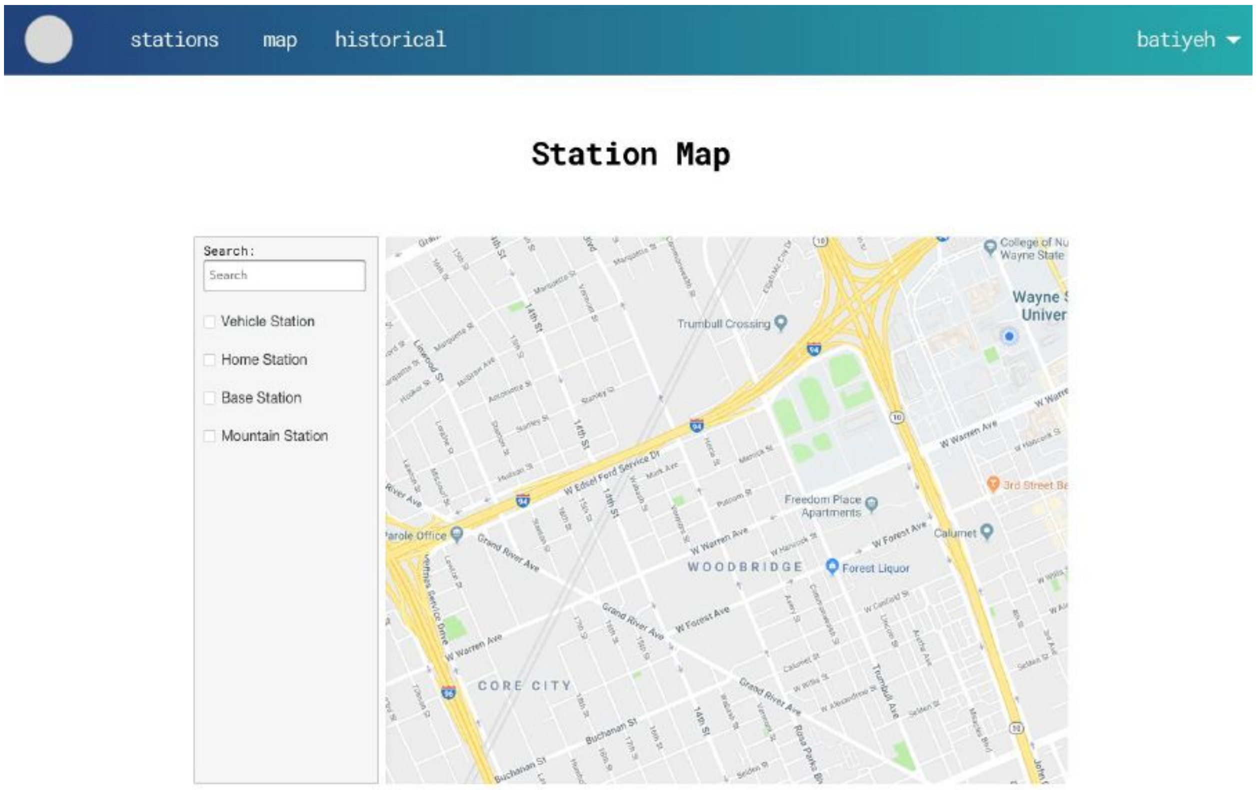


Station Detail View

Name	Test Station
CPU Usage	5.4 %
RAM Usage	31.9 %
Battery Level	100.00 %
Total Battery Life	24:00:00
Remaining Battery	00:00:00

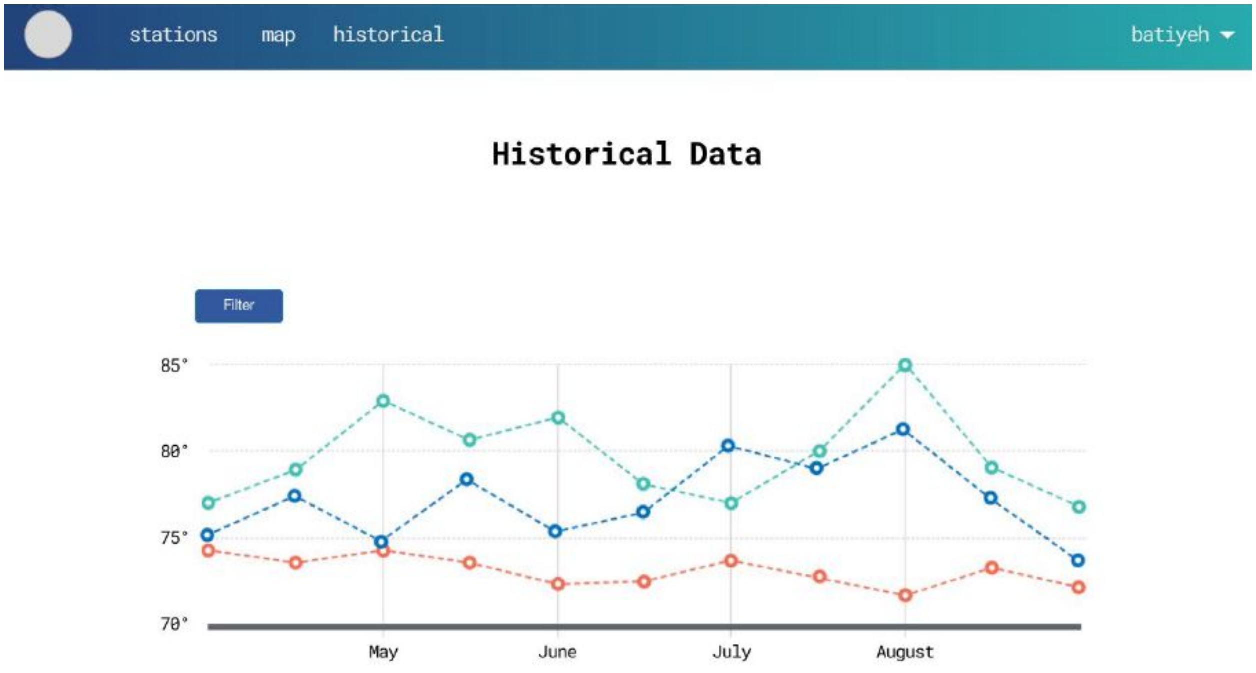
Figure 3.1.1-8: Station Health Additional Information View

The user can navigate to the map page by clicking on the map tab. The map page displays where the weather stations are located utilizing Google Maps API. The user is able to filter the stations based on their name and then selecting the weather station they wish to view on the map.

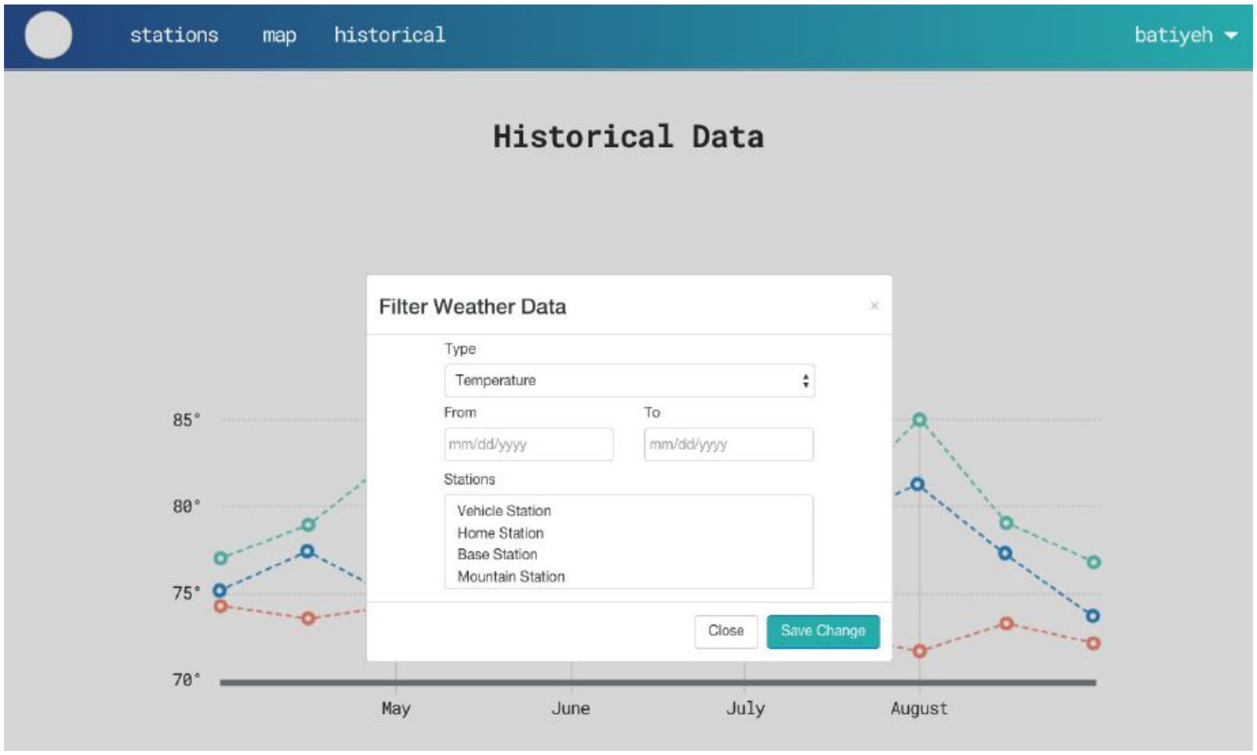


(Jackson et al.,2018) Figure 3.1.1-9: Station Map View

When the user clicks on the historical tab, the user is welcomed with the historical page view. In this view the user is able to see historical data from the weather stations deployed as well as data from Open Weather API. The user is able to filter this data by selecting to view data between two dates, either view the API or station readings and select either a single or multiple stations.



(Jackson et al.,2018) Figure 3.1.1-10: Historical Data View



Filter Weather Data

Type: Temperature

From: mm/dd/yyyy

To: mm/dd/yyyy

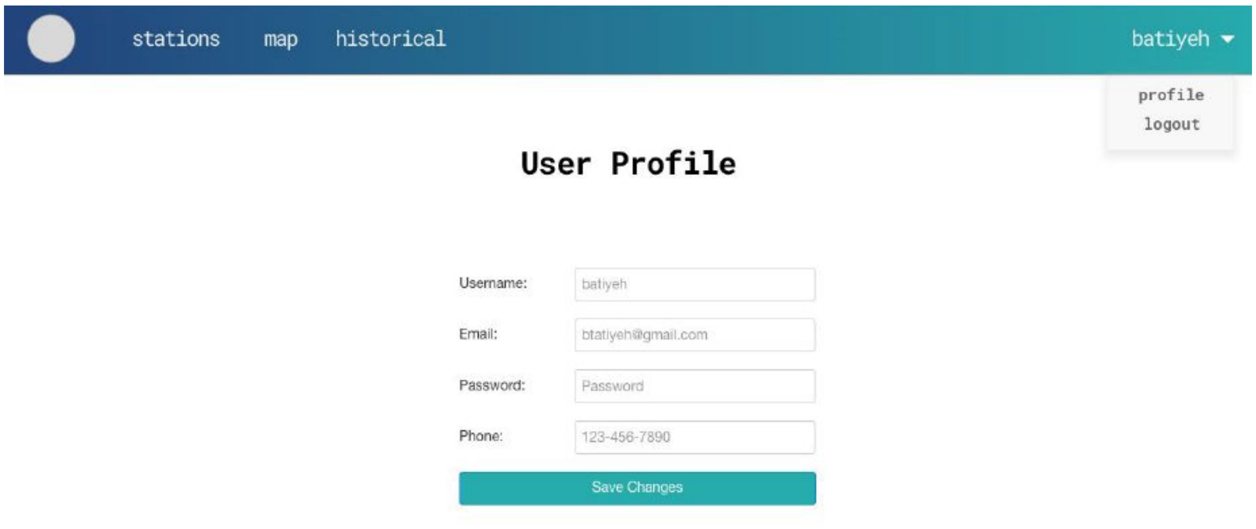
Stations

Vehicle Station
Home Station
Base Station
Mountain Station

Close | Save Change

(Jackson et al.,2018) Figure 3.1.1-11: Historical Data Filter Model

In order for the user to view their profile they are able to click on their username located on the top right hand corner of the navigation bar. A drop down menu will appear allowing the view profile information or logout of the web application. If the user selects “profile,” the the user will be redirected to the user profile page where they can view and change the stored information of their account.



User Profile

Username:	batiyeh
Email:	btatiyeh@gmail.com
Password:	Password
Phone:	123-456-7890

(Jackson et al.,2018) Figure 3.1.1-12: User Profile Page

Once the user selects to logout from the web application, then the user will be redirected to the login page.

3.1.2 Hardware Interfaces

The main component of the weather station is the Raspberry Pi that contains a LoRa HAT. This particular Raspberry Pi will have a connection to the internet and also have communication to the two other Raspberry Pi. The version of the Raspberry Pi being used is the Raspberry Pi 3 Model B. This particular model includes a quad core 1.2 GHz broadcom BCM2837 64 bit CPU, 1 GB of Ram, BCM348 on board wireless LAN, and 40-pin extended GPIO. The operating system and data will be both stored on a SanDisk Ultra 32GB microSDHC card.

The LoRa/GPS Long Range Transceiver HAT contains a built in temperature sensor, low battery indicator, and a gps base. It also has the ability to allow other LoRa connected devices to communicate through long range. The range of communication ranges up to 9 miles on a open field.

3.1.3 Software Interfaces

The weather station web application will be an extension from last semester's. The application has been developed using Node.js, Express, React and MySQL. Node serves as the backend framework using Javascript that enables unblocking I/O. This means that Node is able to handle requests while the application is still able to run. A Node.js framework known as Express is used to handle HTTP requests and page routing. As a user navigates throughout the application, Express handles the route of the web page the user is attempting to access. Node will acquire the necessary data from MySQL database and pass that data over to React. Once React receives the data it will render the interface of the page and display the data to the user. The Raspberry Pi is developed using Python where every three seconds, data from the weather sensors is retrieved and sent to the web server to be stored in a MySQL database.

3.1.4 Communications Interfaces

The application for the weather station is web-based network server that depends on data coming from external hardware and third party APIs. The web application is built around an API that handles HTTP requests. When accessing the website, users request to pull data from the database and the Open Weather API before it is displayed. These HTTP requests are done asynchronously with Node which allows the application to handle multiple requests at the same time. With the use of LoRaWAN technologies, a subscriber Raspberry Pi is only needed to have a internet connection will the broker Raspberry Pi's do not need internet access. The broker Raspberry Pi's will communicate with the subscriber Raspberry Pi using MQTT. Once data from the broker is retrieved by the subscriber, data comes in a text format due to using MQTT, the data will then be parsed for specific weather data and sent to the database for later retrieval.

3.2 Functional Requirements

FR- 1	Communication between one master Raspberry Pi and an undetermined amount of additional Pi's.
Description	This functional requirement is for scalability and overall project necessity, without this requirement completed the project would not be useful in its current state. The expectation without barriers or obstructions in the way, device to device communication should be no less than three miles.
Input	The input will be coming from a sensor device communicates directly with the master device tasked with processing and connecting to the database. There is no set master due to the nature of the project, every device by design will

	have the ability to send and receive messages but a single master device will be connected to the internet.
Processing	The sensor device will be processing data coming from the sensors attached as a HAT or AdaFruit attachments. Additionally the sensor device will be encrypting the data before communicating with a master device. This process is expected to take no longer than five seconds time assuming no loss of communication exists. The master devices will be processing the encrypted data into readable form and forwarding it to the database which will hold an encrypted version of the data from all sensor devices. This should take no longer than five seconds.
Output	The sensor device in its current state is outputting the data to a simple text file. The expectation going forward is all output will be encrypted at rest on the device, during transit and on the database. Additionally, the front end web server will include: ● Map of last known location ● Historical data ● Alerts ● Latest weather
Error Handling	Types of errors expected: ● Loss of communication: ○ Offline device. Caused by loss of power or connection. ■ If master with internet connection: Critical failure with no method of resolution, output alarm. ■ If master that is not internet connected: Determine if any devices nearby are available to route traffic, output alarm. ■ If sensor device: Output alarm.

FR- 2	Alert based on possible incoming storm in the region.
Description	The information being transmitted from each Raspberry Pi will include three different metrics of data being recorded at each location. Alarms will be raised based on factors set by group.
Input	Sense HAT or AdaFruit sensors, providing temperature, humidity and pressure
Processing	All processing of the data gathered will be on the web server based on real-time and historical data.

Output	Alerts will be sent via SMS and Email if an alert is triggered. In this case the alert will be a specific storm warning.
Error Handling	Types of errors expected: ● SMS or Email failing to send. ● Incorrect predictions: Constant refinement of the algorithm will yield more consistent predictions.

FR- 3.1	Alert when CPU usage is between 30-70%
Description	Once the CPU usage for the Raspberry Pi reaches between 30% and 70%, an alert shall be sent to the subscribed user informing the subscriber about the CPU usage.
Input	CPU usage: 50% Identification key for Raspberry Pi
Processing	1. Determine when the CPU usage is at 50% 2. If so, then send alert to subscribed user
Output	Alert sent to subscribed user stating the Raspberry Pi id and message stating that CPU usage is abnormal
Error Handling	Types of errors expected: ● SMS or Email failing to send.

FR- 3.2	Alert when CPU usage is above 70%
Description	Once the CPU usage for the Raspberry Pi goes above 70%, an alert shall be sent to the subscribed user informing the subscriber about the CPU usage.
Input	CPU usage: 85% Identification key for Raspberry Pi
Processing	1. Determine when the CPU usage is at 85% 2. If so, then send alert to subscribed user
Output	Alert sent to subscribed user stating the Raspberry Pi id and message stating that CPU usage is is really high
Error Handling	Types of errors expected: ● SMS or Email failing to send.

FR- 4.1	Alert when RAM usage is between 30-70%
Description	Once the RAM usage for the Raspberry Pi reaches between 30% and 70%, an alert shall be sent to the subscribed user informing the subscriber about the RAM usage.
Input	RAM usage: 50% Identification key for Raspberry Pi
Processing	3. Determine when the RAM usage is at 50% 4. If so, then send alert to subscribed user
Output	Alert sent to subscribed user stating the Raspberry Pi id and message stating that RAM usage is abnormal
Error Handling	Types of errors expected: ● SMS or Email failing to send.

FR- 4.2	Alert when RAM usage is above 70%
Description	Once the RAM usage for the Raspberry Pi goes above 70%, an alert shall be sent to the subscribed user informing the subscriber about the RAM usage.
Input	RAM usage: 85% Identification key for Raspberry Pi
Processing	5. Determine when the RAM usage is at 85% 6. If so, then send alert to subscribed user
Output	Alert sent to subscribed user stating the Raspberry Pi id and message stating that RAM usage is is really high
Error Handling	Types of errors expected: ● SMS or Email failing to send.

FR- 5	Extension of battery life for all devices
Description	Current benchmarks: Large battery - 70 hours Small battery - 23 hours
Input	N/A
Processing	Test plans will be developed to simulate a standard process list of items that should be occuring on the device at any given time and how much power is being drawn.

	The goal is for batteries to show at least 50% improvement with each size of battery.
Output	N/A
Error Handling	

3.3 Non-Functional Requirements

3.3.1 Performance

Given the smaller device and reduced computing power, performance will be a factor when determining where the algorithms and alarming should be taking place. With this in mind all heavy computing of algorithms, alarming and predictions will be done on the web server. Using this schema none of the deployed Raspberry Pi's will reach 100% CPU or memory usage.

3.3.2 Reliability

Application will have robust error handling to keep an uptime of at least 99%. To ensure high reliability in storage, we will have the automatic failover for our RDS instance across multiple location with synchronous data replication.

To minimize downtime, server instances will be deployed into multiple zones that are insulated from each other, in case of failure, one will back up the other server.

3.3.3 Availability

To maintain the high availability of the application, it will be launched on multiple Elastic compute cloud instances from the Amazon Machine Image and later use Load Balancing to distribute the incoming traffic for the application to the reserved instances. Also, spreading the instances into isolated location within a region will improve the fault tolerance in the application. Incase one location experiences an outage, traffic can be automatically routed to a different location.

3.3.4 Security

Data will be encrypted while at rest. The Database instance will be using the industry standard AES 256 encryption algorithm to encrypt the data on the server-side. The relational database system will handle authentication of access to the data and decryption of the data with a minimum impact on the performance. The key that is being is used to encrypt must be stored in a key management system. That system must also be protected by FIPS 140-2 Validation Program which is a U.S. government security standard used to approve cryptographic modules.

While transferring data from one device to another, the data would be secured by encrypting before the actual transfer. Once the data has been successfully transfered from a servant station to the master station, it would be decrypted by the master. Once decrypted it will make a post request to our server, where it would be parsed and inserted into our database.

3.3.5 Maintainability

Hardware: All hardware can be bought from retail outlets and replaced as necessary in case of faults or damage. When replacing or changing the specific HAT’s (Sense and LoRa) care should be taken to avoid bending any of the pins on the Raspberry Pi. With proper drivers and using the install documentation from the previous semester, swapping hardware or setting up a new device should be achievable by someone with no tech experience. In the future this group will write a document explaining how to setup MQTT and LoRa communication starting with the previous semester’s device.

Software: The current version of the software has an output that is easily usable for future functionality. All current modifications are being made using Python scripts that can be changed with knowledge of that specific language. Furthermore the code from the previous semester was done in Python, but the output is in text file format so utilizing the output could technically be done in any programming language preferred.

3.3.6 Portability

The communication protocol using LoRa Transmitter Hat is inherently available on most common operating systems including but limited to: MacOS, Linux, Microsoft Windows. The scripts that are currently implemented are written in Python with the assumption of a Linux based system but can be ported or rewritten for a different operating system if necessary.

Additionally a relational SQL database was chosen as the most suitable database design because of the unchanging data structure coming from each individual Raspberry Pi. Modifying the existing scripts to send the data to different types of databases shouldn’t be a large task if it is necessary.

It may be possible to port this software package to additional microcontrollers, but care would have to be taken to ensure the new microcontroller has the ability to run and execute Python scripts. No testing by this semester's group will take place on this and is purely hypothetical.

3.4 Design Constraints

We encountered no specific design constraints from our client. Referencing code that was written from previous semesters we chose technologies that would avoid conflict while adding features and modifying existing code.

Hardware constraints were also not put forth by the client except to continue using Raspberry Pi's, we were given four different types of sensors and as a group are choosing to add a new Raspberry Pi HAT. Other than that, limited memory and processing power will be taken into considerations. Due to environmental factors such as extreme heat and frigid cold conditions, the Raspberry Pi may not be suitable. According to the manufacturers, the Raspberry Pi is qualified to run in the temperature range of 0°C to 70°C, while the system on chip qualifies for the following temperature range of -40°C to 85°C. The Raspberry Pi may work outside of those temperatures but this is not qualified by the manufacturers. [1]

3.5 Logical Database Requirements

The system must store all the weather information such as timestamp, api key, temperature, humidity, pressure, latitude, longitude in a structured format. Each attribute should be delimited by a comma. The date format must be in year-month-day followed by the time stamp when the file was generated. This file then must be stored into a database. MySQL database will be used to store the information and retrieve data.

The instance which will have the database engine running must have automated backups. Backups retention period must be at least 7 days. In case of data loss, system must be able to recover the database to any point in time during the backup retention period. Storage size of the database shall be 20GiB.

For availability, the relational database system must synchronously replicate the data to a standby instance in a different Availability Zone. Use a security group to control network access to the database. The database instance shall be running on a virtual private cloud, and encrypted using keys to maintain data integrity. For security purpose, the use of SSL will be integrated to secure data in transit.

3.6 Other Requirements

4. Analysis Models

4.1 Data Flow Diagrams (DFD)

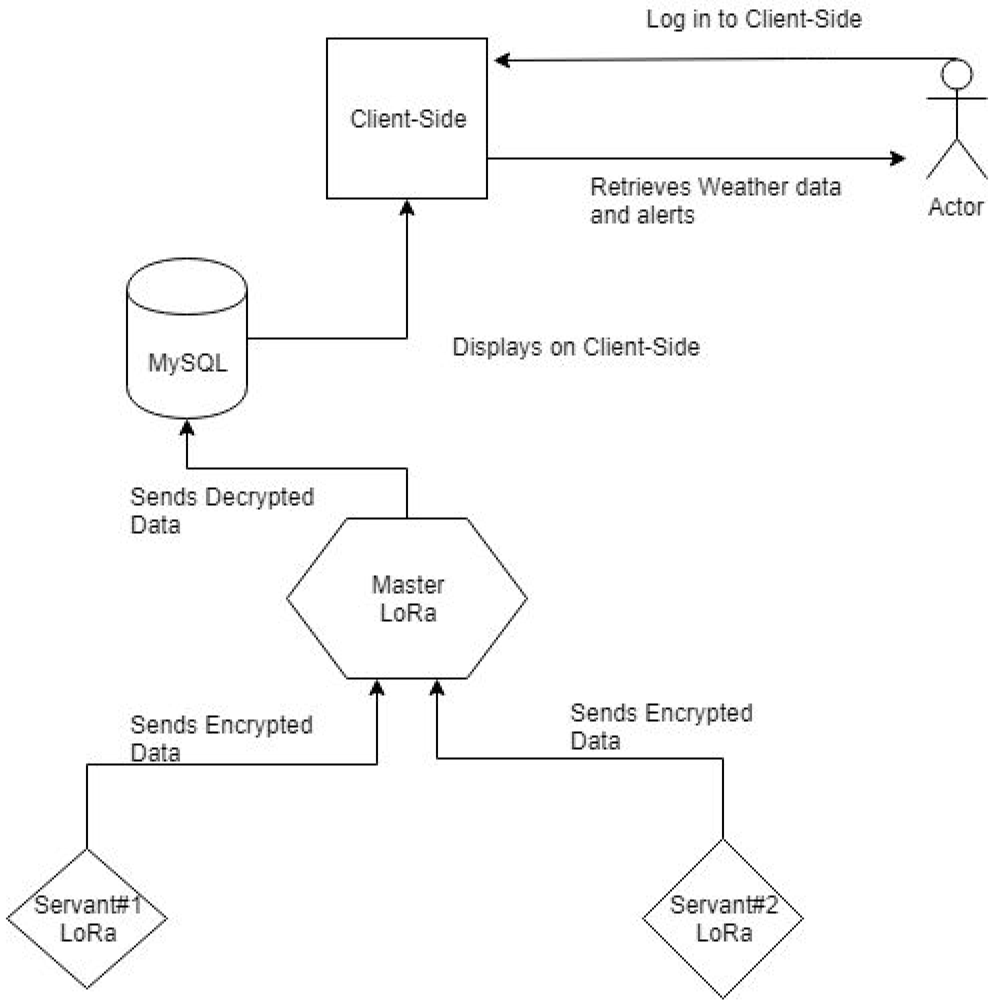
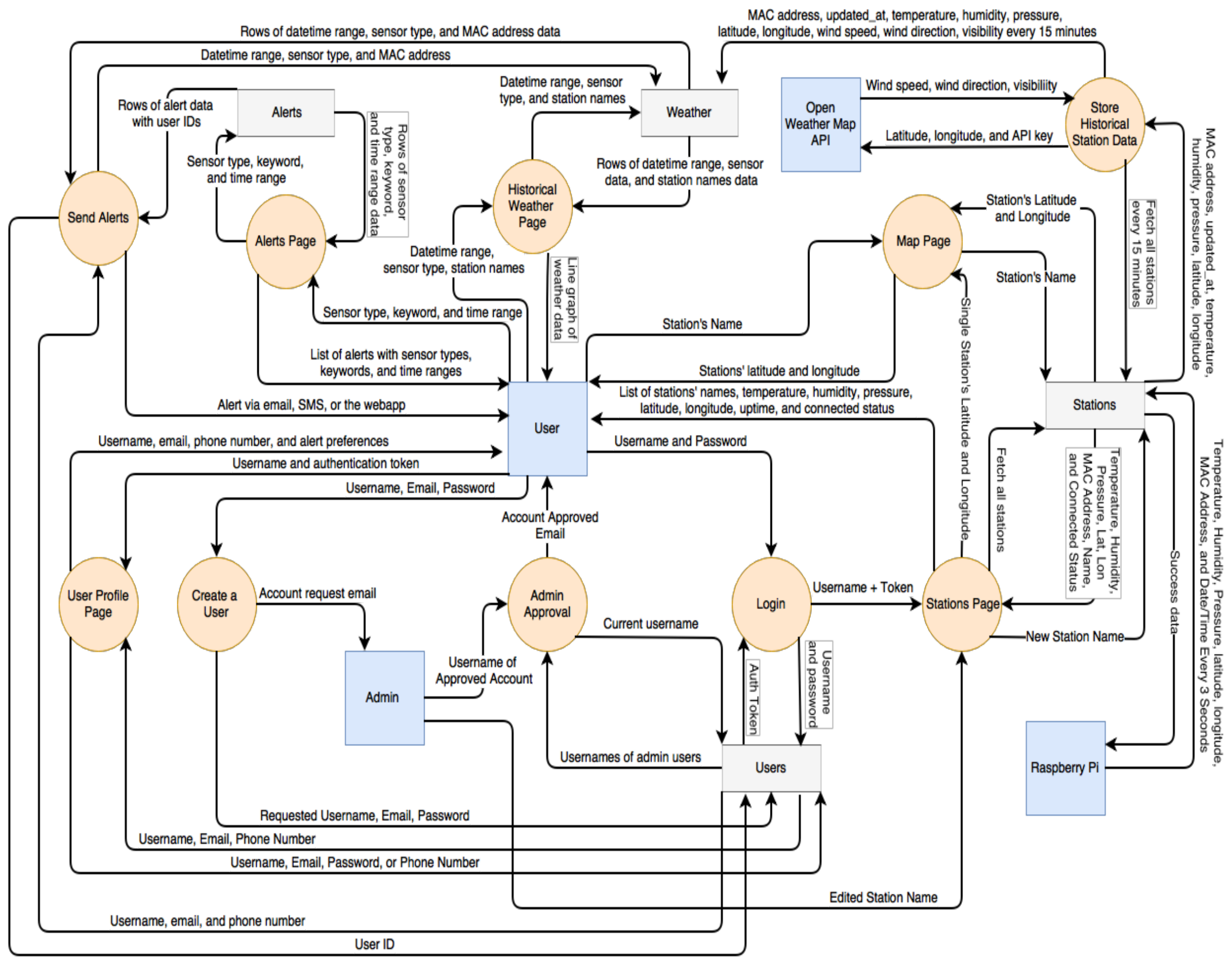


Fig 4.1-1 System's Data Flow Diagram

The data flow for the system is displayed above. It shows the communication between the weather stations and how it will be output data for the user to view. Master will listen for broadcasted data from the servant pi. Each individual pi will have a LoRa Hat to enable communication. Once Master receives data from the servants, it will then make a request to insert into the database, where it will be displayed for end users.



(Jackson et al.,2018) Fig. 4.1-2 Previous semester's Client-Side Data flow

The data flow above shows a flow of the client-side web application. It was done by the group from the previous semester. Since this semester's goal was to add a wireless communication offline to the current network only, the client-side will remain the same. Above it describes how data flows from the Raspberry Pi to the the web application.

A. Appendices

A.1 Appendix 1

Notes from in-person Client meeting, 10-04-2018

- Finish and test LoRa integration: “try not to take away features”
- Encryption at rest/in transit
- Storm warning - if time allows, need warning/alarm system based on possible storm
- Vamp up the Client Side - Make edits to the Client Side to make it better

A.2 Appendix 2

Jackson, B. *Weather Station Software Requirements Specifications* (Vol. V1.3). Wayne State University.