



Weather Station Product Design Specification

February 26

REPRESENTATIVES

Brandon Jackson

Brian Atiyeh

Jeswanth Kodali

Trevor Malarkey

VERSION HISTORY

Date	Version	Author	Comments
2/14/2018	v1.0	Jeswanth Kodali	Document skeleton
2/23/2018	v1.1	Brandon Jackson, Brian Atiyeh, Trevor Malarkey, Jeswanth Kodali	Rough draft sent to TA for feedback
2/24/2018	v1.2	Brandon Jackson, Brian Atiyeh, Trevor Malarkey, Jeswant Kodali	Revised version based on TA feedback
2/25/2018	v1.3	Brandon Jackson, Brian Atiyeh, Trevor Malarkey, Jeswanth Kodali	Presentation version
3/1/2018	v1.4	Brandon Jackson, Brian Atiyeh, Trevor Malarkey, Jeswanth Kodali	Final version

Table of Contents

1	Introduction	4
1.1	Purpose	4
2	General Overview and Design Guidelines/Approach	4
2.1	Assumptions / Constraints / Standards	4
3	Architecture Design	6
3.1	Hardware Architecture	6
3.2	Software Architecture	8
3.3	Security Architecture	10
3.4	Communication Architecture	11
3.5	Raspberry Pi Data Storage Architecture	12
4	System Design	12
4.1	Use-Cases	12
4.2	Use Case Diagrams	32
4.3	Sequence Diagrams	34
4.4	Data Flow Diagram	42
4.5	Database Design	43
4.6	Class Diagrams	44
4.7	Application Program Interfaces	46
4.8	User Interface Design	53
5	Product Design Specification Approval	60
A	Appendices	61
	Appendix A: References	61
	Appendix B: Key Terms	62
	Appendix C: Use Case/Sequence Diagram Mapping	63

1 Introduction

1.1 Purpose

The purpose of the Product Design Specification document is to give a comprehensive description of the architecture and system design for the software to be developed. To achieve this, the document will include an architecture design section that contains all necessary diagrams for hardware, software, security, and communications architecture. Along with the architecture design section a system design section will include all use cases for the site with corresponding sequence diagrams, data flow diagram, database schema, and class diagrams. The final section of the document contains a Product Specification Approval section where upon approval the development team will sign and use this document in the implementation of the project.

2 General Overview and Design Guidelines/Approach

2.1 Assumptions / Constraints / Standards

2.1.1 Assumptions

It is assumed that a user will have access to a computer that has an internet connection. This internet connection needs to be able to support a least a minimum of 1 megabits per second [6]. This type of internet connection will give the user an optimal experience navigating the site. Another assumption is that the Raspberry Pi has access to either a Wifi or ethernet connection to be able to communicate with the server.

2.1.2 General Constraints

The use of the Raspberry Pi is very substantial to this project, as it records the weather using the sensors attached to them. One potential constraint to the use of a Raspberry Pi is environment. The Raspberry Pi has little to no protection against water and dirt while unprotected and thus must be kept in a clean and dry area.

Another constraint of the Raspberry Pi is the portability. Each station will only be able to run on battery for approximately 20 hours of run time with a 20,000 mAh battery pack. Once the battery is depleted, the battery would need to be recharged and the Raspberry Pi would have to be plugged into a separate power supply in the meantime. With a battery pack attached to the Raspberry Pi, there is also a constraint of the battery overheating in the sun which could cause the battery to burst. To keep the battery from getting too hot, the pack must be kept away from direct sunlight.

Another constraint for this project is internet connection. For this device to work, the Raspberry Pi needs an internet connection. This connection is required for data to be offloaded from the Raspberry Pi to the server.

2.1.3 Technical Constraints

API	Constraint
Google Maps	<ul style="list-style-type: none">• 25,000 Javascript API calls per day• 2,000,000 map loads per day
Open Weather Map	<ul style="list-style-type: none">• 60 API calls per minute
Gmail	<ul style="list-style-type: none">• 1,000,000,000 quota units per day or 2 emails per second

3 Architecture Design

3.1 Hardware Architecture

This web application will be hosted on DigitalOcean's cloud server service. This service will provide the application with 1GB of RAM, a 25GB SSD, and 1TB of transfer data. This application is connected to the user and to the Raspberry Pi's via HTTPS internet connection. The database for this project will be hosted together with the server.

Communicating directly to the server will be a Raspberry Pi. The version of Raspberry Pi to be used is the three model B which includes a quad core 1.2GHz broadcom BCM2837 64 bit CPU, 1GB of RAM, BCM4348 on board wireless LAN, and 40-pin extended GPIO [1].

Weather sensors will be attached directly to the Raspberry Pi's 40-pin extended GPIO. To measure temperature and humidity, a Gowoops 2 PCS DHT22 sensor module will be used. This module operates at a voltage of 3-5.5 volts. This module can measure temperatures in the range of -40 degrees fahrenheit to 176 degrees fahrenheit and humidity in the range of 0 to 100 percent relative humidity [3]. To measure pressure an Adafruit BMP280 I2C pressure sensor will be used. This sensor can measure pressure in the range of 300 to 1100 hPa [2].

In addition to the weather sensors, attached to the Raspberry Pi via USB port will be a GPS dongle with a Chip u-blox 7020 chip that will transmit data on the current longitude and latitude of the Raspberry Pi.

To store the operating system and data, a SanDisk Ultra 32GB microSDHC card is going to be utilized. This card will be plugged into the Raspberry Pi's micro sd port and can support transfer speeds of up to 80 MB/s [5].

The power supply the Raspberry Pi will use is a 20000 mAh Anker PowerCore II. This battery pack is plugged into the Raspberry Pi's micro usb port and can supply up to 2.4 amps of power at one time [4].

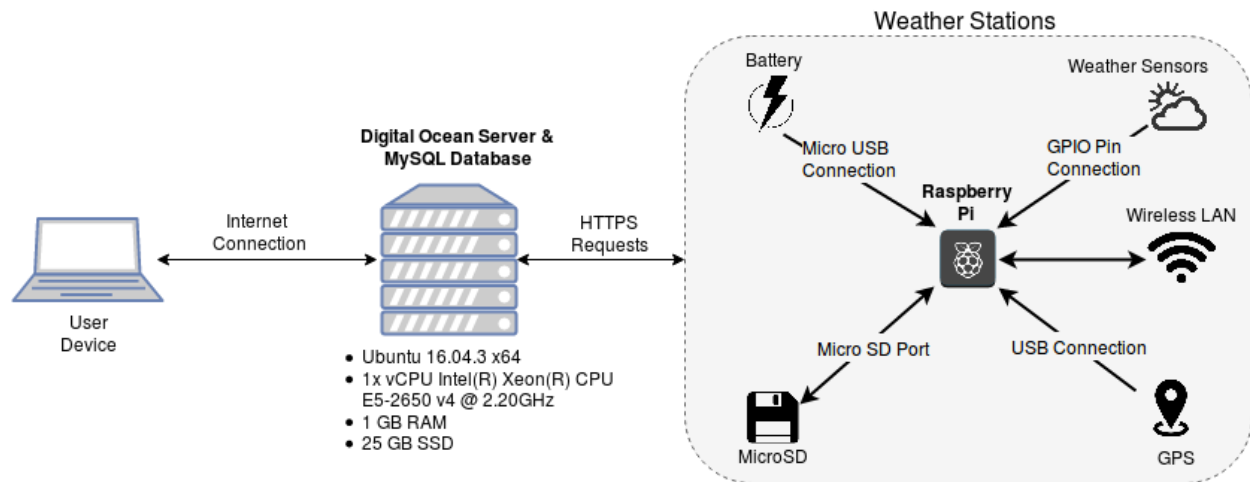


Figure 3.1.1 Hardware Architecture

3.2 Software Architecture

The software architecture for our web application is broken up into three components. There is a front end layer, a back end layer, and a Raspberry Pi layer. The front end layer uses a combination of HTML, CSS, Bootstrap, and the React.js framework to retrieve information from the backend layer and render that information to the user. The back end layer is built using a combination of Node.js, Express.js, and a MySQL Database. The backend layer uses a combination of Node Schedule and javascript scripts to execute features such as user alerts. Communication between these frontend and backend is done through API endpoints. The Raspberry Pi layer uses a series of python scripts to execute it's functions. It also communicates with the backend layer using API endpoints

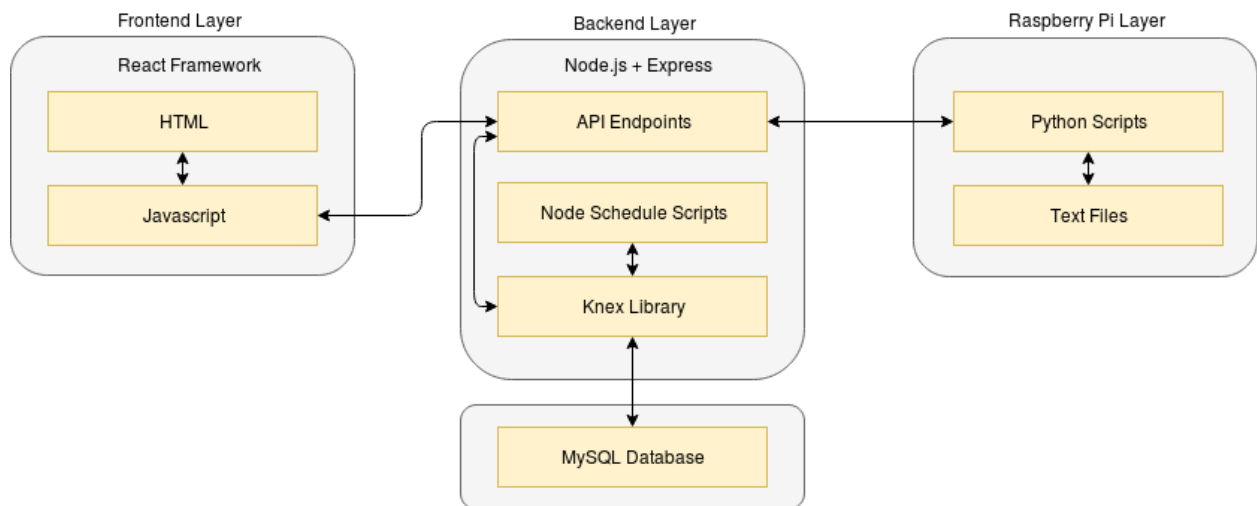


Figure 3.2.1 Software Architecture

Library	Description
React	Primary tool used for building the UI.
React-Router	Used to set up routes for pages.
Reactstrap	Bootstrap components for react.
Google Maps React	React library for displaying Google Maps instances using the Google Maps API.

D3	Javascript library for data visualization.
Babel	Javascript compiler used in this project for writing ES2015 standards.
Webpack	Asset bundler used to create lean builds for deployment.

Table 3.2.2 Frontend Layer Libraries

Library	Description
Node	Javascript runtime that uses asynchronous events.
Express	Javascript framework built upon Node.
Express Session	Express middleware used to create/store user sessions.
Knex	SQL query builder.
Passport	user authentication middleware.
MySQL	Used to create the connection between server and MySQL database.
Bcrypt	Allows the encryption/decryption of user passwords.
Node-Schedule	Used to schedule the times at which specific functions are called.
Body-Parser	Used to parse data sent to the backend via POST, GET, PUT, and DELETE requests.
Crypto	Generates random bytes to create random strings for tokens/api keys.
Nodemailer	Allows the server to send emails.

Table 3.2.3 Backend Layer Libraries

Library	Description
Adafruit_DHT	Library to parse data from the temperature/humidity sensor and make it accessible to our code.
GPSD	GPS library used to parse data retrieved from the Ublox7 USB GPS dongle.

Table 3.2.4 Raspberry Pi Layer Libraries

3.3 Security Architecture

Security is an essential part of anything related to The Department of Defense, especially when it comes to military intelligence. The Weather Station Project is no exception to this standard. To make sure that the website for the project is secured steps such as encrypting user data, using transport layer security, and csrf tokens will be implemented . For encryption the site will be securing users passwords by hashing it with Bcrypt, this includes a prefix such as \$2a\$ with a 128-bit salt and a 184-bit hash value. To ensure a secure connection to the site it will be using the TLS 1.2 making it an HTTPS to protect the information being sent to and from the site. This will be used for both users accessing the website and the Raspberry Pi weather stations sending data to the server.

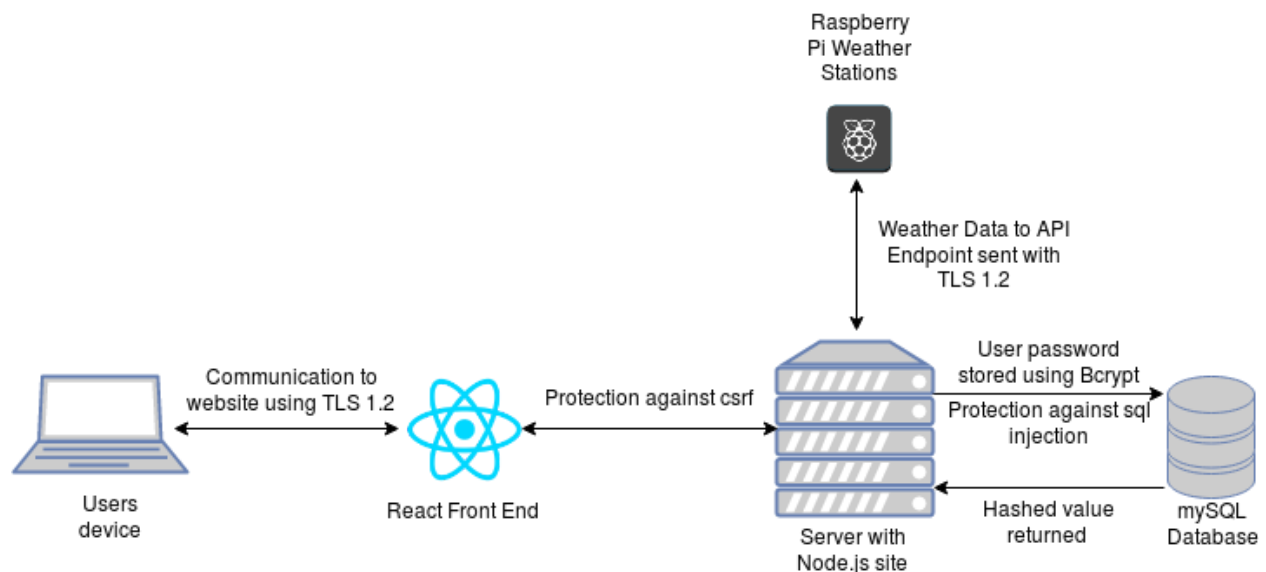


Figure 3.3.1 Security Architecture

3.4 Communication Architecture

The communication between the user's device and the front end of the web application is done through the internet. The front end of our web application will communicate with the back end through a series of API endpoints and POST/GET calls. Communication between the back end and the database will be performed using the Knex library. The Knex library will generate MySQL queries based on the parameters that it is given and return the results of the query to the backend. Communication between the back end and the Raspberry Pi's will be done via POST/GET calls.

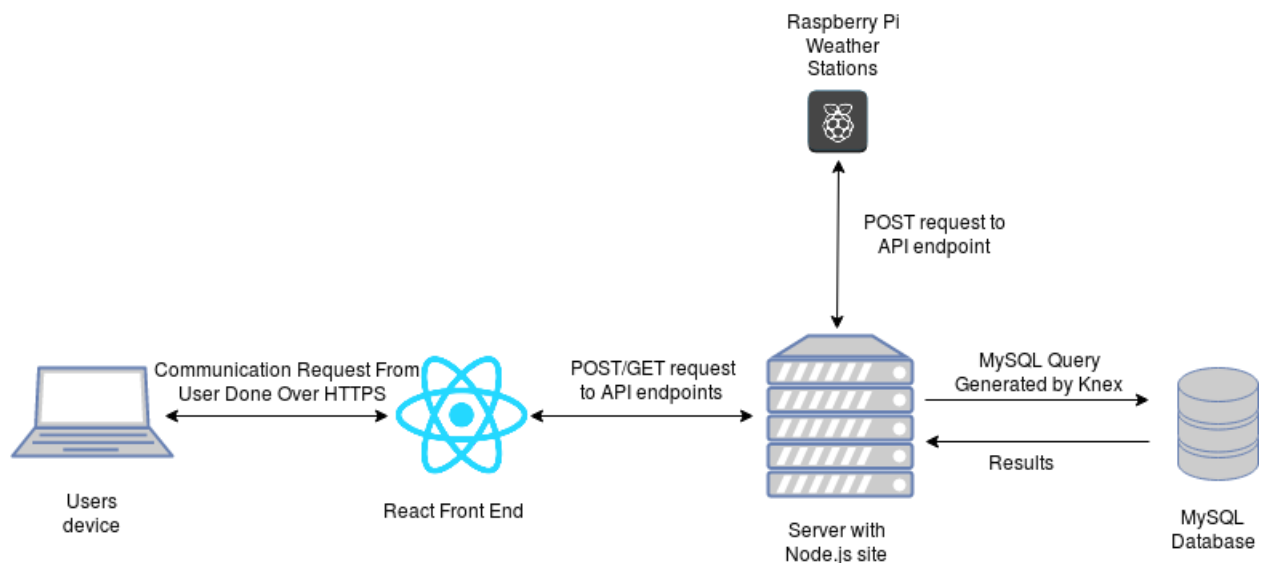


Figure 3.4.1 Communication Architecture

3.5 Raspberry Pi Data Storage Architecture

Weather data storage on the Raspberry Pi will be handled by text files. Each client will first attempt to send a POST request to the location of the server with weather data. If this request is not successfully completed, the weather data will be stored in a text file. Each day without reconnection to the server a new text file will be created. When the connection between the Raspberry Pi and the server is restored, the client will sequentially read through and send the text file data to the server for each day connection was lost in reverse chronological order. Within this text file, each column of data will be comma separated and a new line will be created for each collection time (every three seconds). An example for the data formatting is below:

```
created_at, temperature, humidity, pressure, latitude, longitude»  
2018-02-23 17:15:42, 73.21, 89.5, 987, 42.362955667, -83.072786833  
2018-02-23 17:15:45, 73.8, 89.5, 960, 42.362955667, -83.072046833  
2018-02-23 17:15:48, 72.91, 88.6, 961, 42.362954767, -83.072034833
```

Figure 3.4.1 Data Storage Architecture

4 System Design

4.1 Use-Cases

UC-1	Registers an Account
Actors:	Operator
Description:	The operator will enter their username, email address, and password into the create account page. Once the operator clicks submit, a request to create a new account will be emailed to all admins.
Trigger:	The operator clicks the submit button on the create account page
Preconditions:	The operator has entered a username that is not already in use. The operator has entered a valid email address that is not already in use. The operator has entered a password that is at least 8 characters in

	length, contains 1 letter and 1 number, and does not contain any symbols.
Postconditions:	The operator is sent an email notifying them that their account is pending admin approval.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator enters the username, email, and password for the account they wish to make and submits the form 2. The system sends an email to all admins notifying them of the pending account 3. The operator is redirected to the login page and they are notified that their account is pending admin approval.
Alternative Flow:	2a. The system notifies the operator that the information they entered is invalid and will tell them why.
Use Frequency:	Low - The operator will only visit this page when they wish to make their account for the first time.
Assumptions:	The operator is on the create account page

UC-2	Login
Actors:	Operator
Description:	The operator will enter their username and password to login to their account. The system will then verify that the operator's credentials are correct. If they are correct, the operator's session is created and they are logged in. If they are incorrect, then the operator will receive an error from the system.
Trigger:	The operator submits the form on the login page
Preconditions:	The operator has an account that has been approved by the admins.
Postconditions:	The operator is redirected to the stations page.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator enters the information for the account they wish to log into and submits the form. 2. The system verifies the operator's credentials. 3. The system creates the operator's session and redirects them to the stations page.
Alternative Flow:	3a. The system returns an error to the operator requesting they fix their login credentials.
Use Frequency:	Moderate - The operator will need to login once every time their session expires. The operator will also need to login again every time

	they logout.
Assumptions:	The operator is on the login page.

UC-3	Logout
Actors:	Operator
Description:	The operator will click their name in the top right corner of any screen. A dropdown will appear and from there they can click on logout. The system will end the operator's session and they will be redirected to the login page.
Trigger:	The operator presses the logout button.
Preconditions:	The operator is logged in.
Postconditions:	The operator's session is ended and they are redirected to the logout page.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks their username in the top right corner. 2. The system displays a drop down box containing links to profile, alerts, and logout. 3. The operator clicks logout. 4. The system ends the operator's session and redirects them to the login page.
Use Frequency:	Low - This will only occur when the operator desires to logout of their account.
Assumptions:	The operator is logged in. The operator has an internet connection.

UC-4	Password Reset
Actors:	Operator
Description:	The operator will enter their email address at the forgot password page. The system will find the account with that email address and assign it a randomly generated token. If the system cannot find the email address, the operator will receive an error. The operator is then sent an email containing a link. When the operator clicks this link, they are taken to a page where they can enter their new password. The operator must enter their password twice to confirm the new password. If both passwords match, the page submits successfully. If

	they do not match, then the operator will receive an error. Once the operator successfully submits the page, the system will find the account that has the matching token for the url. If the system successfully finds the operator's account, the new password is assigned. If it is not found, then the operator will see an error. Once the operator account's password is updated then the operator is redirected to the login page.
Trigger:	The operator submits the form on the Password Recovery page.
Preconditions:	The operator has an account that has been approved by the admins.
Postconditions:	The operator's password is updated and they are redirected to the login page.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator enters their email address on the Forgot Password page and submits the form. 2. The system generates a token and assigns it to the account that shares the entered email address. 3. The system sends the operator an email with a link to reset their password. 4. The operator clicks the link in their email and is redirected to the password reset page. 5. The operator enters their new password twice and submits the page. 6. The system matches the token from the url to the token in the database and finds the operator's account. 7. The new password the operator entered is then assigned to the operator's account. 8. The system redirects the operator to the login page.
Alternative Flow:	<ol style="list-style-type: none"> 2a. The system returns an error to the operator asking them to submit a valid email address. 5a. The system returns an error to the operator informing them that their passwords do not match. 7a. The system returns an error saying that their account could not be located.
Use Frequency:	Low - The operator will only visit this page when they forget their password and need to change it.
Assumptions:	The operator has an approved account. The operator has an internet connection. The operator can access their email and can receive emails.

UC-5	Approve Account
------	-----------------

Actors:	Admin
Description:	The admin will receive an email informing them that a new account is awaiting admin approval. The admin can click the link in the email which will take them to a page where they can click approve or deny. Once the admin has made their decision, the operator who requested the account creation will receive an email letting them know the result.
Trigger:	A operator created an account.
Preconditions:	A operator has created an account and it has not been approved by another admin.
Postconditions:	An email is sent to the operator letting them know the result of the admin's decision.
Normal Flow:	<ol style="list-style-type: none"> 1. The admin clicks the link in their email. 2. The admin clicks 'Approve' for the operator account. 3. The system sends an email to the operator letting them know the results of the admins decision.
Alternative Flow:	2a. The admin clicks 'Deny' for the operator account.
Use Frequency:	Low - This will only occur once per operator account that is created.
Assumptions:	The admin is already logged in.

UC-6	View Profile
Actors:	Operator
Description:	The operator views their profile information
Trigger:	The operator clicks the profile page link.
Preconditions:	The operator is logged in.
Postconditions:	The operator will be shown a form with their current profile information.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on their username in the right corner of the navigation bar. 2. The system displays a drop down with profile, alerts, and logout page links. 3. The operator clicks on the profile link.

	<ol style="list-style-type: none"> 4. The system redirects the operator to the profile page. 5. The system displays a form with the operator's username, email, and phone number information.
Alternative Flow:	5a. If the operator is an admin, an additional table will be displayed at the bottom of the screen which gives the admin the option to change other operator's privileges.
Use Frequency:	Moderate - This use case will only occur when a operator needs to change their credentials or an admin needs to change operator permissions.
Assumptions:	The operator is logged in.

UC-7	Edit Profile
Actors:	Operator
Description:	The operator will edit their username, email, phone number, or password on the profile page.
Trigger:	The operator clicks "Save Changes" on the profile page..
Preconditions:	The operator is on the profile page.
Postconditions:	The operator will be notified that their credentials have been saved.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator edits their username, email, or phone number in the input form on the profile page. 2. The operator clicks "Save Changes" 3. The system updates the operator's credentials 4. The system displays an alert notifying the operator that their credentials have been saved.
Alternative Flow:	<ol style="list-style-type: none"> 1a. The operator clicks "Update Password." 2a. The system displays a popup window prompting the operator to enter their current password, new password, and confirm new password. 3a. The operator enters their new password information and clicks "Close." 4a. The operator clicks "Save Changes" 5a. The system updates the operator's credentials

	6a. The system displays an alert notifying the operator that their credentials have been saved.
Use Frequency:	Low - A operator will rarely edit their profile information after creating an account.
Assumptions:	The operator is logged in and on the profile page.

UC-8	Add Custom Weather Alerts
Actors:	Operator
Description:	The operator will create alerts that will notify them via email, SMS, or the webpage based on changes in the weather.
Trigger:	The operator clicks “Add” on the alert page.
Preconditions:	The operator is on the alerts page.
Postconditions:	A new alert is displayed to the operator on the alerts page.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks the “Add” button on the alerts page 2. The system displays the add alert window which contains a form to submit the new alert details. 3. The operator selects the alert keyword, sensor type, and sensor values within the form. 4. The operator clicks “Submit” in the add alert window. 5. The system adds the alert data to the database. 6. The system updates the operator’s alert list.
Alternative Flow:	
Use Frequency:	Low - A operator will add or update their alert preferences from time to time.
Assumptions:	The operator is logged in and already on the alerts page.

UC-9	Modify Alert Methods
Actors:	Operator

Description:	The operator will modify the methods they receive weather alerts through.
Trigger:	The operator clicks one of the alert method checkboxes on the alerts page.
Preconditions:	The operator is on the alerts page.
Postconditions:	The system displays an alert saying “Alert method preferences updated.”
Normal Flow:	<ol style="list-style-type: none"> 1. The operator either checks or unchecks one of the three alert method boxes (email, SMS, webpage) at the top of the alerts page. 2. The system updates the operator’s alert preferences. 3. The system displays a notification to the operator saying “Alert method preferences updated.”
Alternative Flow:	<ol style="list-style-type: none"> 1a. The operator checks “SMS” and does not have a phone number stored in their profile 2a. The system does not update the operator’s alert preferences. 3a. The system displays a notification to the operator saying “Please add a phone number to your operator account before requesting SMS alerts.”
Use Frequency:	Low - A operator will rarely update their alert method preferences.
Assumptions:	The operator is logged in and already on the alerts page.

UC-10	Update Alert
Actors:	Operator
Description:	The operator will modify a currently existing weather alert.
Trigger:	The operator clicks on one of the alert cards on the alerts page.
Preconditions:	The operator is on the alerts page.

Postconditions:	The system updates the list of alerts to reflect the changes to the alert card.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on an individual alert card on the alerts page. 2. The system displays a pop up window containing the keyword, sensor type, and sensor values within an editable form. 3. The operator edits the keyword, sensor type, or sensor values and clicks "Save Changes" 4. The system updates the values in the database for the corresponding alert. 5. The system updates the list of alerts to reflect the changes made by the operator.
Alternative Flow:	<ol style="list-style-type: none"> 3a. The system clicks "Cancel." 4a. The system does not update the values in the database for the corresponding alert. 5a. The system closes the update alert window.
Use Frequency:	Low - A operator will rarely update a pre-existing alert.
Assumptions:	The operator is logged in and already on the alerts page.

UC-11	Delete Alert
Actors:	Operator
Description:	The operator will delete a pre-existing alert.
Trigger:	The operator clicks the "X" icon located on the right side of an alert card.
Preconditions:	The operator is on the alerts page.
Postconditions:	The alert is no longer active and is removed from the operator's alerts page.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator locates the alert card they want to delete within the list of alerts. 2. The operator clicks the "X" on the right side of the alert card 3. The system displays a confirmation window asking if the operator is sure they want to delete the alert.

	<ol style="list-style-type: none"> 4. The operator selects “yes.” 5. The system removes the alert from the database. 6. The system updates the alert list to reflect the changes in the database.
Alternative Flow:	<ol style="list-style-type: none"> 4a. The operator selects “no.” 5a. The system does not remove the alert from the database.
Use Frequency:	Low - A operator will rarely delete their pre-existing alerts.
Assumptions:	The operator is logged in and already on the alerts page.

UC-12	Promote Admins
Actors:	Admin, Superuser
Description:	An admin or superuser will update a operator’s account privileges to that of an admin.
Trigger:	The admin or superuser updates the permissions select box next to the username in the admin table.
Preconditions:	The admin or superuser is on the profile page.
Postconditions:	The admin table is updated to reflect the new operator privileges.
Normal Flow:	<ol style="list-style-type: none"> 1. The admin or superuser searches for the username they wish to update privileges for in the admin table on the profile page. 2. The system filters the table to match the entered username. 3. The admin or superuser changes the select box next to the username from “Operator” to “Admin” 4. The system displays a confirmation box asking if the operator should be promoted. 5. The admin or superuser selects “Confirm.” 6. The system sends an email to the promoted operator notifying them of their new privileges. 7. The system updates the admin table to reflect the new operator privileges.
Alternative Flow:	5a. The admin or superuser selects “Cancel” and the operator privileges are not updated.
Use Frequency:	Low - An admin or superuser will rarely change various operators’ permissions.

Assumptions:	The operator is logged in and has either admin or superuser permissions.
--------------	--

UC-13	Demote Admins
Actors:	Superuser
Description:	The superuser will change an admin back to a regular user.
Trigger:	The superuser updates the permissions select box next to the username in the admin table.
Preconditions:	The superuser is on the profile page.
Postconditions:	The admin table is updated to reflect the new user privileges.
Normal Flow:	<ol style="list-style-type: none"> 1. The superuser searches for the username they wish to update privileges for in the admin table on the profile page. 2. The system filters the table to match the entered username. 3. The superuser changes the select box next to the username from "Admin" to "User" 4. The system displays a confirmation box asking if the operator should be demoted. 5. The superuser selects "Confirm." 6. The system updates the admin table to reflect the new user privileges.
Alternative Flow:	5a. The superuser selects "Cancel" and the user privileges are not updated.
Use Frequency:	Low - A superuser will rarely demote admins.
Assumptions:	The admin is logged in and has either superuser permissions.

UC-14	Receive Weather Alerts
Actors:	Operator
Description:	The operator will receive weather alerts based on the triggers they set in UC-8. The operator will be able to sign up for SMS, email, or webpage alerts. When one trigger from the operator's alerts is activated, the operator will receive an alert in each form they have opted in for.

Trigger:	One or more of the operator's weather alert triggers has been activated.
Preconditions:	The operator has an activated account, they have set at least one customer alert trigger, and they have signed up for at least one method of receiving alerts (SMS, email, or webpage)
Postconditions:	The operator has received an alert for each method they have opted in for. The alert repeats the alert trigger back to the operator and gives the current weather for that station.
Normal Flow:	<ol style="list-style-type: none"> 1. The system tests each operator's alert triggers based on historical data stored from the weather stations. 2. The system matches a operator's alert trigger with historical weather data. 3. The system looks at which alert types the operator has activated, and sends the alert(s) through each method. 4. The operator receives alerts via each method they opted in for. Each alert reads the trigger that the operator initially set and provides the historical data that triggered the alert in a list format.
Alternative Flow:	<ol style="list-style-type: none"> 3a. The system is unable to find any alert types that the operator has activated. 4a. The operator does not receive any alert.
Use Frequency:	Moderate - weather alerts will be sent out depending on the weather, but they will be sent out to many operators and in multiple different formats.
Assumptions:	There is an active internet connection, the operator already has an activated account, and they have signed up for at least one alert trigger and one alert type.

UC-15	View Stations
Actors:	Operator
Description:	The operator will view a list of all currently or previously connected stations. Each connected station will be a card that displays temperature, humidity, pressure, data retrieval time, and a connection quality indicator. If the station is disconnected, the quality indicator will be red and the station will be moved to the bottom of the list. If it is a green plug icon the station is connected and sending data every five seconds.
Trigger:	The operator navigates to the stations page.

Preconditions:	The operator is logged into their account and there is at least one connected station.
Postconditions:	The operator will see a list of all currently or previously connected stations along with each stations' most recent weather data.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator arrives at the stations page 2. The system retrieves a list of all stations. 3. The system retrieves visibility, wind speed, and wind direction from Open Weather Map for each station. 4. The system displays a list of all stations. Each weather station is displayed as a card which shows temperature, humidity, pressure, name, data retrieval time, and a connection quality indicator. The card will also display visibility, wind speed, and wind direction.
Alternative Flow:	<ol style="list-style-type: none"> 2a. The system retrieves no stations. 3a. The system does not attempt to retrieve Open Weather Map data. 3b. The system receives an error while attempting to get Open Weather Map data for its latitude and longitude. 4a. The system displays an alert that reads "There are no currently connected stations" 4b. The system displays a list of all stations. Each weather station is displayed in a card which shows temperature, humidity, pressure, name, uptime, and a connection quality indicator.
Use Frequency:	High - This page will be the main point of contact for logged in operators and will be used most frequently to display station data.
Assumptions:	There is an active internet connection, the operator has already successfully created an account, and their account has been approved by an admin.

UC-16	Filter Stations
Actors:	Operator
Description:	The operator will use the input box above the list of station cards to filter the list. As the operator types in the box, the list is automatically filtered to match the value being typed.
Trigger:	The operator starts typing into the filter input box on the stations page.
Preconditions:	The operator is on the stations page and there is no value typed into the input box.

Postconditions:	The list of station cards is filtered to only display the stations that match the value of the filter input box
Normal Flow:	<ol style="list-style-type: none"> 1. The operator begins typing into the filter input box above the station cards. 2. The system displays the station cards with names that match the value of the filter input.
Alternative Flow:	2a. The system displays an alert that says "No stations match the filter."
Use Frequency:	Moderate - This use case will be used occasionally when the operator wants to view a particular station or needs to filter a larger number of stations.
Assumptions:	There is an active internet connection and the operator is on the stations page.

UC-17	View Station Details
Actors:	Operator, Admin, Superuser
Description:	The operator will click on an individual station card and see additional station data such as uptime, latitude, longitude, and map location in a popup window. The operator will also see the station's current temperature, humidity, name, and pressure in this same window.
Trigger:	The operator clicks on a weather station card on the stations page.
Preconditions:	The operator is logged in and is viewing the stations page.
Postconditions:	The operator will see a popup window with additional station data such as uptime, latitude, longitude, and map location in a popup window. The operator will also see the station's current temperature, humidity, name and pressure in this same window.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on an individual weather station card. 2. The system retrieves the data for the clicked weather station. 3. The system displays a popup window with the clicked weather station's name, uptime, latitude, longitude, map location temperature, humidity, and pressure.
Alternative Flow:	3a. If the operator has admin privileges, the system displays a popup window with the clicked weather station's uptime, latitude, longitude, map location temperature, humidity, pressure, and an input box with the station name as an editable value.

Use Frequency:	Moderate - This use case will be used occasionally when the operator wants to view an individual station's location, uptime, or change the station's name.
Assumptions:	There is an active internet connection and the operator is on the stations page.

UC-18	Edit Station Name
Actors:	Admin
Description:	The admin will click a weather station card to bring up the station detail window. This window will display an input box with the current station name as value. The admin will be able to edit this value and click "Save Changes" at the bottom of the window for the changes to take effect.
Trigger:	The admin clicks on an individual weather station card.
Preconditions:	The admin clicks on a weather station card on the stations page and is currently an admin.
Postconditions:	The weather station's name is changed.
Normal Flow:	<ol style="list-style-type: none"> 1. The admin clicks on a weather station card 2. If the operator has admin privileges, the system brings up the station detail window with an input box for the station name value. 3. The admin changes the value of the station name. 4. The admin clicks "Save Changes" 5. The system updates the value of the station name in the database. 6. The system updates the stations page to display the new station name.
Alternative Flow:	<ol style="list-style-type: none"> 2a. If the operator does not have admin privileges, the system will bring up the station detail window with only text for the station name value. 3a. The operator is unable to change the value of the station name. 5a. The system displays an error saying "This station name already exists"
Use Frequency:	Low - This use case will be used rarely and will only be allowed to occur for admins.
Assumptions:	There is an active internet connection and the operator already has admin permissions.

UC-19	View Historical Data
Actors:	Operator
Description:	The operator will view a graph of historical data from both the weather stations as well as Open Weather API. The default graph displays temperature for each connected station over the last 24 hours.
Trigger:	The operator navigates to the historical page.
Preconditions:	The operator is logged into their account and there is weather data from at least one connected weather station.
Postconditions:	The operator will see a line graph containing the historical temperature data from all currently connected stations.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on the "historical" link in the navigation bar. 2. The system redirects the operator to the historical page. 3. The system displays a line graph of the historical temperature data saved for the previous 24 hours from all connected weather stations.
Alternative Flow:	3a. There is no data stored by connected stations so the system displays an alert saying "No Historical Data."
Use Frequency:	High - Besides the stations page, this will be a common page for operators to get information about previous weather for each station.
Assumptions:	There is an active internet connection, the operator has already successfully created an account, and their account has been approved by an admin.

UC-20	Filter Historical Data
Actors:	Operator
Description:	Once on the historical page, the operator will be able to click the filter button which will bring up the filter window. From this window, the operator will be able to select the type of data, the date range, and which stations they would like displayed on the line graph.
Trigger:	The operator clicks the filter button on the historical page.
Preconditions:	The operator is on the historical page and has clicked the filter button.
Postconditions:	The operator will see a line graph of historical data based on their

	selected filters.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on the filter button. 2. The system brings up the filter window. 3. The operator selects one or more of the following filters: type of data, the date range, and station names. 4. The operator clicks the save changes button. 5. The system displays a line graph based on the selected filters.
Alternative Flow:	<ol style="list-style-type: none"> 4a. The operator decides they no longer wish to filter the graph so they click cancel instead of save changes. 5a. There is no data stored by connected stations so the system displays an alert message saying "No Historical Data."
Use Frequency:	High -In order to get more historical data, the operators will be filtering the graph fairly often.
Assumptions:	There is an active internet connection, the operator has already successfully created an account, and their account has been approved by an admin.

UC-21	View Station Locations
Actors:	Operator
Description:	The operator can view the last known latitude and longitude coordinates of each station displayed on a Google Maps instance..
Trigger:	The operator navigates to the map page.
Preconditions:	The operator is logged into their account and at least one weather station has transmitted GPS coordinates.
Postconditions:	One or more pins will be displayed on the Google Maps instance based on the last known latitude and longitude of each station.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on the "map" link in the navigation bar. 2. The system redirects the operator to the map page. 3. The system displays a map of the last known locations of the weather stations.
Alternative Flow:	3a. There is no connected weather stations so the map shows no pins.
Use Frequency:	Moderate -This page will be accessed by operators looking for the last known location of a weather station.
Assumptions:	There is an active internet connection, the operator has already

	successfully created an account, and their account has been approved by an admin.
--	---

UC-22	Show/Hide Station Locations
Actors:	Operator
Description:	A list of all station names will be next to the map on the map page. operators will click the checkmarks next to the station names. As each station name is checked, the station's last known latitude and longitude coordinates will be displayed on the Google Map instance.
Trigger:	The operator clicks a checkbox next to one of the station names on the checklist.
Preconditions:	The operator is currently on the map page and at least one weather station has transmitted GPS coordinates.
Postconditions:	The Google Map instance either adds or removes a pin signifying the latitude and longitude coordinates of a single station.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator either selects or deselects the desired weather stations to display from the checklist next to the map. 2. The system draws or removes markers based on filter specifications.
Alternative Flow:	
Use Frequency:	Moderate - operators on the map page will often be checking and unchecking station names to see station locations.
Assumptions:	There is an active internet connection, the operator has already successfully created an account, and their account has been approved by an admin.

UC-23	View Individual Station's Location
Actors:	Operator
Description:	The operator will be able to view a station's last known latitude and longitude within the station detail window within a Google Maps instance. Clicking the station card brings up the map display.
Trigger:	The operator clicks on the station's card.
Preconditions:	The operator is currently on the stations page and has clicked on a

	station card to bring up its additional information.
Postconditions:	The operator will be shown the location of the weather station on a map.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on a station card. 2. The system brings up an additional information window that displays last known latitude and longitude. 3. The operator views the station's location on the map below the latitude and longitude.
Alternative Flow:	
Use Frequency:	Moderate - This use case will be used occasionally when operators are on the stations page and want to know where a station is right away. Otherwise, they will be able to navigate to the map page and find it on there.
Assumptions:	There is an active internet connection and at least one weather station has latitude and longitude data.

UC-24	Add a New Station
Actors:	Admin
Description:	To add a new station, an admin will click the 'Add Station' button on the Admin page. The system will request the admin enter a station name. Once the admin has entered a valid station name, the system will generate an API key and display it to the admin.
Trigger:	Admin clicks "Add" on the admin page under the stations section.
Preconditions:	The admin is on the Admin page
Postconditions:	The system adds the station to the stations table in the database. Admin is notified that their station is ready to use.
Normal Flow:	<ol style="list-style-type: none"> 1. The admin clicks the 'Add' button on the admin page 2. The admin will be prompted to enter a station name by the system. 3. Once the admin has submitted a valid station name, the system will generate a 20 character key from 20 random bytes and display it to the admin.
Alternative Flow:	3a. System alerts admin the station name they entered is invalid.
Use Frequency:	Low - This flow will occur every time a new station is to be added.

Assumptions:	The admin is logged in and on the admin page.
--------------	---

UC-25	Open Weather Map API
Actors:	System
Description:	The system will be retrieving wind speed, wind direction, and visibility data from Open Weather Map's API if a connected station is transmitting its GPS coordinates.
Trigger:	A weather station with latitude and longitude data sends sensor information to the server.
Preconditions:	A weather station with GPS capability has connected to the site and is transmitting its location.
Postconditions:	The system stores the wind speed, wind direction, and visibility within each row transmitted from the weather station.
Normal Flow:	<ol style="list-style-type: none"> 1. The system receives the weather station coordinates through API endpoints. 2. The system calls Open Weather Maps API based on latitude and longitude coordinates. 3. The system receives the wind speed, wind direction, and visibility in JSON format. 4. The system stores that data in the database with the weather station's sensor data.
Alternative Flow:	
Use Frequency:	High- This will occur for every station that is connected and has valid GPS coordinates.
Assumptions:	There is at least one weather station transmitting latitude and longitude coordinates.

UC-26	Save Station Data Locally
Actors:	System
Description:	When connection to the server is lost, weather data is automatically stored via text files on the Raspberry Pi. The title of each text file corresponds with the day the weather was retrieved in the format MMDDYYYY.txt.
Trigger:	Connection to the server is lost while a weatherstation is attempting

	to send data.
Preconditions:	The weather station client is installed on the Raspberry Pi
Postconditions:	Weather data is stored in a text file within the data folder at the same file directory as the weatherstation client
Normal Flow:	<ol style="list-style-type: none"> 1. System prints "Lost connection to server...storing weather data locally" to log file. 2. System generates a weather data string to write to a file. 3. System checks if the data file for today's date exists. 4. If the data file exists, append it to the end of the file.
Alternative Flow:	4a. If the data file does not exist, create the file and write the data string to the file
Use Frequency:	High - Any time the station loses internet connection or the server is down all weather data will be stored in text files.
Assumptions:	The weather station is added on the web server, has the weather station client properly installed, and has a valid API key.

UC-27	Connect Stations Automatically
Actors:	Operator
Description:	Upon turning on the station it will automatically connect to the website and begin sending weather data.
Trigger:	The station is turned on.
Preconditions:	The station has been configured correctly with either a Sense Hat or Individual sensors. The station has also installed the binary file from the website and has had its API key entered. In the binary file the setting to run the binary on start up has been set as well.
Postconditions:	When the station is turned on it will automatically connect and send weather data to the site.
Normal Flow:	<ol style="list-style-type: none"> 1. Operator plugs Raspberry Pi weather station. 2. Weather station connects to the website and is sending data.
Alternative Flow:	
Use Frequency:	High
Assumptions:	Operator has installed the weather station and connected the weather sensors correctly.

UC-28	Install new Station
Actors:	Operator
Description:	Inorder to connect a weather station it must have the client code installed on it. A new API key must be created as well to connect and name the station on the site.
Trigger:	The “Download Client” is clicked on the admin page and a station has been added by clicking the add station button.
Preconditions:	The station has been configured correctly with either a Sense Hat or Individual sensors and the station has been added to the site. Operator is accessing the site through the Raspberry Pi weather station.
Postconditions:	The station will have the client binary installed on it and will be able to connect to the website.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator Navigates to the Admin page. 2. The operator clicks the “Download Client” button. 3. The system sends the client code to the Station. 4. The Station begins downloading the client code. 5. The operatore navigates, on the Raspberry Pi, to the folder where the file has downloaded. 6. Inside the folder the operator double clicks the file named “weatherstation” 7. The client code prompts the operator for the API key 8. The operator enters the API key for the station. 9. The Station begins sending data.
Alternative Flow:	
Use Frequency:	High
Assumptions:	The operator is an admin on the site and has gone through the add a station process. The station is connected to the internet.

UC-29	Webpage Alerts Display
Actors:	Operator
Description:	Operator has the option of viewing the weather data at the time a webpage alert was triggered or dismissing the alerts.
Trigger:	Webpage alert is triggered for the operator.

Preconditions:	Alert has been created and the weather conditions to trigger that alert have been met.
Postconditions:	All alerts are marked as read. Alerts are dismissed when dismiss alert button is pressed.
Normal Flow:	<ol style="list-style-type: none"> 1. Operator clicks the webpage indicator 2. Dropdown opens and is populated with all webpage alerts 3. Operator clicks on one of the webpage alerts and views the weather information
Alternative Flow:	<ol style="list-style-type: none"> 3a. Operator clicks on dismiss alerts button 4a. All alerts are removed from dropdown and operator is notified they have no alerts.
Use Frequency:	Moderate
Assumptions:	Operator is logged in. There is at least one connected station sending weather data.

UC-30	Send Weather Data
Actors:	Weather Station
Description:	The weather station sends weather data to the website asynchronously through an API endpoint every five seconds.
Trigger:	The station is connected to the website and is running the client code.
Preconditions:	The station has been added to the website, either has the Sense Hat or Individual sensors connected to it, and has the client installed on it.
Postconditions:	The website is update every five seconds with weather data from the station.
Normal Flow:	<ol style="list-style-type: none"> 1. The weather station is turned on. 2. The weather station reads the sensor or sensors. 3. The data is sent to the website via API endpoint. 4. The new data is stored in the sites database. 5. The website updates the stations page with the new data.
Alternative Flow:	<ol style="list-style-type: none"> 3a. There is no internet connection so the data is saved to a text file. 4a. The internet connection is restored. 5a. The stored weather data is sent to the site via API endpoint.
Use Frequency:	High
Assumptions:	The client has been correctly installed on the weather station and has

	an internet connection
--	------------------------

UC-31	Average Weather Data
Actors:	Operator
Description:	The operator has the option to average multiple stations' weather data by drawing a circle on the map page. Any station markers within the circle's radius will have their temperature, humidity, and pressure averaged together and displayed in the center of the circle.
Trigger:	A circle is drawn on the map page.
Preconditions:	The operator is logged in and on the map page.
Postconditions:	A box is displayed in the center of the circle with the averaged temperature, pressure, and humidity.
Normal Flow:	<ol style="list-style-type: none"> 1. The operator clicks on the "Average Weather" button. 2. The operator clicks, holds the mouse down, and drags until the circle is of satisfactory size. 3. The system averages together the temperature, pressure, and humidity of all station markers within the circle. 4. The system displays a box in the center of the circle with the averaged temperature, pressure, and humidity.
Alternative Flow:	
Use Frequency:	Moderate: This feature will be used any time a user wants to average weather together for a certain area or compare the weather for one area against another.
Assumptions:	The operator is logged in and there is at least one station that has sent latitude and longitude coordinates.

UC-32	Historic Alerts
Actors:	Operator
Description:	The operator has the option of filtering historic alerts by alert or by date. When one of the historic alerts is clicked it will display the weather data at the time the alert was triggered in a modal.
Trigger:	The operator is on the alerts page.
Preconditions:	An alert has been triggered.

Postconditions:	The alerts rendered on the page are changed depending on how the operator has set the filters.
Normal Flow:	<ol style="list-style-type: none">1. The operator selects a day from the calendar filter2. The operator clicks on an alert and views the weather data fro that alert.
Alternative Flow:	1a. The operator selects an alert from the alert filter
Use Frequency:	Low
Assumptions:	The operator is logged in. There is at least one alert that has been triggered.

4.2 Use Case Diagrams

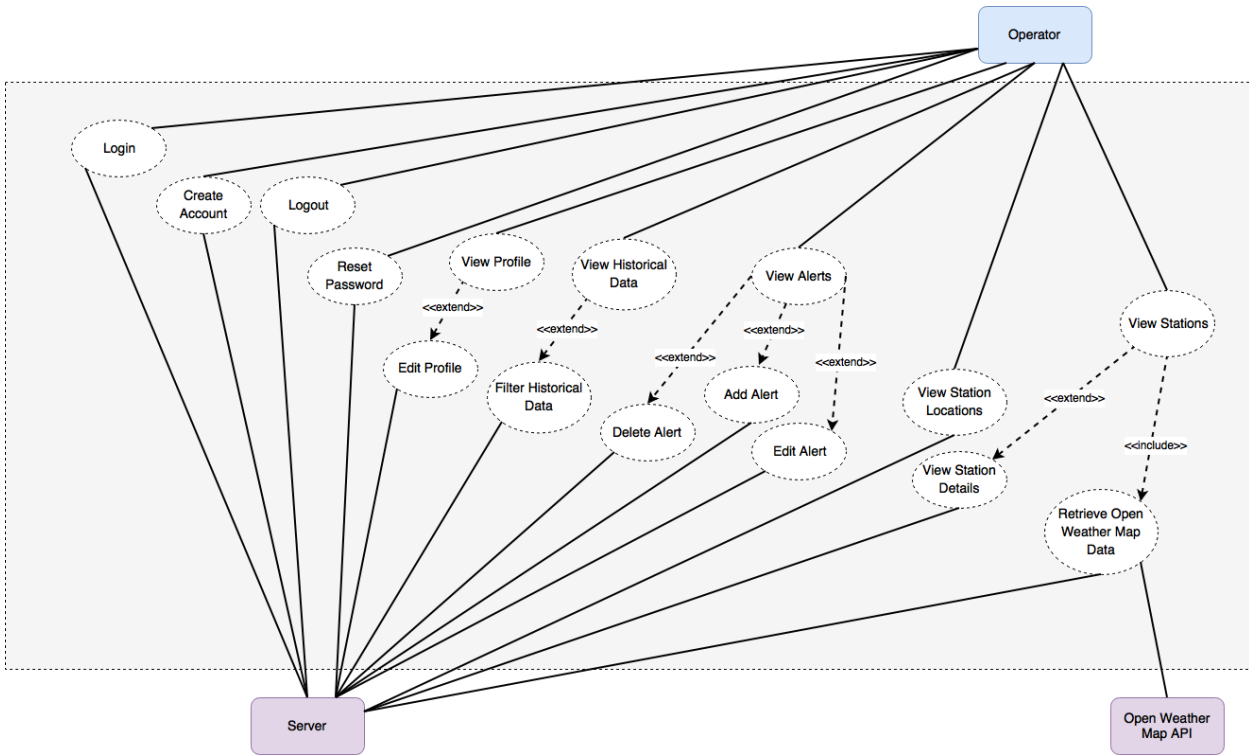


Figure 4.2.1: Operator Website Use Case Diagram

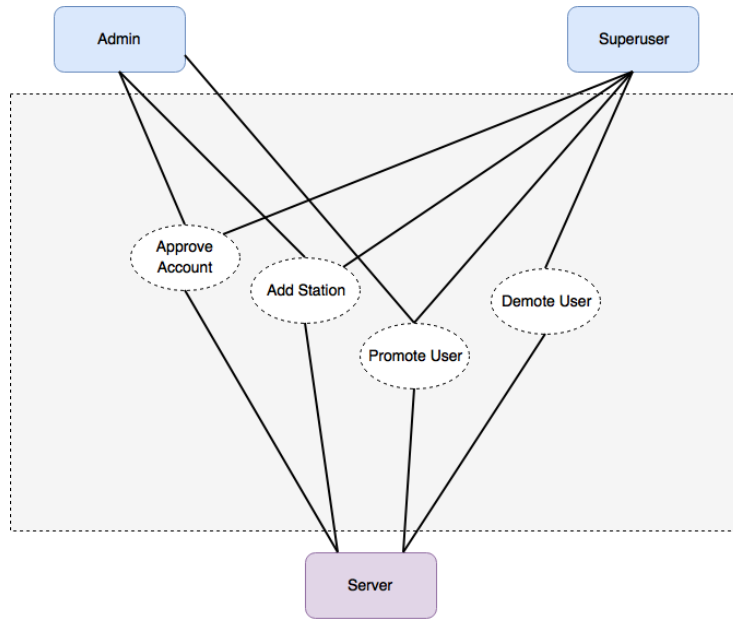


Figure 4.2.2: Admin Website Use Case Diagram

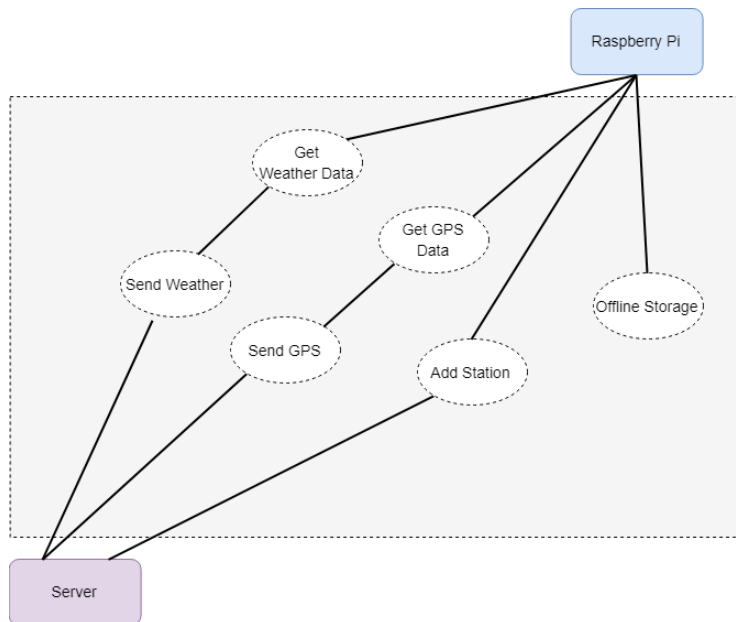


Figure 4.2.3: Raspberry Pi Use Case Diagram

4.3 Sequence Diagrams

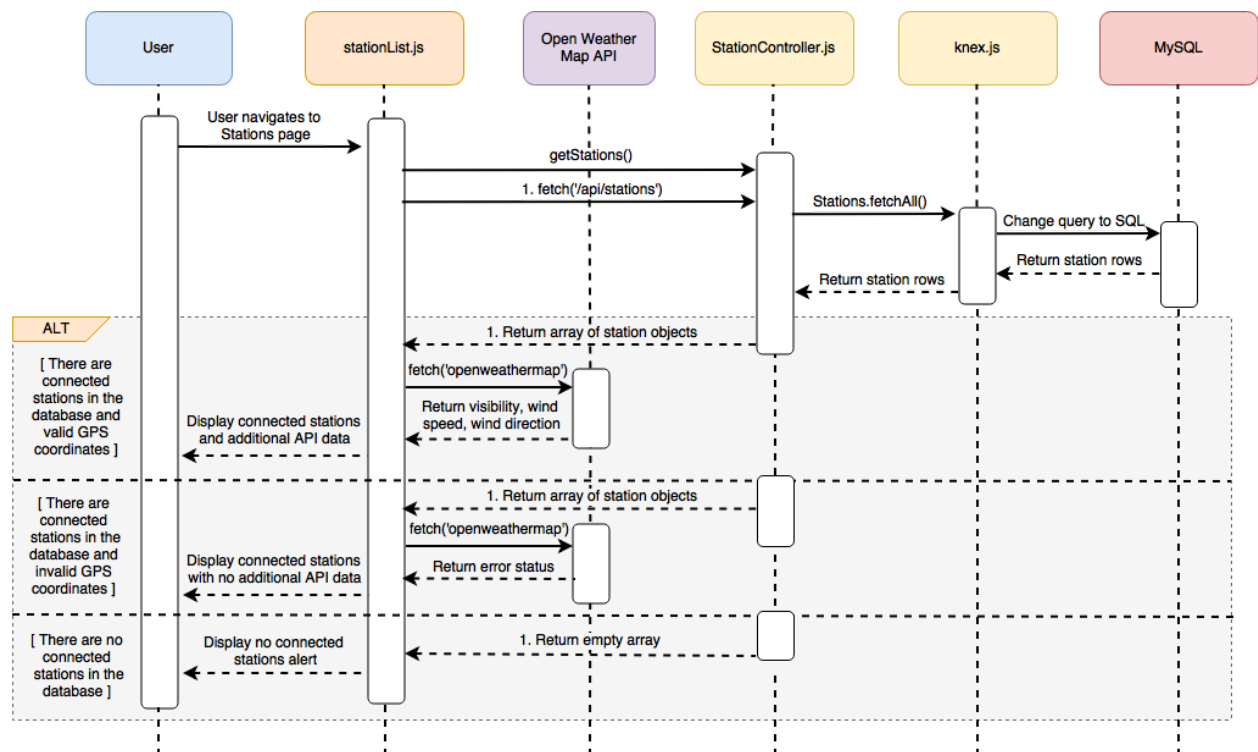


Figure 4.3.1: View Stations

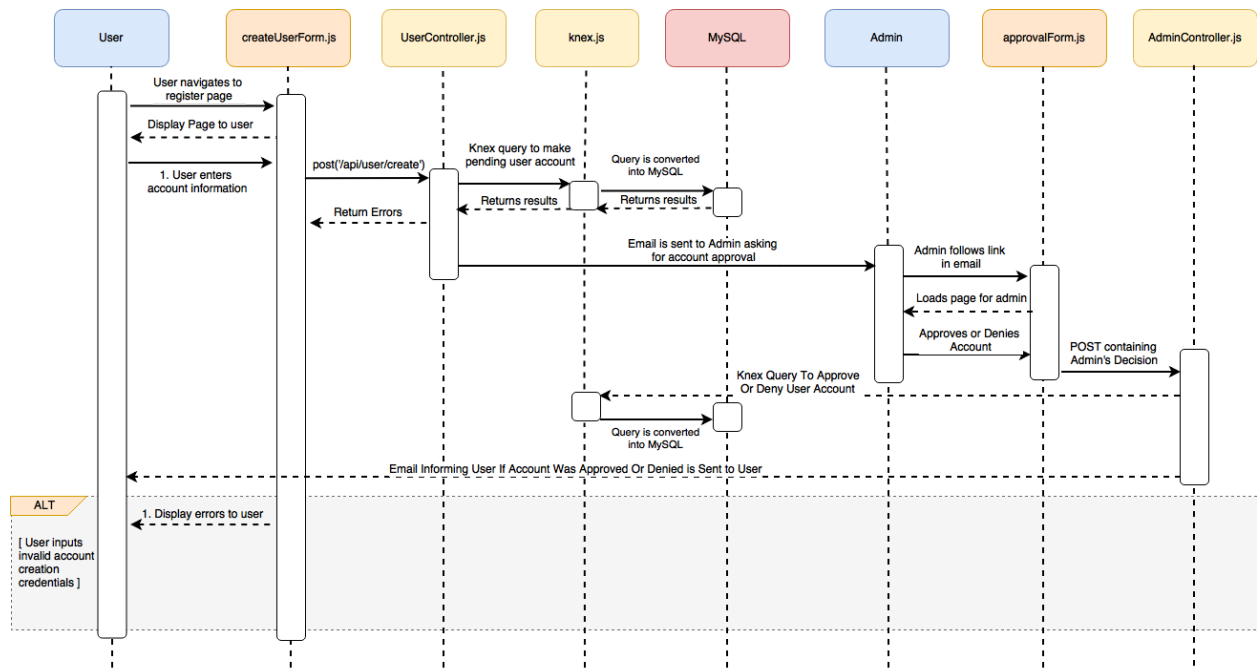


Figure 4.3.2: Register Account And Admin Approves/Denies Account

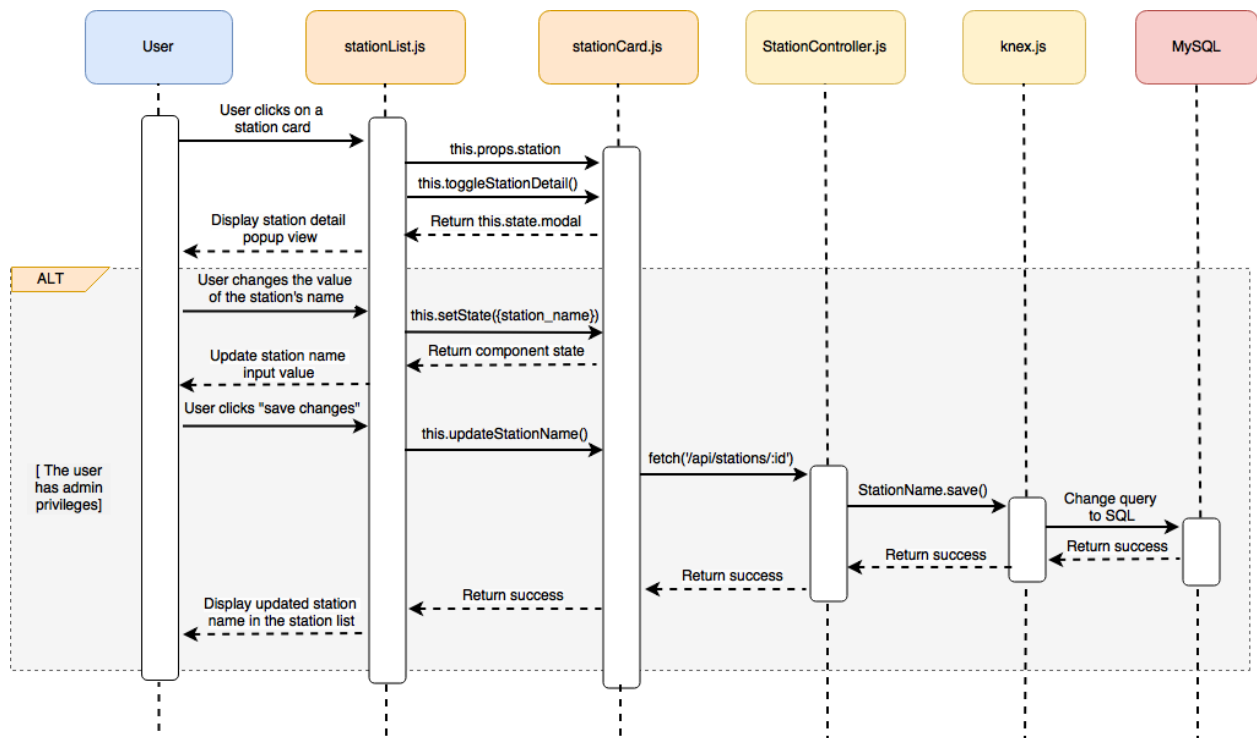


Figure 4.3.3: View Station Details & Edit Station Name

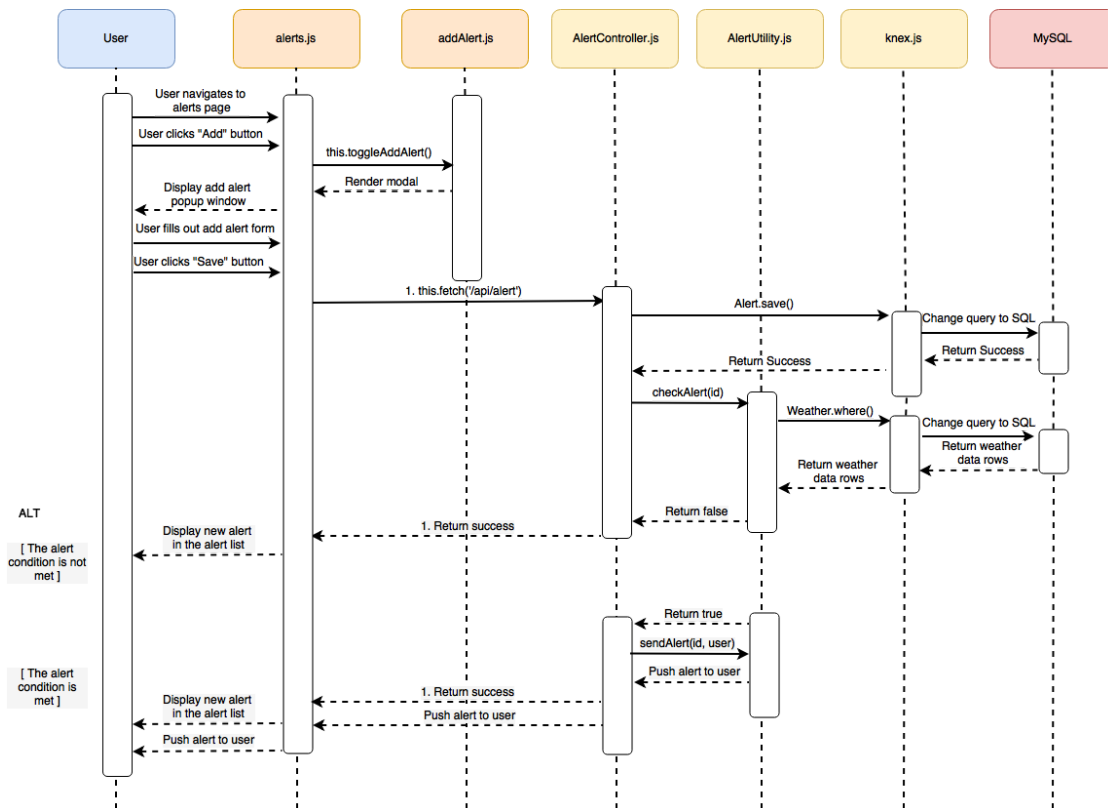


Figure 4.3.4: Add Alert Trigger

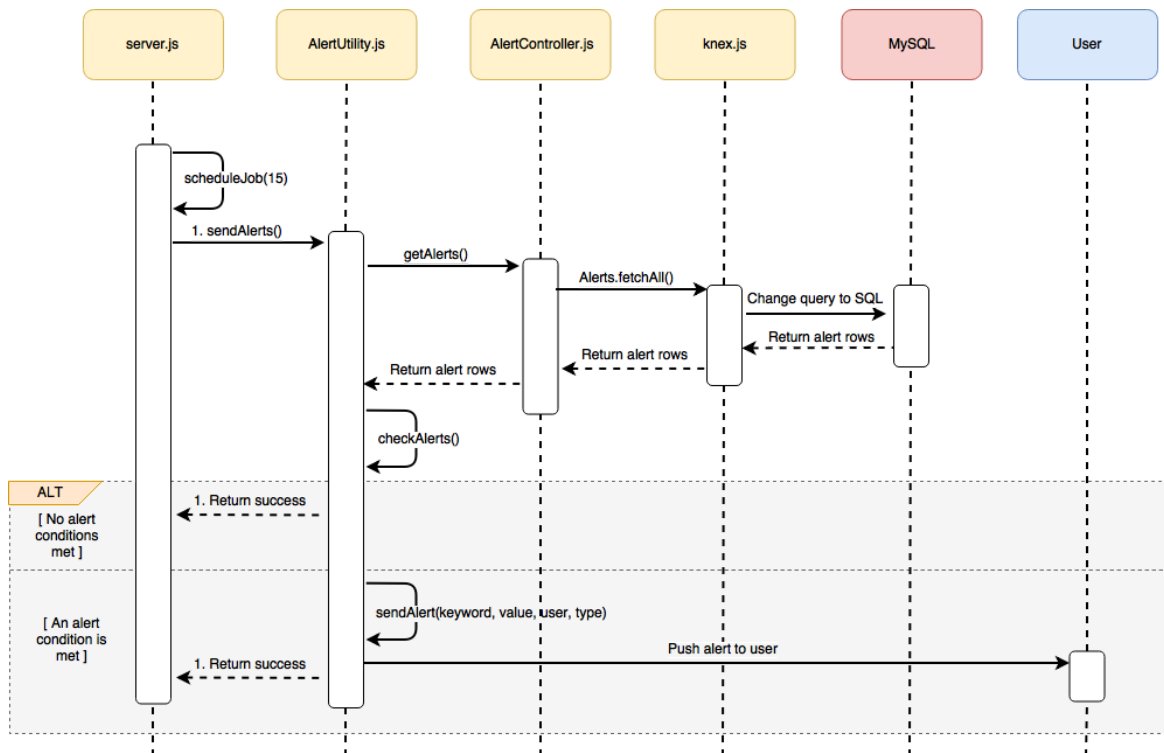


Figure 4.3.5: Receive Alert

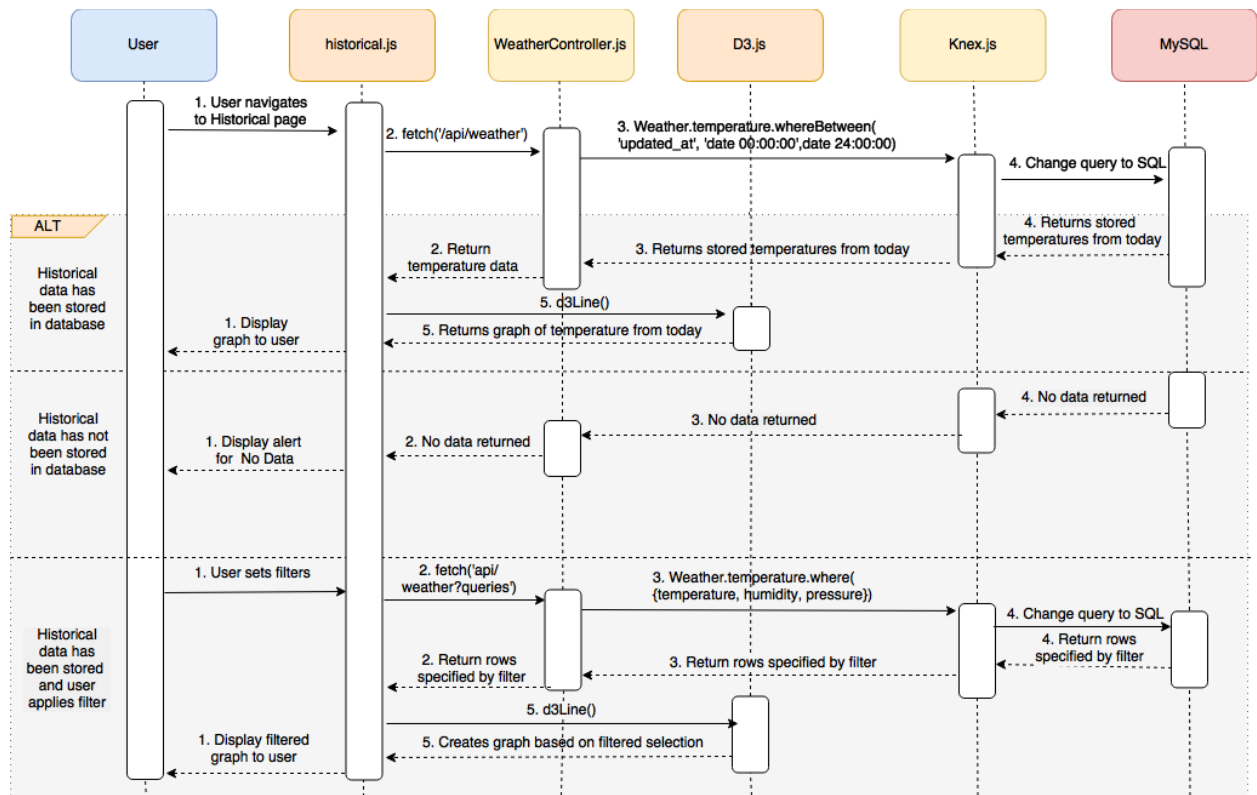


Figure 4.3.6: View Historical Data

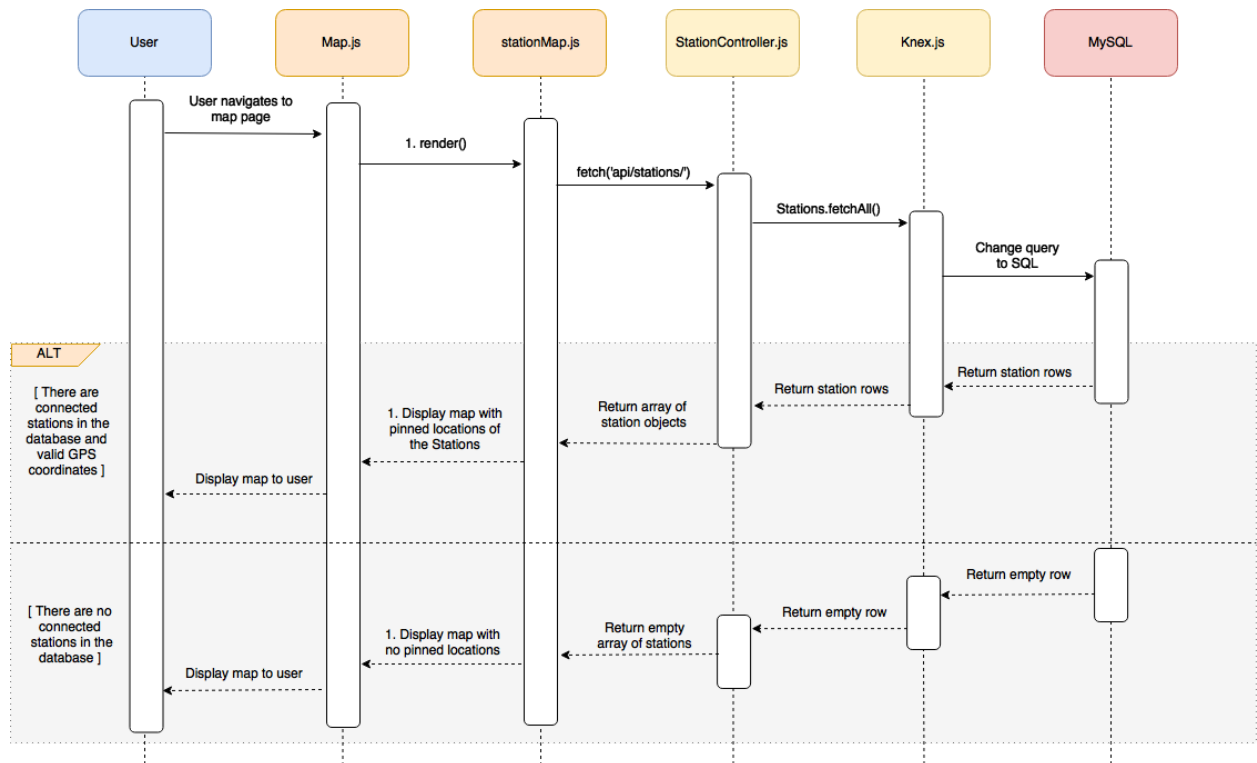


Figure 4.3.7: View Station Location

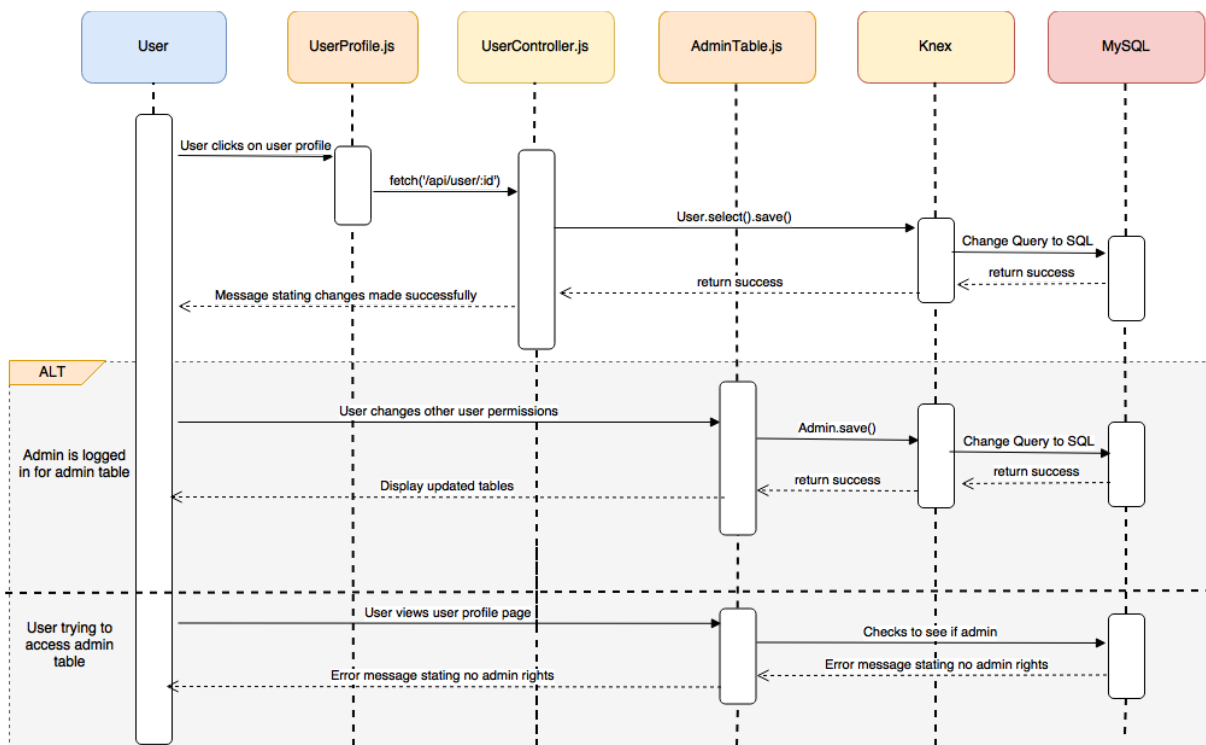


Figure 4.3.8: Update Profile and Admin Table

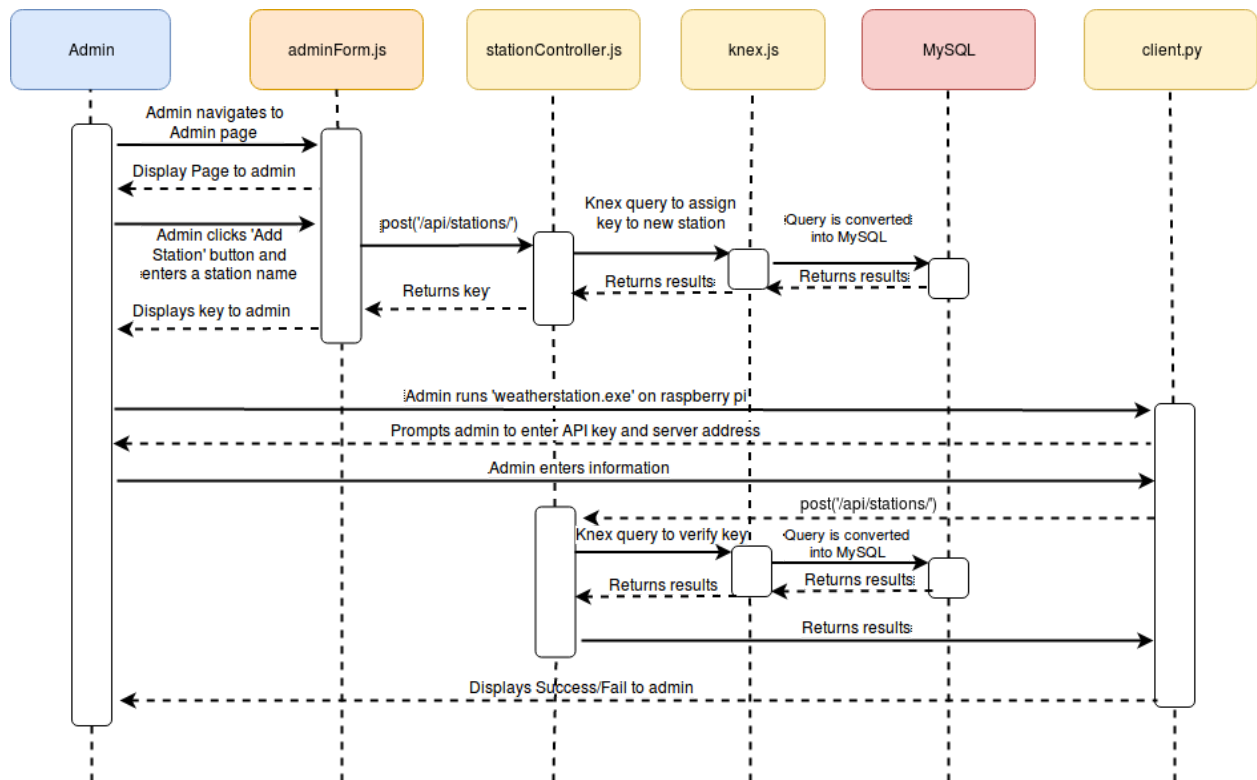


Figure 4.3.9: Admin adds new station

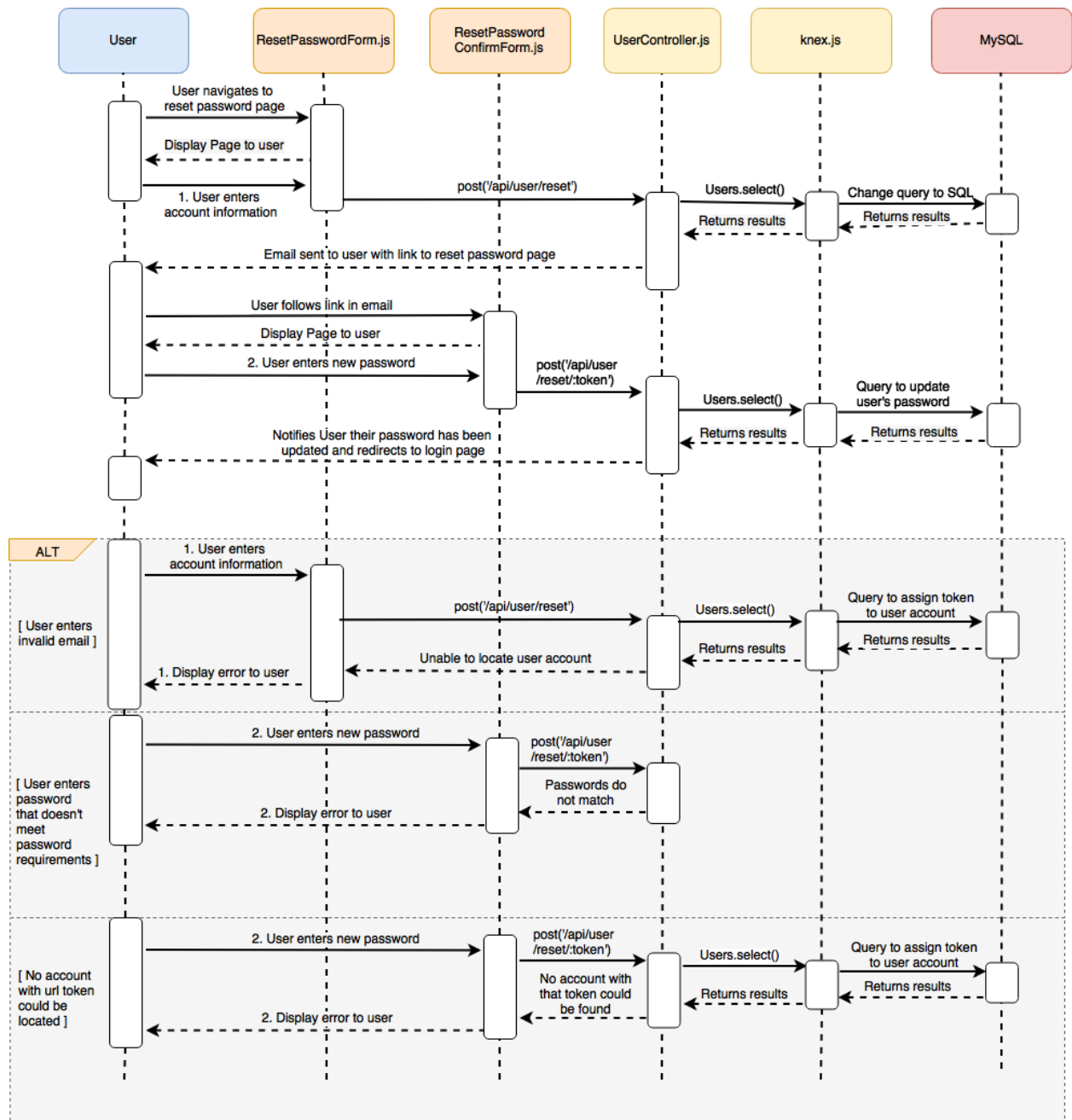


Figure 4.3.10: User Resets Forgotten Password

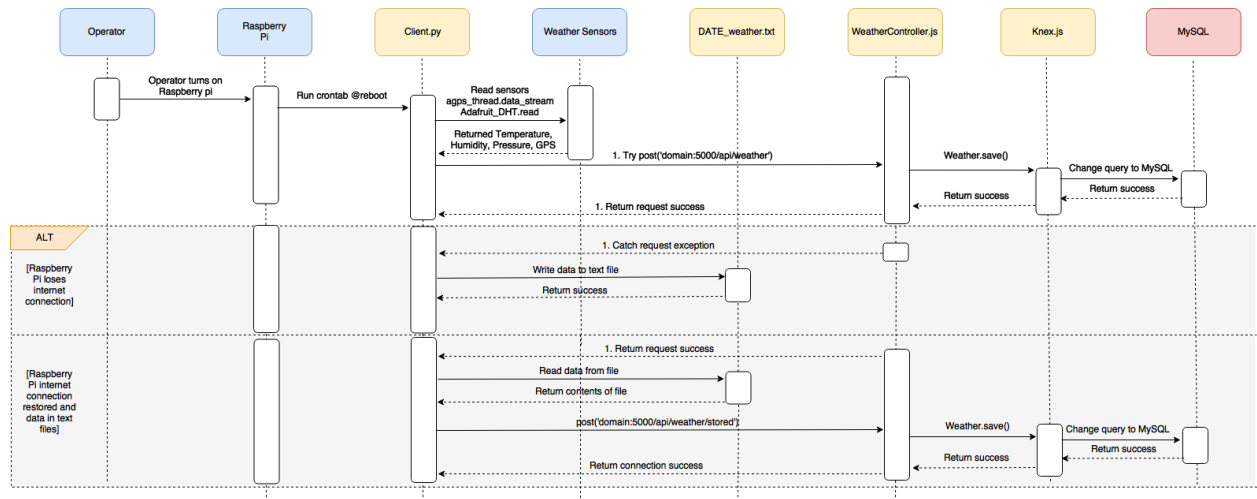


Figure 4.3.11: Raspberry Pi Sending Data

4.4 Data Flow Diagram

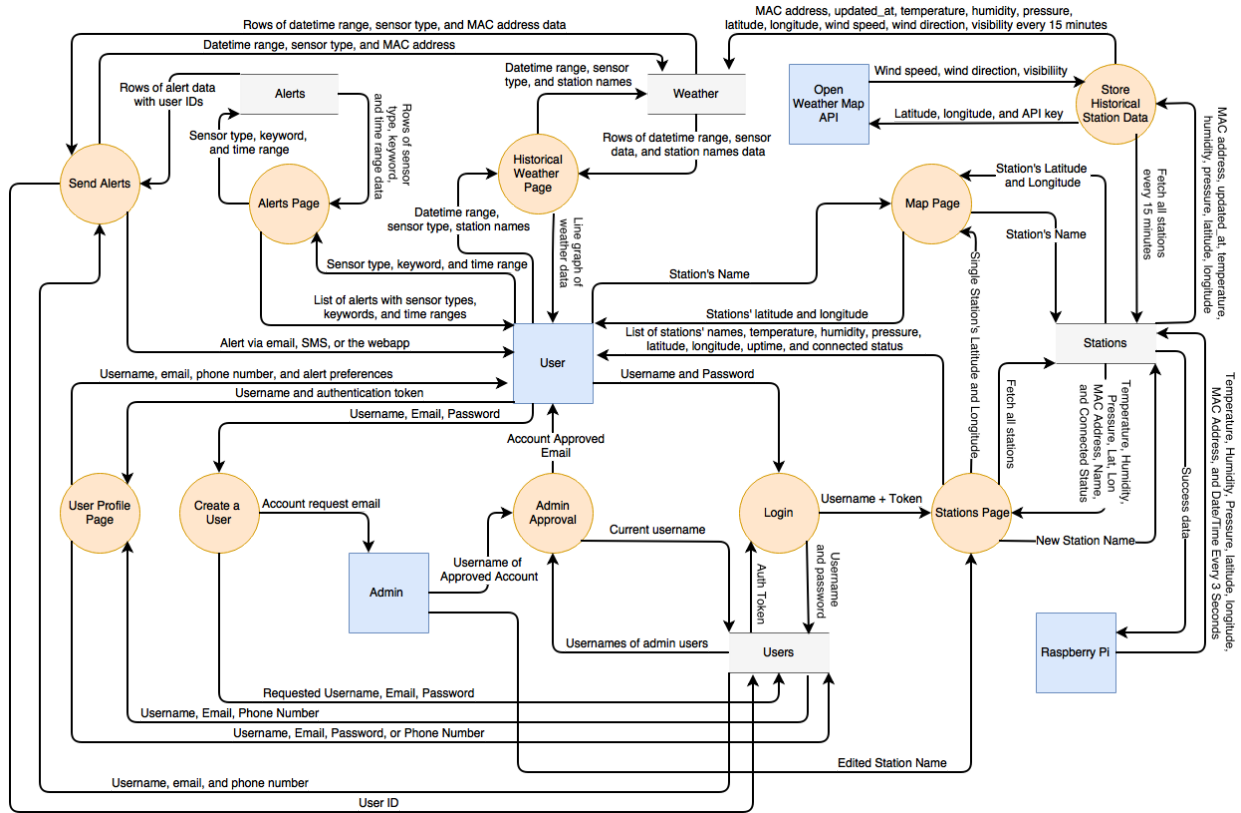


Figure 4.4.1: Data Flow Diagram

4.5 Database Design

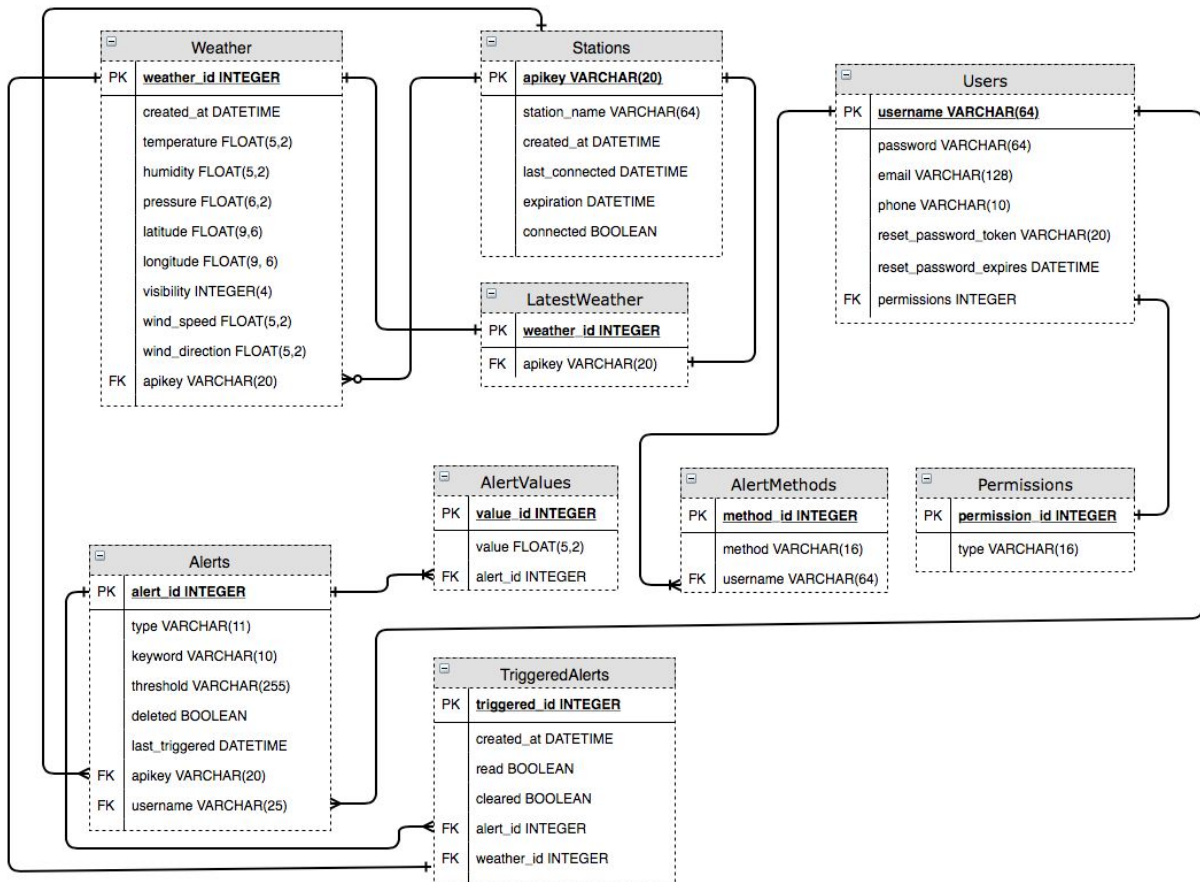


Figure 4.5.1: Database Diagram

4.6 Class Diagrams

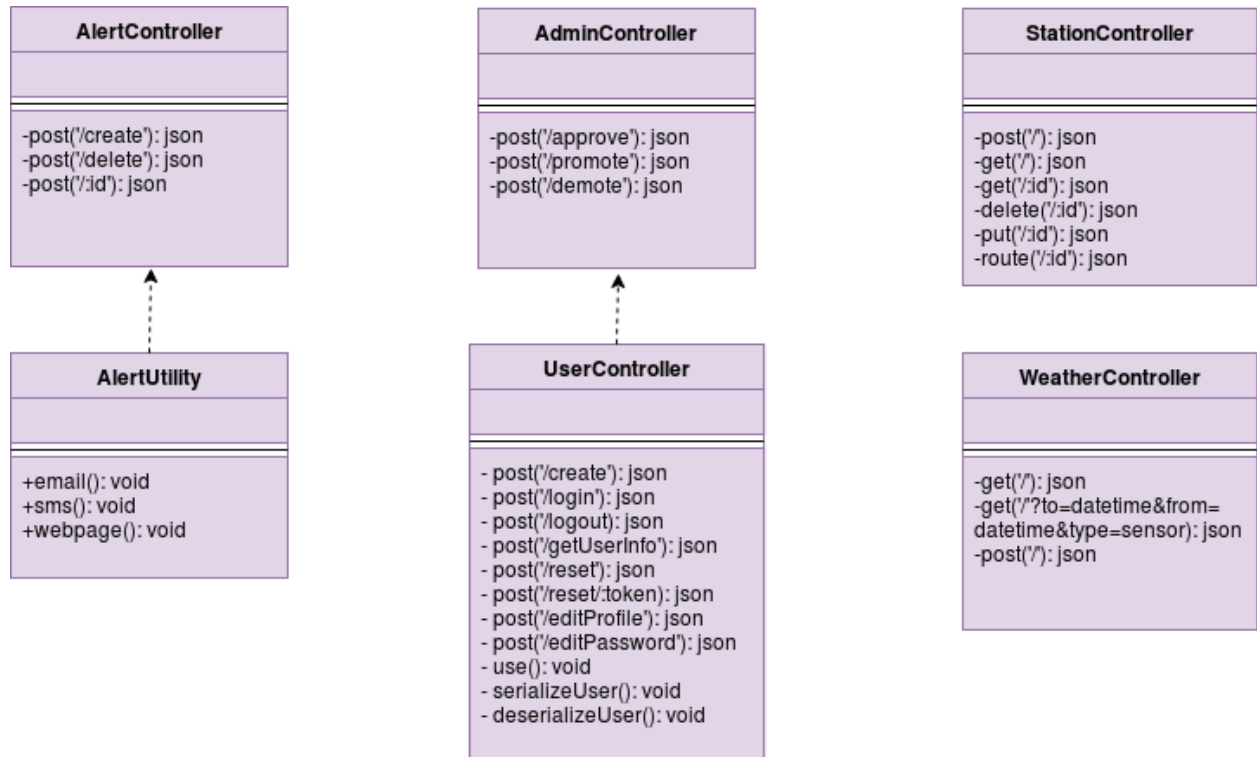


Figure 4.6.1: Controller Class Diagram

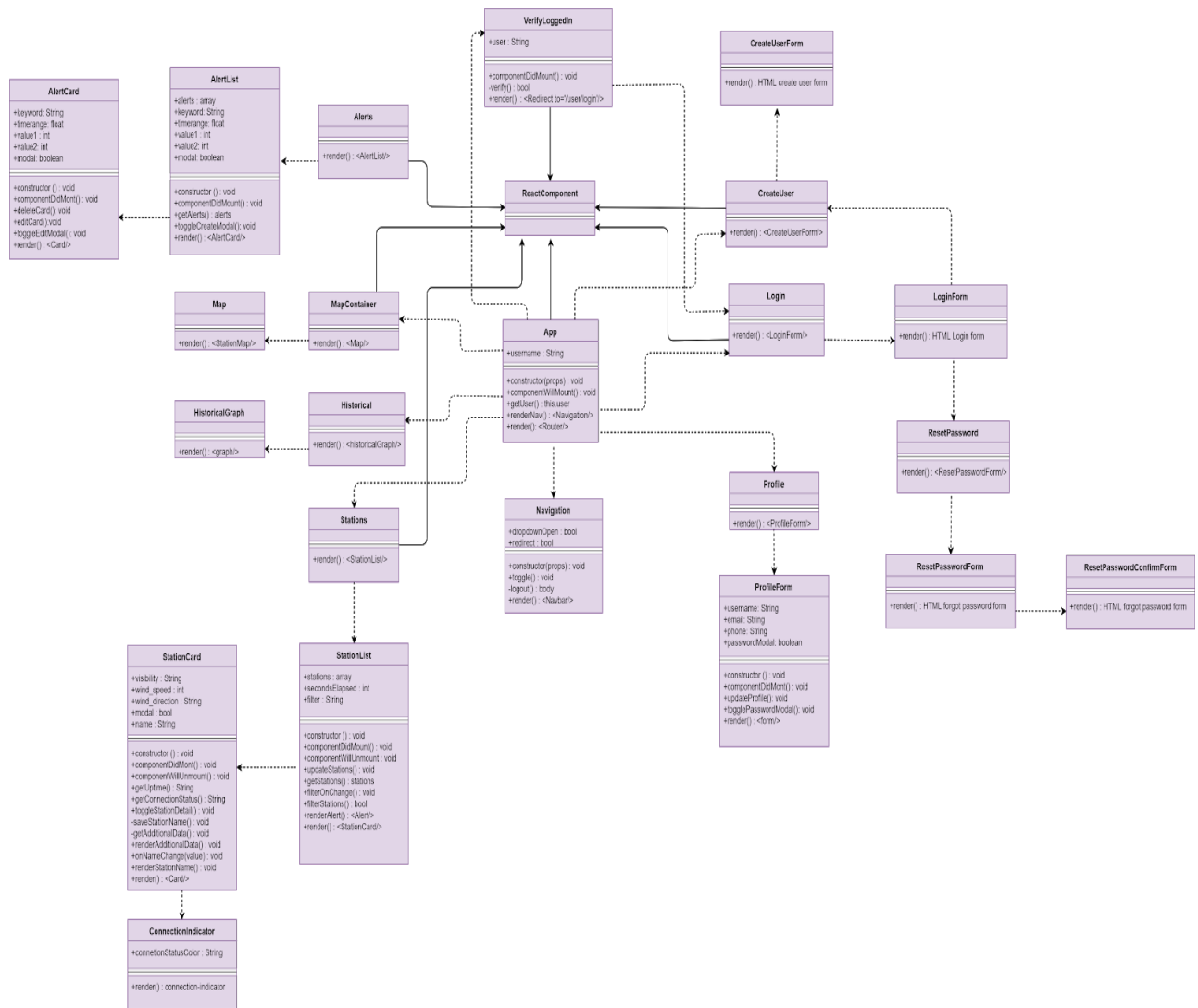


Figure 4.6.2: React Component Classes

4.7 Application Program Interfaces

Title:	Create User
URL:	/api/user/create
Method:	POST
Params:	username[string], email[string], password[string], confirmPas[string]
Returns:	response [string]

Title:	Login
URL:	/api/user/login
Method:	POST
Parameters:	username[string], password[string]
Returns:	response [string]

Title:	Logout
URL:	/api/user/logout
Method:	POST
Parameters:	username[string]
Returns:	response[string]

Title:	Get User Information
URL:	/api/alert/getuserInfo

Method:	POST
Parameters:	
Returns:	username[string], email[string], phone[string], permissions[string]

Title:	Request Forgotten Password Reset
URL:	/api/user/reset
Method:	POST
Parameters:	email[string]
Returns:	response[string]

Title:	Forgotten Password Reset
URL:	/api/user/reset/:token
Method:	POST
Parameters:	token[string], password[string], confirmPass[string]
Returns:	response[string]

Title:	Edit Profile
URL:	/api/user/editProfile
Method:	POST
Parameters:	email[string], phoneNumber[string]
Returns:	response[string]

Title:	Edit Password
URL:	/api/user/editPassword
Method:	POST
Parameters:	currPass[string], newPass[string], confirmPass[string]
Returns:	response[string]

Title:	Add Alert
URL:	/api/alert/create
Method:	POST
Parameters:	keyword[string], type[string], value1[int], value2[int], to[string], from[string]
Returns:	response[string]

Title:	Delete Alert
URL:	/api/alert/delete
Method:	POST
Parameters:	alertID[int]
Returns:	response[string]

Title:	Edit Alert
URL:	/api/alert/:id
Method:	PUT
Parameters:	keyword[string],

	type[string], value1[int], value2[int], to[string], from[string]
Returns:	response[string]

Title:	Admin Approve/Deny user
URL:	/api/admin/approve
Method:	POST
Parameters:	username[string], approved[boolean]
Returns:	response[string]

Title:	Promote To Admin
URL:	/api/admin/promote
Method:	POST
Parameters:	username[string]
Returns:	response[string]

Title:	Demote Admin
URL:	/api/admin/demote
Method:	POST
Parameters:	username[string]
Returns:	response[string]

Title:	Add Station
--------	-------------

URL:	/api/stations
Method:	POST
Parameters:	name[string], key[string], expiration[string], connected[boolean], username[string]
Returns:	response[string]

Title:	Get All Stations
URL:	/api/stations
Method:	GET
Parameters:	
Returns:	stations[array]

Title:	Get Single Station
URL:	/api/stations/:id
Method:	GET, DELETE
Parameters:	
Returns:	response[string]

Title:	Update Station
URL:	/api/stations/:id
Method:	PUT
Parameters:	updated_at[datetime], mac_address[string], temperature[float], humidity[float],

	pressure[string], latitude[string], longitude[string], wind_speed[float], wind_direction[int], visibility[int], connected[boolean]
Returns:	response[string]

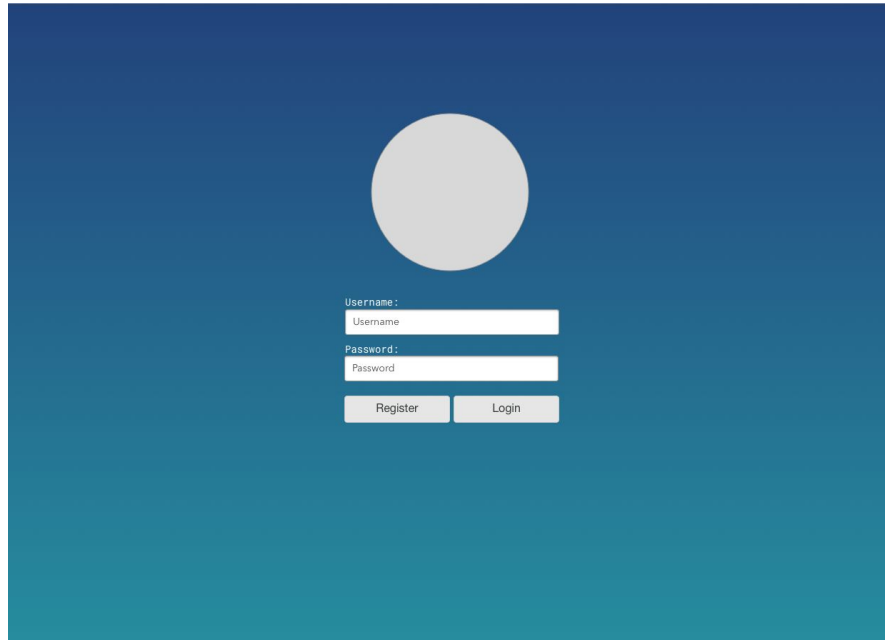
Title:	Get All Weather Data
URL:	/api/weather
Method:	GET
Parameters:	
Returns:	updated_at[datetime], temperature[float], humidity[float], pressure[string], latitude[string], longitude[string], wind_speed[float], wind_direction[int], visibility[int],

Title:	Filter Weather Data
URL:	/api/weather?to=datetime&from=datetime&type=sensor
Method:	GET
Parameters:	
Returns:	updated_at[datetime], temperature[float], humidity[float], pressure[string], latitude[string], longitude[string],

	wind_speed[float], wind_direction[int], visibility[int],
--	--

Title:	Add Weather Data
URL:	/api/weather
Method:	POST
Parameters:	updated_at[datetime], temperature[float], humidity[float], pressure[string], latitude[string], longitude[string], wind_speed[float], wind_direction[int], visibility[int],
Returns:	response[string]

4.8 User Interface Design

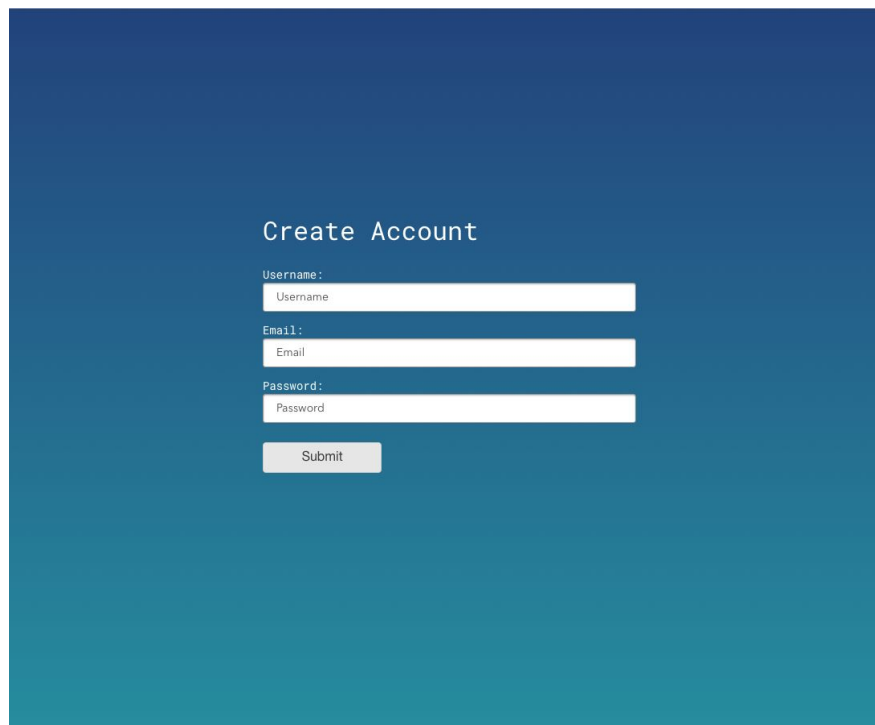


A login form on a blue gradient background. At the top center is a large, light gray circular placeholder for a profile picture. Below it, the form contains two input fields: 'Username' and 'Password', each preceded by its respective label. At the bottom of the form are two buttons: 'Register' and 'Login'.

Username:

Password:

Login View



A 'Create Account' form on a blue gradient background. The title 'Create Account' is centered at the top. Below it, the form contains three input fields: 'Username', 'Email', and 'Password', each preceded by its respective label. At the bottom of the form is a single 'Submit' button.

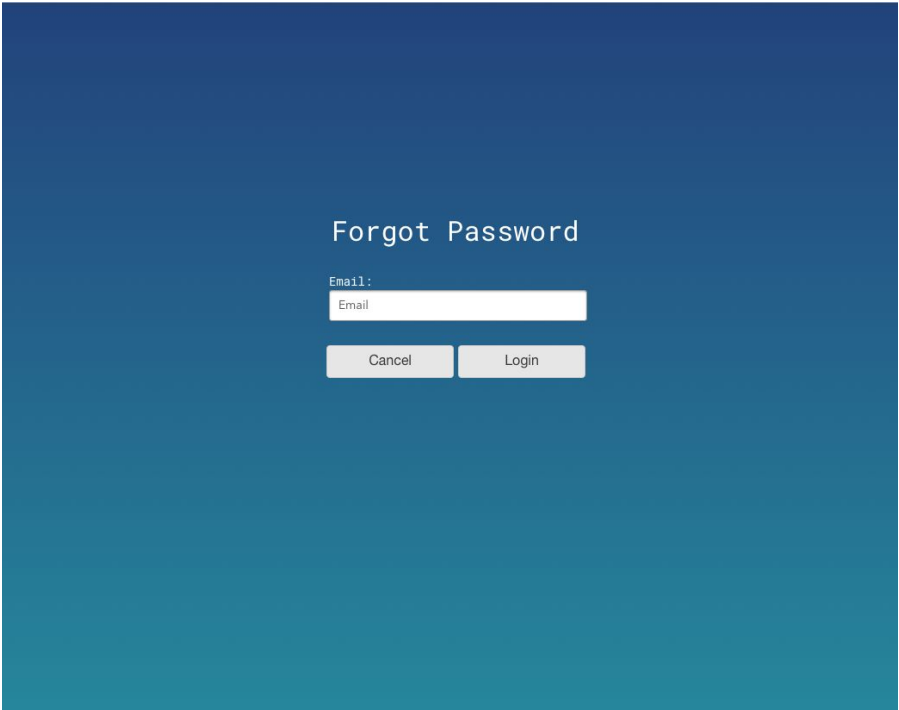
Create Account

Username:

Email:

Password:

Create Account View

A screenshot of a web form titled "Forgot Password" on a blue gradient background. The form includes a label "Email:" above a text input field containing the placeholder "Email". Below the input field are two buttons: "Cancel" and "Login".

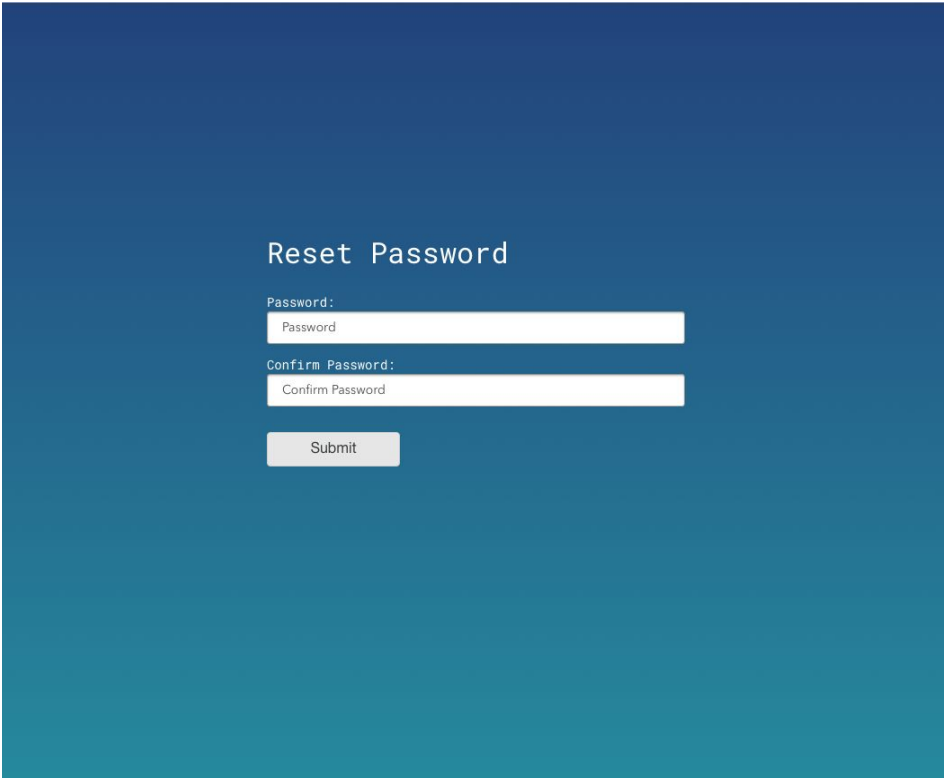
Forgot Password

Email:

Email

Cancel Login

Forgot Password View

A screenshot of a web form titled "Reset Password" on a blue gradient background. The form includes two labels: "Password:" and "Confirm Password:", each followed by a text input field containing the placeholder "Password" and "Confirm Password" respectively. Below the input fields is a single "Submit" button.

Reset Password

Password:

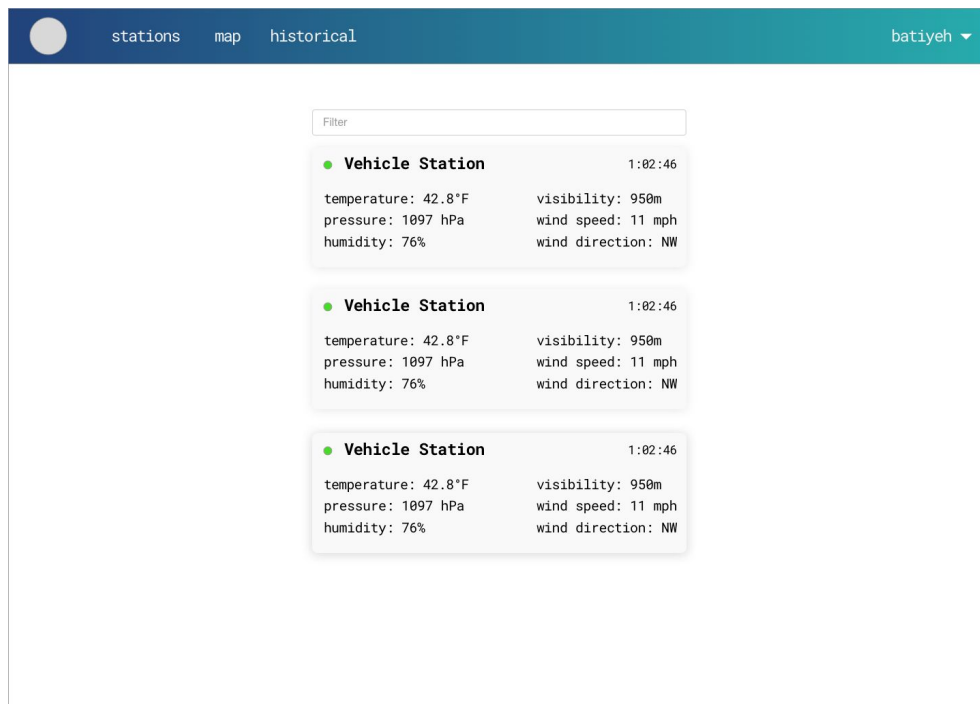
Password

Confirm Password:

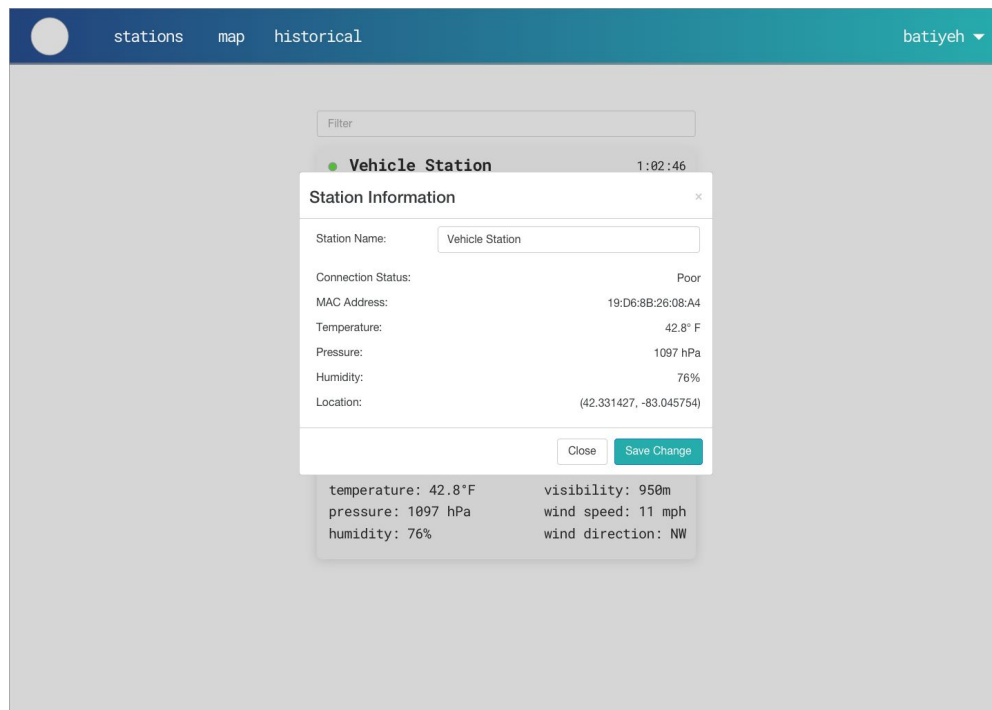
Confirm Password

Submit

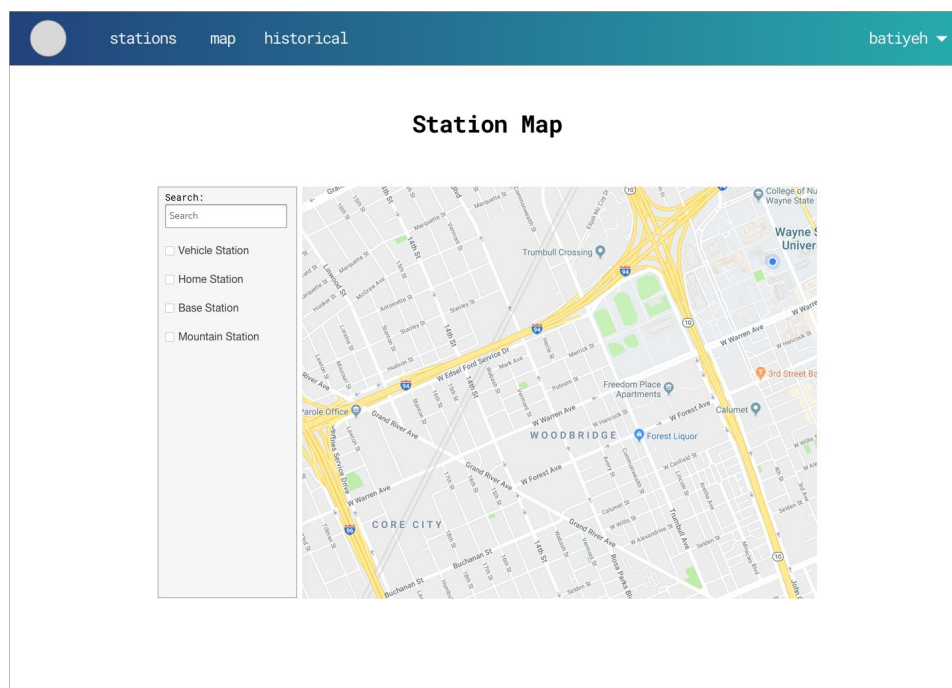
Reset Password View



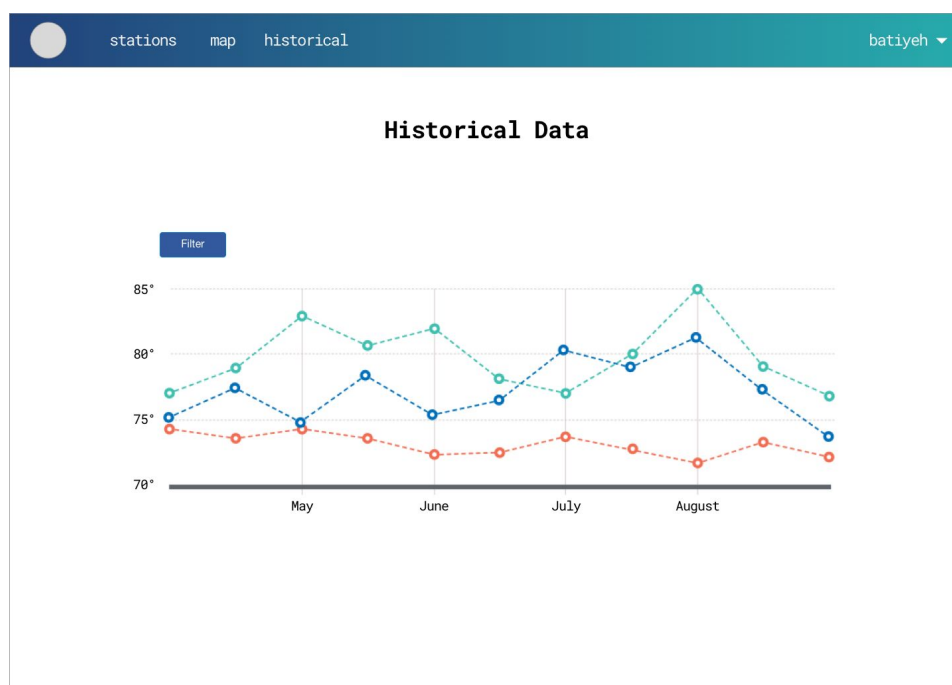
Stations View



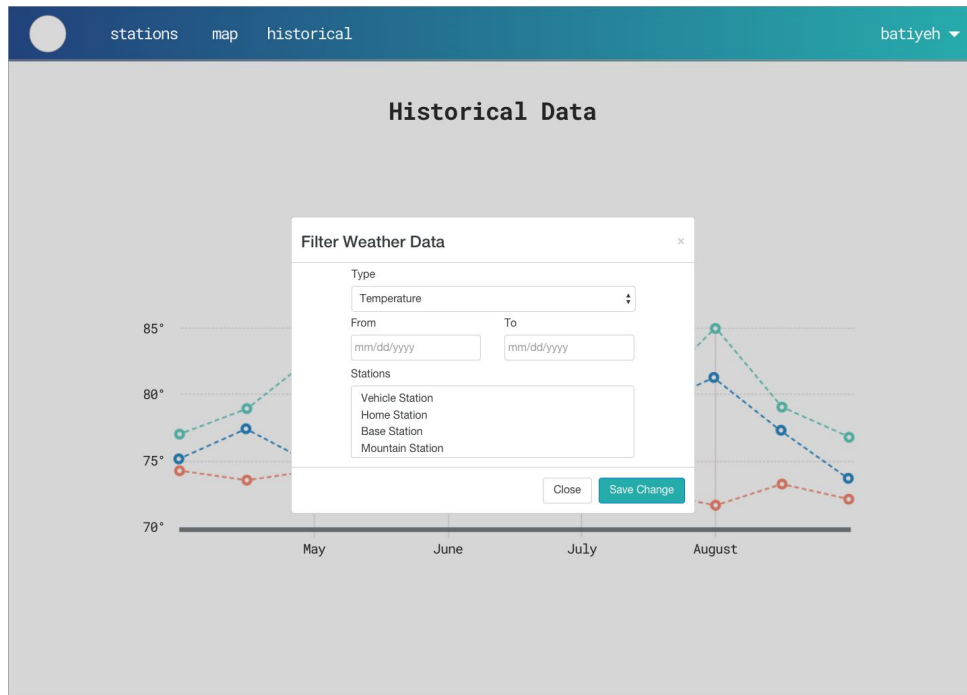
Stations Additional Information Modal



Station Map View



Historical Data View

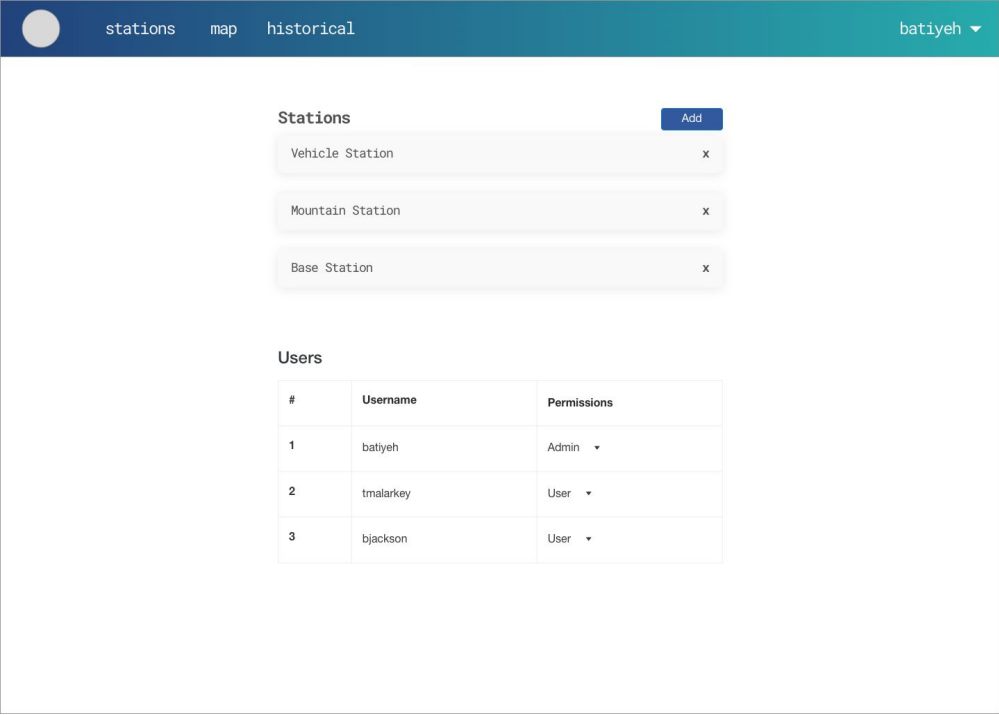


Historical Data Filter Modal

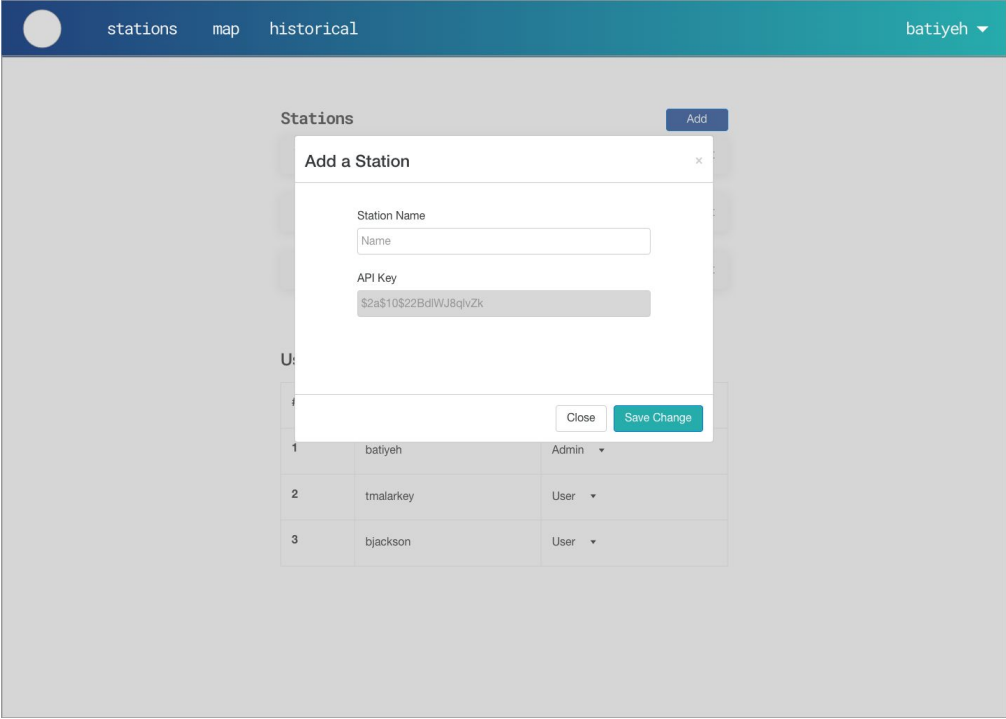
The image shows a web application interface with a teal header bar containing navigation links: 'stations', 'map', and 'historical'. The user 'batiyeh' is logged in, indicated by a dropdown arrow. The main content area is titled 'User Profile' and displays a form for editing user information. The form includes fields for Username, Email, Password, and Phone, each with a corresponding label. A 'Save Changes' button is located below the form. A dropdown menu in the top right corner shows 'profile' and 'logout' options.

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

User Profile View



Admin View



Add / Edit a Station View

stationsmaphistorical

batiyeh

☐ email☐ sms☐ webpage

send an alert when...

Add

the pressure is between 1100 hPa and 1200 hPa.x

the humidity is equal to 100%.x

the temperature has changed more than 10° in 2 hours.x

Alert Triggers View

stationsmaphistorical

batiyeh

☐ email☐ sms☐ webpage

Add Alert Trigger

Data Type

Temperature

Keyword

Between

Values

90to100

Close

Save Change

Add Alert Trigger View

5 Product Design Specification

Approval

The undersigned acknowledge they have reviewed the Weather Station **Product Design Specification** document and agree with the approach it presents. Any changes to this Requirements Definition will be coordinated with and approved by the undersigned or their designated representatives.

Signature:	_____	Date:	_____
Print Name:	_____		
Title:	_____		
Role:	_____		

Signature:	_____	Date:	_____
Print Name:	_____		
Title:	_____		
Role:	_____		

Signature:	_____	Date:	_____
Print Name:	_____		
Title:	_____		
Role:	_____		

A Appendices

Appendix A: References

Description	Location
[1] Raspberry Pi Model 3B Motherboard	https://www.amazon.com/Raspberry-Pi-RASPBERRYPI3-MODB-1GB-Model-Motherboard/dp/B01CD5VC92
[2] Adafruit BMP280 I2C Barometric and Altitude Sensor	https://www.amazon.com/Adafruit-BMP280-Barometric-Pressure-Altitude/dp/B013W0RR6Y
[3] Gowoops 2 PCS DHT22 Temperature Humidity Sensor Module	https://www.amazon.com/Gowoops-Temperature-Humidity-Measurement-Raspberry/dp/B073F472JL/ref=sr_1_3_a_it?ie=UTF8&qid=1516931662&sr=8-3&keywords=raspberry+pi+temperature+and+humidity+sensor
[4] Anker PowerCore 20100mAh Portable Charger	https://www.amazon.com/Anker-PowerCore-20100mAh-Portable-Compatible/dp/B01LQ81QR0
[5] SanDisk MicroSD	https://www.amazon.com/SanDisk-microSDHC-Standard-Packaging-SDSQU NC-032G-GN6MA/dp/B010Q57T02
[6] How to Decide What Internet Speed You Need	https://www.nerdwallet.com/blog/utilities/how-to-decide-what-internet-speed-you-need/

Appendix B: Key Terms

Term	Definition
Department of Defense	Branch of the United States government whose primary duty is supervising national security and the United States Armed Forces.
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.
Node	Node is a backend Javascript framework built to use a non blocking I/O system. 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.
CSS	Cascade style sheets, a style sheet language used to format html.
Bootstrap	Open source toolkit for front end development.
Knex	SQL query builder
MySQL	Database management system
Bcrypt	Password hashing function
CSRF	Cross-Site Request Forgery
TLS	Transport Layer Security
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.
HTTPS	Hypertext Transfer Protocol for secure communication over the internet.
SSD	Solid state drive
mAh	Milliamp Hour
hPa	Hectopascal
MB/s	Megabits per second

Appendix C: Use Case/Sequence Diagram Mapping

Requirements Name	Use Case	Sequence Diagram
Create a user Account	UC-1	4.3.2
Login	UC-2	
Logout	UC-3	
Password Reset	UC-4	4.3.10
Approve Account	UC-5	4.3.2
View Profile	UC-6	
Edit Profile	UC-7	4.3.8

Add Alert	UC-8	4.3.4
Modify Alert Methods	UC-9	
Update Alert	UC-10	
Delete Alert	UC-11	
Promote Admin	UC-12	4.3.8
Demote admins	UC-13	
Receive weather alerts	UC-14	4.3.5
View stations	UC-15	4.3.1
Filter Stations	UC-16	
View Station Details	UC-17	4.3.3
Edit Station Name	UC-18	4.3.3
View Historical Data	UC-19	4.3.6
Filter Historical Data	UC-20	
View Station Location	UC-21	4.3.7
Show/Hide Station Locations	UC-22	
Click Station Longitude/Latitude	UC-23	
Add new station	UC-24	4.3.9

Open Weather API	UC-25	
------------------	-------	--

Appendix D: Traceability Matrix

	Requirement ID	Requirement Name	Use Case ID	Priority	Test Case ID
Requirements	FR-1	User Login	UC-2	High	TC-1
					TC-2
					TC-3
					TC-4
	FR-2	Create a User Account	UC-1	High	TC-5
					TC-6
					TC-7
					TC-8
					TC-9
					TC-10
					TC-11
					TC-12
					TC-13
					TC-14
	FR-3	Password Requirements	UC-4	Moderate	TC-15
					TC-16
					TC-17
					TC-18
	FR-4	Reset a Password	UC-4	Moderate	TC-19
					TC-20
					TC-21

					TC-22
					TC-23
					TC-24
	FR-5	User Logout	UC-3	High	TC-25
					TC-26
					TC-27
					TC-28
					TC-29
					TC-30
			UC-15	High	TC-31
	FR-6	View Stations	UC-25	High	TC-123
					TC-124
	FR-7	View Station Details	UC-17	Moderate	TC-32
					TC-33
					TC-34
	FR-8	View Individual Station Location	UC-23	Low	TC-35
					TC-36
			UC-21	High	TC-37
					TC-38
					TC-39
					TC-40
					TC-41
					TC-42
					TC-43
					TC-44
	FR-9	View Stations Map	UC-22	Moderate	TC-45
	FR-10	Administrator Permissions	UC-12	High	TC-46
					TC-47

			UC-13	High	TC-48
					TC-49
					TC-50
					TC-51
					TC-52
	FR-11	New User Account Approval	UC-5	High	TC-53
	FR-12	Edit Station Name	UC-18	Moderate	TC-54
					TC-55
					TC-56
	FR-13	View Historical Weather Data	UC-19	High	TC-57
					TC-58
	FR-13				TC-59
					TC-60
					TC-61
					TC-62
					TC-63
					TC-64
					TC-65
					TC-66
					TC-67
	FR-13	Filter Historical Weather Data	UC-20	High	TC-68
			UC-6	Low	TC-69
					TC-70
					TC-71
					TC-72
					TC-73
	FR-14	User Profile	UC-7	Low	TC-74
					TC-75

					TC-76
					TC-77
	FR-15	Weather Alerts	UC-8	High	TC-78
					TC-79
					TC-80
					TC-81
					TC-82
					TC-83
					TC-84
			UC-10	High	TC-85
			UC-11	High	TC-86
	FR-16	Email Alerts	UC-9	High	TC-87
					TC-88
	FR-17	SMS Alerts	UC-14	High	TC-89
					TC-90
	FR-18	Webpage Alerts	UC-9	High	TC-91
					TC-92
					TC-93
	FR-19	Station Connection Quality Indicator	UC-14	High	TC-94
					TC-95
					TC-96
	FR-20	Save Station Data Locally	UC-15	Moderate	TC-97
					TC-98
	FR-21	Filter by Station Name	UC-26	High	TC-99
					TC-100
	FR-22	Connect Stations Automatically	UC-27	Low	TC-101
					TC-102
				High	TC-103

					TC-104
					TC-105
	FR-23	Add a Station	UC-24	High	TC-106
					TC-107
	FR-24	Install new station	UC-28	High	TC-108
					TC-109
					TC-127
					TC-128
					TC-129
	FR-25	Webpage Alerts Display	UC-29	Moderate	TC-110
					TC-111
					TC-112
					TC-113
					TC-114
	FR-26	Send Weather Data	UC-30	High	TC-115
					TC-125
Stretch Goals	FR-27	Average Weather Data	UC-31	Low	TC-126
					TC-116
					TC-117
	FR-28	Historic Alerts	UC-32	Low	TC-118
					TC-119
					TC-120
					TC-121
		Multi weather type alerts		Low	TC-122