Pi Weather Station with Progressive Web Application

Team Members:

Ronald Jaglal- 816003245

Vinod Lochan Dassrath – 816117511

Jose Bravo Mata – 816011156

Course code: INFO3604 Project course

Department: Department of computing and information technology(DCIT)

Course Name: Project Course

Lecturer : Ms. Cynthia Cudjoe

Date: 15th May 2020

Table of Contents

Abstract

In Trinidad and Tobago, accurate, ongoing, locale-specific weather data is difficult to obtain. Even in developed cities like New York, the forecasted temperature may be -1 degree, but when you're walking through the city, it can be as low as -5 degrees because of the wind channels that flow through the cities. A local weather station, however, would give the user a more accurate reading. The current direction of human development is steering towards efficiency, reliability and availability of information and data in daily living. In keeping with this trend it will be beneficial to undertake a project for the creation of a system whereby consumers can track and keep a log of precise weather data viz. Rainfall, Humidity, Temperature, Pressure, UV Index, Pollen/Dust Level and Wind Speed readings. Through the use of a Pi weather station to obtain the conditions aforementioned, a database and web app were assembled to handle the delivery of the required weather conditions to relevant stakeholders via displaying conditions as logs, charts and allowing for user expansion of stations available. Through this, we have successfully developed a progressive web app(PWA) capable of providing the necessary weather condition readings to the relevant stakeholders. The user has the ability to access the data from any device to get location-accurate weather data depending on where the weather stations are deployed.

Goals

1. Provide accurate, locale specific weather readings to users of our weather app, WeatherPal.
2. Allow users to save and view logs of weather conditions.
3. Allow users to add more stations to expand the number of locations covered by stations for attaining more locale specific data.
4. Allow users to easily attain the data they need in multiple formats.
5. Allow users to utilize custom notifications for monitoring conditions.

Introduction

Problem description:

The project seeks to develop a Pi Weather Station to be used anywhere in Trinidad and Tobago. The general aim is to have more accurate and localized weather readings and logs. Thus the general population, especially those whose livelihood is impacted by the weather or has the potential to be improved with the availability of more accurate weather readings, will benefit from this project. Identifying key stakeholders was a process undertaken in this project. In order to perform this process we met with various people we believed to be stakeholders for our project around the University of the West Indies St.Augustine campus. We met with people from various facilities which include the Faculty of Science and Technology(FST), the Faculty of Engineering(FoE) and the Faculty of Food and agriculture(FFA). After which we came to the conclusion that agricultural and science sections benefitted from our project. Research was done to determine our system architecture and determine the tools necessary for integrating our system as a weather app via interviews with lecturers at the campus, the use of material provided from the project course lecturers and online research. We worked on requirements specification, producing use cases, system diagrams and context diagrams. In terms of resources, financial investment in a raspberry pi with relevant sensor components was necessary to achieve our purpose. Care had to be taken when building the weather station so that it could function properly in the necessary environments. In order to collect weather data we needed to purchase equipment to take readings of the values. The equipment we used included: Raspberry Pi 3 B+, Adafruit BME280 Sensor, Adafruit VEML6070 Sensor, SDS011 Nova PM Sensor, Adafruit MCP3008 ADC, Argent Data Systems Wind/Rain Sensor Assembly, MicroSD Card,USB over RJ45 Extender, RJ11 Breakout Boards, 4.7KΩ resistor, Adafruit Perma-Proto HAT – with EEPROM and GPIO Female Header. Resources required for our web app included a database(Cloud Firestore), an authentication service(Firebase Authentication) and application programming interfaces(apis) for handling geocoding and chart display. We used Cascading Style Sheets(CSS), Hypertext Mark-up Language(HTML), Javascript(js) and Python to code our progressive web app, with several apis and prebuilt stylesheets to assist in development(Jquery, OpenCage, Materialize, Ajax and Firebase auth.ui). Our project faced risks such as pi station equipment/sensor failure, running out of time before we could complete our app, possibly facing damage to the station due to environmental factors(the station is desired to be atleast water resistant) and equipment importing delays. We faced some delays in our task schedule but otherwise everything appeared to have proceeded in order. We started by working on our proposal document and handled requirement specification in the beginning weeks . After the requirements were defined(a major milestone, as we knew the functions our system needed to have) we started to work on our project design, which involved researching architectures and technologies we needed for our app. Eventually we decided on a serverless architecture and divided work among our members so that one was in charge of

creating the UI, one developed the main code and another worked on assembling the weather station. We met a major milestone when the station was fully built where we could now start sending data to cloud firestore and retrieving data from the database for the app. We managed to finish up most of the code as we came close to the deadline and integrate the UI and main app code. During the period of coding we had to use the firestore documentation as well as the authentication documentation in order to build our apps.

Positioning:

Problem statement

The main problem being addressed is the need to get location-accurate weather data in the country of Trinidad and Tobago. There are issues where weather conditions given over the news may be inaccurate as they only focus on some areas and do not account for the whole island. The opportunity here is that, given that people certainly need to know the weather, we can produce a means of attaining more decentralized weather information for the public and produce a service that would be generally used/accepted.

Product Position Statement

In addition to stakeholders who are involved in data analytics, the general population of Trinidad and Tobago especially those whose livelihood is impacted by the weather or has the potential to be improved with availability of more accurate weather readings would benefit from the weather station. It would assist in giving feedback to relevant personnel who rely on weather updates to carry out their various functions. This includes farmers, transporters, delivery facilities, police and security personnel, arts and theatre in terms of finding various settings to do videography/photography, researchers, community planners and developers, sports facilities, businesses and in the field of preventative medicine, e.g., those with allergies or sun sensitivity. Due to the expansion capabilities of our weather reading system, more users may be benefitted over time.

Stakeholder Descriptions

Stakeholder Summary

- Researchers/Lecturers (Mechanical Engineering, Zoology and Food Production)
- Farmers
- Fishermen
- Transporters
- Delivery Facilities
- Police and Security personnel
- Planners and developers
- Lifeguards
- Sports personnel
- Persons who have health conditions that are triggered by environmental factors
- Businesses that are weather sensitive

User Stakeholders

Researchers/Lecturers (Mechanical Engineering, Zoology and Food Production), farmers, fishermen, transporters in particular but also the general population.

Non-user stakeholders

The general public who are the consumers of the fisher folk and farmers produce. Also workers

who have employees who have allergies to dust or restrictions with UV exposure would be able to be alerted and use protective and avoidant techniques to manage and prevent disease flare-ups which can result from unavoidable time away from work and loss of productivity.

User Environment

The environment of Trinidad and Tobago is the pilot location to be tested , This is a tropical climate with two recognised seasons, rainy season and dry season. The intention is for this system to be implemented in environments around the world with different patterns. The user environment should have internet access, any computer device that has a browser supporting HTML5 and JavaScript to use the web application.

Product Overview

Overall our product provides the user an interface that displays weather data from pi weather stations that can be set up all over Trinidad. The station displays weather data(ultra-violet light, barometric pressure, rainfall, airQuality(pm10, pm2_5, aqi10 & aqi2_5), wind direction, wind speed, humidity and air temperature) to the user in the form of charts and gauges. The app allows the user to add stations that the app can obtain weather conditions from using a cloud firestore database, and then provides a simple ui implementation for displaying this dara. The app allows for reading of logged values and the user to disable his account if he no longer wishes for his email to be linked to the account). The app allows ease access and meets loadtime in sometimes 1 second. Many search functions have been implemented so that user can get the station that meets their needs quickly, such as getting the user's favourite stations, getting the user's owned stations and getting stations by name/location. The app also provides information about the various condtions it provides on individual pages.

Product Perspective

The web application for the system can run on any device that has a modern browser with HTML5 and JavaScript support.

Needs and Features

- A user shall be able to search for a weather station using location, name and proximity
- A user shall be able to register an account for interacting with the system's weather web app
- The system shall be able to attain weather information from a weather station and store it in a database.
- The system shall be able to take weather data from a database and send it to a web app for display.
- A user shall be able to add a weather station to the system's available weather stations.
- A user shall be able to add a weather station to his list of favourite stations for later viewing.
- A user shall be able to get a list of weather log data from each station.
- The system shall be able to keep a store of logs of weather data for each station on a database.
- The user shall be able to view weather data in a graph format.
- The user shall be able to view a list of the weather stations that he owns and has added to the system's available weather stations.
- The user shall be able to initiate notifications that will alert them when sensor readings fall out of or into a certain (user specified) range.
- The user shall have the ability to deactivate their account.

Alternatives and Competition

There are many existing international companies that gather weather data and stakeholders can gain information from those companies, but this project is aimed at providing localised weather data for all to benefit. Thus users can have more precise and accurate data. Competition and alternatives may include the TT Met Office app(by the T&T meteorological office) and the

AccuWeather site for getting weather data.

Other Product Requirements

Browser Compatibility

Users can use any browser that supports HTML5 and JavaScript.

Usability

The application should be user friendly, easy to learn and navigation should be simple. New or infrequent users should encounter minimal difficulty when using the system. Users should be able to perform tasks quickly with little to no errors. If errors do occur, the user should be able to easily recover. The system should supply consistent user interface design standards to invoke a sense of familiarity, eliminate confusion and promote learnability for the user. Users should take no longer than 2 minutes to learn to use the application interface.

Responsiveness

The app should allow the user to be active on the pages instead of waiting on them to load and so the web application's user interface should ideally take no longer than three seconds to load. Search results should be returned within two seconds. Database updates should be done within two seconds. Login validation should be done within three seconds.

Visual design

The website would be attractive to users in terms of appropriate fonts used to communicate the required weather information.

Long term Scalability

System should be flexible enough to easily adapt and meet storage and performance requirements with minimal need of extra infrastructure and continue to be functional as users of the system and load on system resources increase.

Requirements Specification

Functional Requirements;

- A user shall be able to search for a weather station using location, name and proximity.
- A user shall be able to register an account for interacting with the system's weather web app.
- The system shall be able to attain weather information from a weather station and store it in a database.
- The system shall be able to take weather data from a database and send it to a web app for display.
- A user shall be able to add a weather station to the system's available weather stations.
- A user shall be able to add a weather station to his list of favourite stations for later viewing.
- A user shall be able to get a list of weather log data from each station.
- The system shall be able to keep a store of logs of weather data for each station on a database.
- The user shall be able to view weather data in a graph format.
- The user shall be able to view a list of the weather stations that he owns and has added to the system's available weather stations.
- The user shall be able to initiate notifications that will alert them when sensor readings fall out of or into a certain (user specified) range.
- The user shall have the ability to deactivate their account.

Non-Functional:

- Availability
    - System should ideally be available all of the time for targeted users.
    - Some level of monitoring should be considered for quick detection and fixes of any downtime occurring within the system.

- Maintainability
    - Bugs or any defects in the system should be easy to detect and resolve.
    - Code should be easy to read and understand.
    - Code should have comprehensive comments and be written in simplified language
    - Different versions of the system should be easy to maintain.
    - System should be easy to extend or enhance in the future.
    - A test driven development approach should be used for superb low level design and unit tests.

- Interoperability
    - If required the system should be easy to interface with other 3rd party components and to exchange data or services with other systems
    - Different modules within the system should seamlessly work on different operating system platforms, different databases and protocol conditions.

- Performance
    - Web application's UI should ideally take no longer than 3 seconds to load.
    - Search results should be returned within 2 seconds.
    - Database updates should be done within 2 seconds.
    - Login validation should be done within 3 seconds.
    - Code should be tuned to the maximum with the appropriate use of data structures and algorithms.
    - Databases used should be suitable and well optimized for the type of queries required by the system.
    - Caching should be used to improve performance where applicable.

- Portability

- Code should be well defined, with portability in mind so that as technology changes in the future, changing platforms can be easily accomplished.
- Specific dependencies on certain technologies and their features available only in specific forms should be avoided where necessary.
- The system application should run on any device that has a modern browser with HTML5 and JavaScript support.

- Reliability
  - Data registered in database and/or served to user should be accurate and dependable.
  - Database should be well designed, thereby reducing the chances of having errors with data stored.
  - The system should allow for logged data to be periodically backed up and restored in case of data loss. Logged data backups should be done either hourly, daily, weekly or monthly as specified by the user.

- Reusability
  - Development should follow a modular approach whereby the build is broken down into finer components each having specific and focused responsibilities. Therefore these can then be easily reused where necessary within the system and so increase work efficiency.

- Security
  - System security should be sufficient to prevent unauthorized access to system functions, avoid information loss, ensure that the software is protected from viruses, and protect the privacy of data transactions within the system.
  - User login information should be encrypted.
  - Vulnerabilities such as to SQL injection and Cross-site Scripting should be eliminated.
  - Accounts should have limited permissions and privileges accordingly delegated based on their necessary requirements.

- Scalability
    - System should be flexible enough to easily adapt and meet storage and performance requirements with minimal need of extra infrastructure and continue to be functional as users of the system and load on system resources increase.

- Testability
    - Architecture should incorporate a level of modularity so that instead of being restricted to testing the entire system tests can be conducted on individual singular components.
    - Unit tests should be well designed and used as much as possible for the various components within the system.
    - If time permits, tests should be automated where applicable.

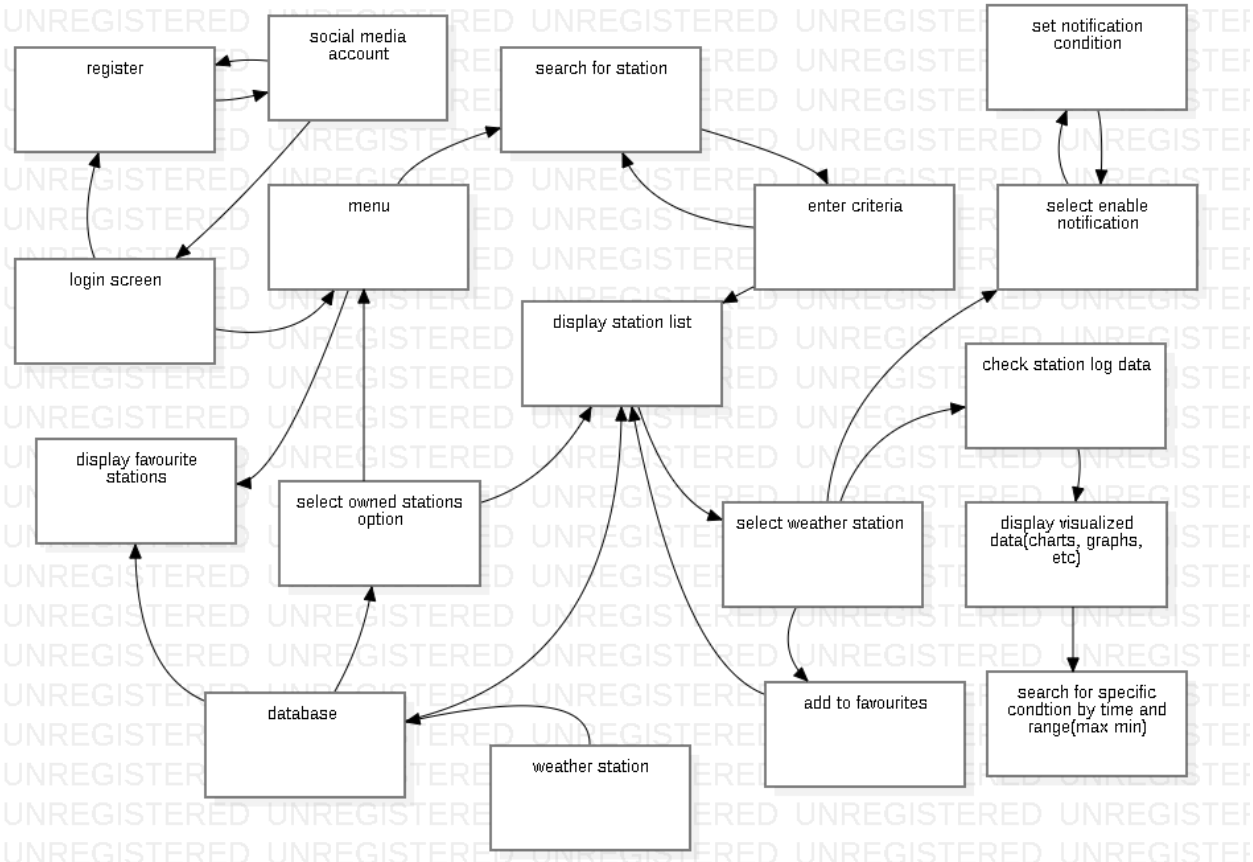
- Usability
    - Application should be user friendly, easy to learn and navigation should be simple.
    - New or infrequent users should encounter minimal difficulty when using the system. Users should be able to perform tasks quickly with little to no errors. If errors do occur the user should be able to easily recover.
    - Users should be able to recover from errors in 10-20 seconds.
    - The system should supply consistent user interface design standards to invoke a sense of familiarity, eliminate confusion and promote learnability for the user.
    - User should take no longer than 2 minutes to learn to use the application interface

<u>User Stories:</u>

1. As a user, I want to check for weather stations near me so that I can get the weather conditions in my area.
2. As a user, I want to search for a particular weather station by name so that I can find the station I want easier.
3. As a user, I want to search for stations by location so that I can get weather conditions at stations in or near that location.
4. As a user, I want to be able to register an account for the weather web app so that I know that my data is confidential and binded to my account.

5. As a station owner, I want to add a weather station to the map so that I can monitor weather conditions at my own stations and for others to use the data it gathers.
6. As a station owner, I want to be able to see a list of weather stations I own and have added, so that I do not have to search for the stations manually
7. As a power user, I want to be able to add a station to a list of my favourite stations so that I can easily access my preferred stations later.
8. As a power user, I want to be able to get a list of previous weather conditions or logs from a station so that I can check the weather pattern at that station.
9. As a power user, I want to be able to see weather pattern data as a graph so that I can get an easy grasp of weather pattern data.
10. As a farmer, I want to schedule notifications to activate whenever the weather conditions my crop needs fall out of the ranges I want them to be in so that I can know if I need to tend to the crop.
11. As a user, I want to deregister my account from the app so that I no longer get any notifications or have the app linked to my account.
12. As a user, I want to be able to get old weather data so that I can analyze or predict weather conditions.
13. As a user, I want to be able to search through weather data to get the condition that I want so that I can know when or if the value I want was encountered.

System Diagram:



Use Cases:

1. Use case: get location weather

Actors: app user

Type:Primary

Description: The user opens the app and logs into the main screen. He selects the option to get weather conditions and is greeted by a search bar with several search options and a button that gets weather stations at his location. He selects to have the bar search by location and inputs the relevant location name for the app to search for. The app displays a list of stations found to the user at or around that location. The user selects the station he is looking for, and is greeted by a screen displaying weather conditions and logs. After he is done checking the conditions and logs he closes the app.

2. Use case: check own station

   Actors: weather station owner

   Type:Primary

   Description: The weather station owner opens and logs into the app to look for weather conditions at one of his own stations. The station owner taps the option to display his pi weather stations and is greeted with a list of weather station he owns. The user checks his stations and closes the app when he finishes.

3. Use case: log in

   Actors: app user

   Type:Primary

   Description: The user starts the app to perform some task. The app opens up and greets him with a login screen. The user types in his credentials into the input boxes and after the app has validated his credentials, it takes him to the main screen.

4. Use case: add to favourites

   Actors: app user

   Type: Optional

   Description: The app user starts the app, logs into the main screen and chooses the option to get weather conditions. After using the function to get a list of weather stations he selects a station and the app displays the weather conditions and logs of that station. He selects the

option to add the station to his favourites which adds the weather station to a list of his personal favourite weather stations, and exits when he is finished.

5. Use case: get favourites

   Actors: app user

   Type:Optional

   Description:The app user starts the app and logs into the main screen. The user selects the function to view his favourite stations. The app displays a list of the app user's favourite weather stations. When he is done checking his favourite station, he exits and closes the app.

6. Use case: enable notification

   Actors: app user

   Type:Optional

   Description: The app user clicks on a weather station to get its data and the web app displays the station's data. The user touches the button that allows him to set notifications and the web app displays options that allow him to specify his notification. He specifies the weather condition he wants to be notified of and he also sets a conditional range for the condition to fall in or out of before being notified. After the user is done setting up his notification, he exits the app.

7. Use case:get condition value

   Actors:app user

   Type:Primary

   Description: The app user clicks on a weather station to get its data and the web app displays the station's data. The user then taps the "view logs" option to have the app display a list of that station's logged data. The user clicks the option to search for a particular condition and the web app displays a screen that allows him to select his search criteria. After entering the user finishes entering his criteria the web app displays his sought data. Once satisfied, the user closes his app.

8. Use case: register

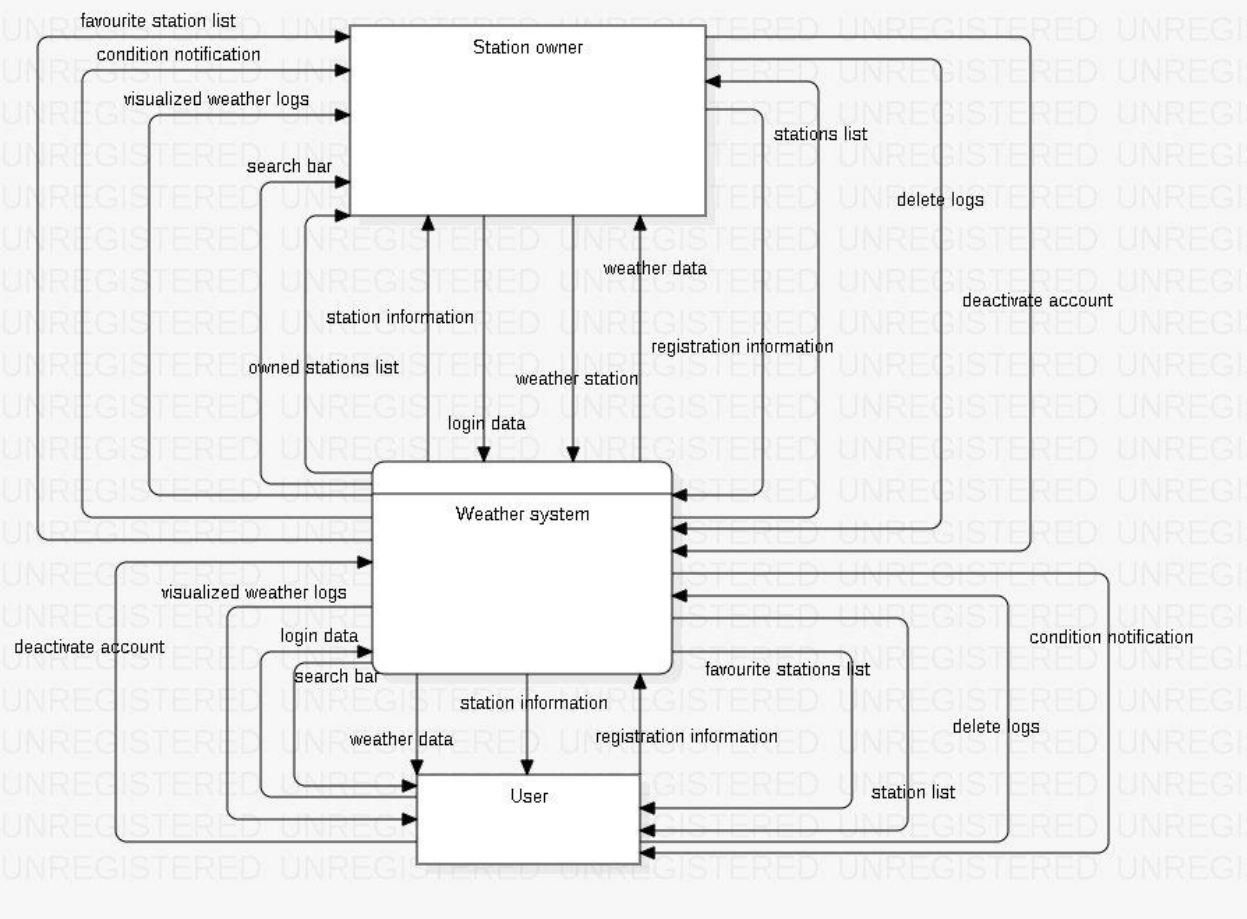
   Actors:app user

Type:Primary

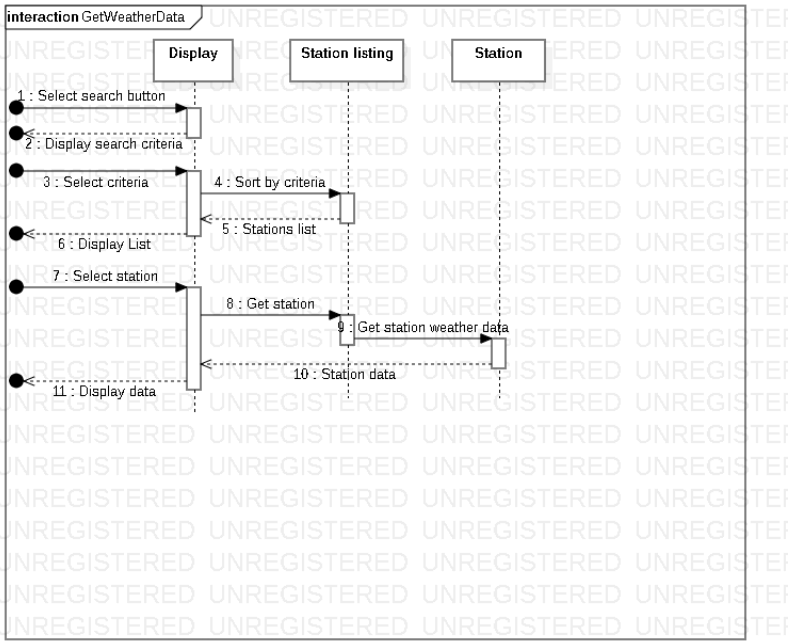
Description: The user opens up the web app and is the web app greets him with a login screen. The user selects the option to register on the login screen and the app displays options to register via making a new account or using his social media account. The user makes his account using the create an account option, fills in the required criteria, and is afterward led back to the log in screen after his inputs are validated.

Use Case Diagram:

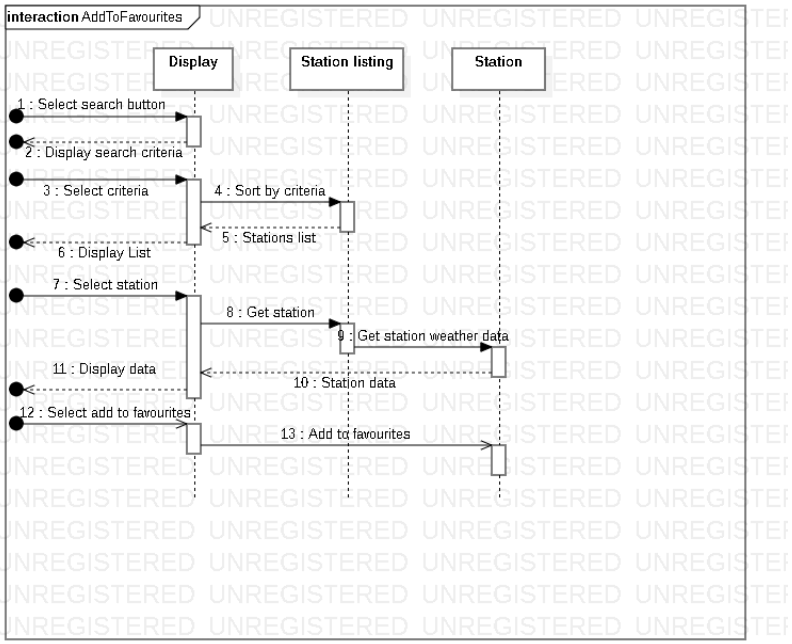
Context Diagram:



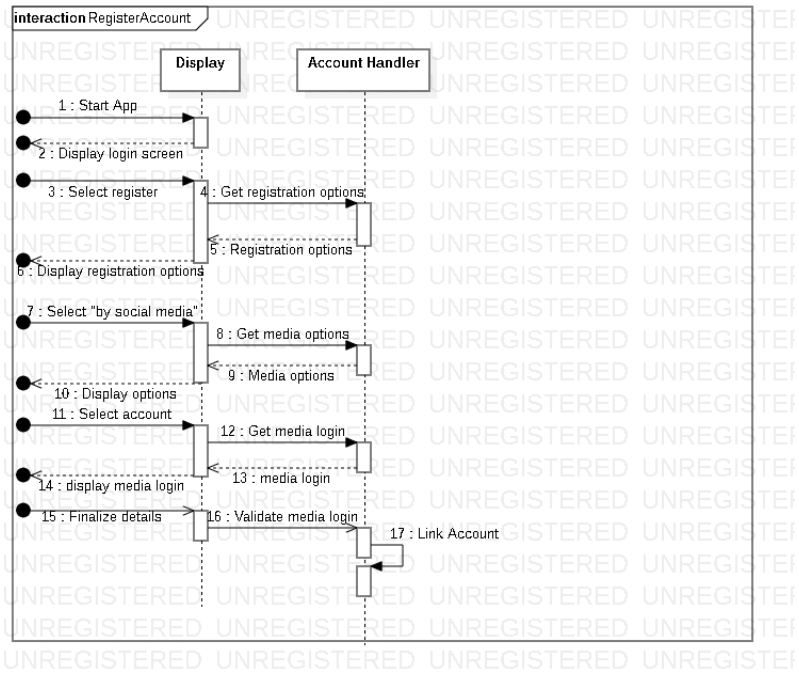
Sequence Diagram for Weather system(GetWeatherData):



Sequence Diagram for Weather system(AddToFavourites):



Sequence Diagram for Weather system(Register):



Technical Constraints:

- The technical constraints we faced involved hardware, where we had to rely on imported hardware to function well for our system and we only managed to obtain 1 working station. Some hardware was not successfully imported and some were not functional.
- Time was a crucial factor to our ability to complete this app, with constraints from other courses to keep in mind.
- Certain apis we intended to use had limitations when encorporated into our web app which led to several set backs.
- Learning was also a problem, new code elements had to be learned, new functions from apis and learning how to handle certain errors had to be undertaken.

Statistics or specific numbers

Average minimum annual temperature:22.7

Average maximum annual temperature:31.3(MET Office 1997-2000)

Statements made by specific individuals, companies or reports

Opinions made by others:

Design Specification

Architecture:

For our system architecture we decided to go with a serverless architecture. Our system consists of a frontend progressive web app(PWA) and a Cloud firestore(Cloud) database. The frontend displays data it receives from the database which, in turn, receives data from (currently) one weather station. Should any stations be added the database can auto-scale to hold the added stations. The app is mainly developed in HTML, CSS and Javascript. The pi weather station code is developed in Python. In terms of flow the pi weather station takes weather condtion readings and sends it to firestore, where the web app queries firestore for the data it needs. In total view, the station act as nodes connected to a firestore database, and send their data which is accessible by the web app.

Alternative designs:

The Progressive web app could have been designed as an android focused app instead. An alternative ui could have existed where the user could have had access to everything on a single page. A previous version of the current app displayed current weather condition data on the pages specified by weather condition. Another alternative design may have included a website focused design instead of a progressive web app where the user can choose to receive notifications and predictions. The project could have also primarly been developed to focus on farmers and agricultural practices.

System interface description:

The system interface of weather pal consists of the functions necessary to display weather station data to the user. The location data of the user is gathered from the user's device to display, which requires calls to the innate location feature of the user's device, this is done via a browser call for

geolocation. Since our app only requires browser compatibility with html, css and javascript we have little requirements for system interface other than support for the required browsers.

Components and subsystems description:

Components:

The progressive web app(PWA) consists of 12 separate screens:

- Login screen: This is where the user chooses his authentication method to log in. It handles authentication functionality only.
- Home screen(index): The main screen in the app, this displays the condition data gathered from the pi weather station and consists of 3 separate tabs: The station tab, the current tab and the logs tab.
    - Station tab: The station tab displays station information(last update, station name and owner) to the user and allows the user to add the station to his favourites for ease of access.
    - Current tab: This displays the weather conditions obtained from the selected weather station. It displays information such as station location, air temperature, barometric pressure, uv, rainfall, wind direction, wind speed, humidity and various factors for air quality in the form of text and gauges.
    - Logs tab: This allows the user to view logs of a station in the form of charts. Log data is also downloadable.
- Search screen: This screen allows the user to use various methods to search for a station such as by name, location(nearest user or specified by user input), owned(user's own station), favourites(stations user has favourited) or just get all stations.
- UV-index, Rainfall, Wind, Air Temperature, Humidity, Air Quality & Barometric pressure screens: These screens display explanations about these weather conditions.
- Alerts screen: Allows the user to set up notifications to receive when the conditions he specifies(via a pop up menu) fall out of the ranges the user sets.
- Account screen: Allows the user to perform user account related functions such as deleting their account, changing their password, adding new stations under their account and enabling/disabling their stations.

The app is structured where extra css stylesheets are placed in file labelled css. The main css file is in the main directory.
The css components are as follows:

- configurealerts.css: Contains style details for the alert html page layout.

- guages.css: Contains style information for the gauges on the home page.
- logcharts.css: Contains the style information for the log charts on the logs tab of the home page.
- propeller.css, propeller.css.map, propeller.min.css, propeller.min.css.map: Used for styling information for the alerts pop up menu, as well the configure alerts button.
- style.css: Main stylesheet for all pages in the app.

Most javascript(js) files are in the directory js. Script.js and the service workers are the only js scripts present in the main directory.
The javascript components are as follows:

- accountfunctions.js: Handles functions related to manipulating the user account(such as deleting the user account, changing password, getting user data).
- accountmain.js: Used by the Account page to verify user input and control elements on the page.
- authenticationsetup.js: Sets up the authentication options(using firebase's pre-built ui)of the Login page.
- cloudmessaging.js: Used to set up cloud messaging functionalities which are used for making alerts and in the cloud-messaging service worker.
- configurealerts.js: Used in the Alerts page pop up for configure alerts to set up the alerts options and functionality.
- firestorefunctions.js: Used for sending data to firestore, more particularly in the event a basic function is necessary to send data to firestore.
- formatfunctions.js: Formats data retrieved from firestore for easy access through an array. It currently specifically handles realtime data from the weather station.
- gauges.js: Handles the creation of gauges to display weather data to the user using Highcharts api.
- geocoding.js: Handles forward geocoding(turning a place name to latitude and longitude coordinates) Using the OpenCage api. Used for converting a user specified place name on the search page to coordinates(for finding stations) and is also re-purposed for reverse geocoding to get the current station's location name from coordinates.
- Initializefirestore.js: Configures firebase and allows for attaining a firestore instance. Used for attaining the database instances necessary on most pages.
- logcharts.js: Handles the creation of log charts to display to the user. These charts are on the logs tab of the Home page.
- main.js: Sets up the service worker for a page.
- pageloading.js: Handles common functions performed between pages such as storing station id and loading up the signout button.
- propeller.js and propeller.min.js: Used for the configuration of the configure alerts popup of the Alerts screen.

- searchfunctions.js: Contains functions for finding stations. Used by the Search screen for finding stations.
- stationfunctions.js: Functions that handle operations related to station data retrieved from firestore. Contains functions such as receiving updates to the current station data.
- stationregister.js: Contains functions for adding a station to the user account and app, adding a station to favourites, removing from favourites.
- userfunctions.js: Contains functions related to user authentication data. Used for storing user data in a document after the user has registered.
- yahooauthentication.js: Allows the user to authenticate using Yahoo.
- firebase-messaging-sw.js: Works in the background to deliver notifications to the user using cloud messaging from firebase.
- sw.js: Works in the background to cache pages for speeding up load times and preserving user data.

The images folder contains images for the web app.
Images are as follows:

- icon.png: The original WeatherPal icon.
- uv_descriptions.png: For the uv page for describing index values.
- cloud.jpeg: Used as a background for certain styling.
- propic.png: Used as part of the navigator menu.

The Fonts folder contains the styling information for Roboto fonts.

The functions folder contain firebase functions for interaction with the cloud messaging service worker.

<u>Weather station components</u>:

The equipment we used included:

- Raspberry Pi 3 B+
- Adafruit BME280 Sensor
- Adafruit VEML6070 Sensor
- SDS011 Nova PM Sensor
- Adafruit MCP3008 ADC
- Argent Data Systems Wind/Rain Sensor Assembly
- MicroSD Card
- USB over RJ45 Extender

- RJ11 Breakout Boards
- 4.7KΩ resistor
- Adafruit Perma-Proto HAT – with EEPROM
- GPIO Female Header.

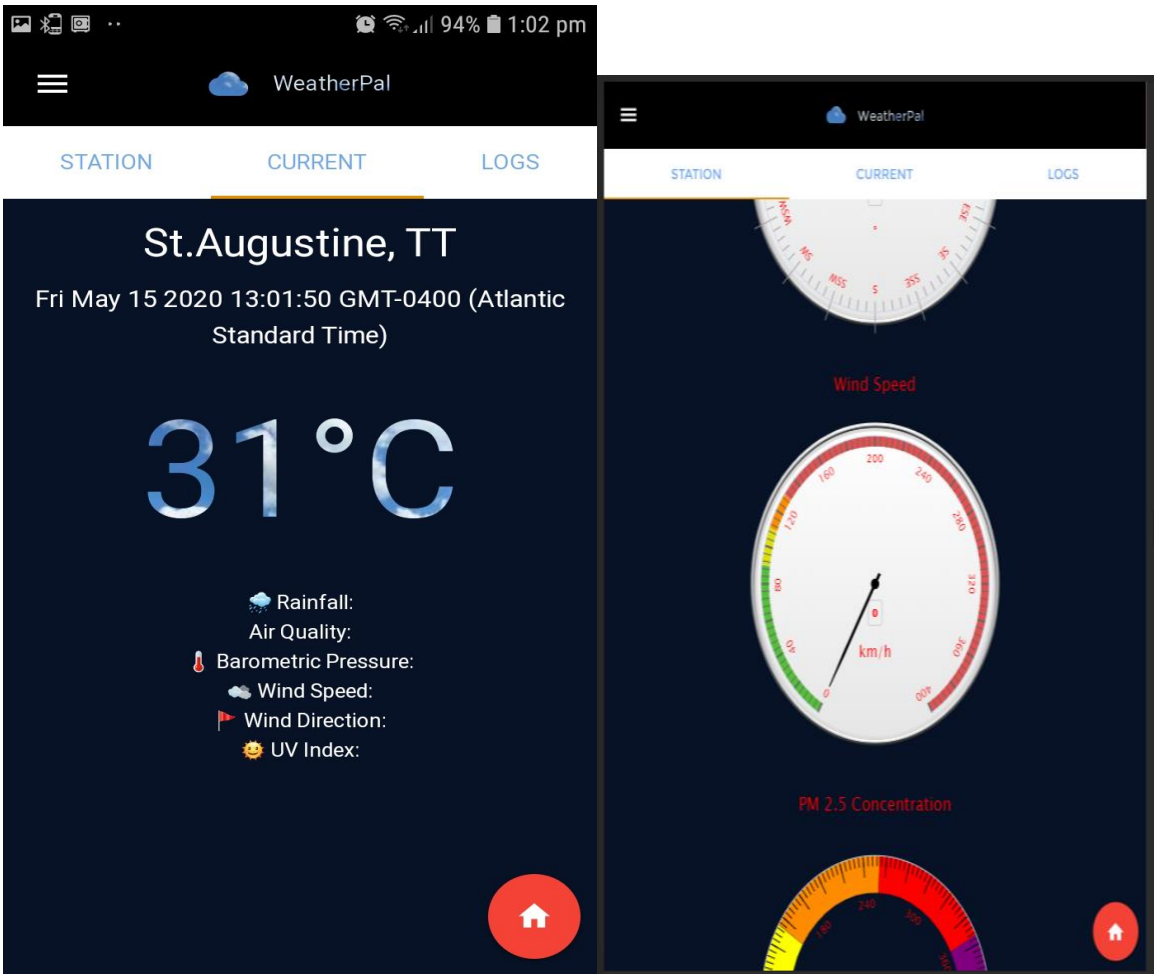
Subsystems:

- Firestore storage: Storage between the firestore database and user end web app. Operates by calls to the firestore instance where the app receives and instance that is configured from initialize app to access firestore.
- Cloud messaging: In charge of the notifications that a user receives. Set up by node.js code set to run as a cloud function. Works offline once the service worker(cloud-messaging-sw.js) has been ran on the user end and the user has set up alerts on the Alerts page.

User Interface Design:

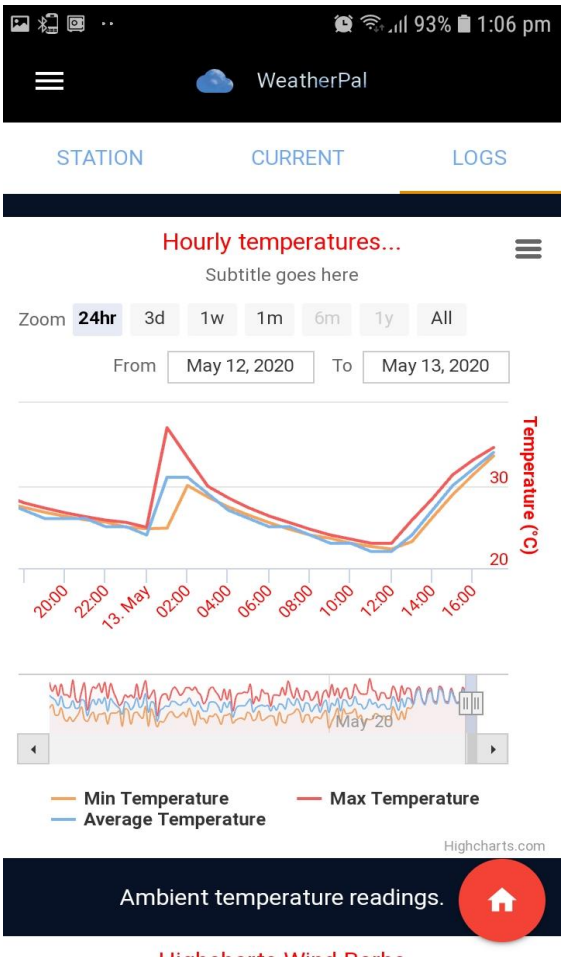
Description of the User Interface **;**

The user interface was designed specifically for the user to be able to get as much data as they can with the least amount of work done (clicks). On the home screen, there are three tabs that are swipeable. This makes it very simple for the user to switch between *stations, Current Weather* and *logs.* Reasons are that the majority of users engage in apps with similar gestures, therefore making the learning curve low. In the image below, the home screen is shown.



These pictures above show the center tab of the home screen. Current shows all the real time data. The temperature is set to the biggest size, to make the app more appealing to the user. When the user scrolls down the current page, they are exposed to a number of gauges and meters that show the real-time metrics in a graphical way. This is one of the features that would stand out to the user as it is the same data they would see in another app, but it's done in a more innovative way. On the left tab, the stations that the user saved are shown. On the right tab, the logs are shown to the user. A snapshot of the logs page is shown below.

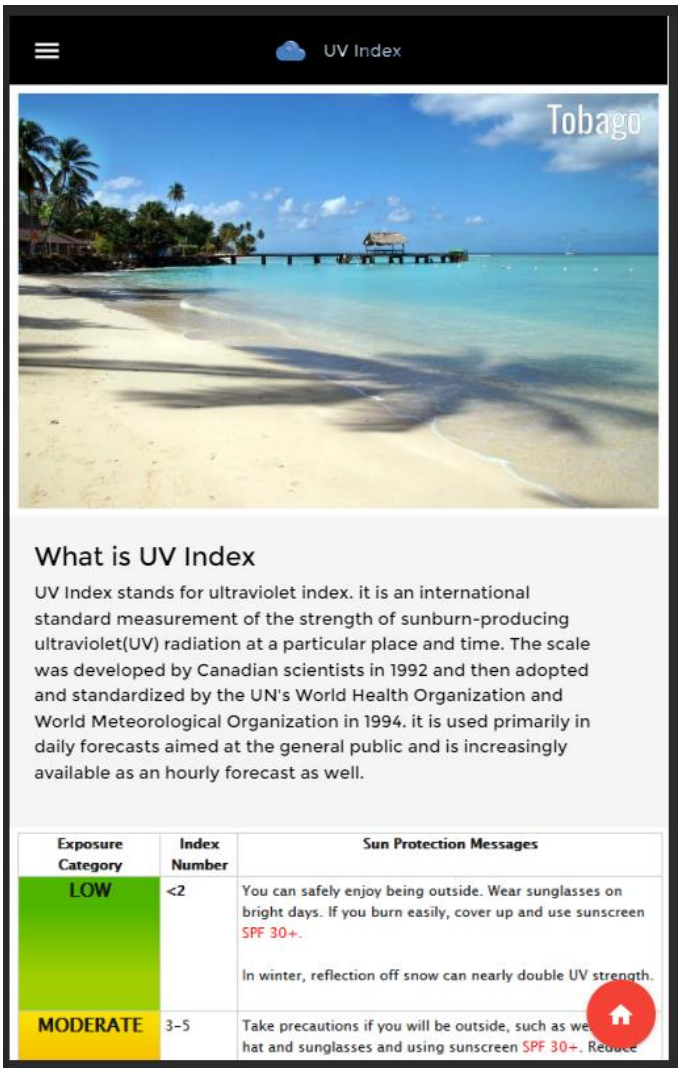
On this page, the user can view and interact with a number of charts which show weather data for a specified duration of time. This is another vary innovative way of displaying weather data, as the user can play around with the data and see the changes as they happen. There is a menu button to the right of the page, that when clicked it gives the user more options such as to download the graph values or download as an image.



With just the home screen alone the user can access all this data by simply swiping or scrolling. The team of the app is also appealing as it has some users have said that they like it, especially the fact that the temperature writing is the image rather than being white writing on a picture. To the bottom of the screen is a red icon with a white home icon. This is a well known symbol known to the vast majority of users as the home button. Its bright red color makes it distinguishable from whatever content it may be floating over, and it's easy to find. The user gets alot of functionality from this little button as it can bring them back to main stuff with just one click.

To the top left corner of the app is a menu button, that when clicks give a satisfactory animation of the side navigation panel opening. This is shown on the next page. The user is then exposed to the other features of the application. They can log in, search stations, they can manage their account. With that functionality out of the way, the user can click on the various pages and read interesting facts about weather, so that the next time someone asks them about the weather, they can have facts and make their conversation more interesting.


92% 1:13 pm
WeatherPal
CURRENT
LOGS
t.Augustine, TT
Fri May 15 2020 13:13:13 GMT-0400 (Atlantic Standard Time)
Login
Search
Air Temperature
Rainfall
Barometric Pressure
Air Quality
Humidity
UV-index
Wind
31°C
Rainfall:
Air Quality:
Barometric Pressure:
Wind Speed:
Wind Direction:
UV Index:


UV Index
Tobago
What is UV Index
UV Index stands for ultraviolet index. it is an international standard measurement of the strength of sunburn-producing ultraviolet(UV) radiation at a particular place and time. The scale was developed by Canadian scientists in 1992 and then adopted and standardized by the UN's World Health Organization and World Meteorological Organization in 1994. it is used primarily in daily forecasts aimed at the general public and is increasingly available as an hourly forecast as well.
Exposure Category
Index Number
Sun Protection Messages
LOW
<2
You can safely enjoy being outside. Wear sunglasses on bright days. If you burn easily, cover up and use sunscreen SPF 30+.
In winter, reflection off snow can nearly double UV strength.
MODERATE
3-5
Take precautions if you will be outside, such as we
hat and sunglasses and using sunscreen SPF 30+. Reduce

Implementation

Approach and Methodology:

For the purpose of this project a progressive web app(PWA) was implemented. In terms of design we decided on (details in architecture) a serverless architecture with a frontend that receives data from a cloud firestore firebase. The main idea behind our app was to ensure that the frontend user can get the data he wants. We also pushed towards fulfilling our requirements of an expandable weather reading system, which is where cloud firestore came into use. Once the user has established his weather station he can simply link it to our firestore database with the rules and templates we specify. We decided apriori development to meet our stakeholders who might have the most potential desire for our system. Focus was made on work division as we split work between members as frontend code, station code and ui development. Requirements were quickly gathered as soon as possible to focus on coding, where we would compile the three divisions of work mentioned before into the completed final version. Mid-development a changelog was created to monitor the frontend code developers progress(see appendix).

Libraries(scripts and styles) used:

- Materialize
- Bootstrap
- Propeller
- Toastr
- Jquery

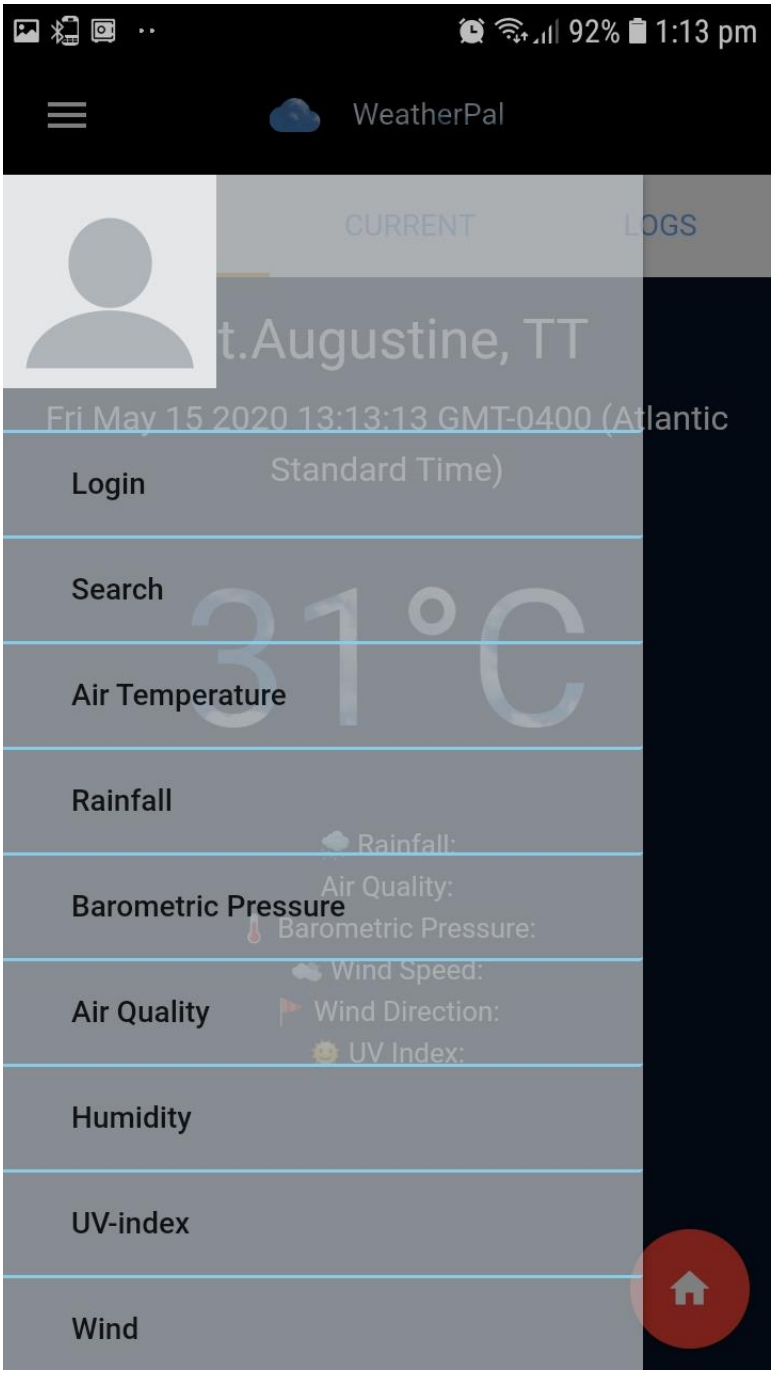
- Fontawesome
- GoogleApis
- OpenCage
- Firebase(app, messaging, firestore, auth and auth.ui)

Languages used:

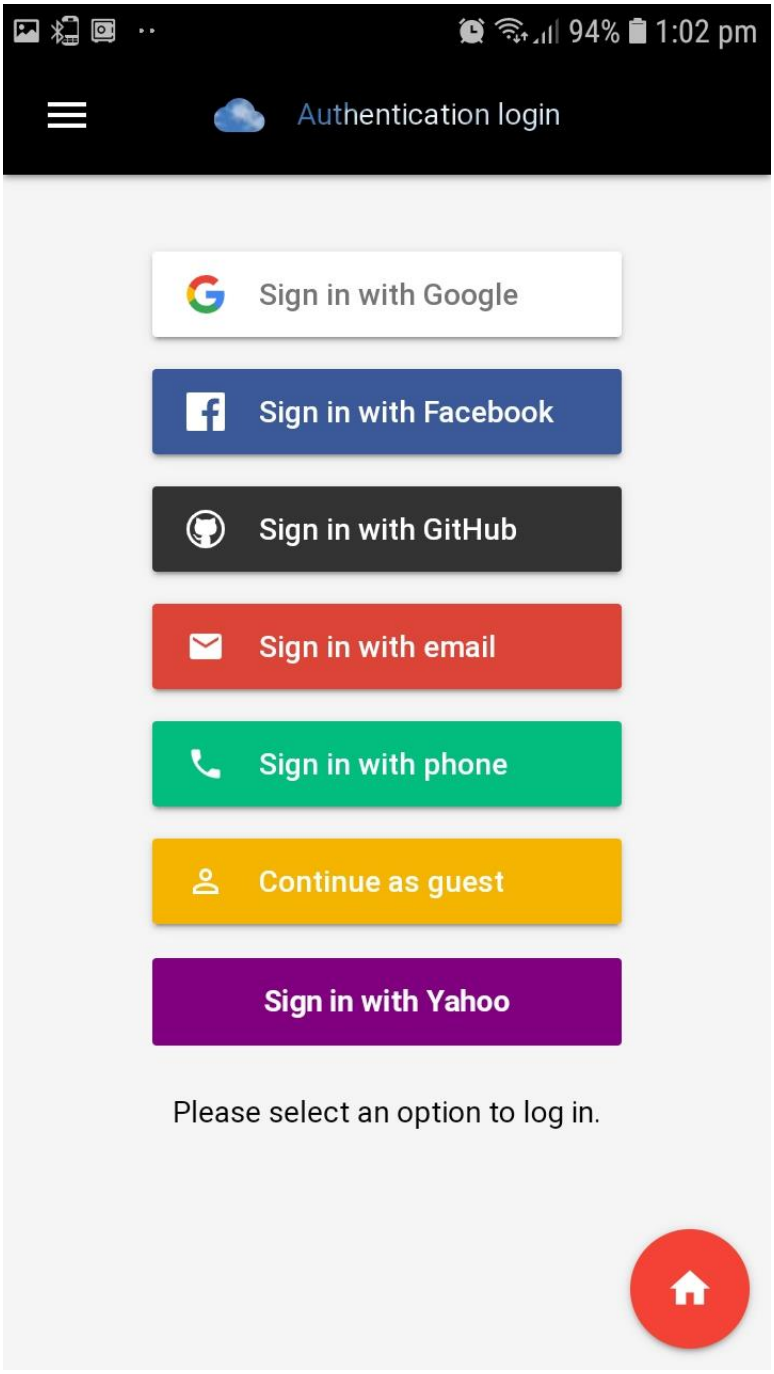
- HTML(HyperText Markup Language)
- CSS(Cascading Stylesheets)
- JS(Javascript)
- Python
- Node.js

Screenshots with descriptions:

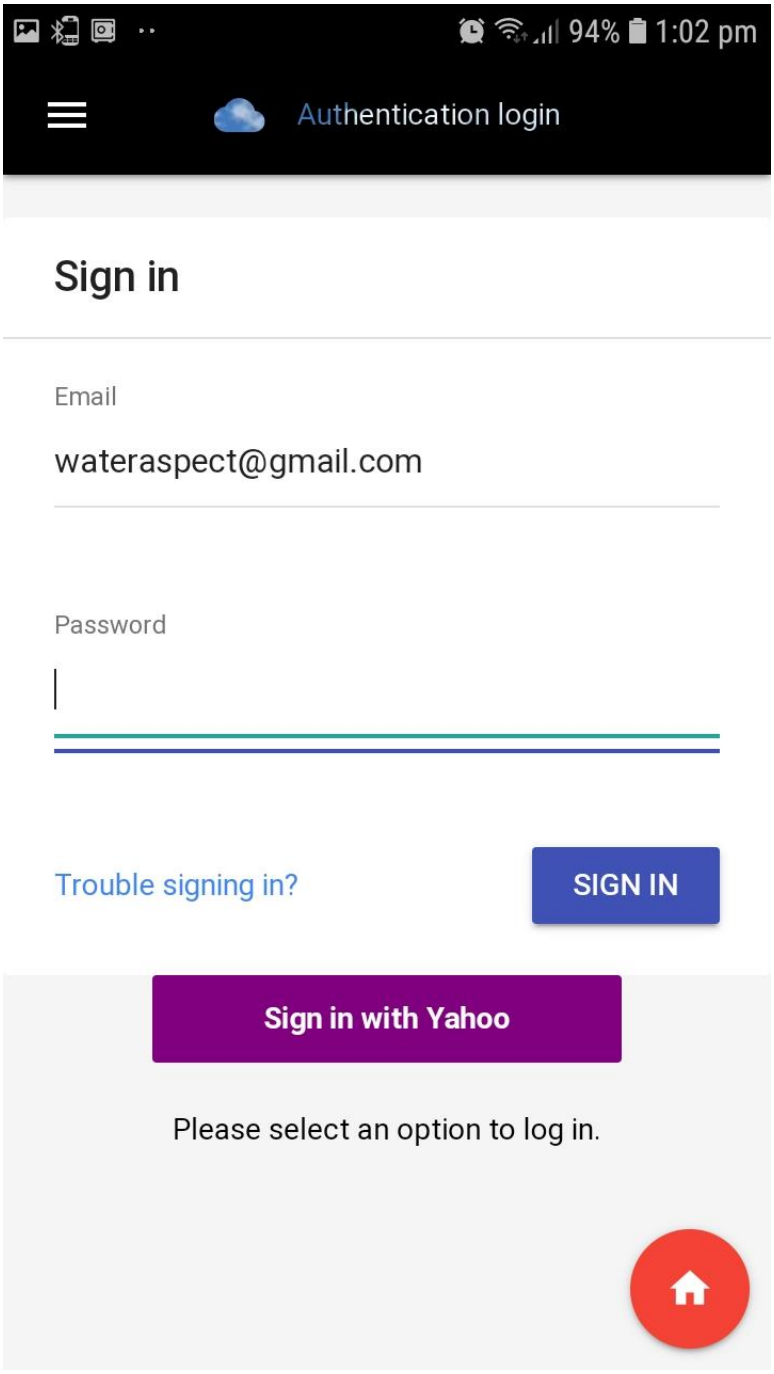
The main screen after the user has started the app.



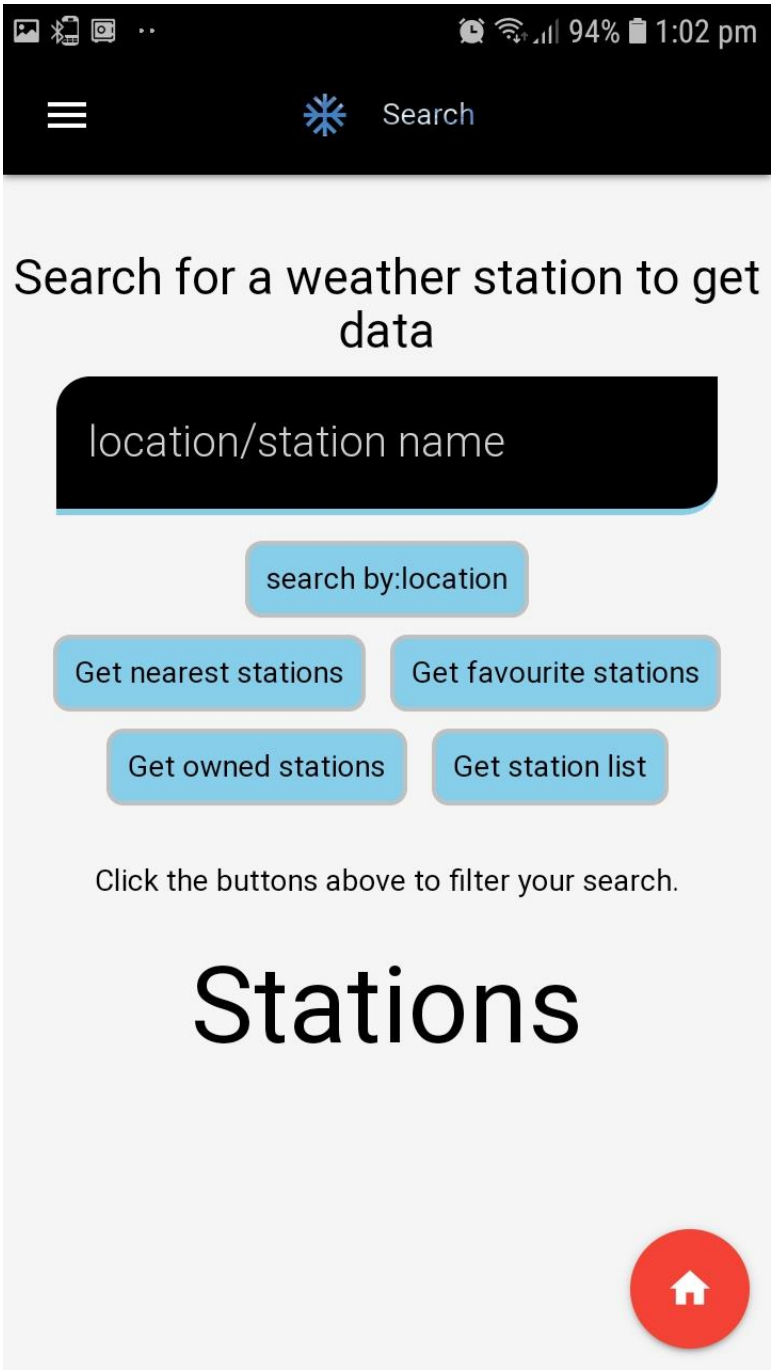
The navigation bar, which is opened using the 3 bars to the top left of the app.


94% 1:02 pm
Authentication login
Sign in with Google
Sign in with Facebook
Sign in with GitHub
Sign in with email
Sign in with phone
Continue as guest
Sign in with Yahoo
Please select an option to log in.

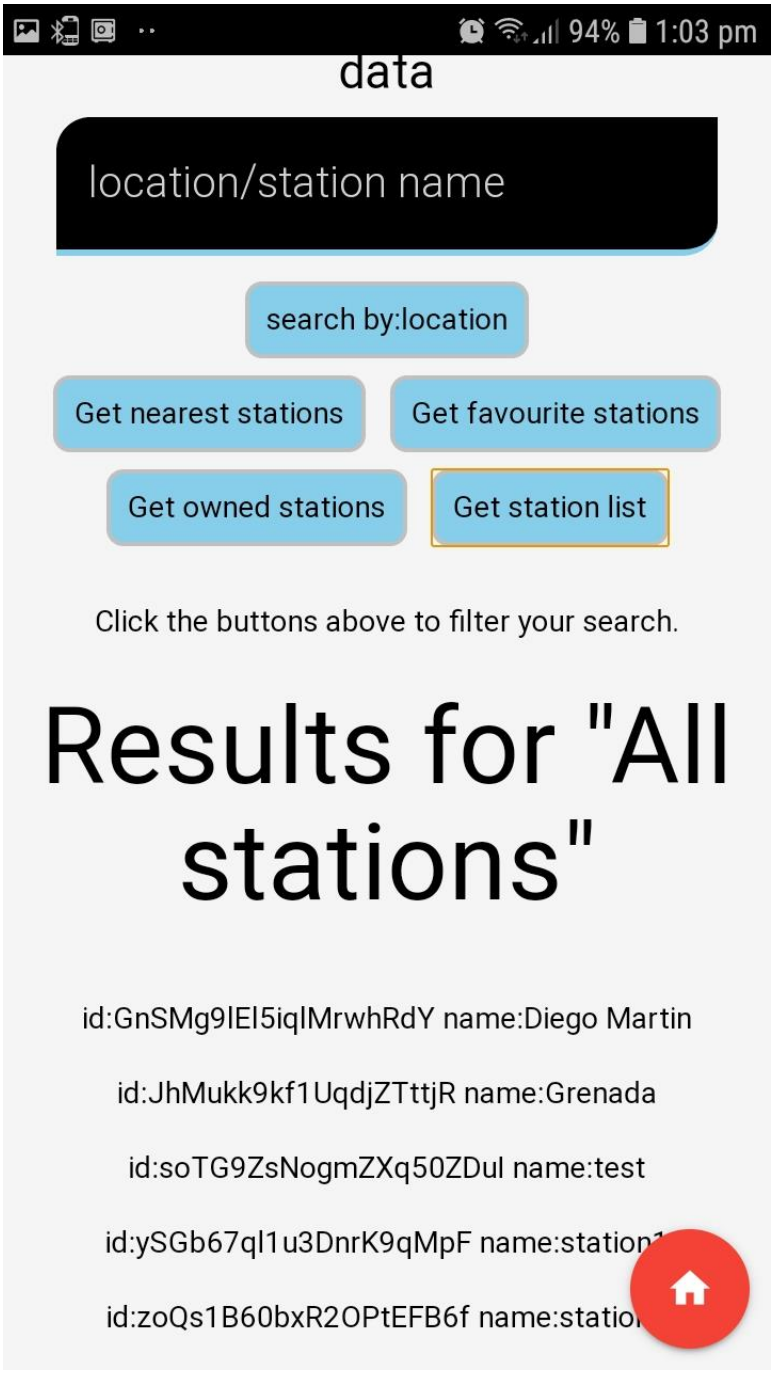
The Login screen after the user has navigated to it using the navigation bar.



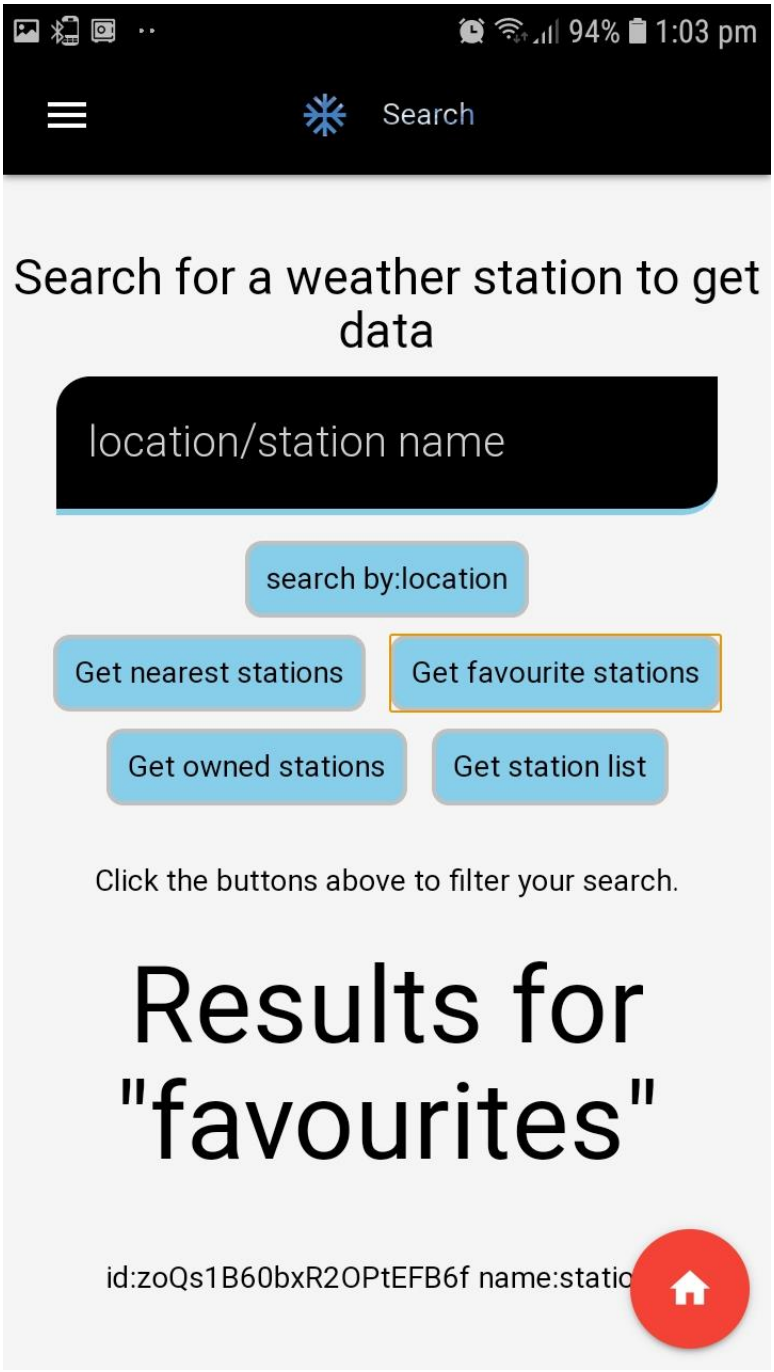
Login screen after the user has selected to sign in with email.



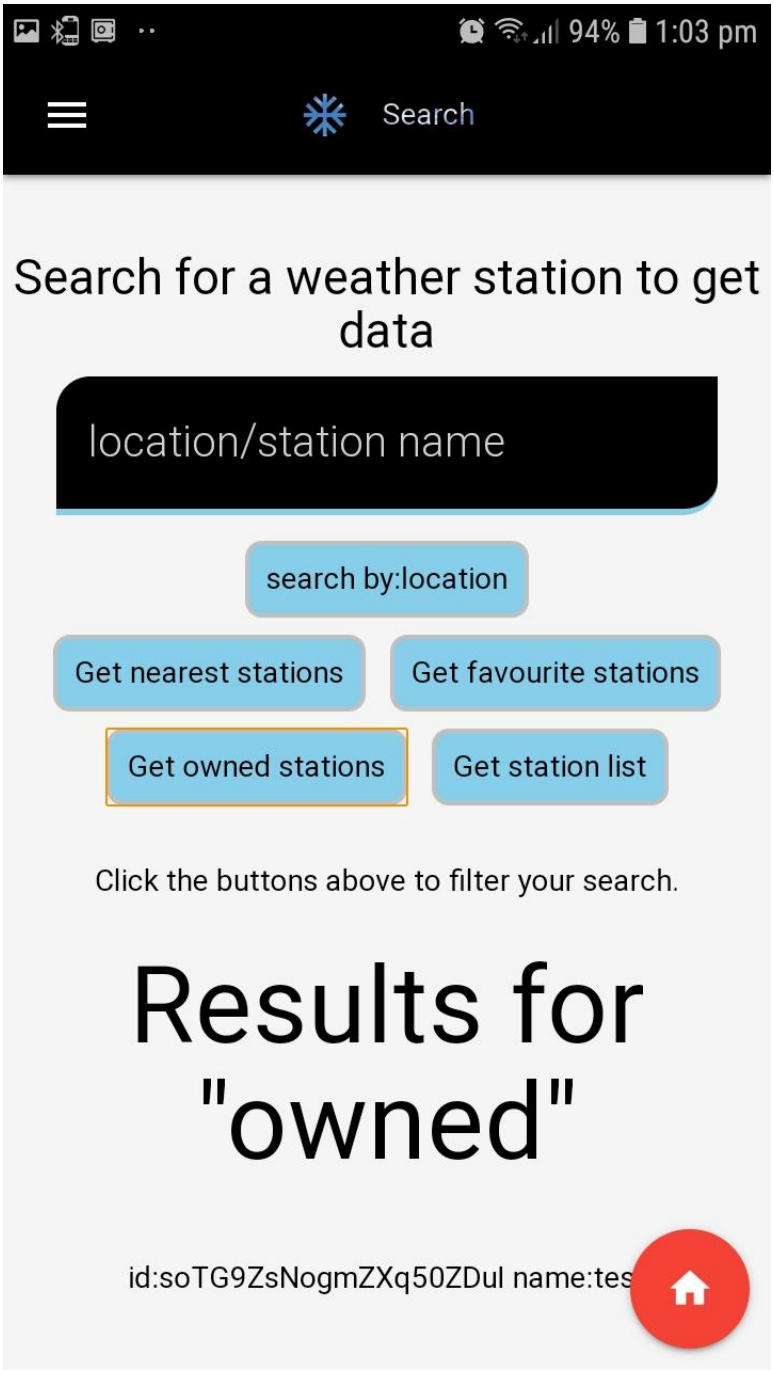
User redirected to search for stations after he is finished logging in.



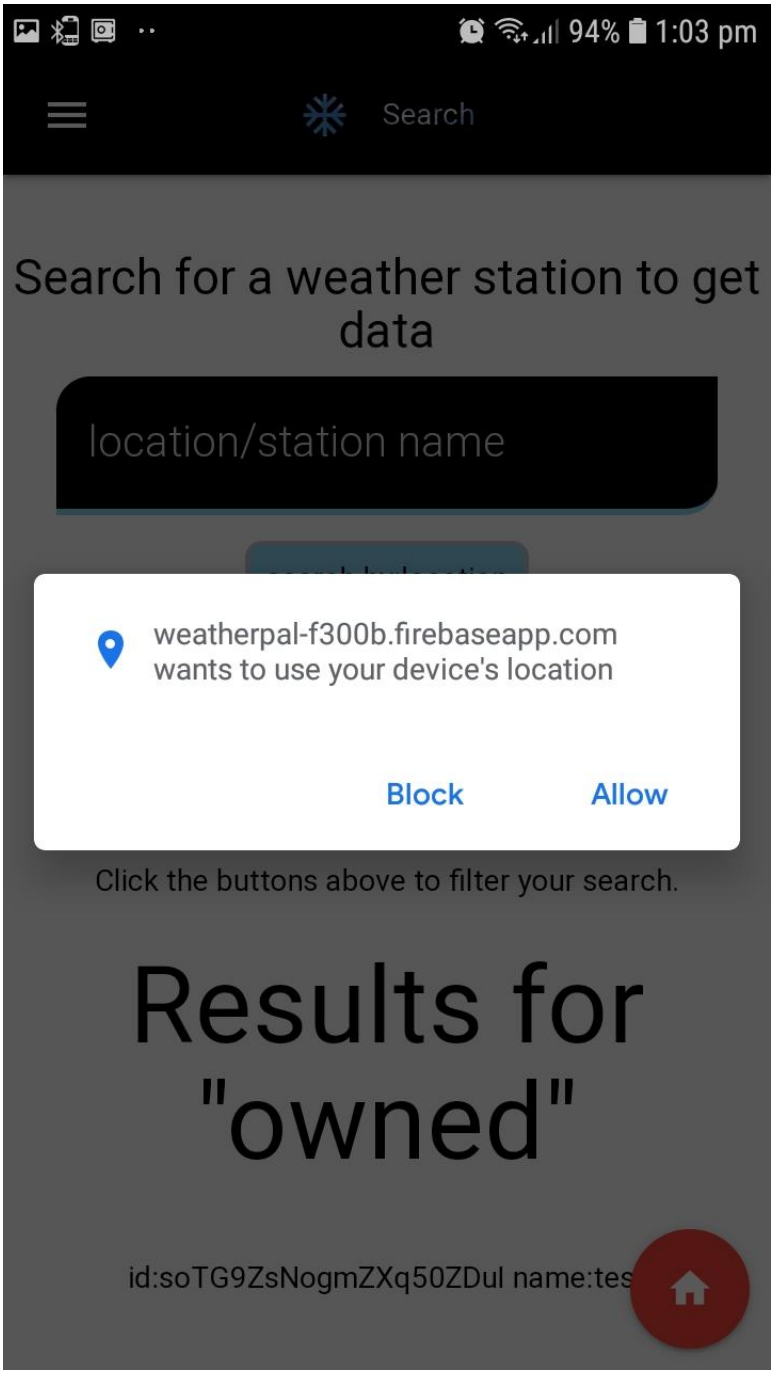
The user has selected to search for all stations.



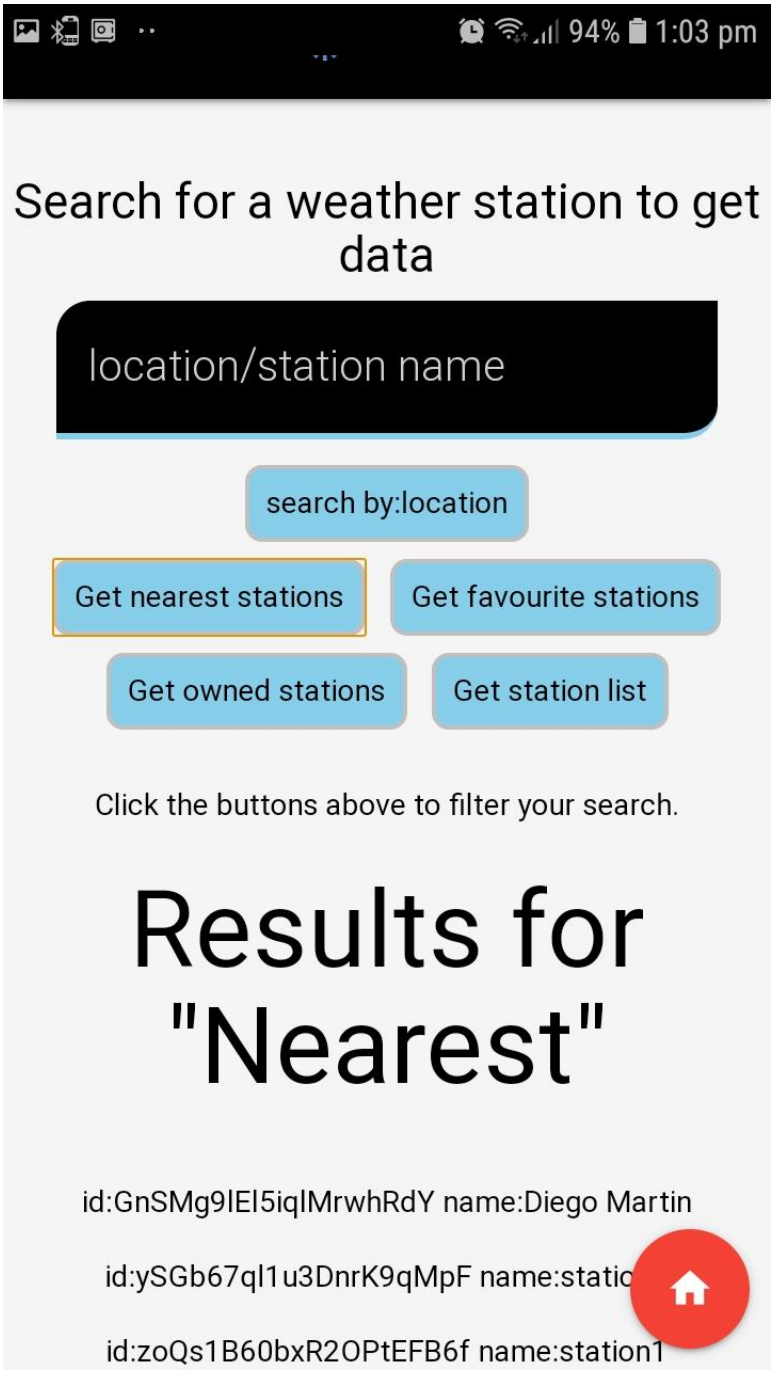
User has selected to search for his favourite stations.



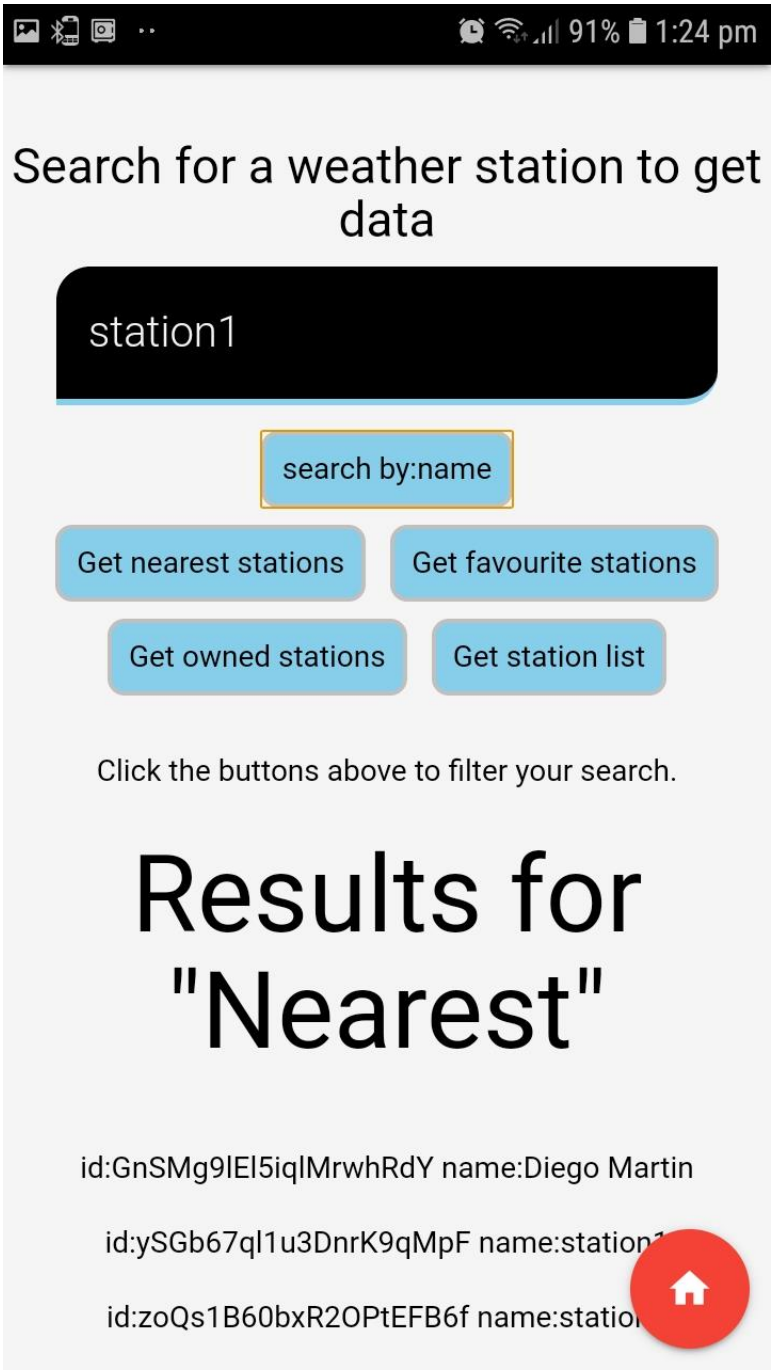
User has selected to search for stations that he owns.



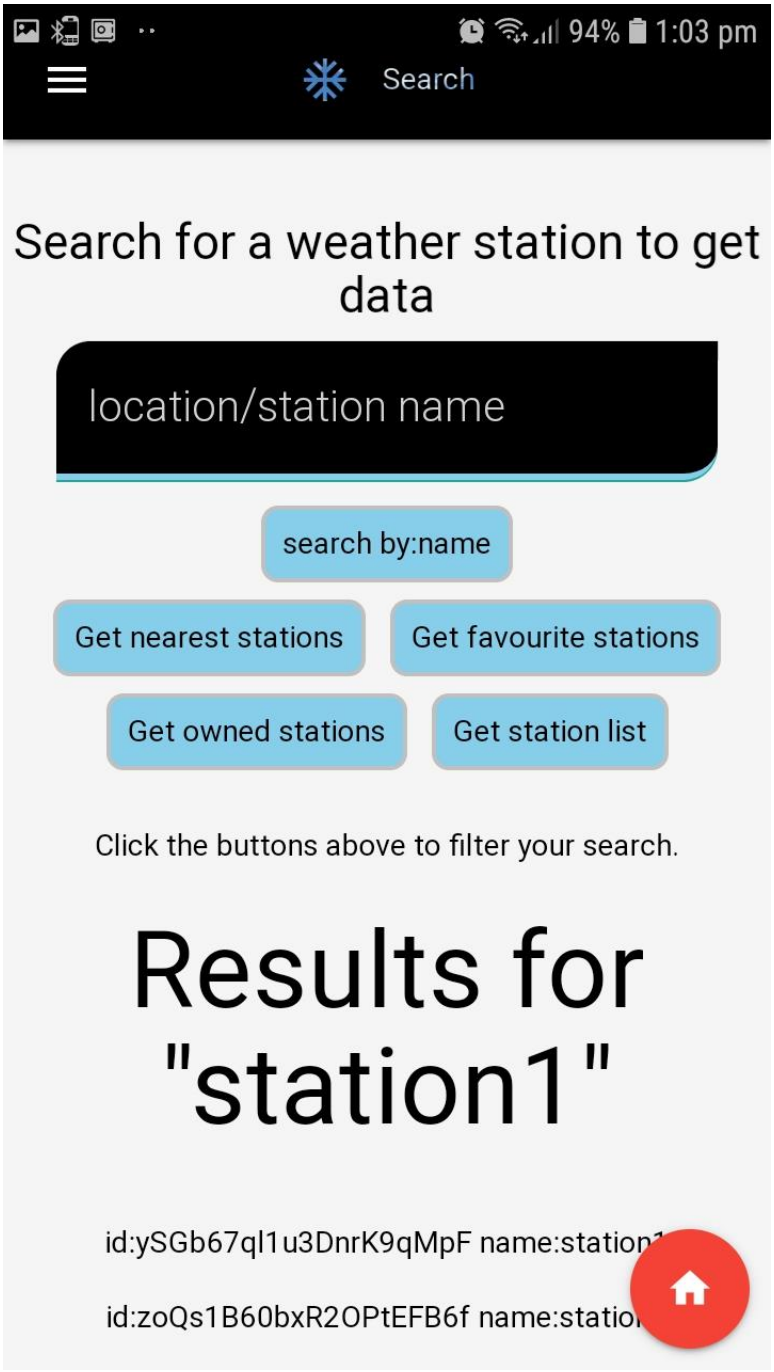
User selects to search for stations nearest to him and gets prompted for location features.



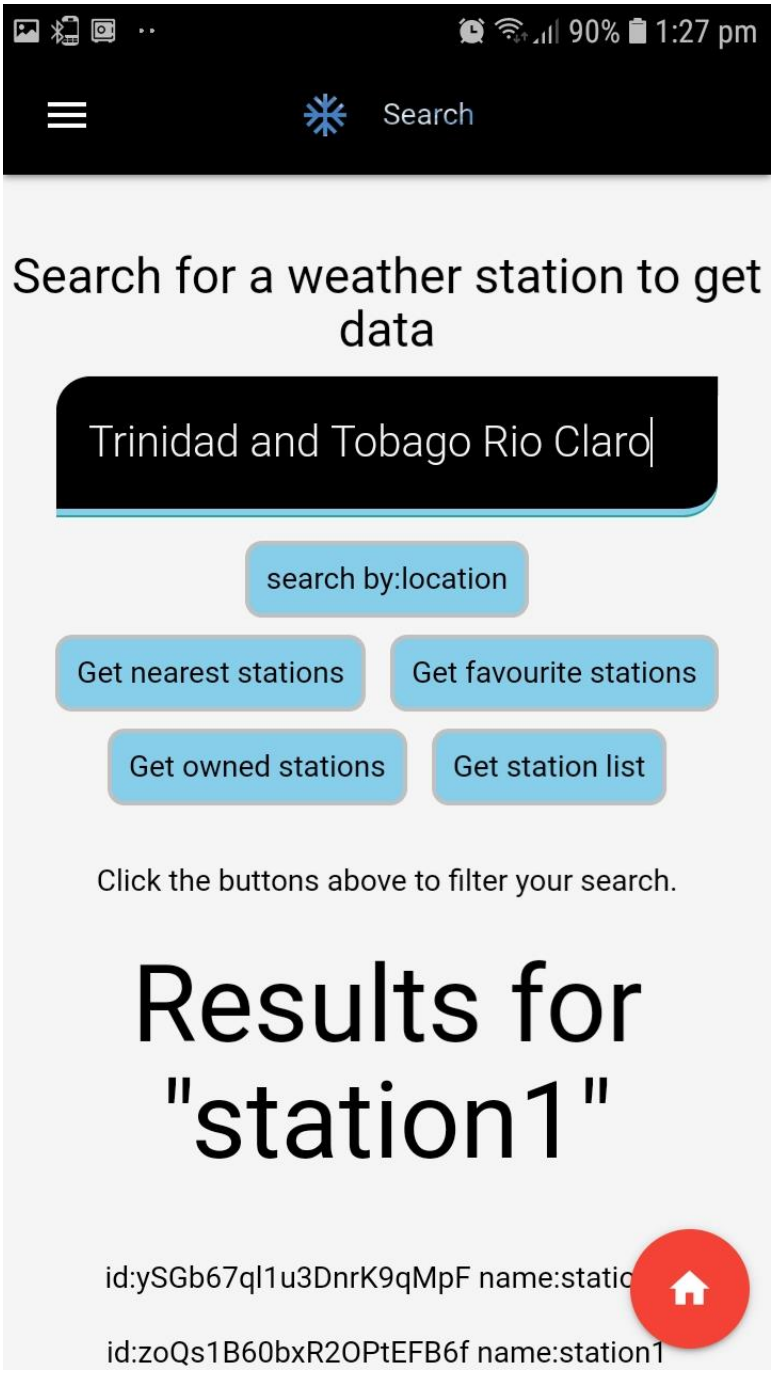
User has enabled locations for the app and gets the stations nearest to him.



User changes the search criteria in search by to name by clicking it and then types station1.



The user obtains a list of stations that match station1 after he presses enter.

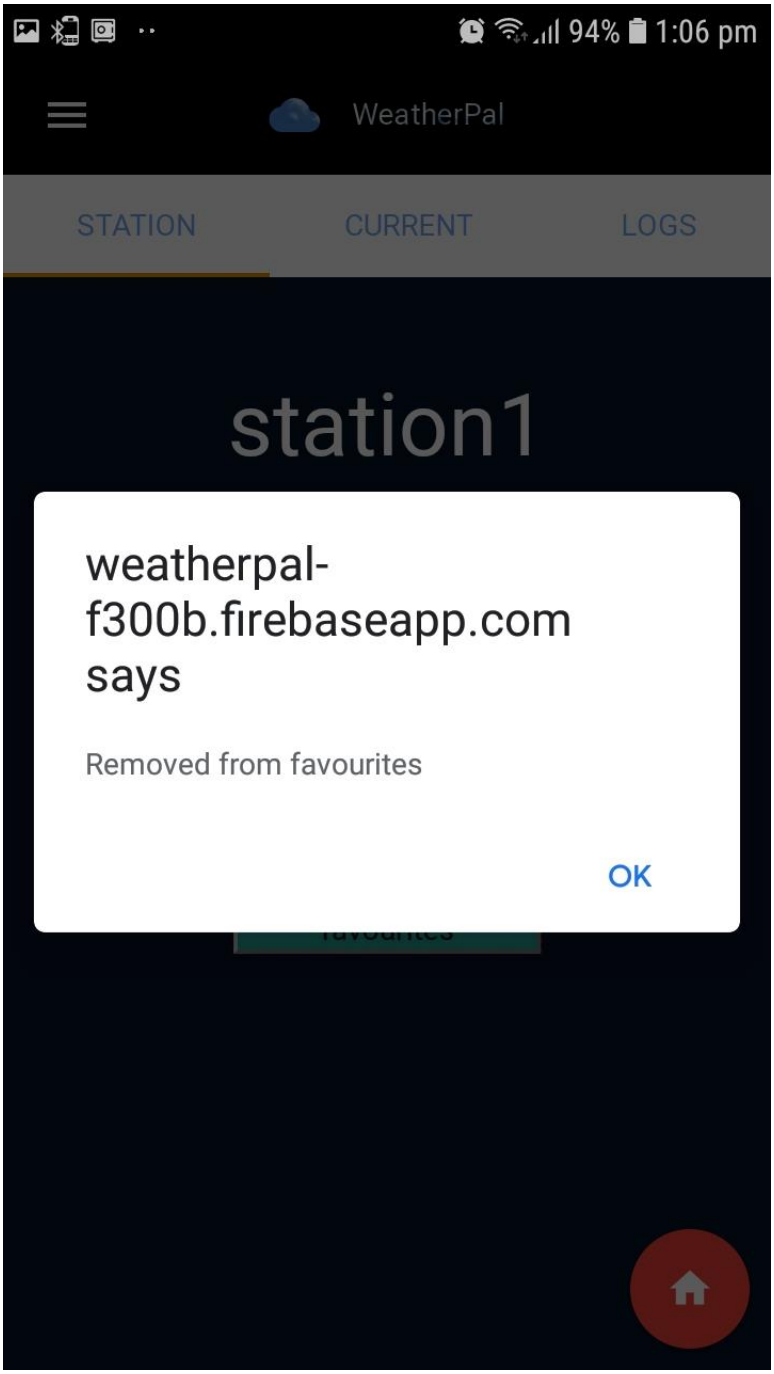


The user changes search by criteria to location and enters Trinidad and Tobago Rio Claro.

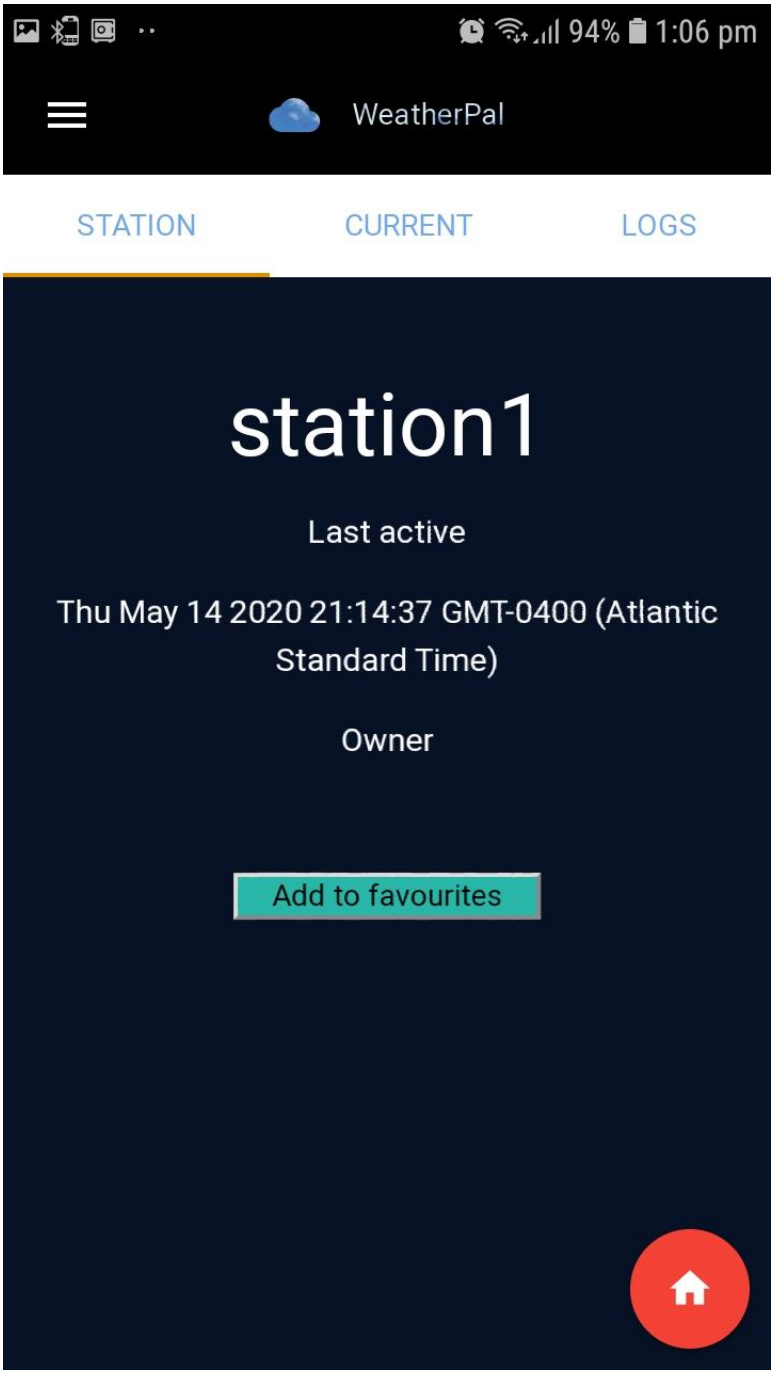
The user obtains a list of stations that are near rio claro(the result is wide as the range value is wide, which is adjustable).

After the user selects station1 from the search results the user is redirected to the Home page which has loaded in station data.

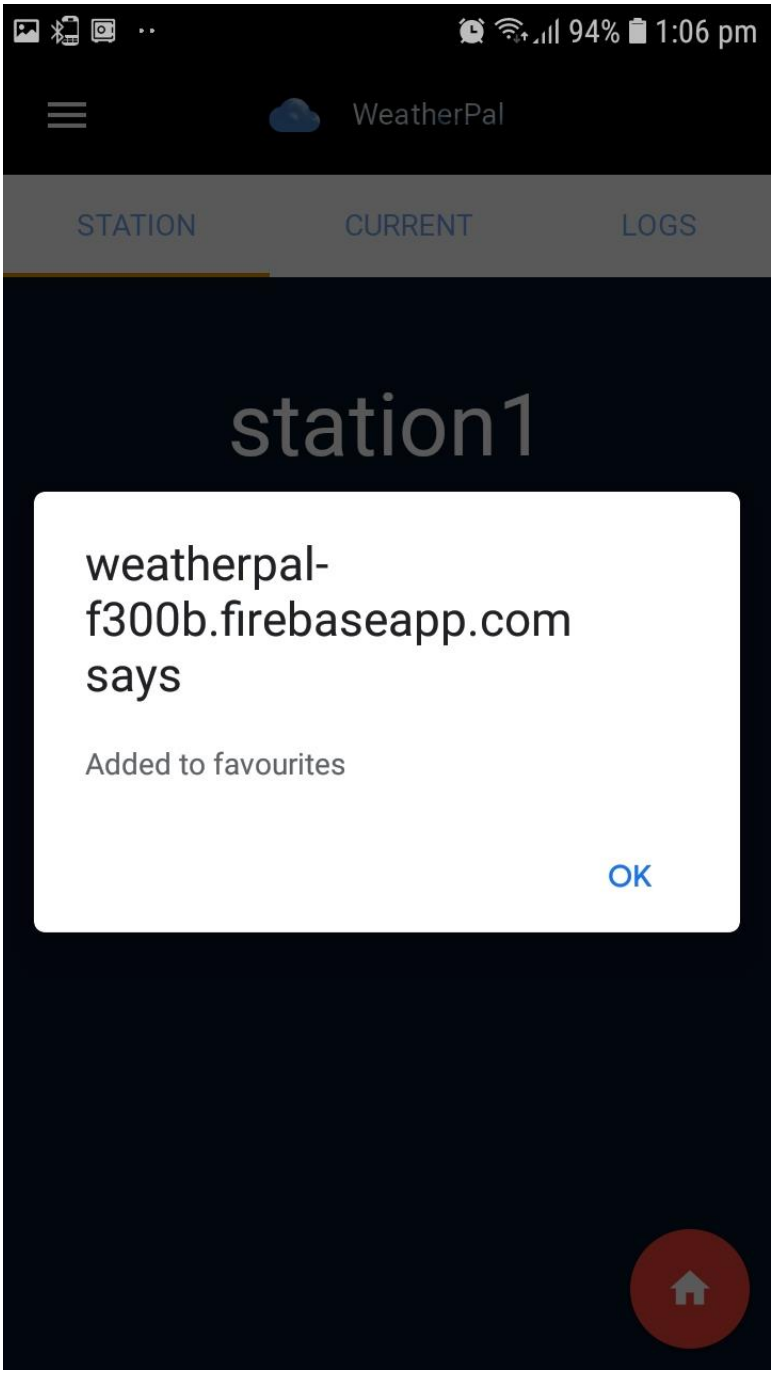
The stations tab, user has selected the tab by pressing it.



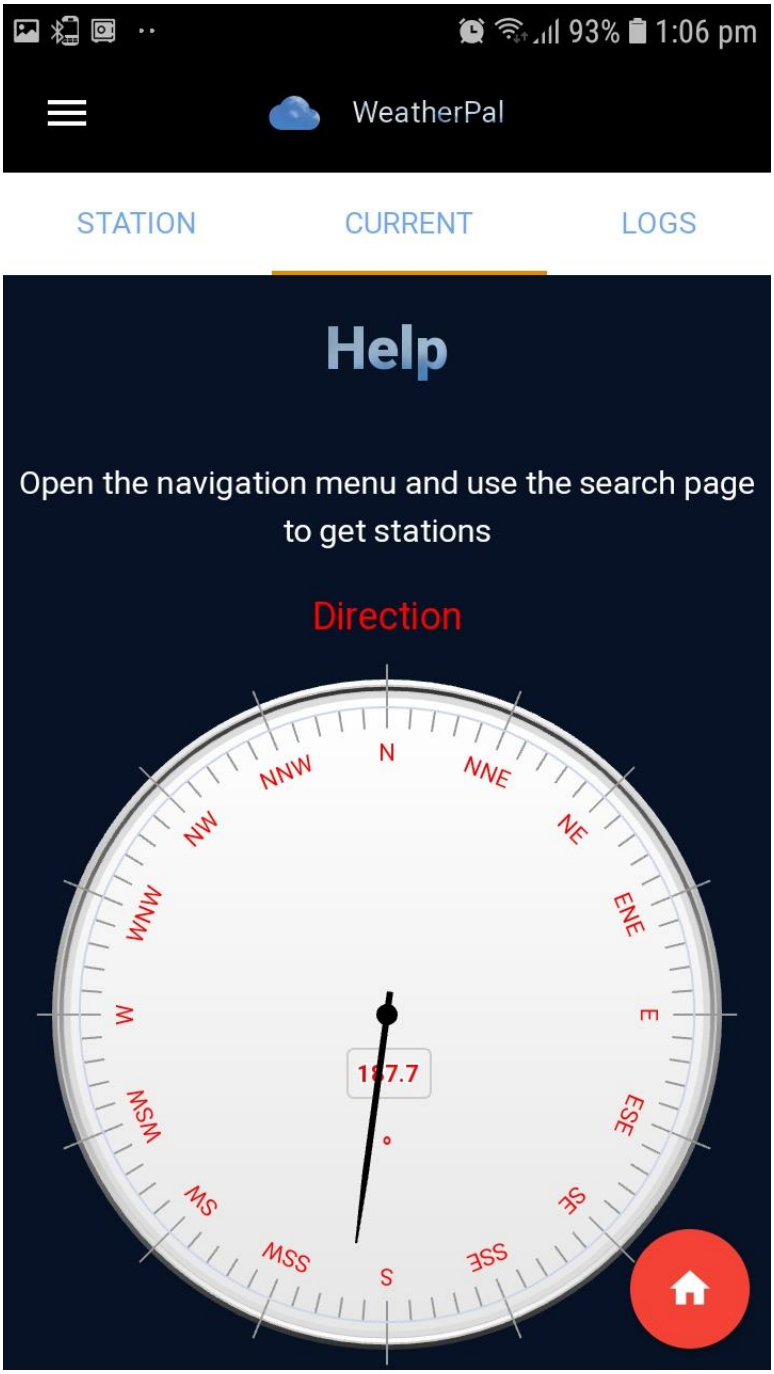
The user has selected to remove the station from favourites.



User has then selected to re-add to favourites.



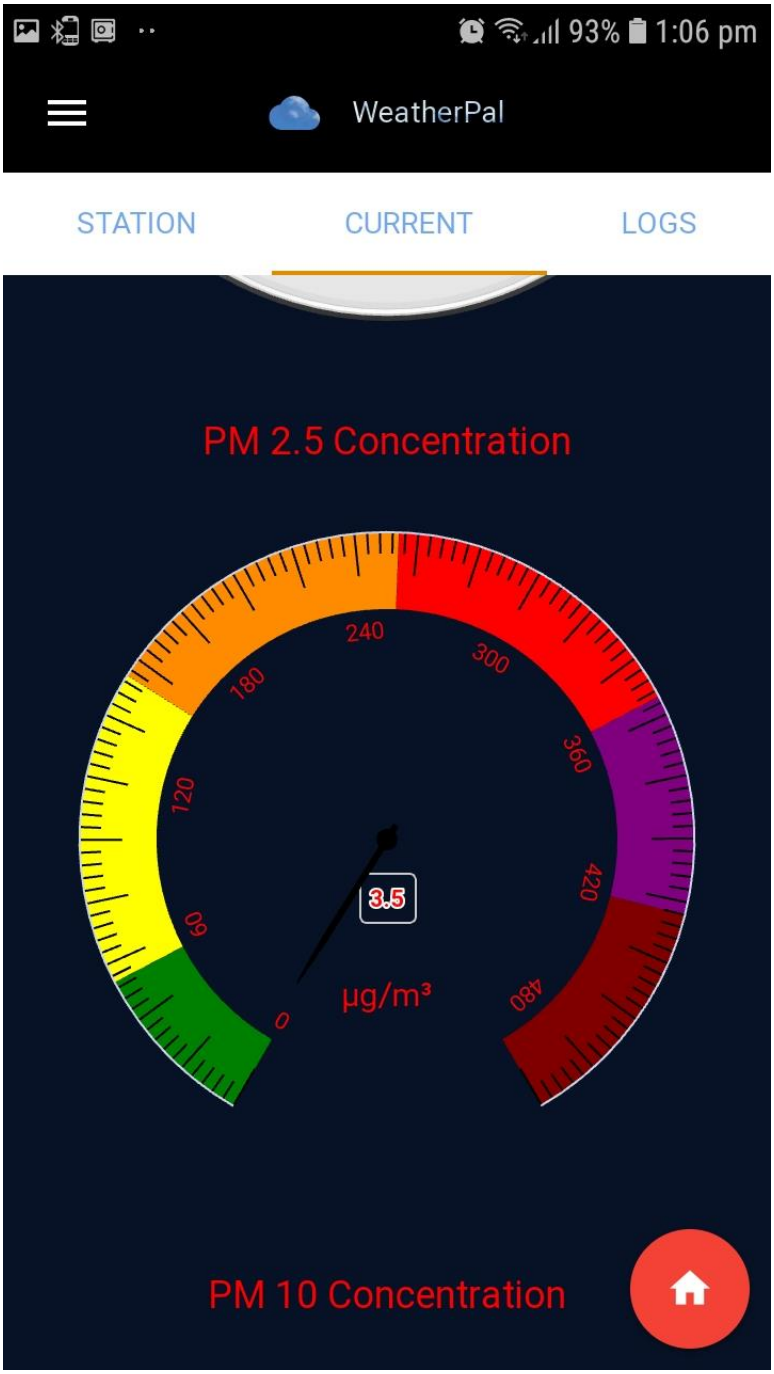
After the station is done adding to favourites this is displayed.

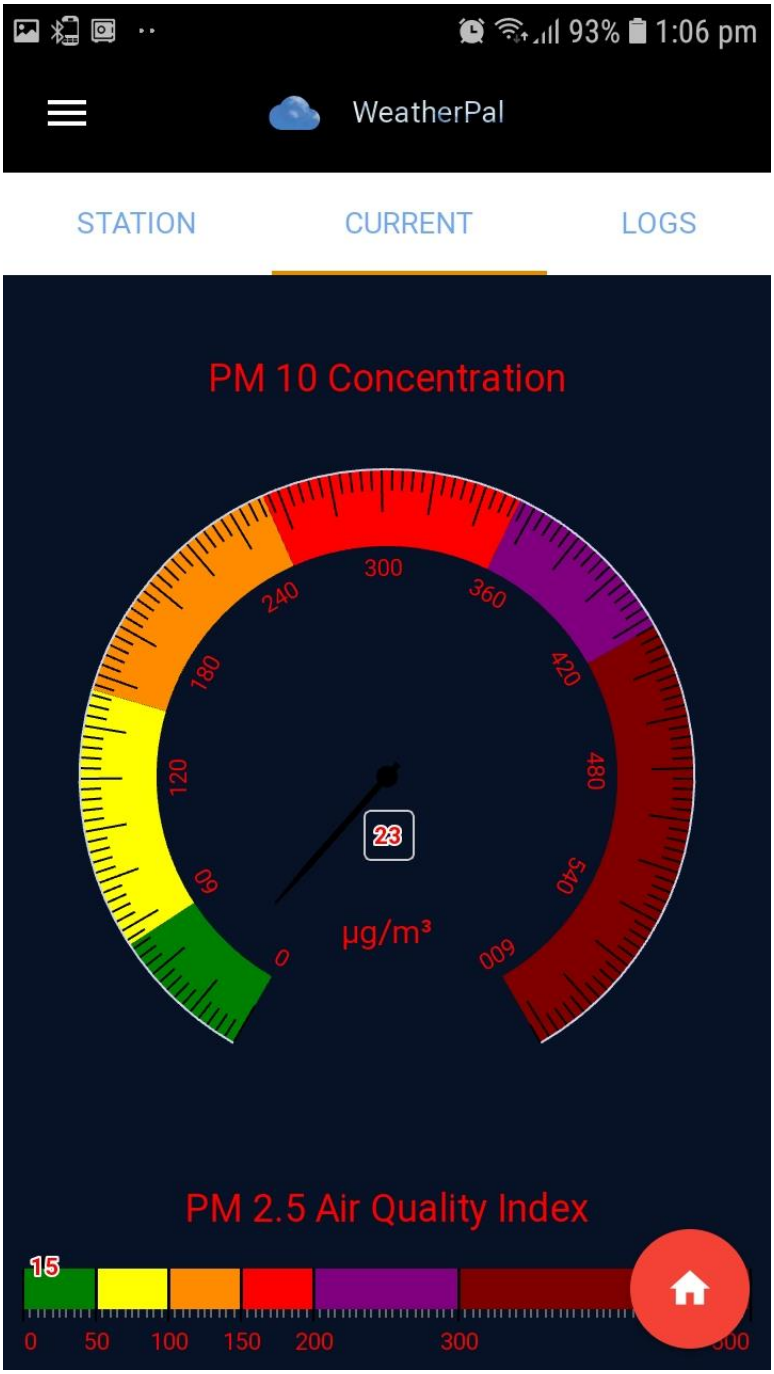


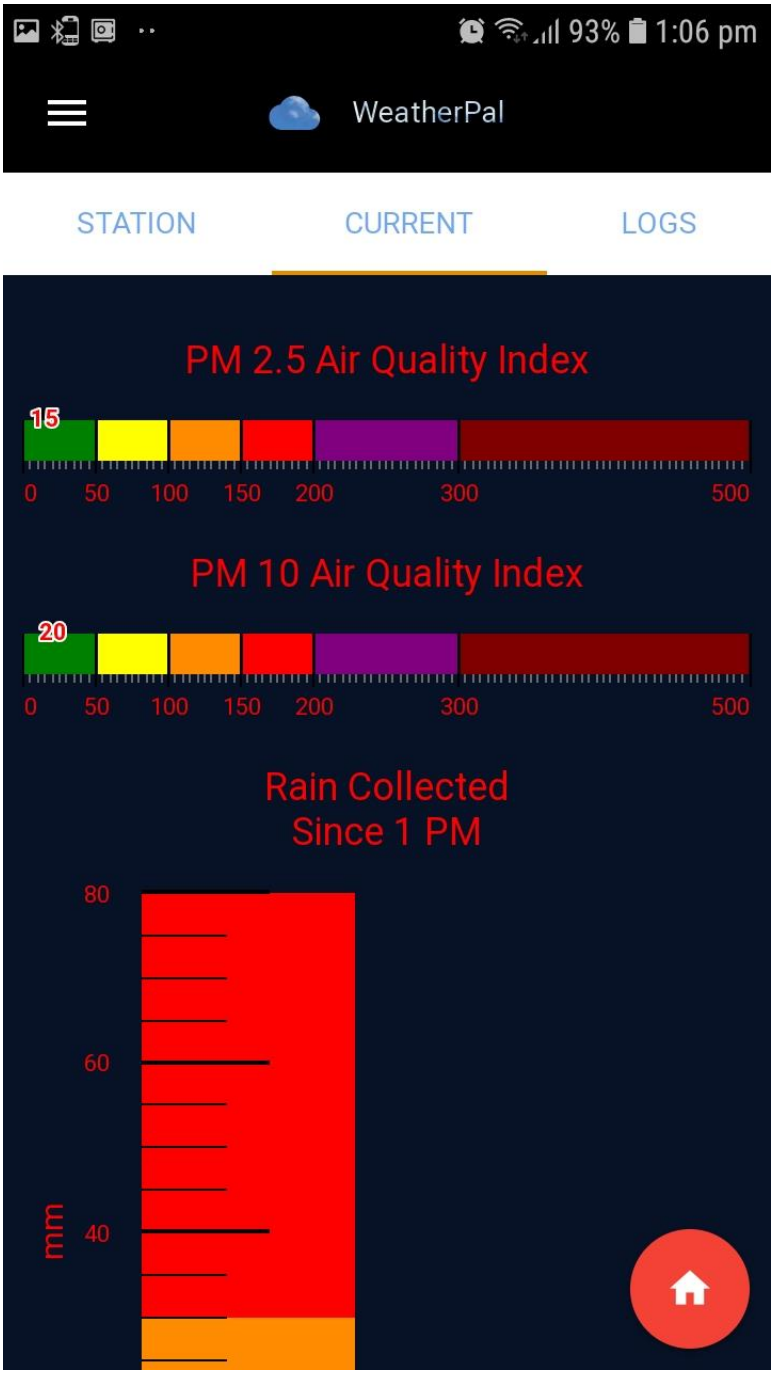
The user goes back to the current tab and scrolls down to the first gauge.

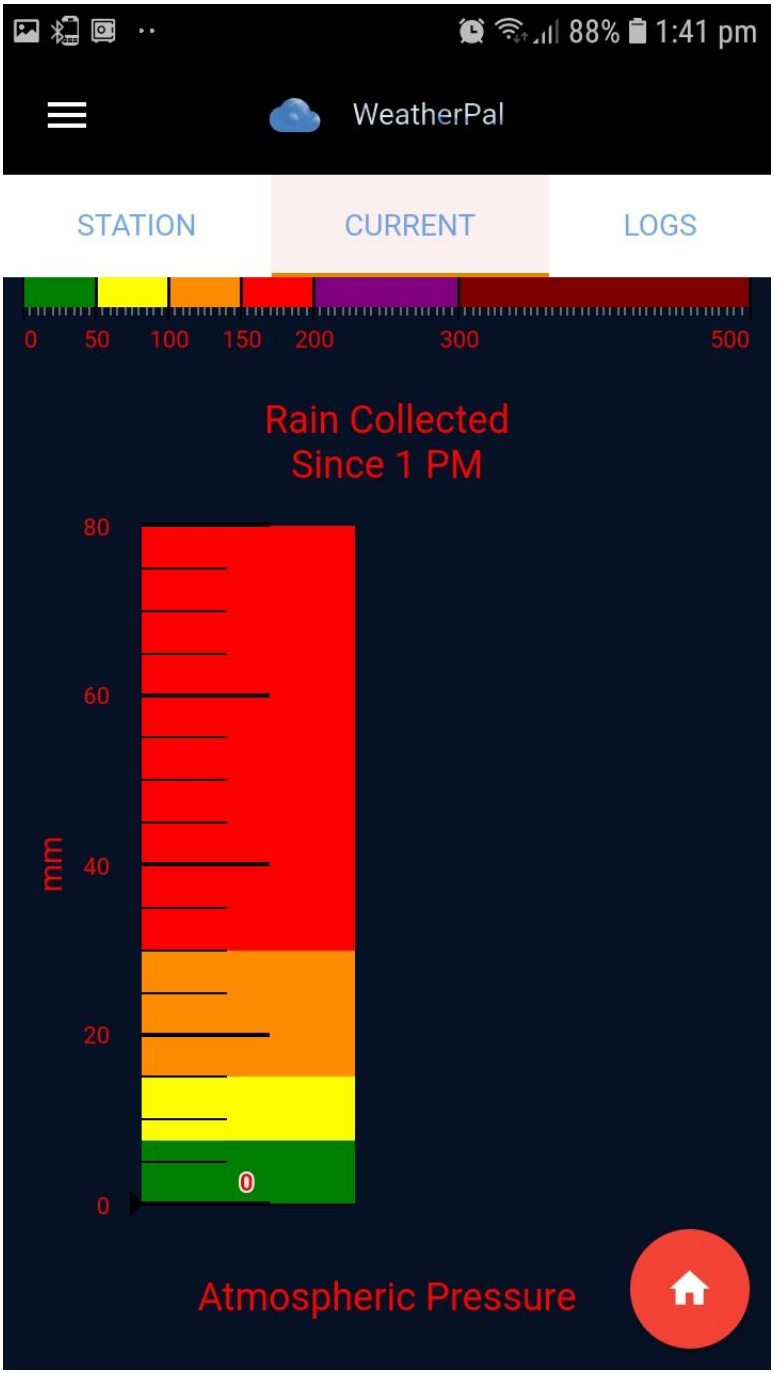
The user can continue to scroll to view more gauges.
Gauges displayed:

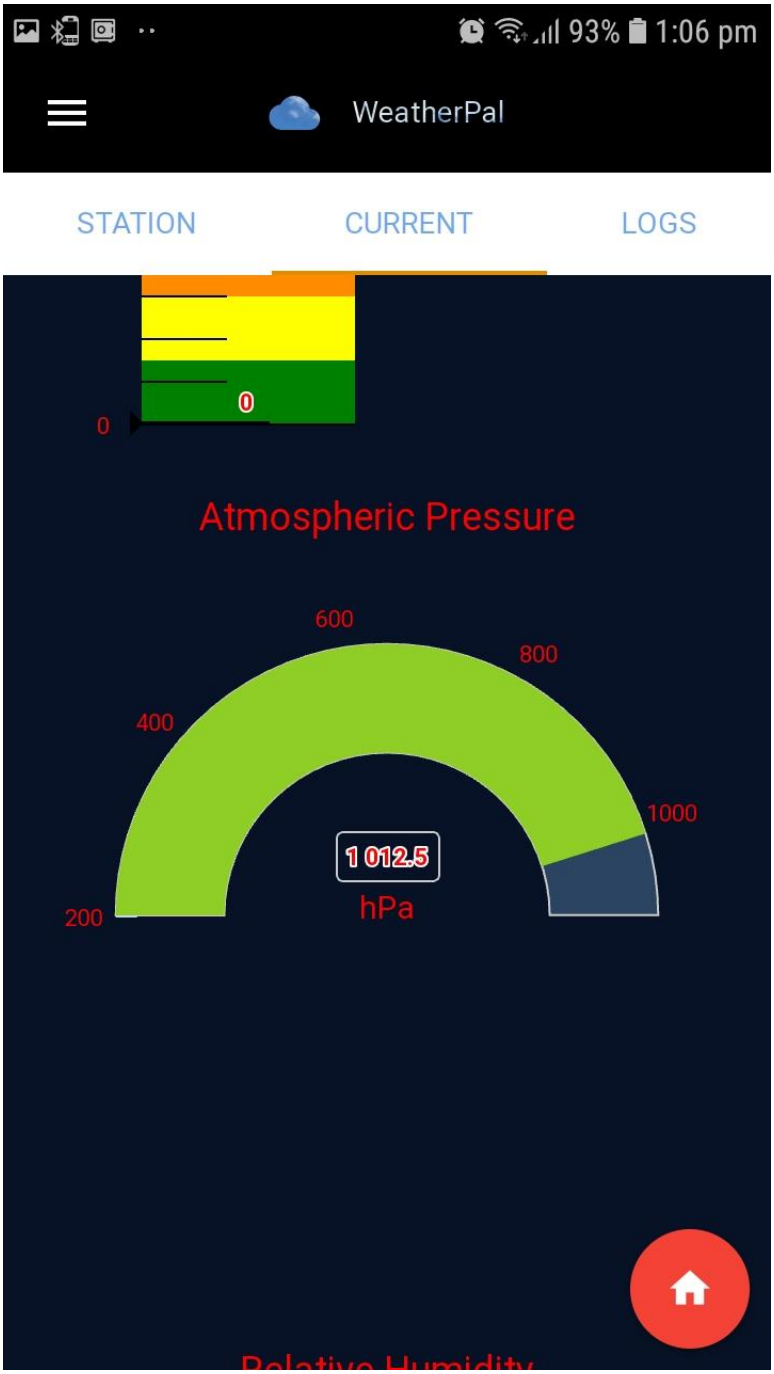

93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
Wind Speed
0
40
80
120
160
200
240
280
320
360
400
1.3
km/h
PM 2.5 Concentration

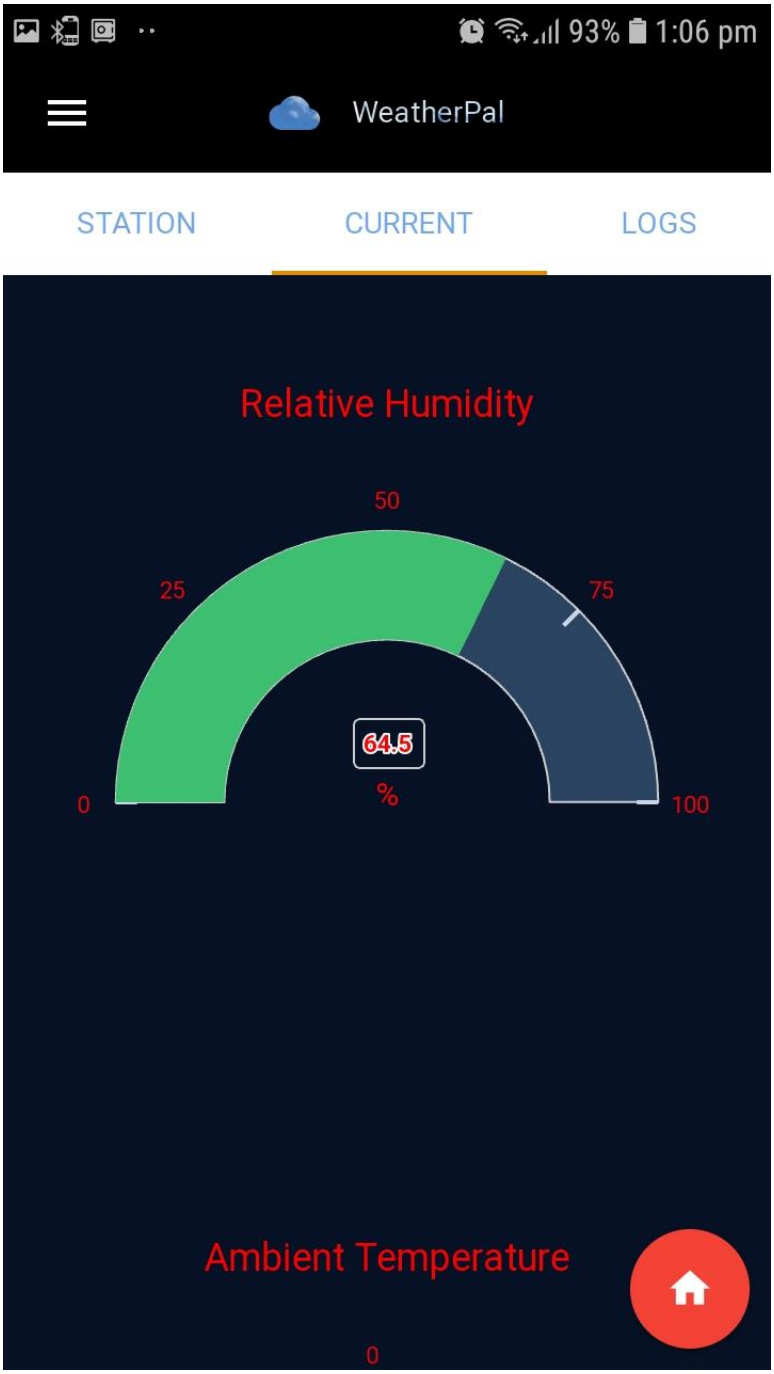

93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
PM 2.5 Concentration
0
60
120
180
240
300
360
420
480
3.5
µg/m³
PM 10 Concentration

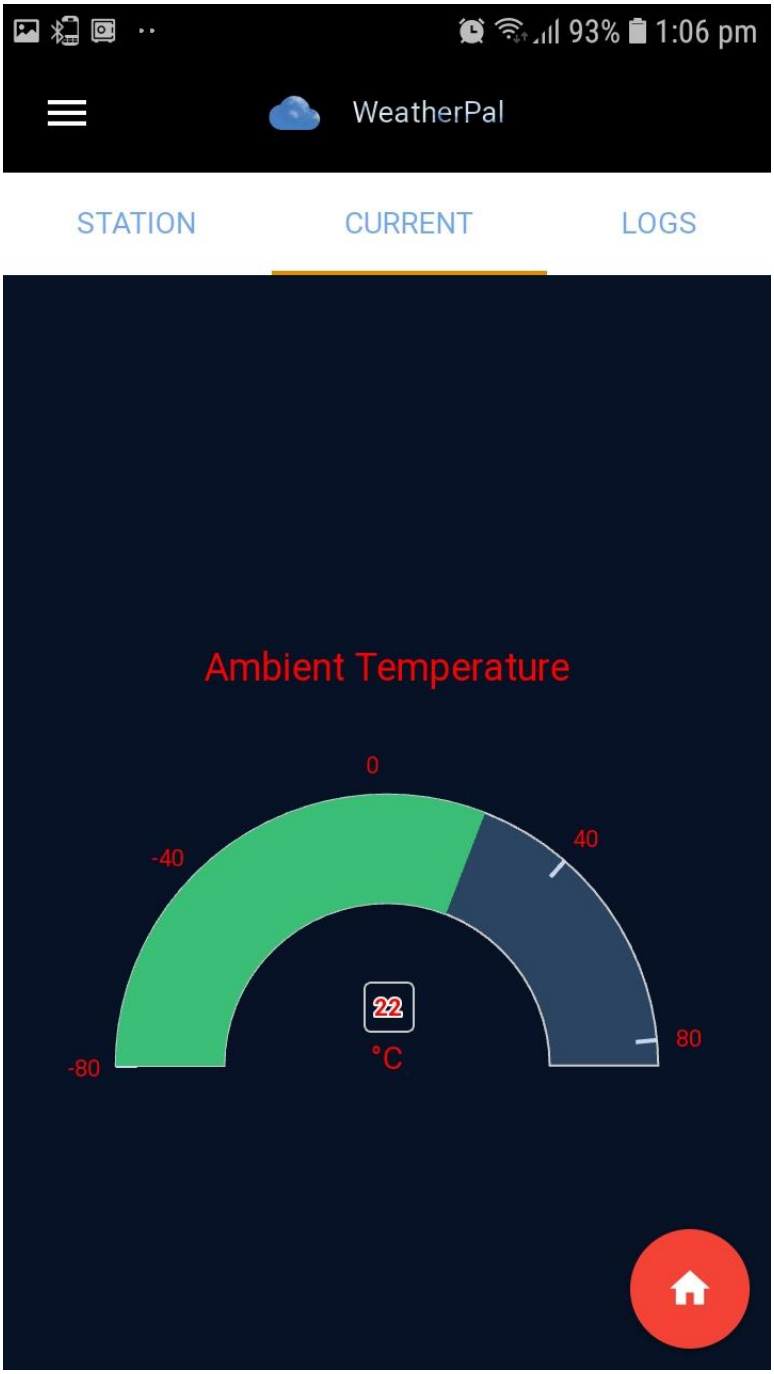

93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
PM 10 Concentration
0
60
120
180
240
300
360
420
480
540
600
23
µg/m³
PM 2.5 Air Quality Index
15
0
50
100
150
200
300

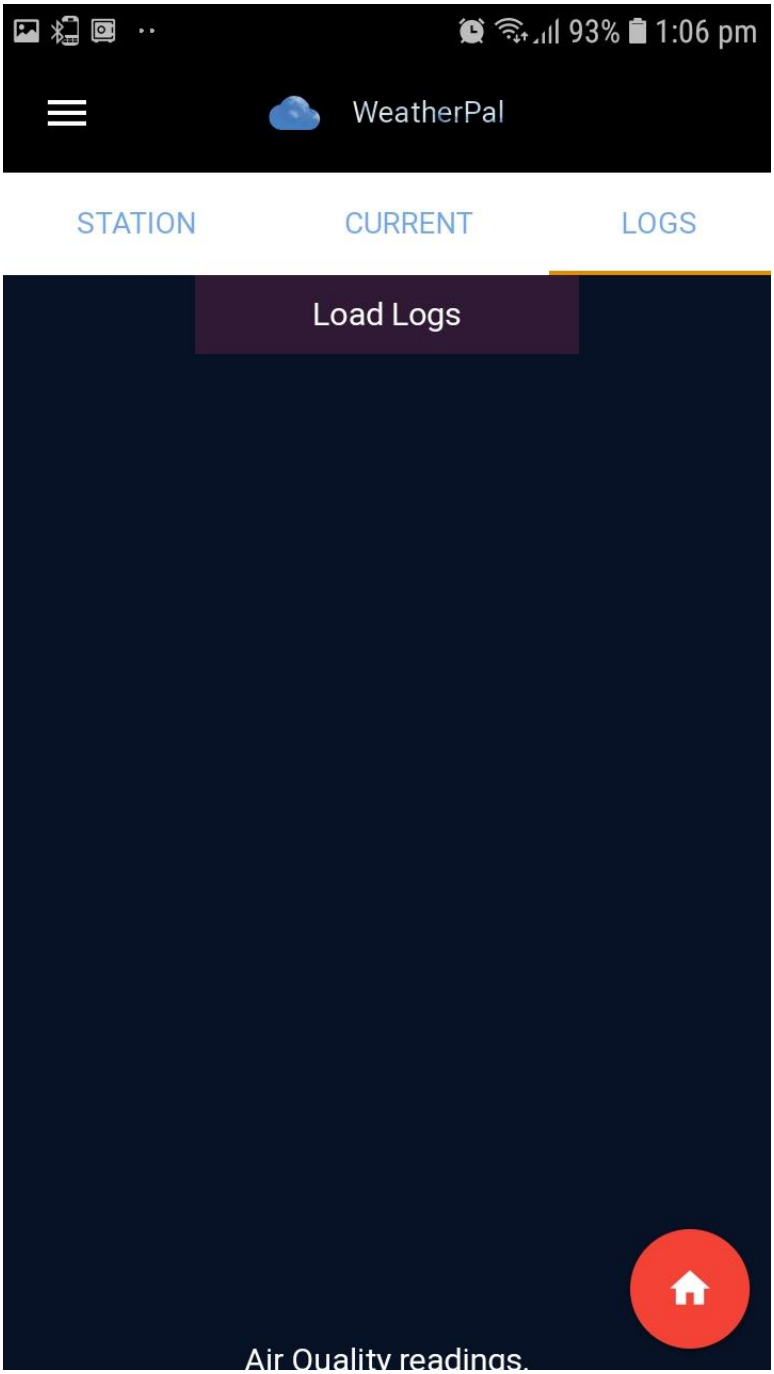

93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
PM 2.5 Air Quality Index
15
0 50 100 150 200 300 500
PM 10 Air Quality Index
20
0 50 100 150 200 300 500
Rain Collected
Since 1 PM
80
60
40
mm


88% 1:41 pm
WeatherPal
STATION
CURRENT
LOGS
0
50
100
150
200
300
500
Rain Collected
Since 1 PM
80
60
40
20
0
mm
0
Atmospheric Pressure

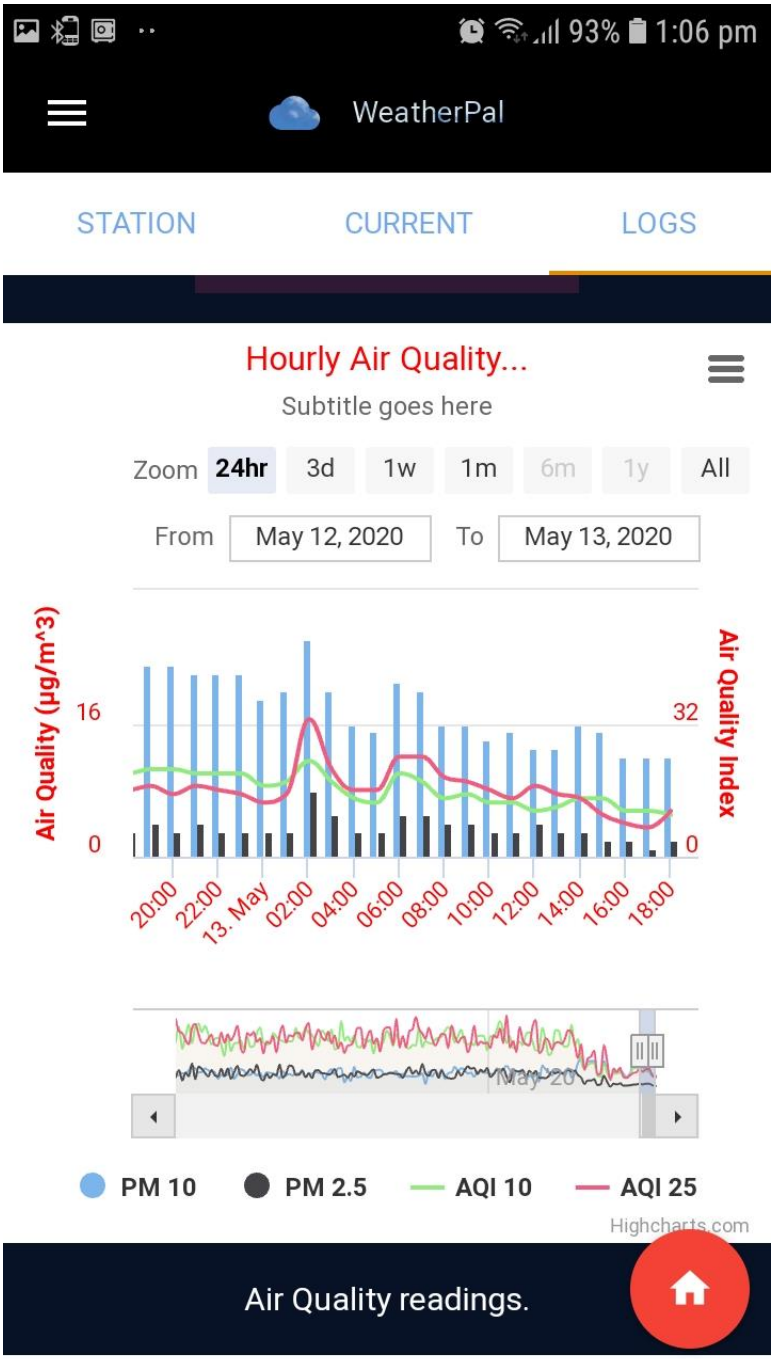

93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
0
0
Atmospheric Pressure
600
800
400
1000
1012.5
hPa
200
Relative Humidity


93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
Relative Humidity
50
25
75
64.5
%
0
100
Ambient Temperature
0


93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
Ambient Temperature
0
-40
40
-80
80
22
°C

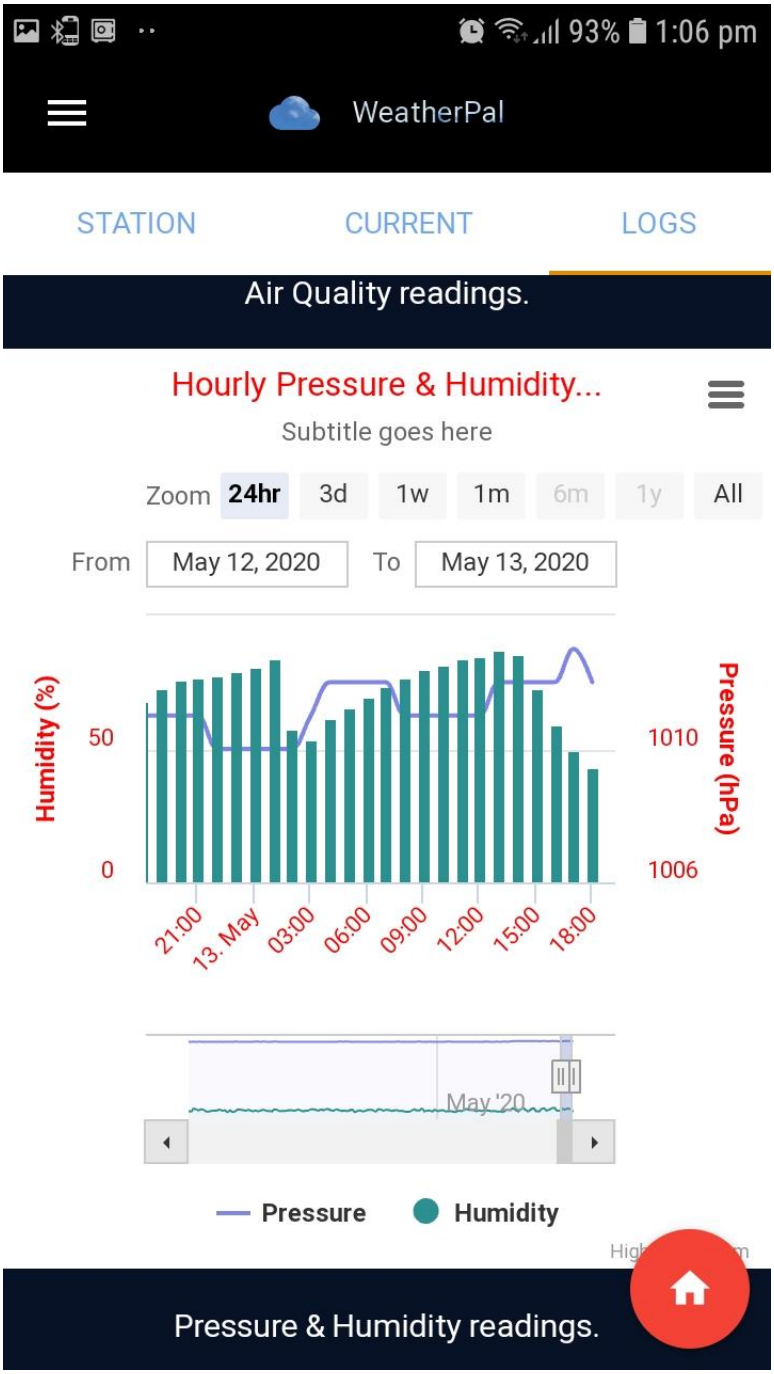


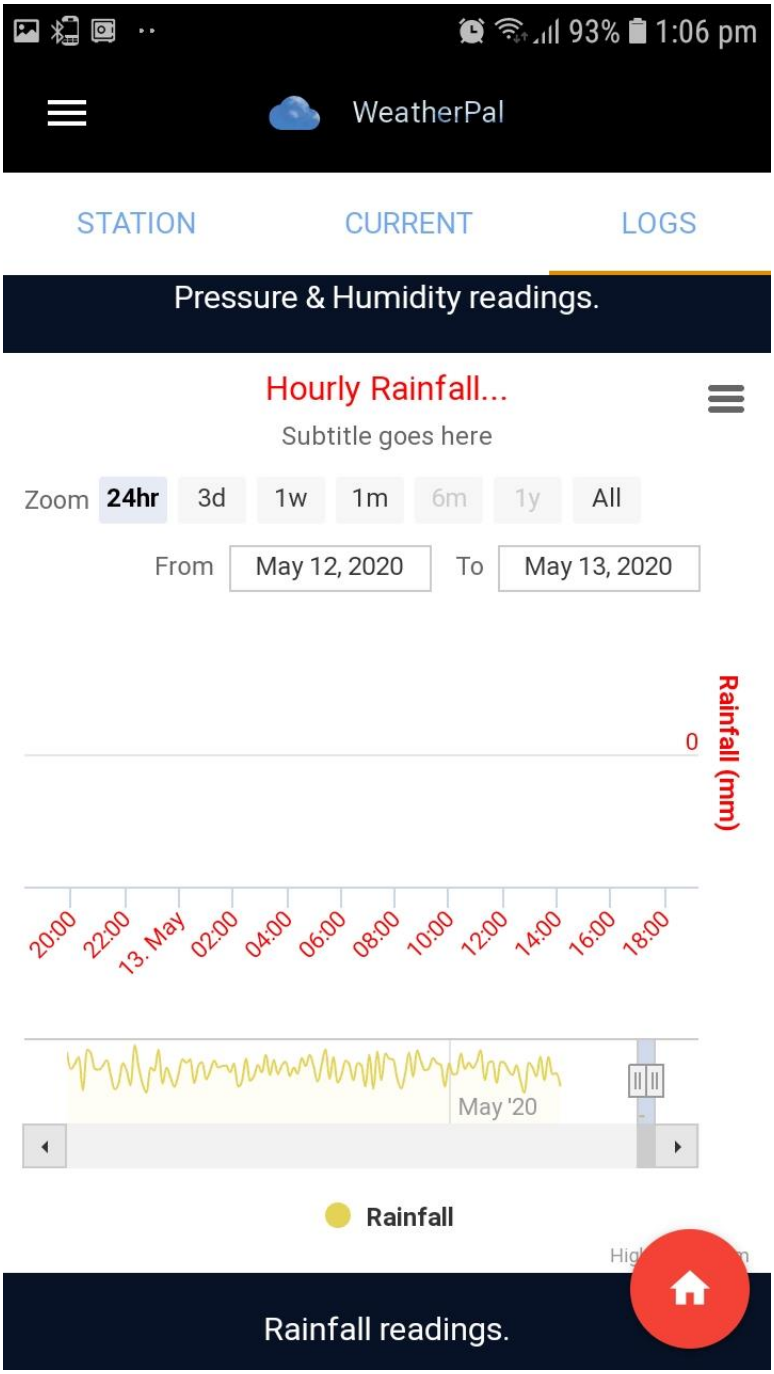
The user has selected to go to the logs screen and selects the load logs button.

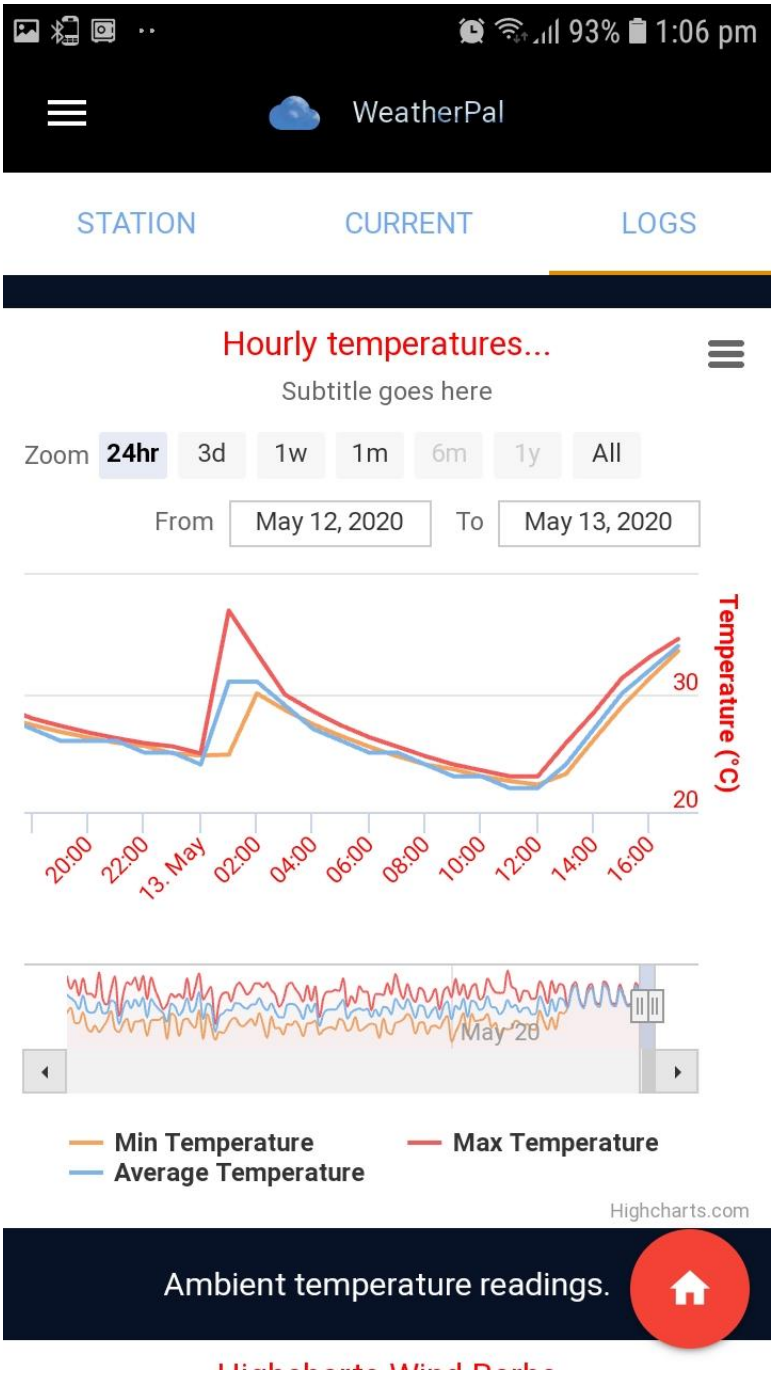


The first log chart displayed after the logs have loaded in.

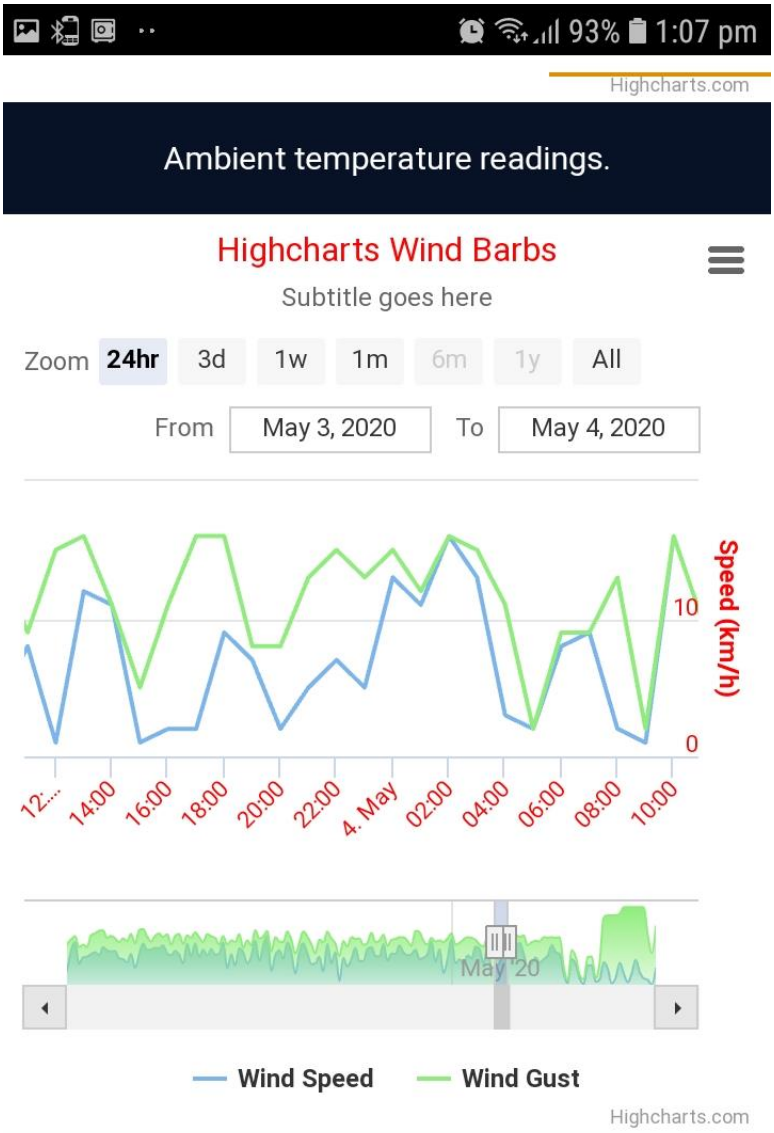
The user can scroll down to view log charts.
Log charts displayed:

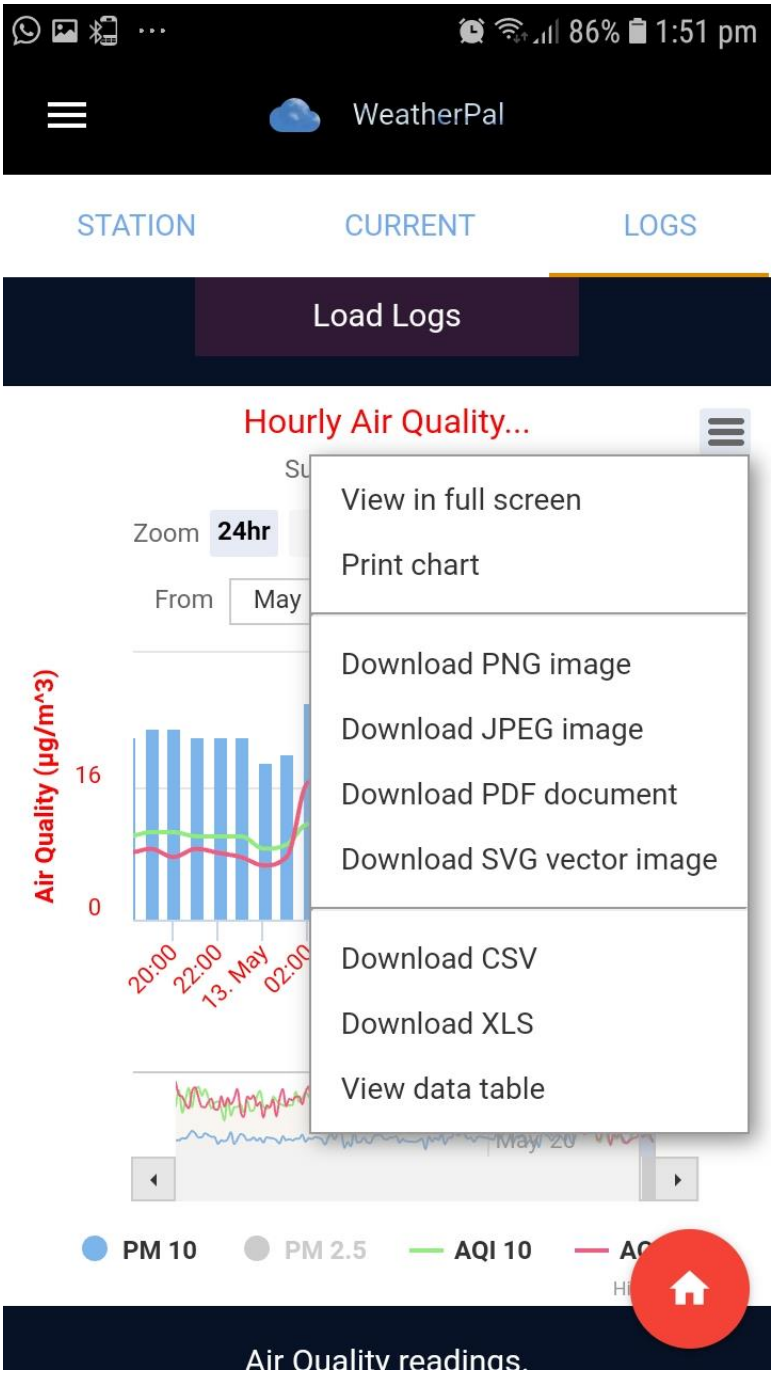

93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
Air Quality readings.
Hourly Pressure & Humidity...
Subtitle goes here
Zoom 24hr 3d 1w 1m 6m 1y All
From May 12, 2020 To May 13, 2020
Humidity (%)
50
0
1010
1006
Pressure (hPa)
21:00
13. May
03:00
06:00
09:00
12:00
15:00
18:00
May '20
Pressure
Humidity
Pressure & Humidity readings.


93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
Pressure & Humidity readings.
Hourly Rainfall...
Subtitle goes here
Zoom 24hr 3d 1w 1m 6m 1y All
From May 12, 2020 To May 13, 2020
Rainfall (mm)
0
20:00
22:00
13. May
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
May '20
Rainfall
Rainfall readings.

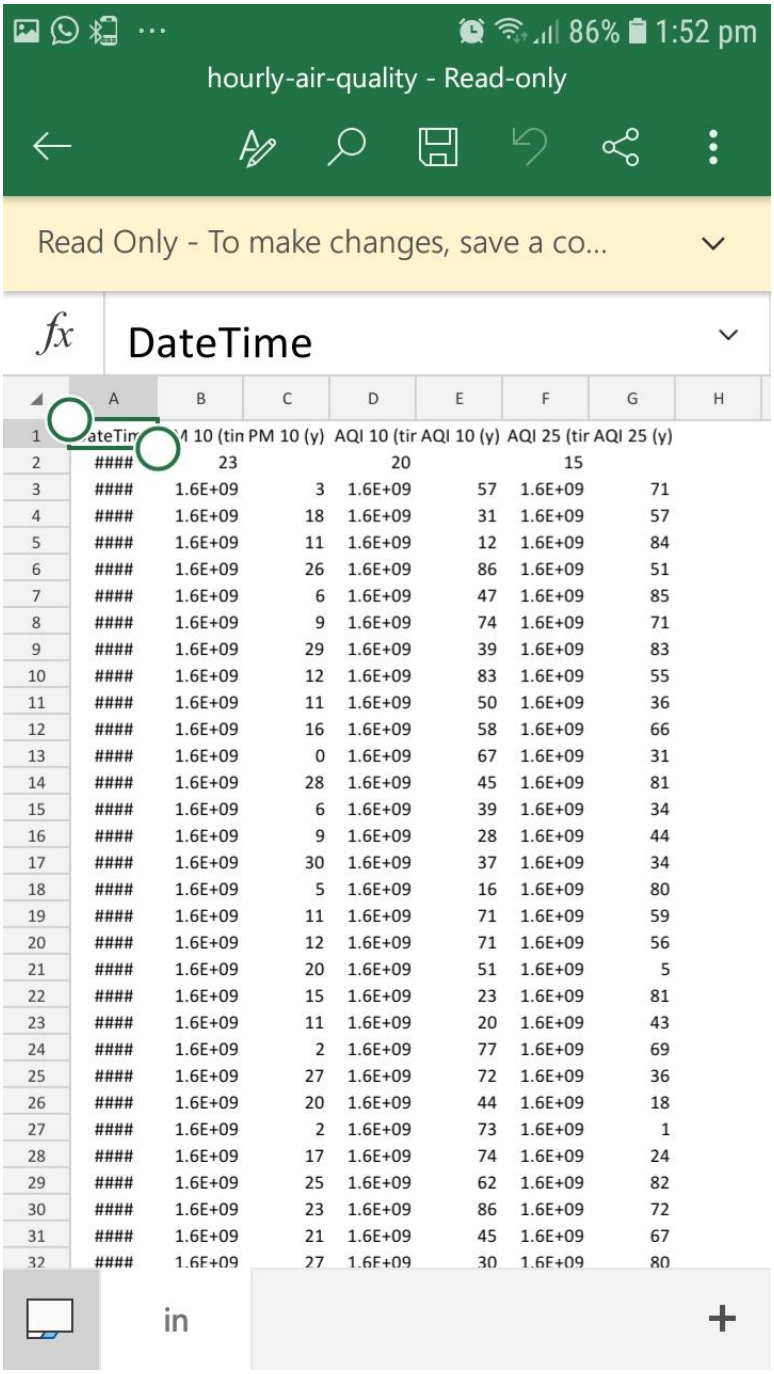

93% 1:06 pm
WeatherPal
STATION
CURRENT
LOGS
Hourly temperatures...
Subtitle goes here
Zoom 24hr 3d 1w 1m 6m 1y All
From May 12, 2020 To May 13, 2020
Temperature (°C)
30
20
20:00
22:00
13. May
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
May '20
Min Temperature
Max Temperature
Average Temperature
Highcharts.com
Ambient temperature readings.


93% 1:06 pm
STATION
CURRENT
LOGS
Highcharts.com
Ambient temperature readings.
Highcharts Wind Barbs
Subtitle goes here
Zoom 24hr 3d 1w 1m 6m 1y All
From May 12, 2020 To May 13, 2020
Speed (km/h)
20
10
0
20:00
22:00
13. May
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
May '20
Wind Speed
Wind Gust
Highcharts.com
Wind barbs are used to visualize wind direc
speed on a chart. As seen in this example,
often combined with other series types to provide


93% 1:07 pm
Highcharts.com
Ambient temperature readings.
Highcharts Wind Barbs
Subtitle goes here
Zoom 24hr 3d 1w 1m 6m 1y All
From May 3, 2020 To May 4, 2020
Speed (km/h)
10
0
12:...
14:00
16:00
18:00
20:00
22:00
4. May
02:00
04:00
06:00
08:00
10:00
May '20
Wind Speed
Wind Gust
Highcharts.com
Wind barbs are used to visualize wind direction and speed on a chart. As seen in this example, th re often combined with other series types to additional information.

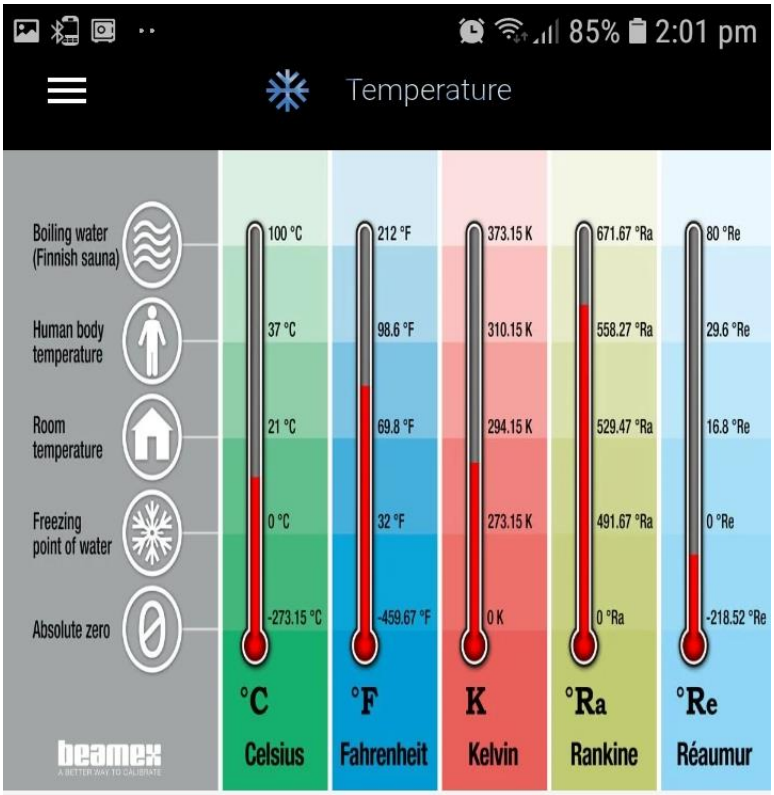


The user selects the bars in the top right corner and is greeted with a menu where he can download log data.



	A	B	C	D	E	F	G	H
1	ateTim	M 10 (tin	PM 10 (y)	AQI 10 (tir	AQI 10 (y)	AQI 25 (tir	AQI 25 (y)	
2	####	23		20		15		
3	####	1.6E+09	3	1.6E+09	57	1.6E+09	71	
4	####	1.6E+09	18	1.6E+09	31	1.6E+09	57	
5	####	1.6E+09	11	1.6E+09	12	1.6E+09	84	
6	####	1.6E+09	26	1.6E+09	86	1.6E+09	51	
7	####	1.6E+09	6	1.6E+09	47	1.6E+09	85	
8	####	1.6E+09	9	1.6E+09	74	1.6E+09	71	
9	####	1.6E+09	29	1.6E+09	39	1.6E+09	83	
10	####	1.6E+09	12	1.6E+09	83	1.6E+09	55	
11	####	1.6E+09	11	1.6E+09	50	1.6E+09	36	
12	####	1.6E+09	16	1.6E+09	58	1.6E+09	66	
13	####	1.6E+09	0	1.6E+09	67	1.6E+09	31	
14	####	1.6E+09	28	1.6E+09	45	1.6E+09	81	
15	####	1.6E+09	6	1.6E+09	39	1.6E+09	34	
16	####	1.6E+09	9	1.6E+09	28	1.6E+09	44	
17	####	1.6E+09	30	1.6E+09	37	1.6E+09	34	
18	####	1.6E+09	5	1.6E+09	16	1.6E+09	80	
19	####	1.6E+09	11	1.6E+09	71	1.6E+09	59	
20	####	1.6E+09	12	1.6E+09	71	1.6E+09	56	
21	####	1.6E+09	20	1.6E+09	51	1.6E+09	5	
22	####	1.6E+09	15	1.6E+09	23	1.6E+09	81	
23	####	1.6E+09	11	1.6E+09	20	1.6E+09	43	
24	####	1.6E+09	2	1.6E+09	77	1.6E+09	69	
25	####	1.6E+09	27	1.6E+09	72	1.6E+09	36	
26	####	1.6E+09	20	1.6E+09	44	1.6E+09	18	
27	####	1.6E+09	2	1.6E+09	73	1.6E+09	1	
28	####	1.6E+09	17	1.6E+09	74	1.6E+09	24	
29	####	1.6E+09	25	1.6E+09	62	1.6E+09	82	
30	####	1.6E+09	23	1.6E+09	86	1.6E+09	72	
31	####	1.6E+09	21	1.6E+09	45	1.6E+09	67	
32	####	1.6E+09	27	1.6E+09	30	1.6E+09	80	

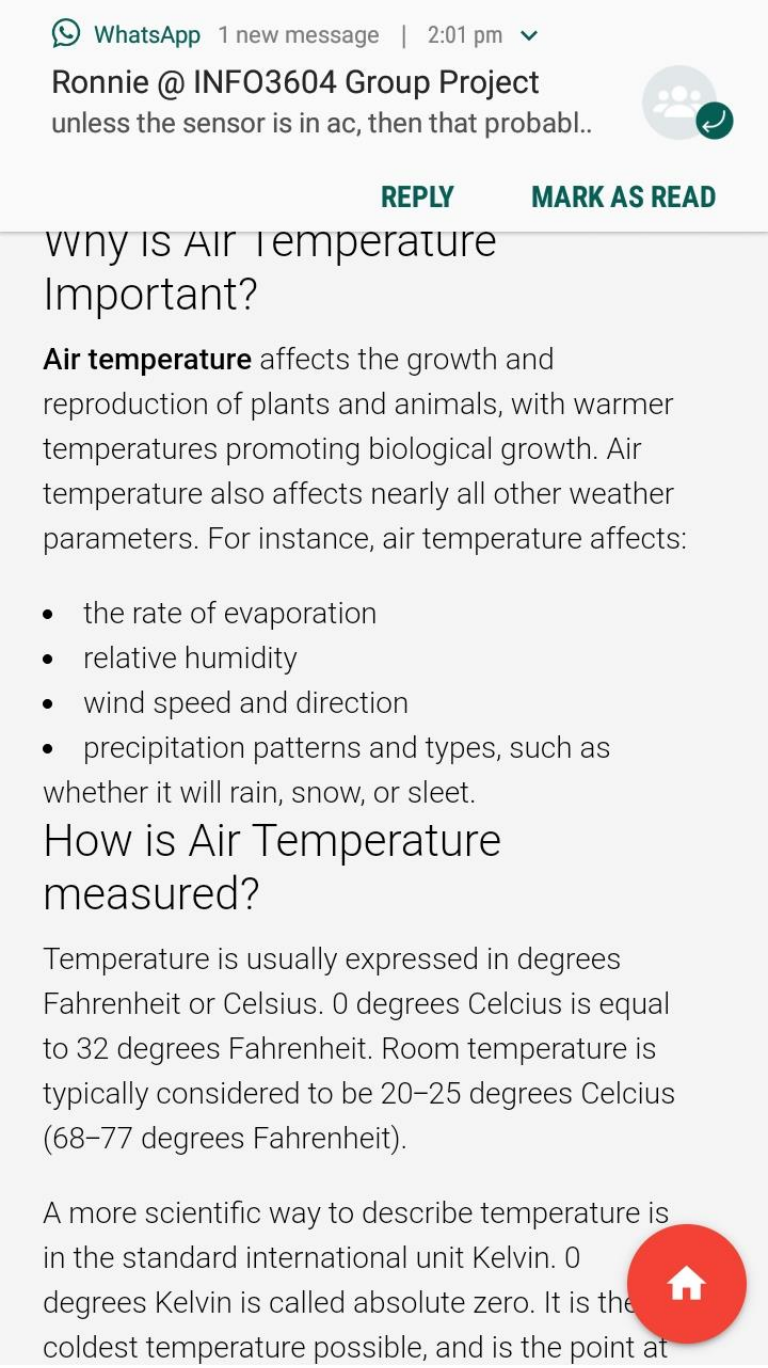
After the user has downloaded the data as a CSV



Here are some Facts on Air Temperature

- Air temperature is a measure of how hot or cold the air is.
- It is the most commonly measured weather parameter.
- Air temperature describes the kinetic ener or energy of motion, of the gasses that make up

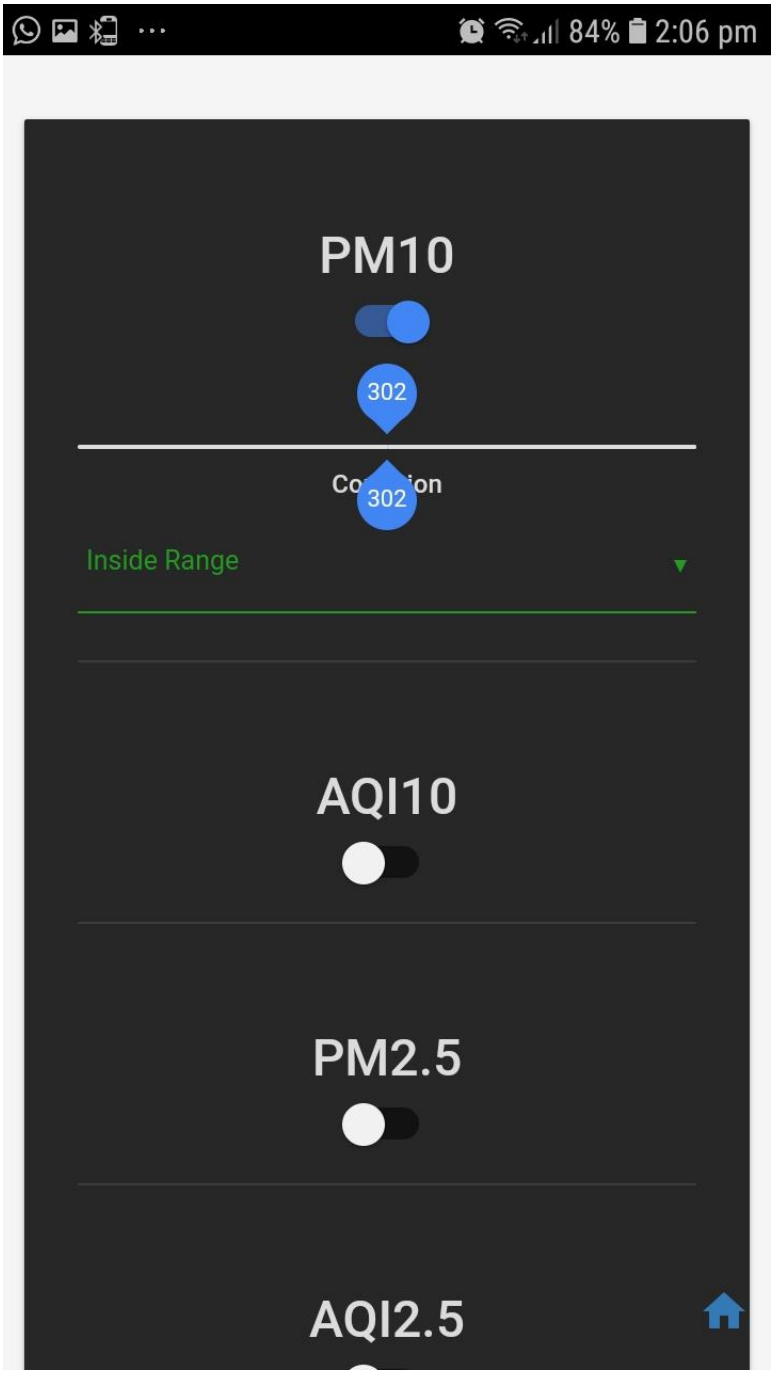
The user has navigated to the Air Temperature screen.



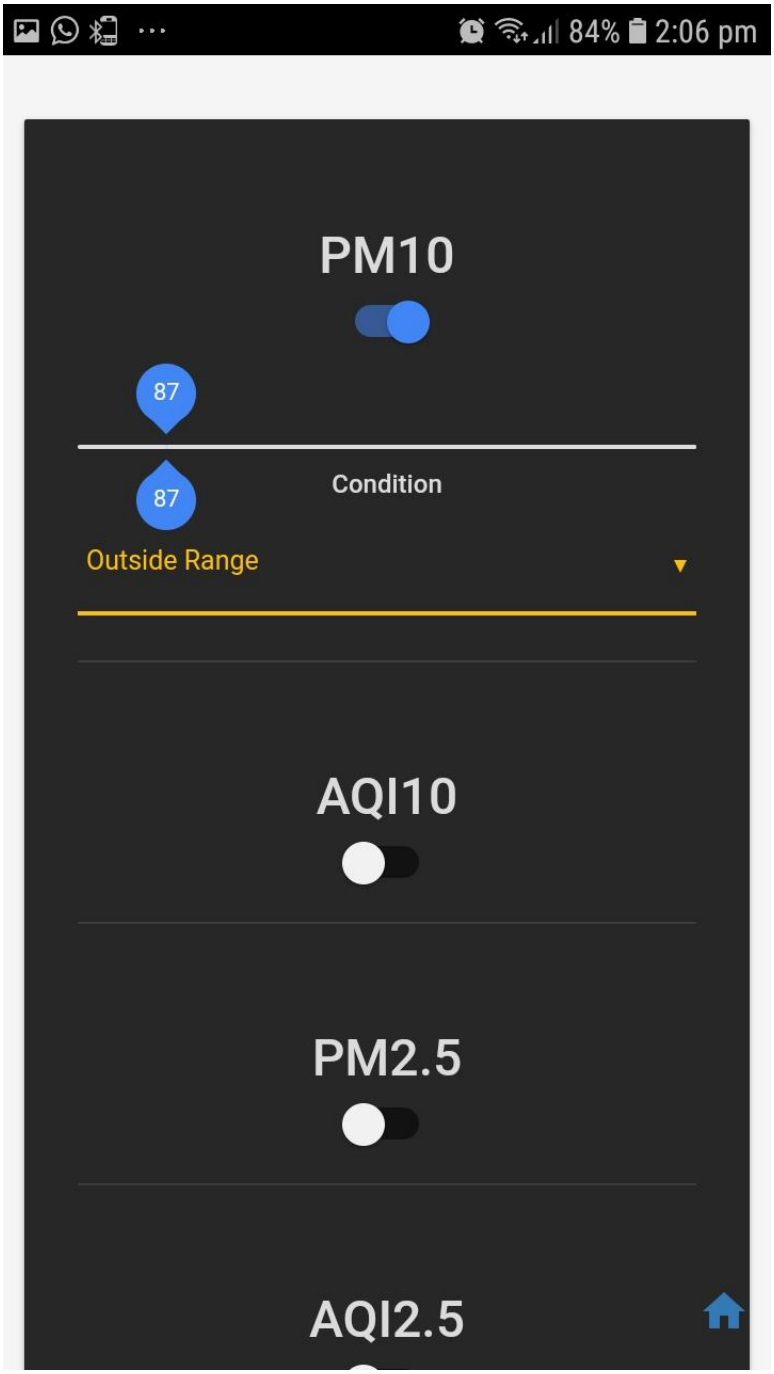
The user has scrolled down and can see description on information on air temperature.

Similarly the user can see information on barometric pressure if they navigate to the Barometric pressure screen.

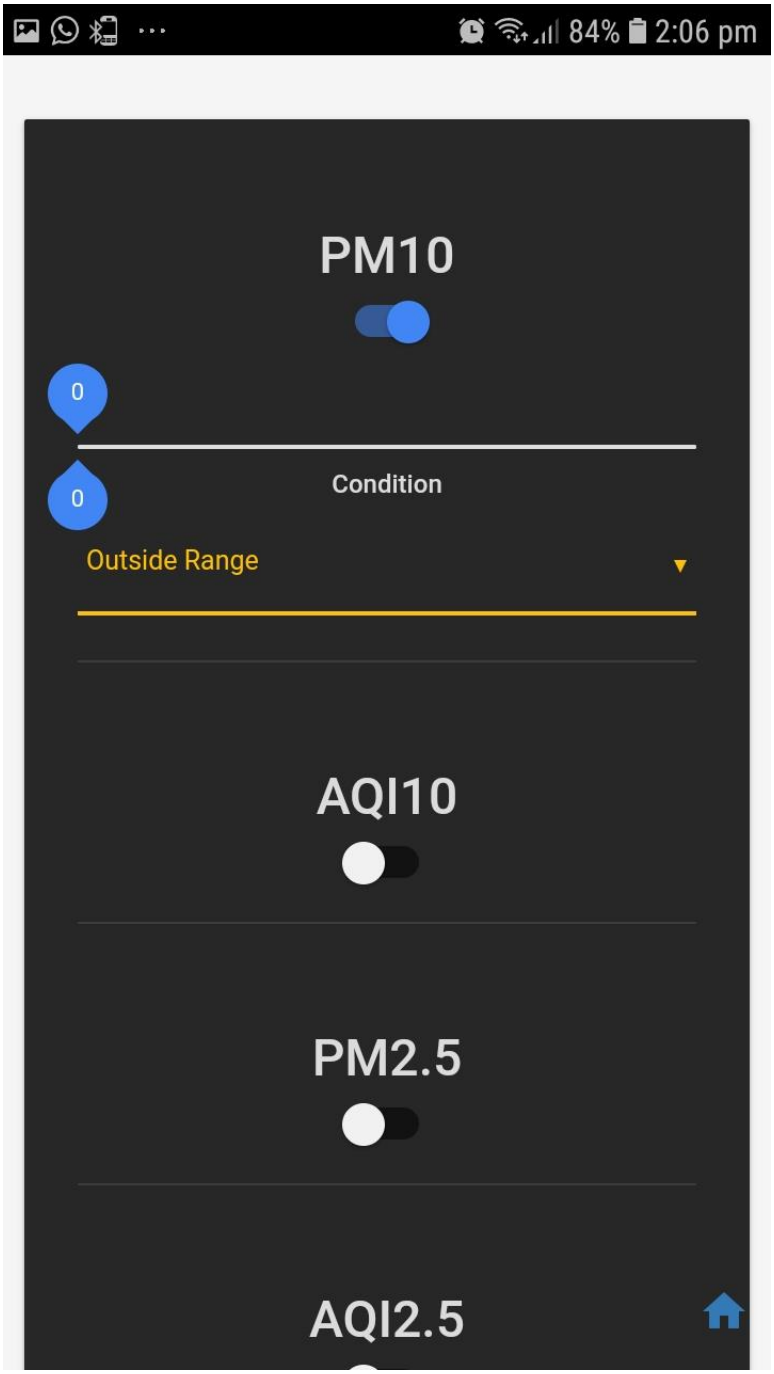
The user can use the other screens to get information on that condition.

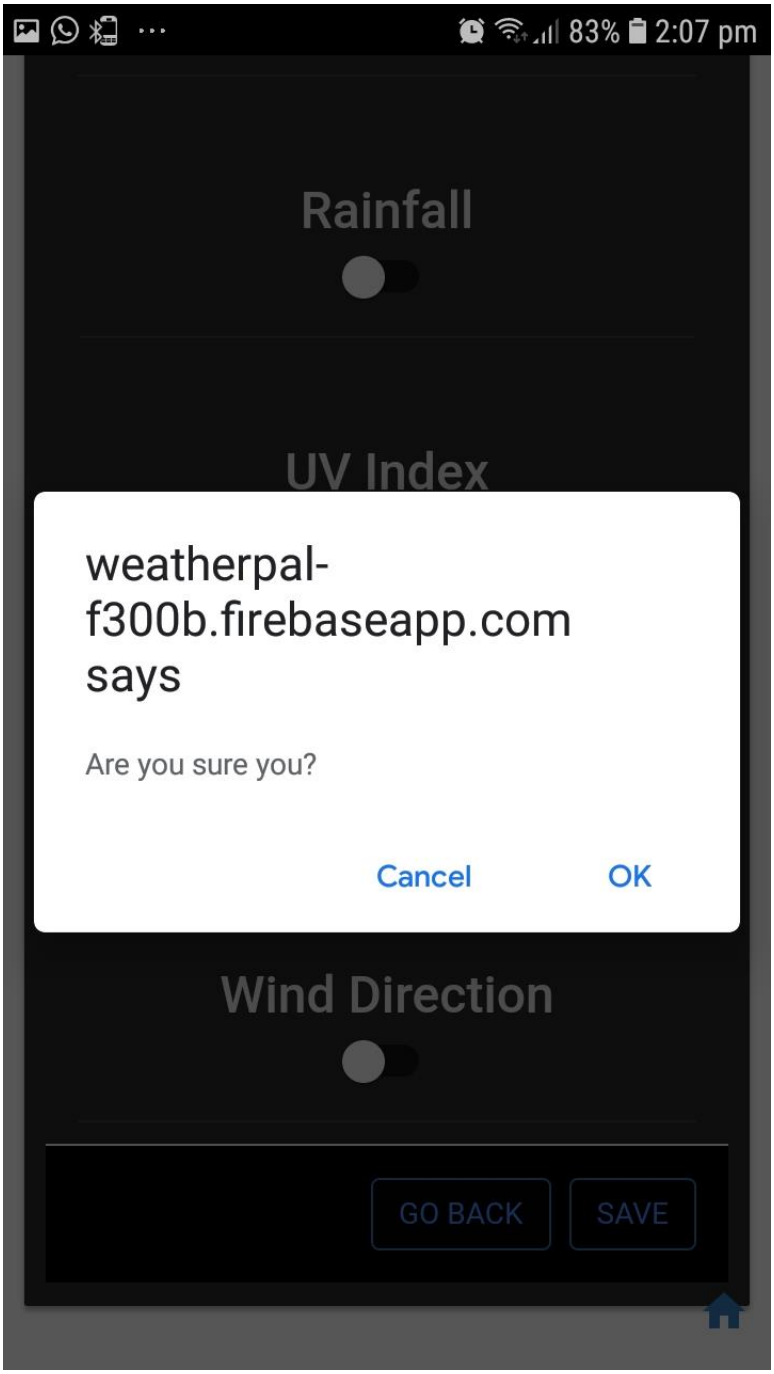


The user has navigated to the alerts screen where he can set a condition to monitor for notifications.

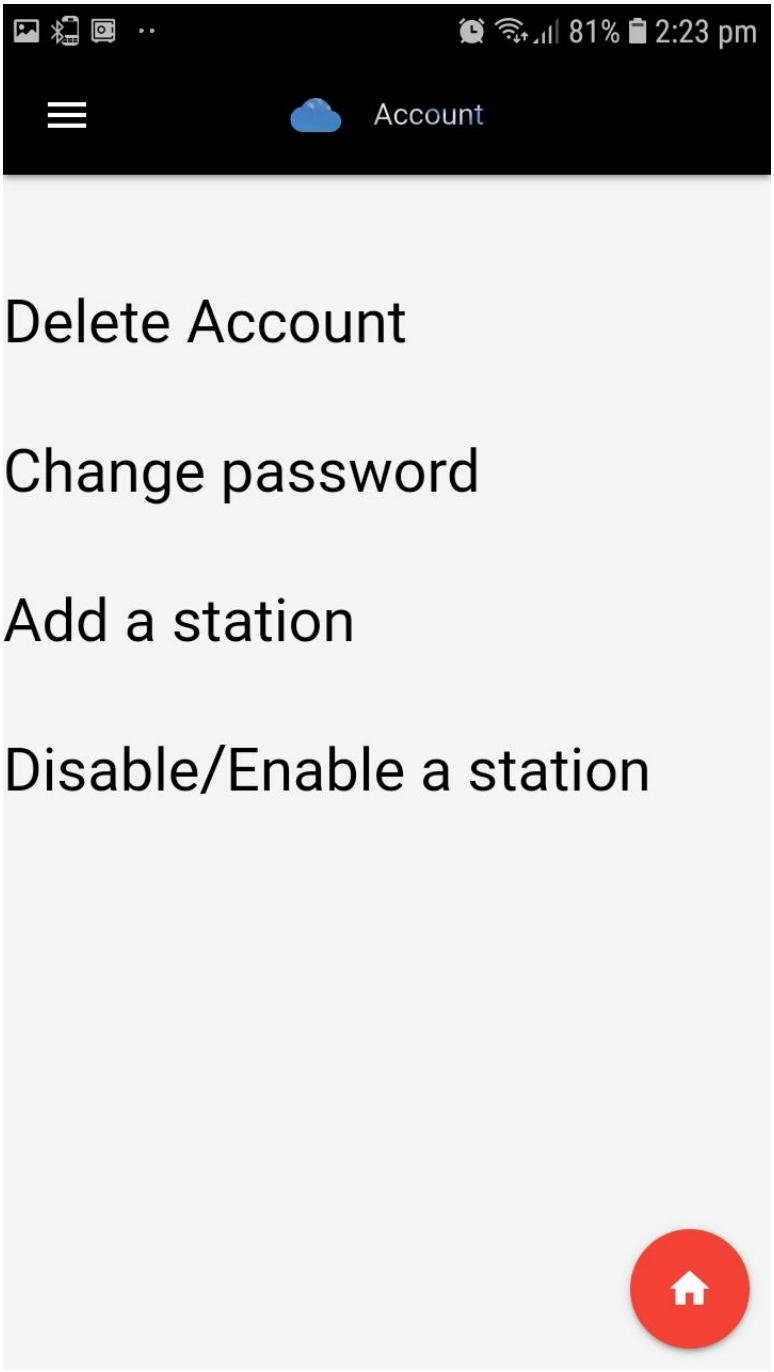


The user is sliding the bar values to indicate the range he wants.


84% 2:06 pm
PM10
0
0
Condition
Outside Range
AQI10
PM2.5
AQI2.5

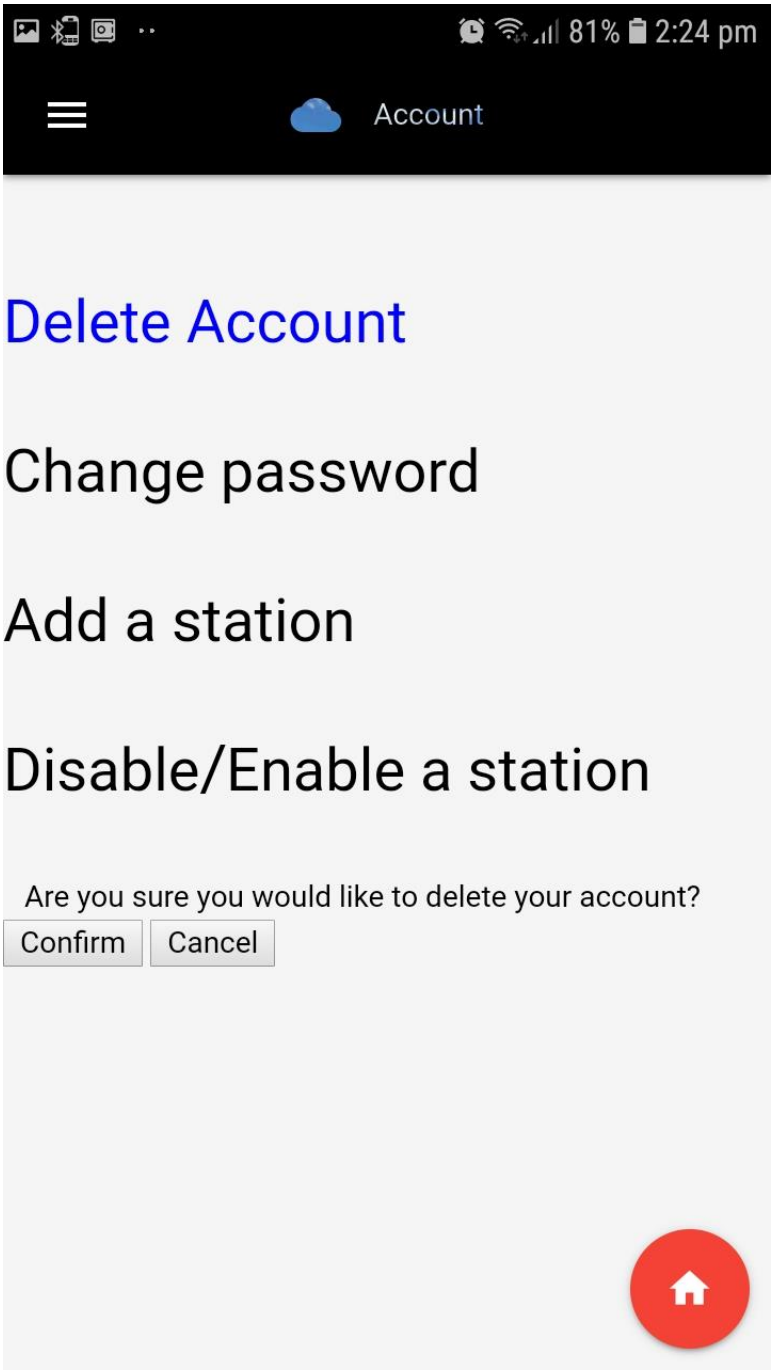


The user sets the values he want to use and presses go back at the bottom of the page and is greeted by a confirmation prompt.

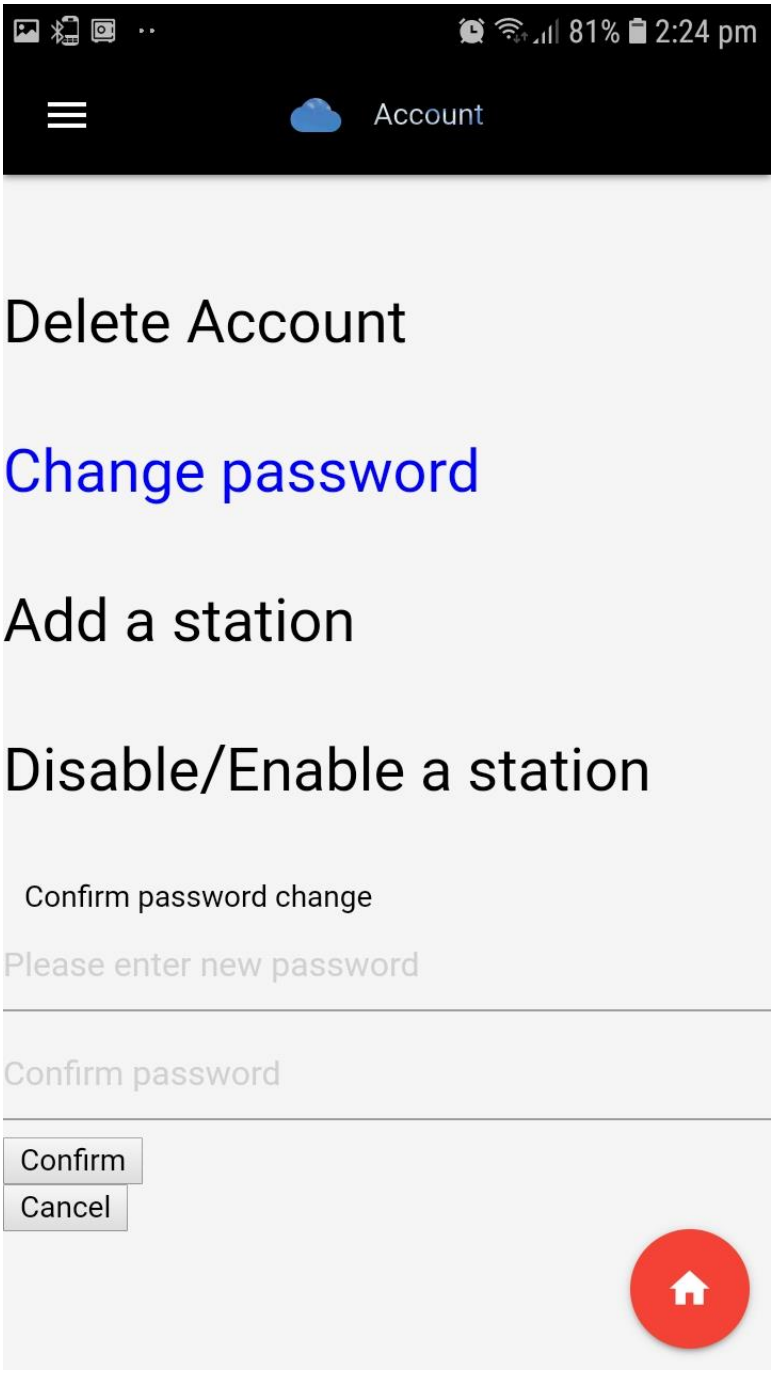


The user has navigated to the accounts page.

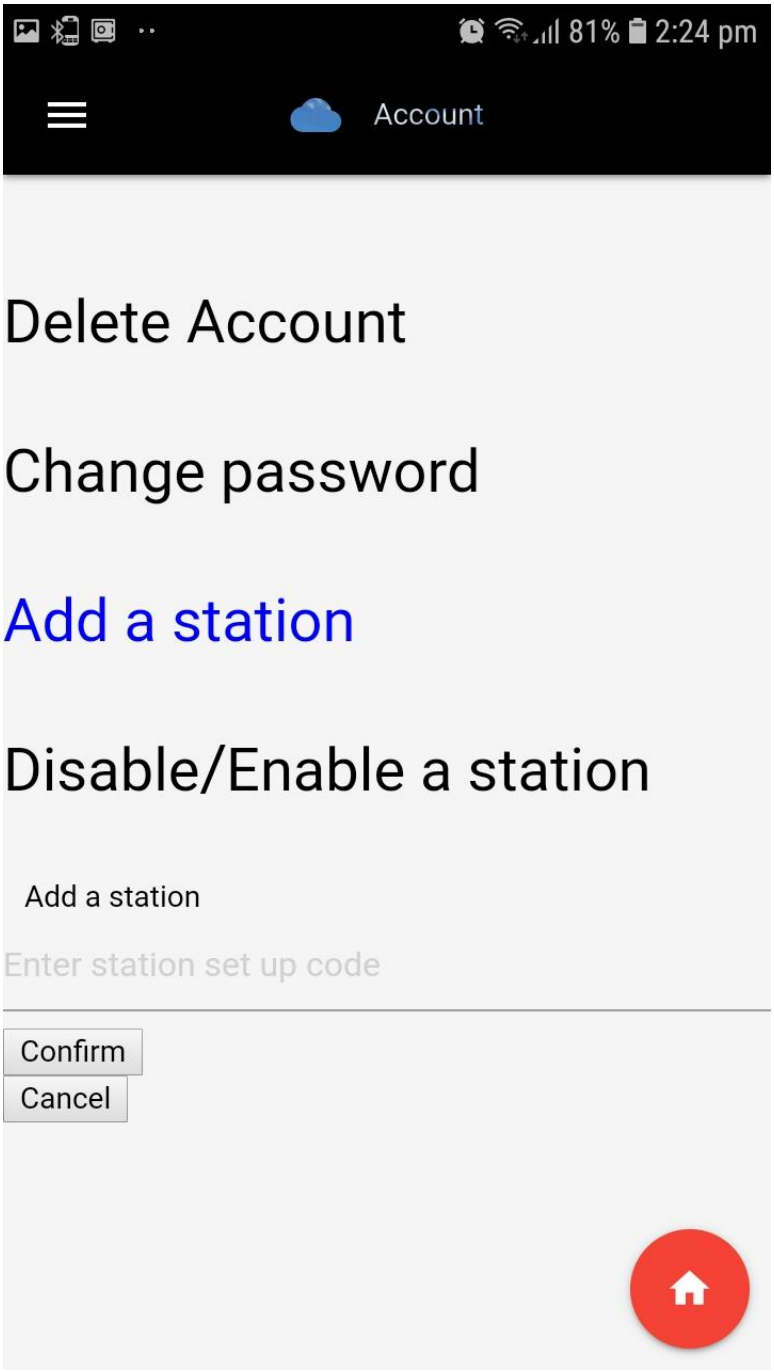
When an action is selected the page displays an area to input user data.
Areas displayed are as follows:



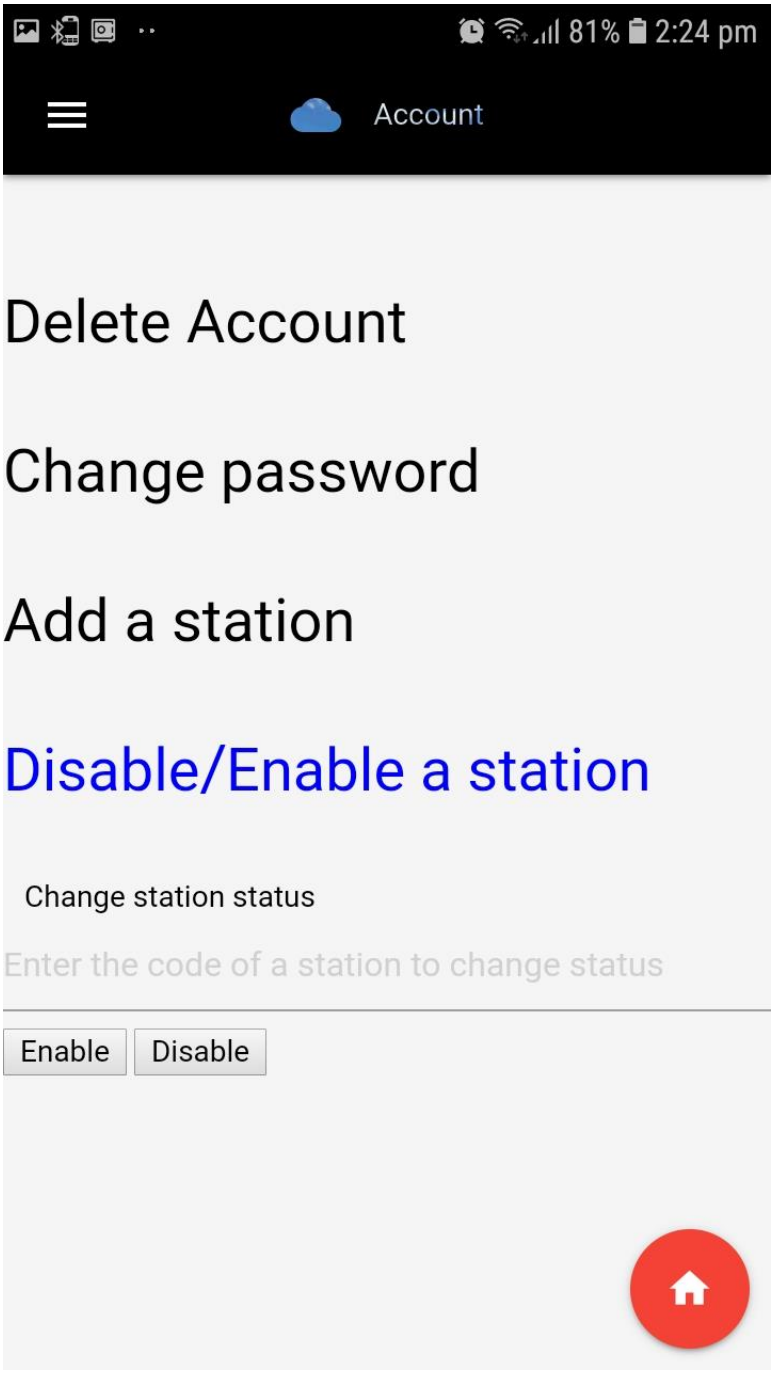
The user has selected to delete his account.



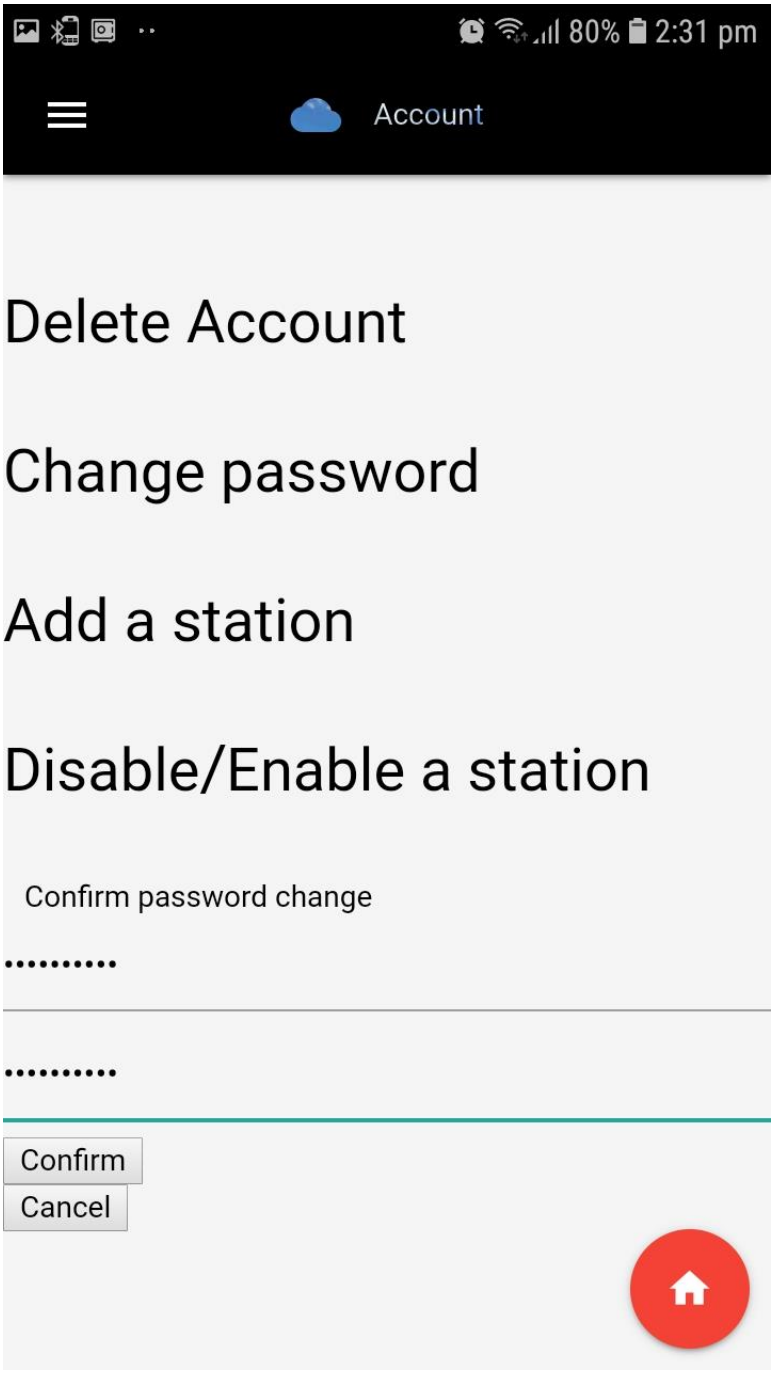
The user has selected the option to change his password.



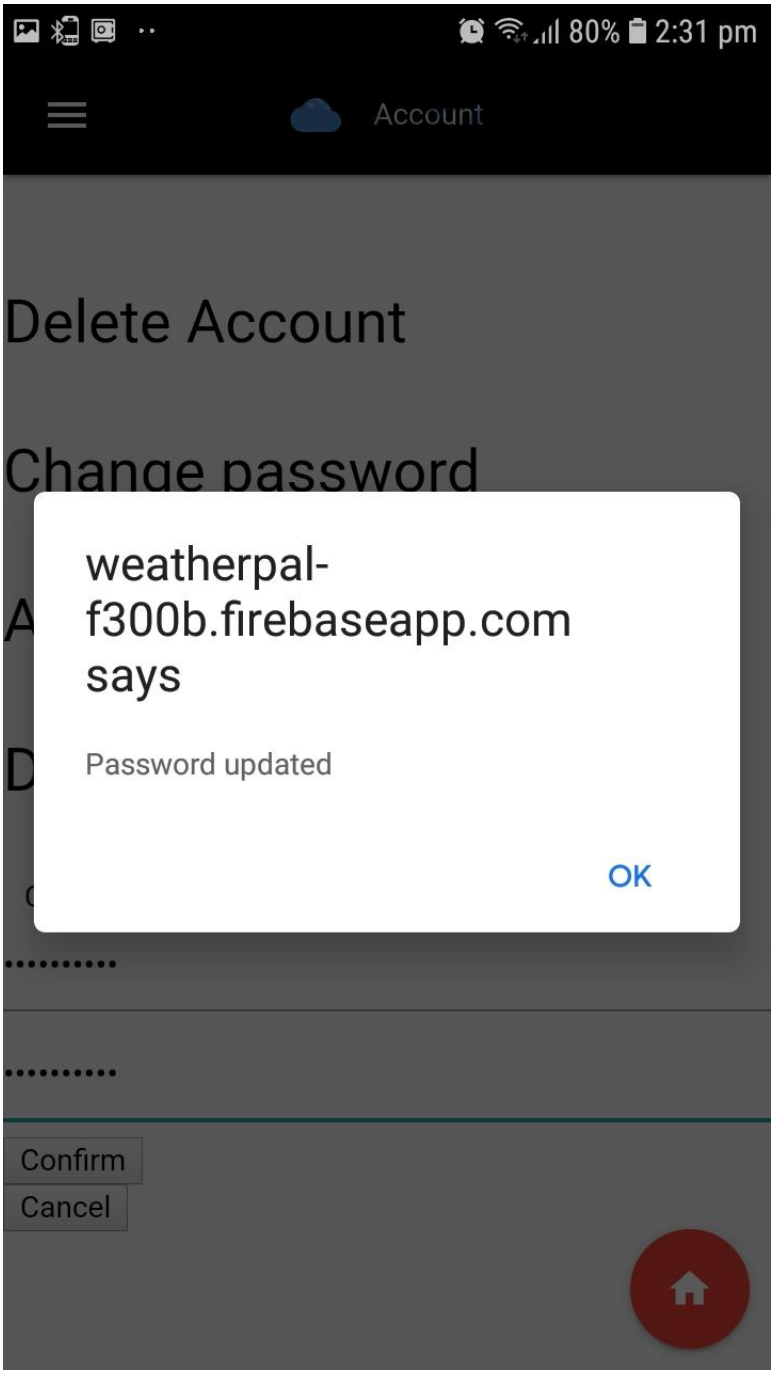
The user has selected to add a station to his account and the firestore database.



The user has selected to disable one of his stations.



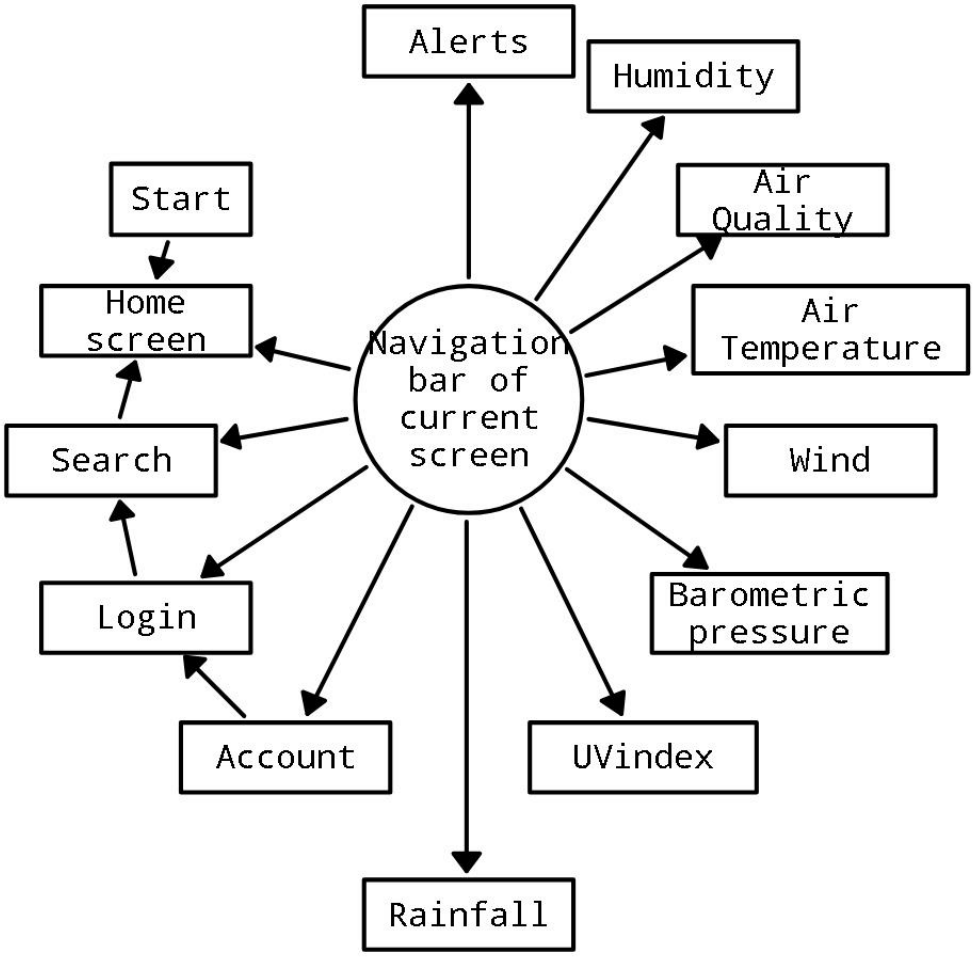
The user has entered the details of his new password and confirms it.



The users password has changed, the user can now login to his account using his new password and is redirected back to the login screen.

Navigation map

Navigation map showing the possible navigations to every other screen.



Details:

Alerts take you back to the last visited screen or the home screen and has no navigation menu.

The Home screen button can be used to jump to the Home screen from any page.

The navigation bar is used to navigate to every screen.

The signout option can be used to instantly log out to the main screen.

Test

Test plan:

The original test plan was to develop tests for each increment as we go along. This was not implemented, but, due to the app's code design, the system and unique components could be easily tested. So the new test plan was to implement tests for unique functions and smaller components of the systems but ensure that the functional requirements were the primary test targets. After the functional test non functional tests would be performed where evaluation of app performance would be conducted. It would be determined that our project had succeeded based on whether or not we fulfilled our requirements and displayed the required data. The requirements and conditions displayed constitute what we believe to be the products the user we expect to use our app need.(See the appendix for exact tests performed)

Test approach

Tests were broken down into categories of testing, where the functional requirements would be tested separate to the non functional requirements. The Ui would be inspected to test for any bugs and errors. Certain tests were not conducted and therefore certain errors were passed through, however, to make use of what time we had left, certain condition values and the functional requirements were ensured to be tested, as well as performance. Condition tests were done to ensure that the correct data was displayed to the user were predefined that was passed through to display to the user. The functional requirements outputs were tested based on test stations set up on firebase. The tests were done to ensure correctness of output, that firestore functioned well and that the data correctly got to the user from cloud firestore. All functional requirements required the interaction and use of firebase. Unique conditions were tested first then firebase functions, the requirements and the ui. The system was also to be tested against errors. Tests were also performed against the Trinidad and Tobago MET(meteorological service) office.

Features tested:

All of the required functionalities were tested but here is the list:

- A user shall be able to search for a weather station using location, name and proximity
- A user shall be able to register an account for interacting with the system’s weather web app
- The system shall be able to attain weather information from a weather station and store it in a database.
- The system shall be able to take weather data from a database and send it to a web

app for display.

- A user shall be able to add a weather station to the system's available weather stations.
- A user shall be able to add a weather station to his list of favourite stations for later viewing.
- A user shall be able to get a list of weather log data from each station.
- The system shall be able to keep a store of logs of weather data for each station on a database.
- The user shall be able to view weather data in a graph format.
- The user shall be able to view a list of the weather stations that he owns and has added to the system's available weather stations.
- The user shall be able to initiate notifications that will alert them when sensor readings fall out of or into a certain (user specified) range.
- The user shall have the ability to deactivate their account.
- (Perfromance)Page load times e.g the page shall load within 2 seconds

Features not tested:

- The user shall be able to install the web app
- The user shall be able to view the web app on any device(mobile or desktop).
- Most non-functional requirements did not meet tests

Business Aspects

This model can be applied by interested parties to improve the efficiency of their businesses. For instance, a farmer will know that he may benefit on a certain day, based on weather conditions, by planting or harvesting as opposed to focusing on sales. Likewise a fisherman may be able to judge if it is safe to venture out to fish and make informed decisions that minimize cost and risk of life. Other groups that can benefit include persons who are affected by the outcomes of weather changes especially flooding such as researchers, planners and developers in various private and state institutions, lifeguards, water sportspersons and other sportsperson and all other businesses that are weather-sensitive. Similarly transporters, delivery facilities, police and security personnel, arts and theatre stand to gain in terms of finding various settings to do videography/photography. The field of preventative medicine, e.g., those with allergies or sun sensitivity can find useful information using this weather service. Thus it has the potential to improve productivity, efficiency, health and safety to name some of the benefits.

Individual contributions

- Jose Andrez Bravo Mata: Integrating firebase firestore function, frontend code, all station searching functions, displaying weather conditions(see the status reports for more detail on contribution.)
- Vinod Lochan Dassrath:
- Sourcing, assembling and connection of all hardware
- Document non-functional requirements, class diagram etc.
- All the code for setting up the alerts for users, all the charts, all the gauges, all the weather station sensors. Fixing weather station when broken down, 2 dead up sensors, reducing/changing my fully function code to integrate into the app UI. Everything I did, I ensured worked 100% in terms of functionality before integrating except for an error in the log charts degree to cardinal function. I still haven't completed adding the code to set up loading the unique id for the weather station from a file and also for using the status value in the database to check if registered before updating the database and also making the code auto run at boot when the pi boots. They were easy to do but did not have enough time as I had to devote elsewhere just to integrate my already working code.
- Ronald Jaglal: Designing and implementing the User Interface. Gathering content and assembling the information/facts pages about the various readings.

Financial Considerations

Project budget:

Careful consideration was put into the project budget, the focus was on getting material that would not be costly. The budget was assumed to be not very high as expectations were made of easily attaining materials for assembling the pi weather station. This material consisted of the raspberry pi 3 and various sensors as well as gear for ensuring the protection of the station.The budget projection and true cost were around 400 to 500 used.

Cost projections:
This project is nonprofit, however, there are expansions that can be added that would increase further costs to the user but not from the app. This is when the user seeks to add his own station he would have to acquire the material itself.
One should consider the costs of building the weather station(see project implementation report for more details on materials required to replicate our system). Since it costed around 400 to 500 USD to construct our weather station(this includes replacing broken hardware) it is suggested the typical budget for this project should be around 800 USD, mainly for expanding condition capabilities(ph levels of soil, ground temperature, etc).

Conclusions, Lessons Learned and Recommendations

State of completion:

Our project is, by technicality, 100% complete as we have fulfilled all our functional requirements. It is to note that we did not get to implement certain features that would have been useful to our web app. The web app currently does not fully support desktop browser(You cannot view the app in full screen only in windowed mode and not at full screen length). There is a desire to fix up certain features(Alerts page had trouble integrating). Our app is essentially complete, the only things we would have desired is more time to complete certain ui styling and ensure top, error free performance.

Summary of feasibility:

The app is very feasible, we suspect farmers may get the most use out of our app for monitoring their crops, however, the app can be distributed to anyone. Due to the expansiveness of the app, and it's ability to provide relevant data stored over time to users, the app can expand beyond into research and other domains with minor additions such as adding extra sensors. It may be even possible to set the weather stations to monitor conditions in wildlife areas with the proper equipment, and this would not require a change in the app as long as the user can aquire the correct materials.

Future work:

The next thing we would have continued working on is likely the alerts page. After that we would have handled ui issues and small bugs to the system(such as the desktop browser compatibility issue). After this the ui can be further developed and the system further optimized. It may have been possible to have expanded the number of stations available to us in order to test more readings all across Trinidad but we decided on one station alone for this project. Many optimizations could be made, including how we handle the storing of logs on firestore to overcome the limits with regard to usage. Furthermore the possibility of implementing the changes that are mentioned in the project extention.

Project extension

The app can be expanded in to:

- Handle more condition data such as ph, salinity
- Support browser fullscreen and installation
- Implement weather predictions
- Support a lowered functions model(less conditions, less sensors) for lowered cost.
- Handle farming focus conditions only.

- True real-time updates to a user's mobile device.
- Using the station for implimenting an effector system in argiculture(adjust conditions for crop growth).

References

"What is Air Temperature? ", Fondriest Staff on August 12, 2010

"Atmospheric Pressure: Definition & Facts", Joe Rao - Meteorologist, Astronomer August 29, 2013

"What You Should Know About Air Quality Alerts? ", Renee Cho on June 26, 2018

Links

Web App site:
https://weatherpal-f300b.firebaseapp.com

Project Site: https://github.com/ronaldjaglal/Coding-Squad

Appendix

Tests:

Testing

Table 1:Tests for weather conditions

Test	Unit evaluated	Expected result	Actual result
Evaluate unit outputs correct data received from firestore	Barometric pressure	A float number is returned	Proper float value is obtained from station
Evaluate unit outputs correct data received from firestore	Humidity	A float number is returned	Proper float value is obtained from station
Evaluate unit outputs correct data received from firestore	Rainfall	A float number is returned	Proper float value is obtained from station
Evaluate unit outputs correct data received from firestore	Dust level	A float number is returned	Proper float value is obtained from station
Evaluate unit outputs correct data received from firestore	Air temperature	A float number is returned	Proper float value is obtained from station
Evaluate unit outputs correct data received from firestore	UV Index	A float number is returned	Proper float value is obtained from station
Evaluate unit outputs correct data received from firestore	Wind direction	A float number is returned	Proper float value is obtained from station
Evaluate unit outputs correct data received from firestore	Wind speed	A float number is returned	Proper float value is obtained from station

Table 2:Tests for condition retrieval

Test	Unit evaluated	Expected result	Actual result
Evaluate if station correctly reads in data and sends to firestore	Barometric pressure	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes
Evaluate if station correctly reads in data and sends to firestore	Humidity	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes
Evaluate if station correctly reads in data and sends to firestore	Rainfall	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes
Evaluate Evaluate if station correctly reads in data and sends to firestore	Dust level	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes
Evaluate if station correctly reads in data and sends to firestore	Air temperature	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes
Evaluate if station correctly reads in data and sends to firestore	UV Index	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes
Evaluate if station correctly reads in data and sends to firestore	Wind direction	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes
Evaluate if station correctly reads in data and sends to firestore	Wind speed	Firebase Firestore station document is updated with a float value	Station document properly updates value every 5 minutes

Table of unique functions tested

Test	Unit evaluated	Expected result	Actual result
Evaluate if app returns correct results when used to retrieve stations by name	Search by name	List of stations with that name	Stations retrieved successfully
Evaluate if app returns correct results when used to retrieve stations by location	Search by location	List of stations near that location	Most values are obtained correctly, however, when searching by country some queries may require radius adjustment
Evaluate if app returns correct results when used to retrieve nearby stations	Find nearest stations	List of stations near user location	Nearby stations are retrieved correctly
Evaluate if a user can register an account for the app	Register	User is successfully authenticated	User can successfully authenticate with all options
Evaluate if a user can add a weather station to his list of favourite stations	Favourite station	User can retrieve a list of his favourite stations after adding them	Errors produced when picking stations with no logs and when stations share the same name. User can add and retrieve a station they have favourited
Evaluate if app can correctly display logged data to a user	Get logs	User can view logged data	User can view log data
Evaluate if user can view logged weather data in a graph format	View logs(graph)	Log data is displayed on graphs	User is able to view logged data in a graph format, the data may be difficult to understand and graph

			function may be unkwown to user.
Evaluate if user can view realtime data on graphs(gauges)	View realtime data(graph)	Realtime values are displayed on gauges	User can correctly view data on gauges
Evaluate if the user can deactivate their account properly	Delete account	User account is deleted/deactivated	User can successfully delete his account
Evaluate if the user can set a notification on a particular value of the station	Set notification	Notification is set and triggers	Notifications triggers appropriately
Evaluate if user can add a station to the available weather stations	Register station	Station collection on firestore updated the station document to be the user's	Station successfully registered
Evaluate if user can view stations that he owns	View my stations	App displays user's station list	The app correctly displays the user's station list.
Evaluate if the user can change his password properly	Change password	User's password is changed.	User can successfully change his password
Evaluate if the app can display all stations to the user	Get all stations	User obtains a full list of existing stations	User can obtain a list of all stations
Evaluate if a user can download log data	Download logs	User obtains log data for a particular period	User can download logs successfully. Issue using "view table data". Time stamps are not properly stored in files.
Evaluate if the app can sort log data appropriately	Sort logs	User can see sorted log data	User is able to see logs sorted by time intervals and

			files.
Evaluate if the app can sort log data appropriately	Sort logs	User can see sorted log data	User is able to see logs sorted by time intervals and condition, however the user cannot attain the "highest values or lowest values" without manually finding them.
Test to see if the user can signout of the app	Signout	User is successfully signed out	User can successfully log out
Evaluate if a user can register an account on the system	Register	User can register successfully with any of the options available	User can successfully register with any of the authentication options available

Test cases for weather conditon display and values

	Values used		
Unit evaluated	**Result when input=100**	**Result when input=5000**	**Result when input=100000**
Barometric pressure	Output displayed correctly	Output displayed correctly	Output displayed correctly
Humidity	Output displayed correctly	Output displayed correctly	Output displayed correctly
Rainfall	Output displayed correctly	Output displayed correctly	Output displayed correctly
Dust level	Output displayed correctly	Output displayed correctly	Output displayed correctly
Air temperature	Output displayed correctly	Output displayed correctly	Output displayed correctly
UV Index	Output displayed correctly	Output displayed correctly	Output displayed correctly
Wind direction	Output displayed correctly	Output displayed correctly	Output displayed correctly
Wind speed	Output displayed correctly	Output displayed correctly	Output displayed correctly

Table showing error handling tests

Test	Unit evaluated	Input	Result
Evaluate how searching by location handles erroneous input	Search by location	2781738 278 mars Avenue	Some search results are autocorrected or aligned properly even with vague input.
Evaluate how searching by name handles erroneous input	Search by name	Home Station1 123	Results do not autocorrect, however, as expected, no results should have been returned
Evaluate how loading in data handles erroneous input	Weather condition screens, logs	Replaced numbers with strings Negative values for certain numbers	
Evaluates how missing values affect the app	System	No station name Missing condtions Missing logs	Station can still be found with the get all stations function even if it has no name. Missing condtions or logs prevent any values from displaying regardless of if the particular value is not missing on that station.

Table showing user interface performance tests

Test	Unit evaluated	Expected result	Actual result
Evaluate load time of condition pages	Weather conditions	Conditions should load within 3 seconds	Around 1.5 seconds
Evaluate login time	Login	User should be able to login within 3 seconds. Login times are variable due to processes.	Login times are as follows: 6 seconds Google 4 seconds Facebook 3 seconds Github 8 seconds Email 35-40 seconds Phone Less than a second anonymous 12-15 seconds Yahoo
Evaluate station search time	Search functions	Search results return within 2 seconds	Less than a second
Evaluate main, login and account page load times	Index, login, account pages	Main page should load within 3 seconds	Around a second
Evaluate app start up time	System	UI loads within 3 seconds	Around 1.5 seconds

Some test cases used for testing

Test Case: View condition
Description: A user should be able to view a condtion obtained from firestore
Pre-requistes: User is logged in
Assumptions: Station has data loaded in on firestore.
Uses: Used to test the conditions in table 1
Process:
Navigate to search using side navigator
Search for "station1" by name
Select the station
Navigate from the main menu to the required condition
Success: The condtion on the page should load in a value indicating that data was retrieved from firestore

Test Case: Value update
Description: A station should be able to send values to cloud firestore and update them.
Pre-requisites: Station is set up
Assumptions: Pi station has required hardware for reading weather data
Uses: Used to test the conditions in table 2
Process:
Set up station to read weather data
Using the instructions from firebase, have the data sent to a document for your station.

Success: The weather conditions on the cloud firestore database should be updated with the station's collected values.

Test Case: Search for stations
Description: A user should be able to search for stations using one of the search features
Pre-requisites: Each station has a form of identification(name, location) stored on the firestore database for searching.
Assumptions: N/A
Uses: Used to test the search functions in table 3
Process:
From the main menu(index) navigate to the search screen
Select one the search options availabe
If required, input search query(name or location(eg. station1, Trinidad))
Success: A list of the stations matching the query or valid for the search option selected

Test Case: Register account
Description: A user should be able to register an account for the app
Pre-requisites: User has an email address and or social media account.
Assumptions: Only one account is desired
Uses: Used as a prerequisite of the login function in table 3 and to test registration function
Process:
Start the app
Navigate to the login screen(if necessary)
Select a registration method(social media or email) available
Success: An account with the app has been created and can be used for authentication.

Changelogs(from frontend code developer):

4/4/2020
Comments added sheet restyled to be more presentable/understandable.
last function worked on was notifications.

17/04/2020
function added to delete user account
function added to check if user is logged in
added functionality to send user to login screen if logged in returns false
page added for account settings
modularized some elements(removed certain dependencies and separated certain components)

18/04/2020
ensured delete function works
code refactored

29/04/2020
Made change password function
code refactoring

30/04/2020
Updated code for functionality
code refactored(account.html and related files)
error checks for the files aforementioned
minor changes to main.html
added button and function for retrieving the entire station list
removed attributes.js and put content in logfunctions.js

1/05/2020
Implemented function to search for stations by location using geocoding

02/05/2020
Implemented function to search for nearest station
fixed log formatting
refactored code
Implemented function to sort log tables using the headers
started work on function to save logs

04/05/2020
Started fitting code to ui
Created account, login and search pages
Updated ui elements

05/05/2020
Updated ui elements
Fixed pages to display weather data

06/05/2020
Removed un-useful code
Fixed index to display logs and station info
Comments added to files
Log charts added to index page

Added Yahoo for authentication
Removed Twitter

07/05/2020
Fixed authentication issues
Added signout to the ui
Fixed favourite button
Fixed user login verification
Fixed ui issue with scrolling

11/05/2020
Implemented gauges to pages
fixed load time of index using button
refactored code

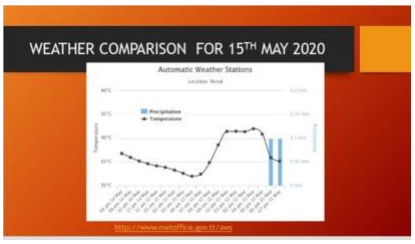
12/05/2020
Fixed code issues

13/05/2020
Implemented functions for adding and activating/deactivating a station

14/05/2020
Implemented alerts to ui
Final version of the app created

15/05/2020
Some big fixes
Updated final version

Station readings vs T&T MET Office


WEATHER COMPARISON FOR 15TH MAY 2020
Automatic Weather Stations
Precipitation
Temperature
http://www.metoffice.gov.tt/awt