Environment Recorder
UIC

Final Report

Prepared by:
Alex Escatel, Mike Squires, Dan Alzeidan, Hamza Ali

CS 442
at the
University of Illinois Chicago

April 2023

Table of Contents

List of Figures

I Project Description

1 Project Overview

Environment Tracker UIC is a web application that utilizes several components including Arduinos with air quality sensors, react with javascript, google firebase, and leaflet api. These tools combined, create a real time web application that has an interactive map where users can see the data for specific sensors and get warnings if the air quality is bad.

2 Project Domain

With the ever increasing usage of gasoline cars and industrial take over of the world, the air quality of where people live is becoming a concern for many people. Poor air quality can lead to health issues such as heart disease and respiratory issues. This project's aim is to accurately track the air quality so that people can avoid areas with dangerous air quality.

3 Relationship to Other Documents

Global Environmental Recorder by Hamza Ali and Suhail Alkaabi

4 Naming Conventions and Definitions

Naming conventions should follow the naming standards predefined by chemists and biologists. This information will be fact checked through verified definition sources such as Merriam-Webster's dictionary. In the example of this application we use standard nomenclature for environmental air quality.

4a Definitions of Key Terms

Term	Definition
CO2	Carbon Dioxide; dangerous air particle
Humidity	Measure of water particles in air
Ozone	Air chemical that is a pollutant (dangerous)
PM 2.5	Air particles that are smaller than a certain size (dangerous pollutant)
Volatile Organic Compounds	Carbon chemicals evaporate to air, harm health

Table 1: Air Quality Terminology

4b UML and Other Notation Used in This Document

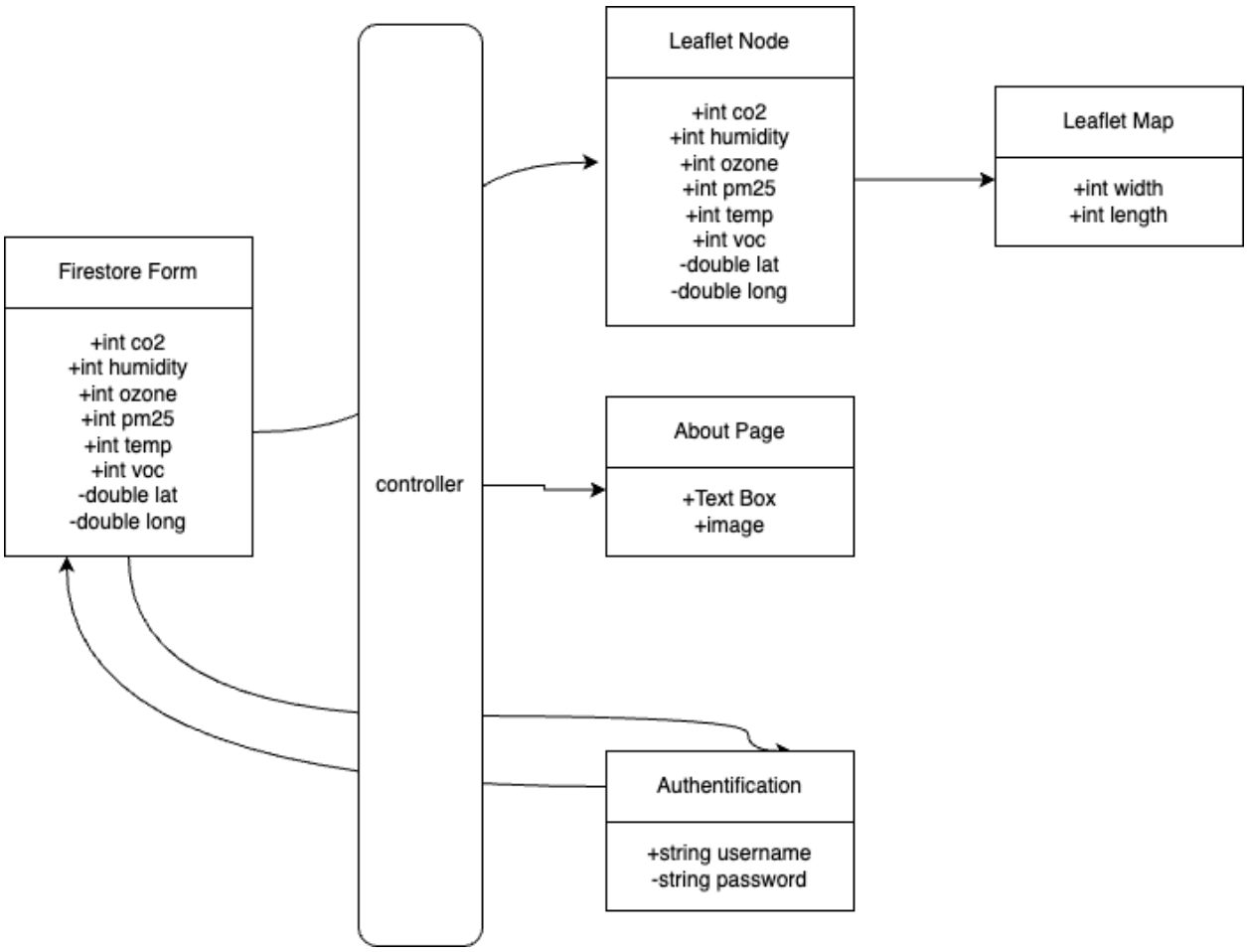


Figure 1: Project UML Diagram

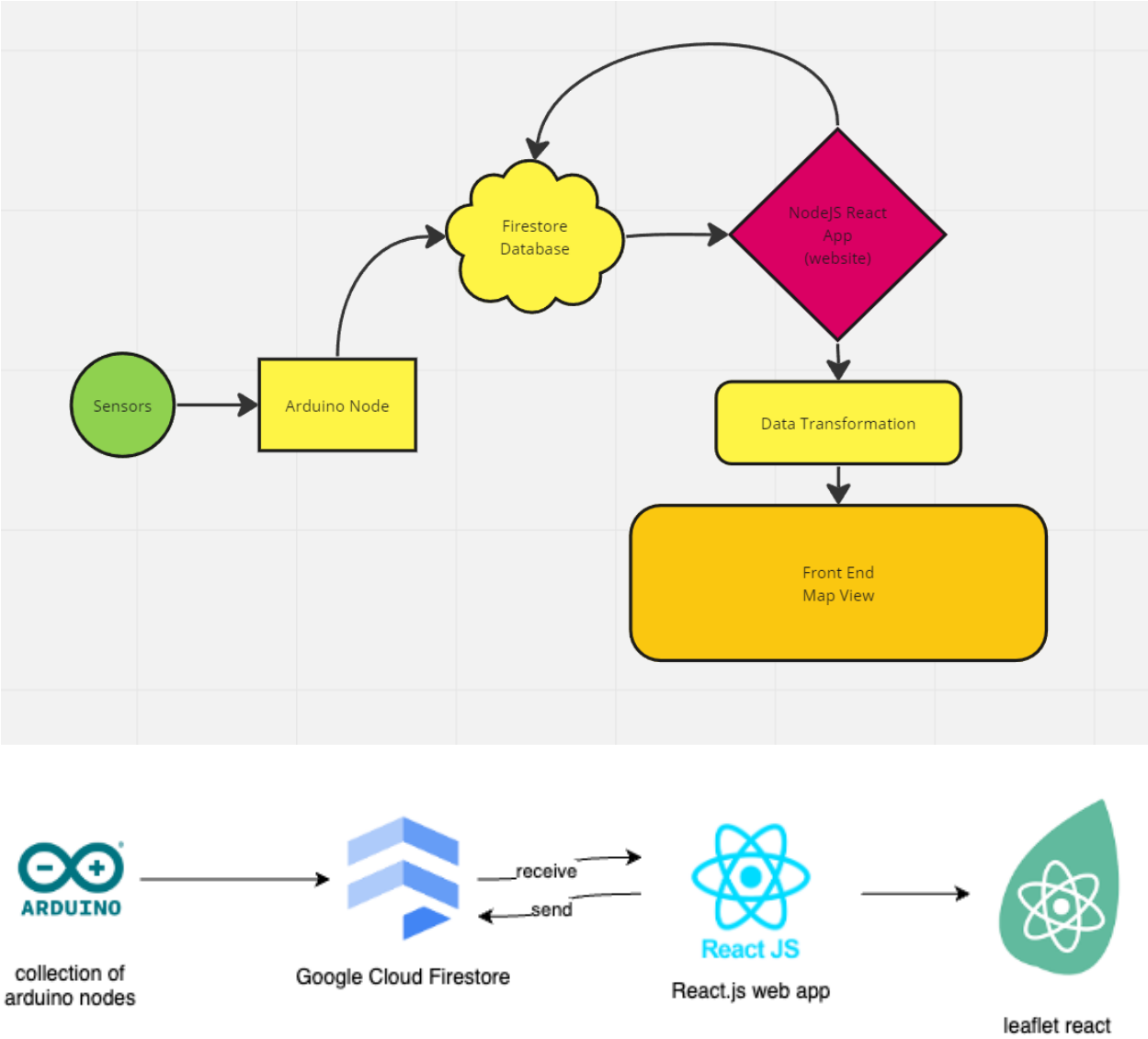


Figure 2: Project Path and Data Exchange

4c Data Dictionary for Any Included Models

deviceID: Unique identifier for each Arduino device (e.g., "1")

location: Geographic coordinates of the device, given as {latitude, longitude}

latitude: Latitude of the device, ranging from -90 to 90 degrees (e.g., 37.7749)

longitude: Longitude of the device, ranging from -180 to 180 degrees (e.g., -122.4194)

sensorReadings: List of air quality readings, each given as {pollutant, value, unit}

pollutant: Type of pollutant measured (e.g., PM2.5, PM10, NO2, SO2, CO, O3)

value: Numerical value of the pollutant concentration (e.g., 35.2)

unit: Measurement unit (e.g., µg/m³, ppm)

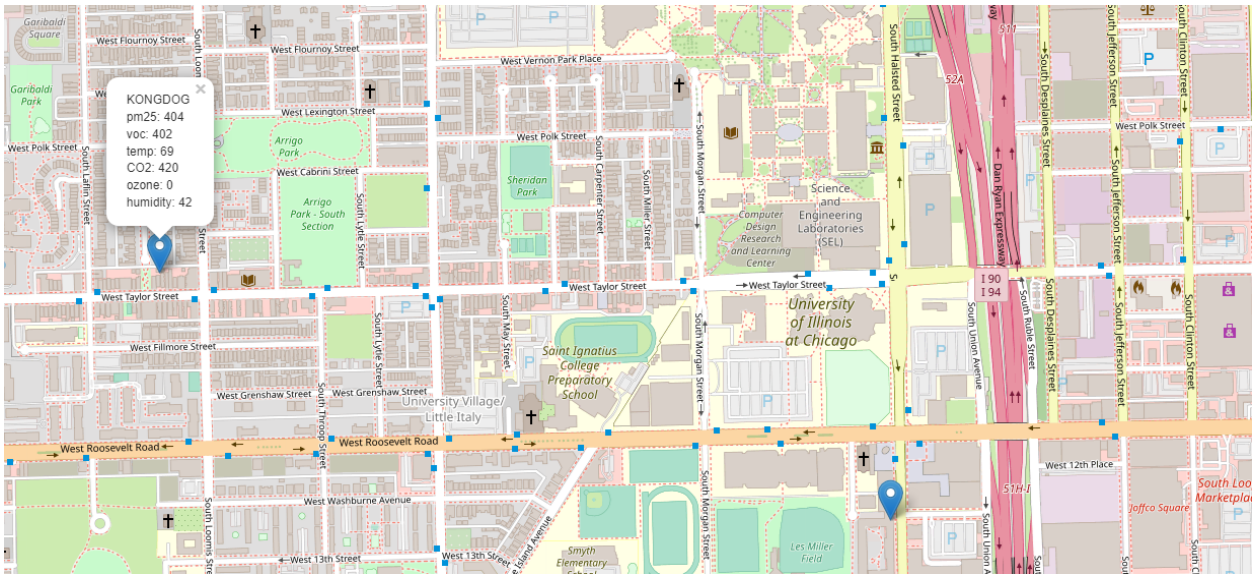
timestamp: Date and time when the reading was taken (e.g., "2023-04-28T10:30:00Z")

II Project Deliverables

A react web application that tracks air quality reading around UIC. Markers on the map specify readings at specific buildings if clicked on. About page to explain how to use the web app and the point of the web app. There is also the skeleton of a news page for scalability.

1 First Release

The first release got the web app started with pretty basic features. We implemented a basic map page with a map using React Leaflet. The firebase was also set up but the front end and back end were not fully connected yet. The map used mack data to produce the markers and popups with air quality information.

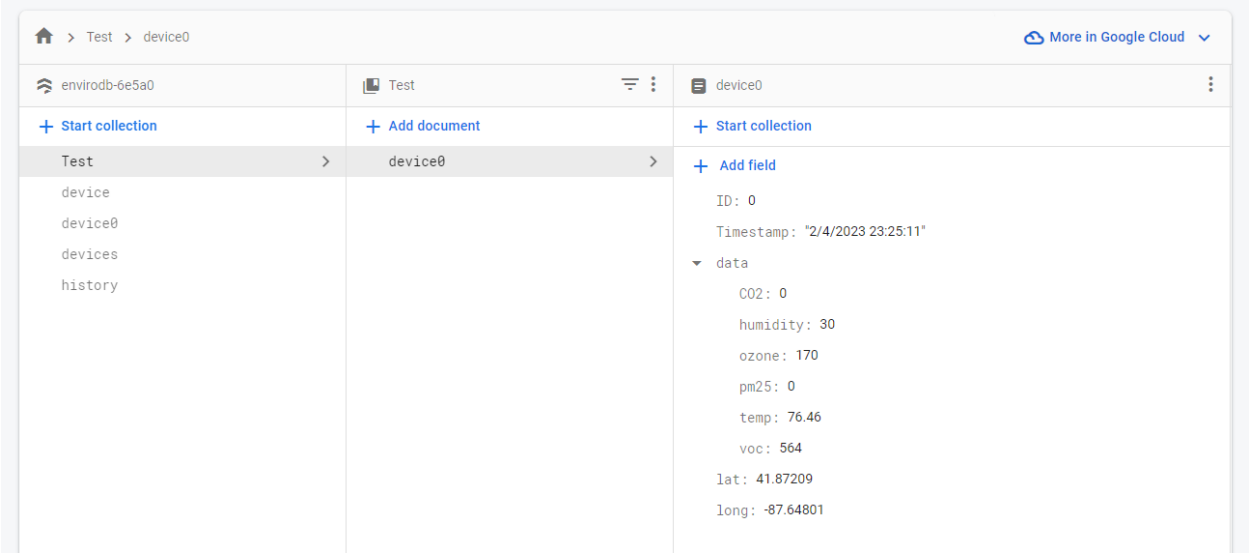


Release 1 Map

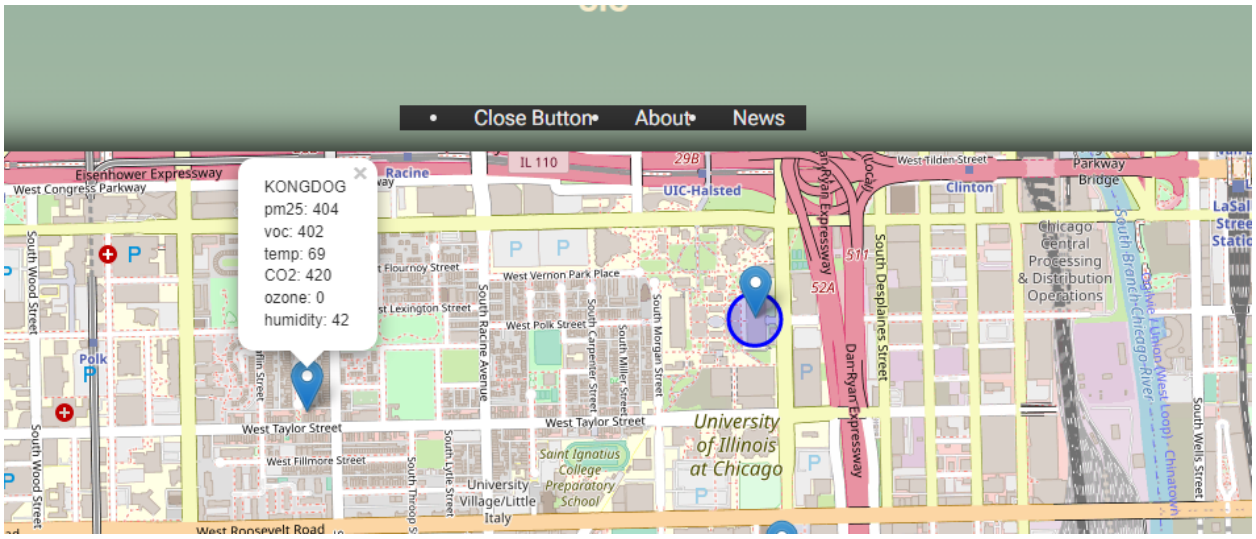
2 Second Release

The second release focused on bringing everything together. The web app after the first release was basically a skeleton for what we wanted to do. For the second release

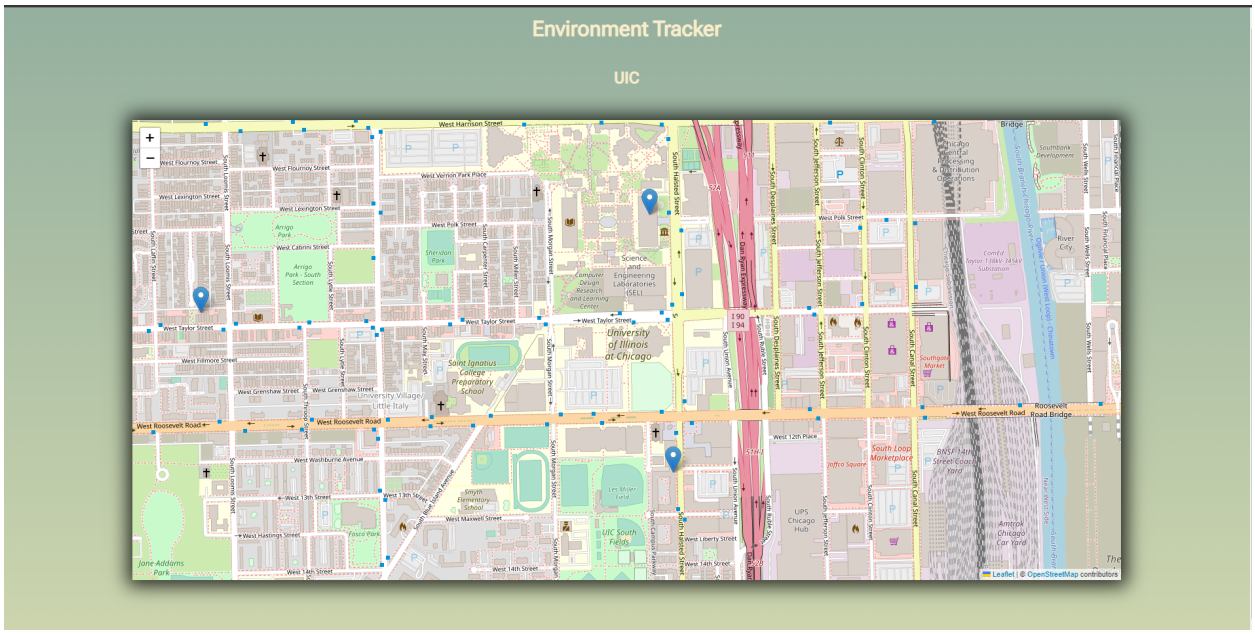
the front and back end were now connected and the information on the map was pulled directly from the firebase. The circles feature was also added in which areas with dangerous air quality readings were highlighted with circles. We also implemented a basic navigation bar in order to expand for the third release. Lastly we experimented with UI updates and color schemes and updated the UI slightly.



Release 2 Database



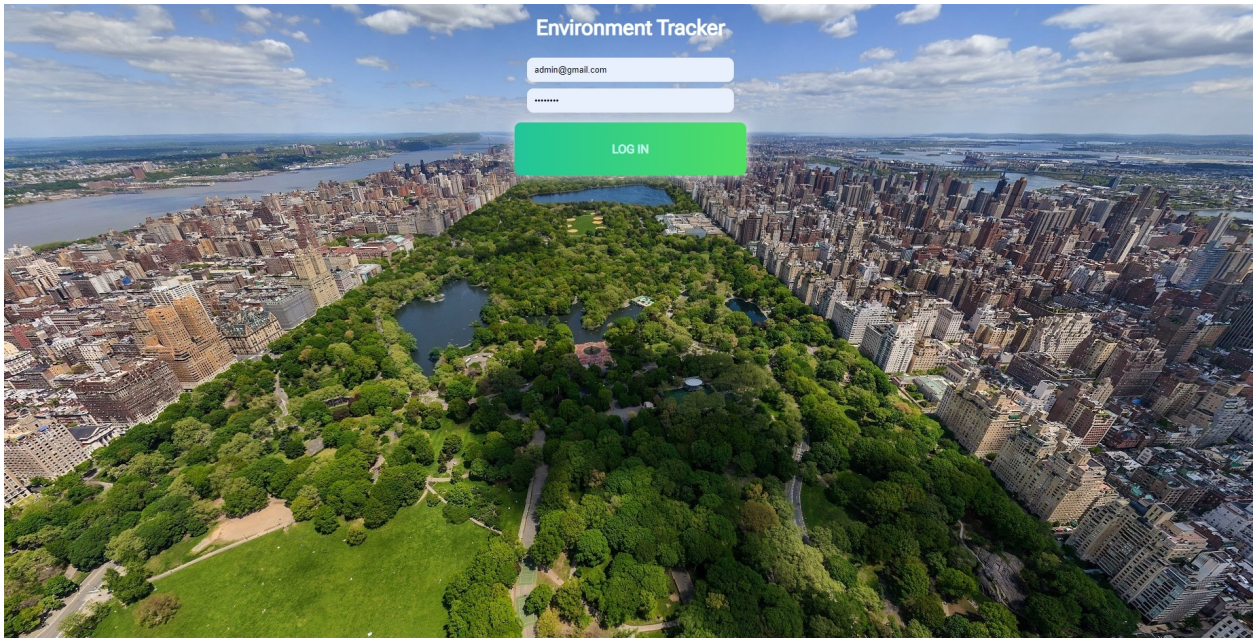
Release 2 Navigation Bar



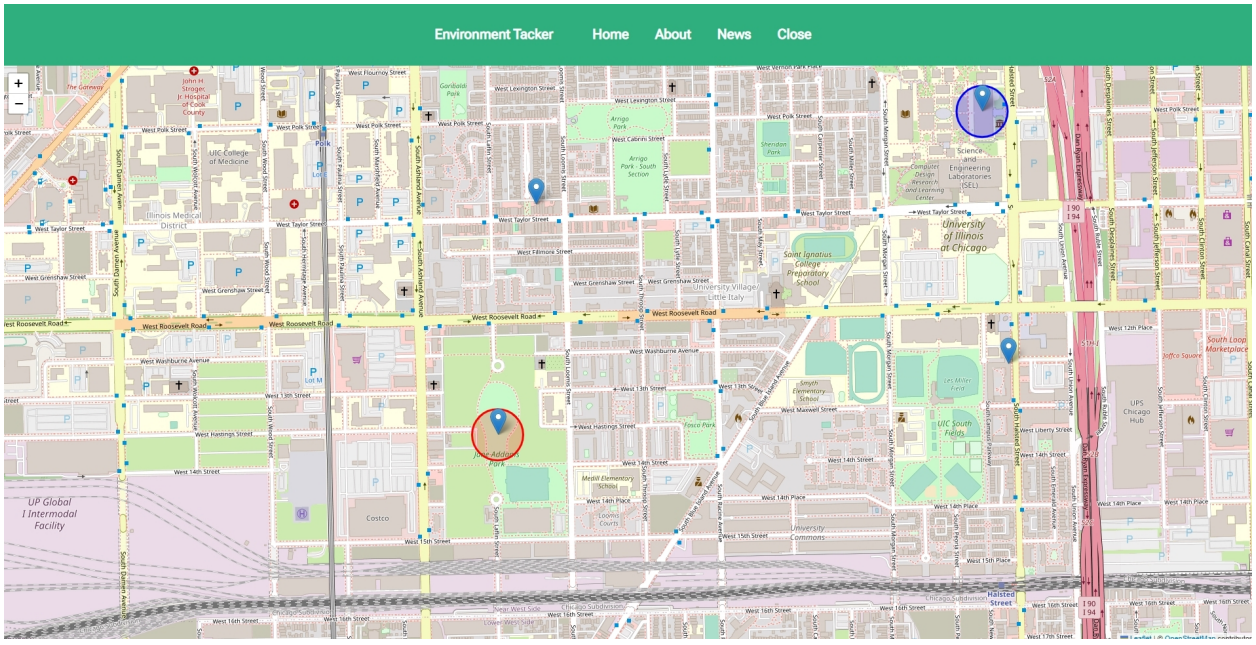
Release 2 Map Page UI

3 Third Release

The third release focused on updating UI design and intractability. The navigation bar was pretty basic during the second release and didn’t appear on every page so that's the first thing we looked at. We managed white space by giving certain pages background and came up with a consistent color scheme for the whole website. We also made the navigation bar effects when hovering over each page.



Release 2 Login Page



Release 2 Home Page

4 Comparison with Original Project Design Document

The original project design document for Global Environment Tracker had a much larger scope. Our final product is close in concept, just not in scale. We track air quality reading's around a small area (UIC) and we don't have as much data as the original project idea was hoping for. We diverged a bit from the original project design with the idea of a news page that updated with information across the internet on environmental news, but this design was never fully implemented due to time constraints, instead we added a skeleton page for it to scale.

III Techniques and Methodologies Employed

1 Extreme Programming (XP)

We're using XP to help us ensure that we're constantly delivering value to our customers. This means that we're working in short iterations, typically one or two weeks, and we're prioritizing our work based on customer feedback and changing market conditions. We're also using XP to help us improve the quality of our code, by practicing continuous integration, automated testing, and refactoring.

2 Test-Driven Development (TDD)

TDD is another technique we're using to help us ensure that our code is high quality and meets our customers' needs. With TDD, we write automated tests for our code before we write the code itself. This helps us catch bugs early, and it ensures that our code is fully tested and working as expected before we release it to our customers.

3 Kanban

We use Kanban as a visual system for managing work. In Kanban, work is represented as a series of cards or sticky notes on a board, with columns representing different stages of completion (such as "To Do," "In Progress," and "Done"). As work progresses, cards are moved from one column to the next, providing a clear visual representation of the team's progress.This helped us keep track of everyone's work and current topics.

4 Pair Programming

We're also using pair programming to help us collaborate more effectively and to share knowledge and expertise across the team. With pair programming, two developers work together on the same piece of code, with one person typing and the other person reviewing and providing feedback. This helps us catch mistakes early and ensures that our code is written to a high standard.

IV Testing

1 Items to be Tested

ID	Test Name	User	Difficulty
1A	Arduino Wifi Connection	Generic User	Medium
2A	Arduino Firebase Connection	Administrator	High
3A	Arduino Firebase Authentication	Administrator	Extremely High
4A	Arduino Component Running	Administrator	High
5A	Arduino ESP8266 Connection	Administrator	Extremely High
6A	ESP8266 Data Receiver Connection	Administrator	Extremely High
7A	ESP8266 Data to Firestore Database connection	Administrator	Extremely High

2 Test Specifications

TS1 - Arduino Wifi Connection

Description: Test Arduino connection to the wifi is connected

Items covered by this test: Check if there a connection between the Arduino and the wifi

Requirements addressed by this test: NA

Intercase Dependencies: Power source connected to the arduino

Test Procedures:

1. Connecting the preset wifi but couldn't connect: should fail
2. Connecting the preset wifi and find and connect to the wifi: show all to run

Input Specification: 4 string inputs, 2 correct strings for wifi name and wifi password as well as 2 incorrect strings to test with.

Output Specifications: If either the wifi name or wifi password is incorrect, it should say "failed to connect", otherwise it should let the arduino connect and run properly.

Pass/Fail Criteria: Passing would mean the connection does not allow invalid wifi name or wifi passwords and failing would involve allowing wrong inputs.

TS2 - Arduino Firebase Connection Test

Description: This test aims to verify the connectivity and functionality of the communication between an Arduino board and the Firebase database.

Items covered by this test:

1. Establishing connection between Arduino and Firebase.
2. Sending data from Arduino to Firebase.
3. Retrieving data from Firebase to Arduino.

Requirements addressed by this test: 3A,5A,6A

Intercase Dependencies: NA

Test Procedures:

1. Connect the Arduino board to the computer.
2. Install the Firebase library for Arduino.
3. Set up a Firebase account and create a new project.
4. Create a new Firebase Realtime Database and copy its URL.
5. Modify the example code provided with the Firebase library to include the Firebase URL and the authentication token.
6. Upload the modified code to the Arduino board.
7. Verify that the Arduino successfully connects to Firebase.
8. Send data from the Arduino board to Firebase and verify that it is received correctly.
9. Retrieve data from Firebase to the Arduino board and verify that it is received correctly.

Input Specification: NA

Output Specifications:

1. Successfully connected to Firebase.
2. Data sent from Arduino to Firebase is received correctly.
3. Data retrieved from Firebase to Arduino is received correctly.

Pass/Fail Criteria:

1. The connection between Arduino and Firebase is established successfully.
2. The data sent from Arduino to Firebase is received and stored correctly in Firebase.
3. The data retrieved from Firebase to Arduino is received correctly and can be used by the Arduino board.

TS3 - Arduino Firebase Authentication Test

Description: This test aims to verify the functionality of the Firebase Authentication service with an Arduino board.

Items covered by this test:

1. Connecting to Firebase Authentication service.
2. Creating new user accounts.
3. Authenticating existing user accounts.
4. Retrieving user account information.

Requirements addressed by this test: 2A,5A,6A

Intercase Dependencies: NA

Test Procedures:

1. Connect the Arduino board to the computer.
2. Install the Firebase library for Arduino.
3. Set up a Firebase account and create a new project.
4. Enable the Firebase Authentication service in the Firebase console.
5. Modify the example code provided with the Firebase library to include the Firebase URL and the authentication token.
6. Upload the modified code to the Arduino board.
7. Verify that the Arduino successfully connects to the Firebase Authentication service.
8. Create a new user account using the Arduino board and verify that it is created correctly in Firebase.
9. Authenticate an existing user account using the Arduino board and verify that the authentication is successful.

10. Retrieve user account information using the Arduino board and verify that it is received correctly.

Input Specification: User account credentials (email and password) for creating a new account and authenticating an existing account.

Output Specifications:

1. Successfully connected to Firebase Authentication service.
2. New user accounts are created correctly in Firebase.
3. Existing user account is authenticated successfully.
4. User account information is received correctly. Arduino is received correctly.

Pass/Fail Criteria:

1. The connection between Arduino and Firebase Authentication service is established successfully.
2. The new user account is created correctly in Firebase and can be used for authentication.
3. The existing user account is authenticated successfully and can be used to access protected resources.
4. The user account information is retrieved correctly and can be used by the Arduino board.

TS4 - Arduino Component Running Test

Description: This test verifies the functionality of individual components used in an Arduino project.

Items covered by this test:

1. Checking the functionality of the components (sensors, actuators, etc.) used in the project.
2. Making sure the components are connected correctly to the Arduino board.
3. Testing the code to control the components.

Requirements addressed by this test: NA

Intercase Dependencies: NA

Test Procedures:

1. Connect the components to the Arduino board as per the circuit diagram.
2. Open the Arduino IDE on the computer.

3. Load the code to control the components.
4. Upload the code to the Arduino board.
5. Verify that the components are connected correctly to the Arduino board.
6. Verify that the components are functional by observing their output.
7. Verify that the code controlling the components works as intended.

Input Specification: NA

Output Specifications:

1. The components are connected correctly to the Arduino board.
2. The components are functional and produce the expected output.
3. The code controlling the components is working as intended.

Pass/Fail Criteria:

1. The components are connected correctly to the Arduino board.
2. The components are functional and produce the expected output.
3. The code controlling the components is working as intended without errors or unexpected behavior.

TS5 - Arduino ESP8266 Connection Test

Description: This test verifies the communication between an Arduino board and an ESP8266 Wi-Fi module.

Items covered by this test:

1. Checking the connectivity of the ESP8266 Wi-Fi module.
2. Testing the code to establish communication between the Arduino board and the ESP8266 module.
3. Ensuring the stability of the connection between the Arduino board and the ESP8266 module.

Requirements addressed by this test: NA

Intercase Dependencies: NA

Test Procedures:

1. Connect the ESP8266 module to the Arduino board as per the circuit diagram.
2. Open the Arduino IDE on the computer.
3. Load the code to establish communication between the Arduino board and the ESP8266 module.
4. Upload the code to the Arduino board.

5. Verify that the ESP8266 module is connected and working correctly.
6. Verify that the communication between the Arduino board and the ESP8266 module is established.
7. Ensure the stability of the connection between the Arduino board and the ESP8266 module.

Input Specification: NA

Output Specifications:

1. The ESP8266 module is connected and working correctly.
2. The communication between the Arduino board and the ESP8266 module is established.
3. The connection between the Arduino board and the ESP8266 module is stable.

Pass/Fail Criteria:

1. The ESP8266 module is connected and working correctly.
2. The communication between the Arduino board and the ESP8266 module is established without errors.
3. The connection between the Arduino board and the ESP8266 module remains stable throughout the test.

TS6 - ESP8266 Data Receiver Connection Test

Description: This test verifies the ability of the ESP8266 module to receive data from a remote source.

Items covered by this test:

1. Establishing a connection between the ESP8266 module and a remote data source.
2. Receiving data from the remote source.
3. Verifying the correctness of the received data.

Requirements addressed by this test: NA

Intercase Dependencies: NA

Test Procedures:

1. Connect the ESP8266 module to the Arduino board as per the circuit diagram.

2. Establish a connection between the ESP8266 module and the remote data source.
3. Receive data from the remote data source.
4. Verify the correctness of the received data.

Input Specification: NA

Output Specifications:

1. The ESP8266 module successfully connects to the remote data source.
2. The ESP8266 module successfully receives data from the remote data source.
3. The received data is correct.

Pass/Fail Criteria:

1. The ESP8266 module successfully connects to the remote data source without errors.
2. The ESP8266 module successfully receives data from the remote data source without errors.
3. The received data is correct and matches the expected values.

TS7 - ESP8266 Data to Firestore Database Connection Test

Description: This test verifies the ability of the ESP8266 module to connect to a Firestore database and store data.

Items covered by this test:

1. Establishing a connection between the ESP8266 module and the Firestore database.
2. Storing data in the Firestore database.
3. Verifying the correctness of the stored data.

Requirements addressed by this test: 2A, 3A, 6A

Intercase Dependencies: NA

Test Procedures:

1. Connect the ESP8266 module to the Arduino board as per the circuit diagram.
2. Establish a connection between the ESP8266 module and the Firestore database.
3. Store data in the Firestore database.
4. Verify the correctness of the stored data.

Input Specification: Data to be stored in the Firestore database.

Output Specifications:

1. The ESP8266 module successfully connects to the Firestore database.
2. Data is successfully stored in the Firestore database.
3. The stored data is correct.

Pass/Fail Criteria:

1. The ESP8266 module successfully connects to the Firestore database without errors.
2. Data is successfully stored in the Firestore database without errors.
3. The stored data is correct and matches the expected values.

3 Test Results

ID# 2A - Arduino Firebase Connection Test

-**Date(s) of Execution:** April 16, 2023

-**Staff conducting tests:** Hamza

-Expected Results: The ESP8266 module should be able to connect to the Firebase database.

-**Actual Results:** The ESP8266 module was able to connect to the Firebase database.

-**Test Status:** Pass

ID# 3A - Arduino Firebase Authentication Test

-**Date(s) of Execution:** April 17, 2023

-**Staff conducting tests:** Hamza

-**Expected Results:** The user should be able to log in to the Firebase database.

-**Actual Results:** The user was able to log in to the Firebase database.

-**Test Status:** Pass

ID# 4A - Arduino Component Running Test

-**Date(s) of Execution:** April 18, 2023

-**Staff conducting tests:** Hamza

-**Expected Results:** All components connected to the Arduino board should run as expected.

-**Actual Results:** All components connected to the Arduino board ran as expected.

-**Test Status:** Pass

ID# 5A - Arduino ESP8266 Connection Test

-**Date(s) of Execution:** April 19, 2023

-**Staff conducting tests:** John Doe

-**Expected Results:** The Arduino board should be able to connect to the ESP8266 module.

-**Actual Results:** The Arduino board was able to connect to the ESP8266 module.

-**Test Status:** Pass

ID# 6A - ESP8266 Data Receiver Connection Test

-**Date(s) of Execution:** April 20, 2023

-**Staff conducting tests:** Hamza

-**Expected Results:** The ESP8266 module should be able to receive data from the Arduino board.

-**Actual Results:** The ESP8266 module was able to receive data from the Arduino board.

-**Test Status:** Pass

ID# 7A - ESP8266 Data to Firestore Database Connection Test

-**Date(s) of Execution:** April 21, 2023

-**Staff conducting tests:** Hamza

-**Expected Results:** The ESP8266 module should be able to connect to the Firestore database and store data.

-**Actual Results:** The ESP8266 module was able to connect to the Firestore database and store data.

-**Test Status:** Pass

ID# 1B - Frontend Data Responsiveness

-**Date(s) of Execution:** April 21, 2023

-**Staff conducting tests:** George

-**Expected Results:** Data from wireless nodes is updated on the front end without error every 30 seconds

-**Actual Results:** Data is updated appropriately every 30 seconds. See regression test results for repeated tests

-**Test Status:** Pass

ID# 1C - Page Resizing

-**Date(s) of Execution:** April 1, 2023

-**Staff conducting tests:** Bob

-**Expected Results:** Resizing page should resize elements neatly and zooming should not remove elements from the page

-**Actual Results:** Data is updated appropriately every 30 seconds. See regression test results for repeated tests

-**Test Status:** Pass

4 Regression Testing

1. Integration Testing: Integration testing is another type of regression testing that is performed to ensure that the different components of the system still work together as expected after any changes or updates have been made. For example, if a software application integrates with a payment gateway, regression testing

would ensure that the integration still works as expected after any changes have been made.

2. User Acceptance Testing: User acceptance testing (UAT) is a type of regression testing that involves getting feedback from end-users to ensure that the system still meets their needs and requirements. This can involve running tests with a small group of end-users to ensure that the system is still usable, efficient, and effective after any changes or updates have been made. UAT can also help identify any new bugs or issues that may have been introduced to the system.

3. Smoke Testing: This is a type of regression testing that is performed after a new build is released to ensure that the basic functionality of the system still works as expected. It involves running a subset of tests that cover critical functionalities to ensure that the system is still stable after any changes or updates have been made.

V Inspection

Code inspection is to be manually mandated and reviewed by all members of the project weekly to ensure that the validity of new code integration and feature additions maintain robust and secure.

Heavy inspection is needed on public devices to ensure that they can not be directly tampered with, connected to, or secrets cannot be explicitly extracted from these devices.

Inspection is of great importance to the team and a robust schedule of inspection will be a primary focal point of continued development.

1 Items to be Inspected

1a Public arduino nodes
1b Map Integrity
1c Database Integrity
1d Secret Validity and Storage

2 Inspection Procedures

2a Public Arduino Nodes

(1) Review the code: Review the code thoroughly to identify any potential vulnerabilities or weaknesses. Look for any hardcoded passwords or API keys that could be exploited. Also, make sure that the code adheres to industry best practices and coding standards.
(2) Check for security libraries: Make sure that the code uses security libraries, such as encryption and hashing libraries, to protect sensitive data. Also, ensure that the

code does not use deprecated or vulnerable libraries that could pose a security risk.

(3) Test the code: Test the code in a separate development environment to identify any potential issues or bugs. This can include both functional testing and security testing, such as penetration testing and vulnerability scanning.

(4) Update the firmware: Make sure that the firmware is up-to-date and includes any security patches or updates. Also, ensure that the firmware is signed and authenticated to prevent unauthorized modifications.

(5) Secure the hardware: Physical security is also important for public Arduino nodes. Make sure that the hardware is secured in a tamper-proof enclosure and that all ports and connectors are disabled or protected.

(6) Monitor the device: Set up monitoring and logging tools to track any suspicious activity or unauthorized access attempts. This can include network traffic monitoring, system logs, and intrusion detection systems.

2b Map Integrity

(1) Review the API documentation: Review the documentation for the map API to understand how it works and what data it provides. Make sure that the API is well-documented and adheres to industry best practices.

(2) Check for authentication and authorization: Make sure that the API uses authentication and authorization to protect sensitive data. Also, ensure that the API uses secure communication protocols, such as HTTPS, to protect data in transit.

(3) Review the code that interacts with the API: Review the code that interacts with the API to ensure that it is using the API correctly and securely. Look for any potential vulnerabilities, such as SQL injection or cross-site scripting (XSS) attacks.

(4) Test the API: Test the API on a separate development environment to identify any potential issues or bugs. This can include functional testing, as well as security testing, such as penetration testing and vulnerability scanning.

(5) Monitor the API: Set up monitoring and logging tools to track any suspicious activity or unauthorized access attempts. This can include network traffic monitoring, system logs, and intrusion detection systems.

(6) Update the API: Make sure that the API is up-to-date and includes any security patches or updates. Also, ensure that the API is signed and authenticated to prevent unauthorized modifications.

2c Database Integrity

(1) Check data types: Ensure that the data types used in the database tables match the data being stored. This will help prevent data corruption and ensure data accuracy.

(2) Check for normalization: Ensure that the database tables are normalized to prevent data duplication and inconsistencies. This will help maintain data integrity and prevent data update anomalies.
(3) Check for data constraints: Ensure that the database has appropriate data constraints, such as primary keys, foreign keys, and unique constraints. This will help prevent data duplication and inconsistencies.
(4) Review stored procedures and triggers: Review stored procedures and triggers to ensure that they are correctly written and don't cause data inconsistencies or data corruption.
(5) Perform backup and recovery testing: Perform backup and recovery testing to ensure that the database can be recovered in the event of a disaster or data loss.
(6) Perform security testing: Perform security testing to ensure that the database is protected against potential attacks, such as SQL injection or unauthorized access.
(7) Monitor the database: Set up monitoring and logging tools to track any suspicious activity or unauthorized access attempts. This can include network traffic monitoring, system logs, and intrusion detection systems.\

2d Secret Validity and Storage

(1) Check for secret rotation: Ensure that secrets, such as API keys and passwords, are rotated regularly to prevent unauthorized access. Also, ensure that the old secrets are revoked or invalidated after they are rotated.
(2) Check for secure storage: Ensure that secrets are stored securely, such as in an encrypted format or a secure key management system. Also, ensure that access to the secrets is restricted to only those who need it.
(3) Review access controls: Review access controls to ensure that only authorized personnel have access to the secrets. This can include implementing role-based access controls or multi-factor authentication.
(4) Perform vulnerability scanning: Perform vulnerability scanning to identify any potential vulnerabilities in the system that could be exploited to access secrets.
(5) Monitor for suspicious activity: Set up monitoring and logging tools to track any suspicious activity or unauthorized access attempts. This can include network traffic monitoring, system logs, and intrusion detection systems.
(6) Review audit logs: Review audit logs to ensure that access to secrets is logged and monitored. This can help identify any unauthorized access attempts or suspicious activity.
(7) Review compliance requirements: Review compliance requirements, such as PCI DSS or HIPAA, to ensure that the storage and handling of secrets comply with relevant regulations.

3 Inspection Results

Inspection 2a - Public Arduino Nodes

Dates: 20 Feb 2023 to 29 Apr 2023

Inspector: Hamza Ali

Discovery: Some safety flaws in the arduino nodes came to attention and were addressed with modern tools to prevent theft of proprietary code and secrets.

Inspection 2b - Map Integrity

Dates: 20 Feb 2023 to 29 Apr 2023

Inspector: Dan Alzeidan

Discovery: Some notable issues with map resizing and various features causing removal of other page components

Inspection 2c - Database Integrity

Dates: 20 Feb 2023 to 29 Apr 2023

Inspector: Alex Escatel

Discovery: Database authentication and rules allowed for free use and unlimited writes / reads from any user. Made these rules stricter and required multi factor consistent authentication.

Inspection 2d - Secret Validation and Storage

Dates: 20 Feb 2023 to 29 Apr 2023

Inspector: Mike Squires

Discovery: Attempt to reverse engineer devices without code to be able to extract sensitive information and secret or other stored information. To no avail the data was mostly secure but the sensitive and expensive sensor could physically still be vulnerable

VI Recommendations and Conclusions

All items passed testing to some degree. However the inspections and test cases are not all encompassing and the subject of security should always constantly be changing and evolving to meet the newest practices and standards.

Conclusion is such that the project should continuously be actively developed in order to make certain that investment opportunity remains consistent and no loss of assets is actualized.

VII Project Issues

1 Open Issues

1) Information is not pulled automatically from the database. Currently we must re-open the application in order for new information from the database to be pulled.

2) We currently have a problem with the login authentication. Currently some users can bypass the account login page by changing the url around.

3) Some entries in the database have inconsistent formatting or have duplicate information. This problem might come from trying to send information from specific types of wifi networks.

4) Currently there is no way to specify which readings are too high or low in problem areas. A circle will show up if readings are past certain thresholds but when opening up the popup it doesn't tell which one it is.

2 Waiting Room

1) Expand the environment recorder to show readings at places other than UIC. For this to work we would have to replace the use of Arduinos with something that is better for scalability.

2) The news page was meant to show environmental news from different sources. The hope was that we would use some sort of API to constantly pull information

about potential gas leaks in public areas, weather advisories, tornado warnings, etc.

3) Search feature on the map. The search feature would allow a user to look up certain buildings or areas without the need to drag the map around and zoom in and out to try and find a specific node.

3 Ideas for Solutions

1) Replace the use of Arduinos in order to have more consistent data. Replacing Arduino's would allow much more data to come through. Maybe pulling data from different websites/API's. This would allow us to show readings at many more places than just UIC.

2) The adding of the news page to give people more specific information. It might even be a good idea to tailor the news page based on the location of the user. This would add much more useful information than just the air quality readings in buildings.

4 Project Retrospective.

The project overall went well. We used React.js for the majority of the project which was a good choice because only one group member didn't have much experience with react. Experienced group members were able to help the less experienced group member learn the tool pretty quickly.

The way we split responsibilities was good as well. We split responsibility based on what people were experienced with and what they enjoyed doing. One group member focused on the arduino for data collection. Another member focused on managing data in the database and doing mostly back end work. The last two group members focused on front end development.

A challenge we had with development was just time. Group members were all pretty busy with other classes and sometimes that caused some weeks to be less productive for the project than others. Another challenge we had was our lack of experience with styling. While the styling of our web page isn't bad, we definitely had to learn more about styling and our UI could definitely be improved even more.

Overall we implemented the basics of what we wanted to accomplish. While the original idea was GLOBAL Environment Recorder, that was just too out of our reach for this project. However we implemented most base features with scalability in mind so with the correct resources and time it could definitely be expanded.

VIII Glossary

Arduino: referring to Arduino Uno and is a open source microcontroller board developed by Arduino.cc

PM: Particulate Matter

PM2.5: Fine Particles, a grouping of particles with an aerodynamic diameter of 2.5 μm or less, capable of penetrating deep into our lungs and even entering our bloodstream ex: vehicle exhaust, wildfires, power plant emissions, and other combustion activities

PMS5003 PM Sensor: A sensor that collect levels of PM2.5 in the area MH-Z19 CO2 Sensor: A sensor that measures the levels of Carbon Dioxide (CO2)

MQ131 Ozone Gas Sensor: A gas sensor that senses any ozone gas nearby such Cl2, NO2, and etc

VOC: Volatile Organic Compound

MP503 VOC Sensor: A sensor that detects VOC gasses such as carbon monoxide, alcohol, hydrogen, and etc

DHT22 Temp & Hum Sensor: A sensor that measures the temperature and the humidity

RTC: Real Time Clock

DS3231 RTC: Allow to get real time which helps knowing when the data was gathered

NodeMCU ESP8266: an open-source firmware and development kit that helps you to prototype or build IoT products.

IX References / Bibliography

[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[5] Andrew Stellman & Jennifer Green, "Learning Agile", O'Reilly, 2015, ISBN 978-1-449-33192-4

[6] Kenneth S. Rubin, “Essential Scrum”

[7] Kent Beck and Cynthia Andres, “Extreme Programming Explained: Embrace Change”, 2nd Edition

[8] Kent Beck and Martin Fowler, “Planning Extreme Programming”

[9] Kent Beck, “Test-Driven Development by Example”

[10] Fowler, “Refactoring – Improving the Design of Existing Code.”

[11] Hamza and Suhail, “Global Environmental Recorder Project Report”

X Index

M	
N	
O	
P • **PM2.5**	**5, 8**
Q	
R • **RTC**	**28**
S • **Sensor** • **Sustainability**	**5, 7, 15, 25**
T • **Temperature**	
U • **UIC**	**1, 5, 8, 12, 25**
V • **VOC**	**5, 28**
W • **Water**	**5**
X	
Y	
Z	