Team Satellite Fire Patrol

Final Report v1.0
April 18, 2024

Team Members:
Aaron Santiago, Ricardo Chairez and Zachary Hallemeyer

Clients:
Benjamin Wiebe, Dr. Camille Gaillard, and Dr. Christopher Doughty

Mentor:
Saisri Muttineni

TABLE OF CONTENTS

Introduction	**2**
Process Overview	**3**
Requirements	**4**
Architecture and Implementation	**6**
Testing	**9**
Project Timeline	**13**
Future Work	**15**
Conclusion	**16**
Glossary	**17**
Appendix: Development Environment and Toolchain	**18**

INTRODUCTION

The Hawaiian Islands hold immense ecological and cultural value but are currently threatened by increasing hot and dry conditions caused by climate change. This threat was evident in August 2023, when multiple wildfires across Maui resulted in at least 106 deaths, billions of dollars in damages, and ecological destruction. These fires begin in grasslands, which are vulnerable to high temperatures due to increasing drought and low moisture. Other Hawaiian ecosystems, such as coral reefs and tropical forest canopies, are also threatened by rising temperatures resulting from climate change.

Team Satellite Fire Patrol is partnering with clients Dr. Christopher Doughty, Dr. Camille Gaillard, and PhD student Benjamin Wiebe to create a GUI web application that uses real-time satellite thermal data to identify warning signs in the Hawaiian Islands and send alerts to the proper authorities. The app will automatically aggregate and process satellite thermal data from multiple sources, compare historical averages to highlight temperature anomalies, and present the data in a user-friendly interface for a range of resource management applications.

Our project is an open source web application that allows users to view thermal data over the islands of Hawaii. The web application receives its data from NASA Ecostress and its goal it to make thermal data easily accessible to researchers and the general public. In this report we will be covering how we processed and approached this project to get a good understanding of the development process. We will giving an overview of of our design process with the changes made along the way.

Our requirements will also be covered and how the client wanted our final product to work and the tools needed for these features. The finished product will be available at Thermalwatch.org and has the potential to be expanded on in the future.

PROCESS OVERVIEW

For our project we used the AGILE approach which involved us doing sprints of about two weeks which would have certain tasks assigned to each person. We started off as a team of four which made the development process manageable but as time went on we kept losing members due to unforeseen circumstances. Initially Nasya chose the role of team recorder where she would record team and client meetings which was especially important in the beginning of the project when determining requirements. Aaron chose the role of team leader because he wanted the leading experience and was in charge of communicating to the clients and mentor most meetints. Zachary chose the role of team architect because he was mainly interested in coding the project as soon as possible. Ricardo chose the role of team release manager. In the first half of capstone we lost Nasya and had to redistribute the role to the rest of the team, we also lost Ricardo halfway through the second half of the project. The roles were redistributed to Aaron and Zachary. For our version control we used github to keep track of all our changes and the entire team used VScode as their coding environment. Our team standards document became irrelevant because our process was completely chaotic.

In the end Aaron and Zach went along logically with what was needed to be done next to stay on the development schedule. Luckily, the clients were very understanding which helped us to develop things as fully as possible with our current capabilities.

REQUIREMENTS

Our project will use multiple technologies and APIs to create an easy-to-use final product in the form of a web application. These technologies will be chosen based on their performance and the project's requirements.

1. A web application that provides visualization of near real time land surface temperature data anomaly compared to historical averages.
2. A system for users to create an account, set up alerts for when a specific area reaches a certain temperature threshold, and receive alerts.
3. A system that processes historical and real time land surface temperature data and stores it.

Functional Requirements

1. **Users will be able to view near real time land surface temperatures of the Hawaiian islands.**
    a. Users can view a map of the Hawaiian islands. This map will be displayed through the MapBox framework. The map will be limited to the Hawaiian islands only.
    b. Colored overlays on the map of Hawaii will represent temperatures in certain areas.
    c. A legend will be displayed next to the map to signify what temperature the color represents.
    d. Numerical Fahrenheit and Celsius temperatures of a given area will be displayed.
    e. Map will update in real time as more recent temperature data comes in.

2. **Users will be able to view historical land surface temperatures of the Hawaiian islands.**
    a. A slider will be implemented for users to view the change in temperature over time. The slider will go as far back as our data goes.
    b. Users will be able to enter a specific date and view the land surface temperature of that time.
    c. Selecting a date will update the map in real time.

3. **Users will be able to create an account through the website.**
    a. If the user is not signed in, the "create an account" button appears on the website.
    b. Accounts will be created through the Google Accounts API.
4. **Users will be able to log into an existing account with the following information:**
    a. Google Email

    b. Password

5. **Users will be able to set up custom alerts with the following parameters:**
    a. Region of the island
    b. Temperature threshold

6. **Users will receive alerts via SMS/email when the system detects custom alert conditions have been met.**
    a. Users will receive SMS alerts when temperatures are excessive
    b. Users will receive SMS alerts with a link to access and view the map
    c. User notifications will default to SMS but have email notifications for backup

Non-Functional Requirements

Ease of Use
Our web application must be user-friendly and easy to use for users of any kind. In the future, our team will create a small walkthrough that walks new users through the website once they have logged in. The application will need to be easy to use to allow as many people as possible to take advantage of its features and alert system users.

Speed
The web application will be relied upon by users to determine if environments are in danger and because of this urgency our system must be fast. Our system must function and complete its task of displaying data within seconds. The alert system must work quickly because if there is the possibility of a fire occurring, time is of the essence. To accomplish this goal, we will attempt to make our scripts on NAU's monsoon work as quickly as possible.

Data Verification
To avoid the occurrence of needless false positives for temperature warnings, we will need to ensure that the data we receive from NASA's ECOSTRESS is correct. Incoming data will be optimized to ensure that the data we receive can be properly compared against historical averages. By cleansing the data we will have much more accurate comparisons, and we will reduce the chance of false positives occurring and alerting users when an alert was not necessary.

Scalability
Our clients have expressed interest in expanding this project to work outside of Hawaii. The goal of our project is to create code that is easily scalable to potentially work in a worldwide setting. To accomplish this, we will be using scalable frameworks and systems for our project to allow for the expansion of this alert system to be used in a worldwide setting. Our data management must be incredibly flexible and able to handle and display data from multiple locations.

ARCHITECTURE AND IMPLEMENTATION

Architectural Overview

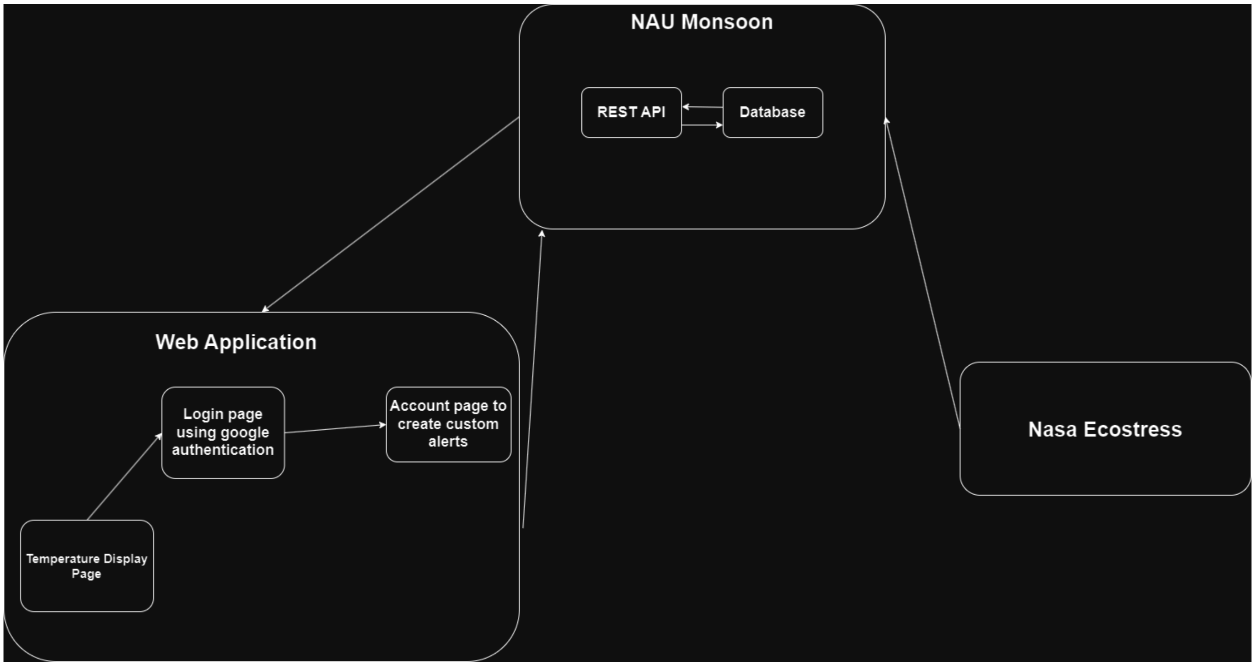


Figure 1: Architecture Diagram

Our system is composed of about two separate parts that must communicate with each other as efficiently as possible to provide the best possible user experience. The first component of our system is the back end, which will be hosted on NAU's monsoon and have these crucial tasks. First it must prompt and store the data from NASA Eco stress, then it organizes the data and places relevant information into folders based on years. It will also hold our backend framework Django, which will communicate with our front end whenever a user requests data on the map to send the correct data for those coordinates. The data will be requested and sent out of the RD port on NAU's monsoon.

Then there will be the main website, which allows users to login using their Google credentials and save up to 5 areas to be monitored. The main website will communicate with the backend and be created using Vue.js as a framework. The front end will be communicating with the RD port to show new data on the website as soon as it is requested. Users will be able to save 5 locations using a select tool, which will be saved with their account. The front end will update users if their thresholds are reached or if their areas have been updated through SMS or email using OneSignal.

NASA Ecostress API will be providing up-to-date information consistently every day to the monsoon database. Setting up our system this way allows for us to organize the raw data from NASA as soon as the satellite completes an orbit. The monsoon database will organize and calculate the averages and push its data to the data portal for the website to grab and use for the maps.

The web application will be hosted on ThermalWatch.org and will contain multiple pages and be responsible for the overall user experience. Signing in with Google accounts will enable users to save and monitor 5 locations with a selection tool. Mapbox is similar to Google Earth in both its user interface and information. This tool allows users to see an accurate satellite view/or graphical representation of Hawaii.

Mapbox also allows for graphical overlays on its maps, which we will use to display the temperature data that has been published from monsoon. The design will be easy to follow and use, which will make it as accessible as possible to the public. The website will also be responsible for using the OneSignal technology to send notifications to users when their temperature thresholds have been reached or surpassed. The notifications will send the user a notification with the information about which zone was breached and a link to access the website.

Implementation Overview

For the implementation, the project will be divided into three main components: a website application, a cloud hosted server, and a database. The web application will be made with a framework called Vue.js.

This web application is where the land surface temperatures can be seen on a virtual map. The virtual will be in a satellite style, allowing the user to move and focus on areas of Hawaii that interest them. The virtual map will be rendered with the help of an API called Mapbox. Along with this, the user will be able to create custom alerts in the web application, with both the user information and alerts being stored in our database.

The database will be contained on NAU's monsoon servers. The monsoon servers have 5 terabytes available for the web application to use. The information that needs to be stored is temperature data, both historical and current, as well as information about users and their custom alerts.

Finally, a backend program hosted on the monsoon servers will be responsible for gathering land surface temperature measurements from NASA Ecostress. Once new data is gathered, it will be processed and stored in monsoon servers. As the new information is stored, the web application will be able to request the information to display

Technologies

The team has researched different technologies and frameworks available to use in implementation and has decided on the following:

- **AWS** - Cloud computing services to host our front end web application. We used AWS to host the website and handle all the frontend services like google accounts and user interaction.

- **Vue.js -** Vue.js is a front-end framework that allows for flexible and rapid development of the web application's front end. It has the capability to effectively render data into the virtual DOM
- **Mapbox** - Mapbox is an API developed for web applications. The API allows for custom rendering of virtual maps, making it ideal for this project.

TESTING

UNIT TESTING

Unit testing is a vital component when building a software application. These tests ensure that all software components from the smallest functions to the biggest functions work as intended. More specifically, unit testing is the practice of confirming if a given function returns the expected result.

1.0 Monsoon

The software written to handle retrieving, processing, storing, and publishing the satellite data is written in Python and hosted in NAU's Monsoon. NAU Monsoon allows for fast computation provided with its vast components of hardware as well as a surplus of storage with the availability to be scaled up if needed. Python also allows for fast data manipulation with libraries such as NumPy and pandas. Because the code for Monsoon is written in Python, Pytest will be used for unit tests. This is because of the low base amount of code required to test each function. Pytest is also widely used in industry.

The code for retrieving data from external sources has the unfortunate nature of preventing 100% test coverage, as there may be unforeseen changes in API's. Therefore, testing for API's must rely on checking for correct status codes and confirming that data is being received. Furthermore, the code must be able to handle errors brought on through network interruptions and continue with computing without exiting execution. This means that the code retrieving data must be dynamic in nature to adjust to unforeseen changes.

In terms of data processing, a large part of validating incoming satellite images is ensuring that the data is from a coordinate on the Hawaii islands. This is accomplished with global_land_mask, which ensures that a given coordinate is a land coordinate rather than an ocean coordinate. Furthermore, the code validates coordinates by ensuring that they are within a bounding box containing Hawaii. Along with this, testing is performed with Pytest to ensure that a given satellite h5 file is converted to a geojson file, with land surface temperature being successfully converted to Fahrenheit or Celsius.

Cloud masking, removing clouds from satellite images, is also an important part of validating the temperature data that the site displays. In order to accomplish this, a cloud mask is also received from NASA ecostress and used to remove clouds.
In regard to data storage, the system is dynamic, so that the place in which data is stored can be easily changed and modified by simply changing data destination paths.

Finally, the processing data scripts run automatically on Monsoon on a time interval (base: 24 hours). Testing if the scripts are correctly scheduled is quite simple, as the scripts will be marked as a queued job.

1.1 Data Retrieval

- Tests for correct status codes from NASA Ecostress to confirm correct operations
- Ensure dynamic handling of errors for network interruptions
- Implement tests to confirm API response form

1.2 Data Processing

- Validate satellite image data with the use of global_land_mask as well as a limited bounding box of Hawaii
- Tests to confirm the temperature conversion from scaled Kelvin to Fahrenheit and Celsius
- Validate satellite temperature data with the use of cloud masking

1.1 Data Storage

- Validate that data retrieval and processing are correctly scheduled with the use of NAU Monsoon's job scheduler
- Test data storage is dynamic by changing data destination and processed data destination paths

2.0 AWS

Luckily for AWS there is only one section that needs to be tested. Inside the thermalwatch->scraper folder there is a scraper.js file which contains a web scraper that pulls data from the published monsoon website. To test the file, you simply run the file, and it will let you know directly from the terminal what is occurring and which files are being grabbed from the website.

For proper unit testing, we used the framework JEST, which is a JavaScript testing framework. This framework tests inputs and expected outputs, which came back looking optimal for the purposes of this project. JEST uses selenium and is widely used in large projects to test for user input which does not necessarily apply here.

INTEGRATION TESTING

Integration testing is to ensure that the different components of a software application work effectively together. For instance, in this project there is an AWS server that hosts a Vue application as well as a backend for said Vue application, and there is also the NAU Monsoon component that handles data processing. These components need to be connected together, and they need to be effectively tested to ensure that the connection can withstand error without crashing either component.

1.0 Monsoon and AWS

The data processing scripts publish the processed data to a constant URL. Therefore, the backend of AWS is able to retrieve the files at that constant URL. More specifically, a directory of available files is constantly updated by the scripts on Monsoon, which allows the backend to receive a dynamic and accurate list of all the data available for use. Therefore, the AWS backend serves as a bridge for the Vue application to use the data on Monsoon.

In order to ensure that errors are properly handled in the event of a requested file not existing or another unseen event. The backend will raise an error and continue with its execution. Similarly, due to the nature of Vue and JavaScript, the Vue application will not crash when a file is requested and not received. Rather, it will revert to a stable state.

2.0 AWS and Vue

Integrating Vue with AWS is an easy task with the built-in tools Vue has. Vue is node-based, so it runs using simple commands such as “npm run build” to build the application, which will not compile if there are any issues present. Then, once we resolve all the issues, we can run the command “npm run serve,” which serves the prebuilt application to the specified URL.

Using these methods, we will constantly test the application while adding features to it and ensure it works as expected. AWS is running Ubuntu, so everything is easy to navigate and test, which is why we will be constantly testing as we implement new features.

With the Vue framework, it also allows us to build the application to different specifications. We are currently building it normally, but once we have a finished product, we can build it for production, which is faster because Vue automatically optimizes our code and processes to run in a commercial environment.

USABILITY TESTING

Usability testing is focused on user interaction with our product. Our goals for this section are to ensure the user has the best experience possible using our product and to remove any bugs or unexpected behaviors. We will be mainly focusing on the website since the users will only have access to it, and we want to make it as easy to understand as possible. For our usability testing, clients and other students outside of our degree program will be testing out the website to provide helpful feedback on its usability. The clients will provide the most useful feedback since this product is being built for their needs. The students outside of our program will provide feedback for the general public, which is also something that needs to be included because the clients want to release this project to the public.

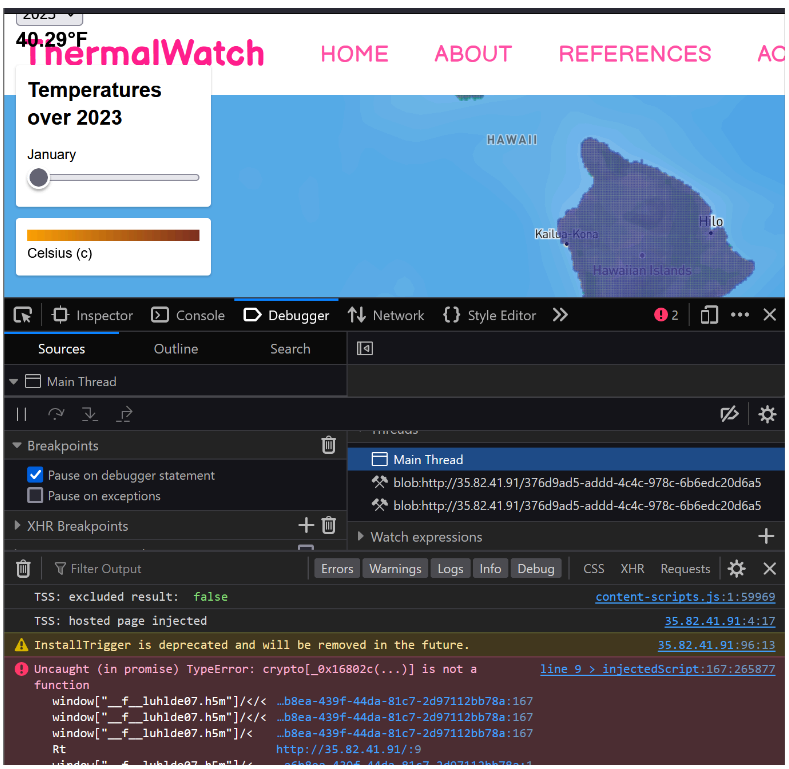
1.0 Monsoon

Monsoon itself will not be interacted with by the user, so we did not do any usability tests for it. Luckily our clients are well versed with monsoon and will feel comfortable using it for this project's purpose. All documentation and help related to monsoon as itself is not managed by us, so it is subject to change without notice.

2.0 AWS

Our website is hosted on AWS at thermalwatch.org. The website is fairly user-friendly, with more updates being applied. To check for user experience testing, we try to simulate what a normal user would do and reduce errors during development by using the inspect tool directly on the website. We will continue to present the website to the clients who can then recommend changes based on usability. Below is an example of what the inspect tool appears as with an error.

Figure 2: Inspect Element Tool



PROJECT TIMELINE

Spring 2024 Capstone
This semester will include the bulk of the software development process. We will be splitting our development process into two parts. The first part will focus on the backend design of the project. For our backend design, we will be setting up NAU's monsoon to manage and organize the data from NASA's ECOSTRESS in a database in a way that is easy to access from. The second part of our project will focus on the Frontend of our project by creating an intuitive and beautiful design.

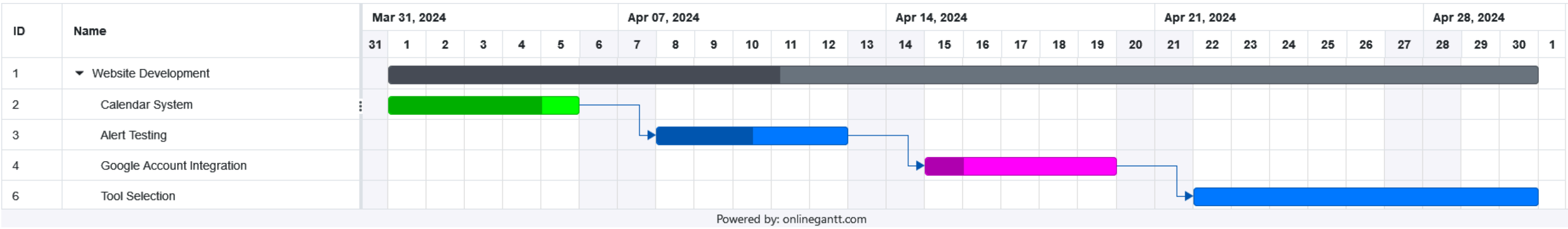
Data Organization ~ January 2024 - March 2024
We used only Python and monsoon to handle our data management because that is what we could handle during development. Zachary Hallemeyer led on this part of the project because he enjoys working on the backend. Aaron Santiago also helped to develop the scripts that organize the data for easy data management.

Alert System ~ April 2024
We will be relying on an API to function for our alert system. The script for this will be included in the backend of development, so it can notify people as soon as possible. Zachary Hallemeyer took the lead on this and completed the implementation using bevo.

User Authentication ~ April 2024
Instead of managing usernames and passwords, we will be incorporating google account's API to ensure users can easily log in and access our web application. Aaron Santiago took the lead on this implementation.

Web Application ~ February 2024 – May 2024
The web application is a crucial part of the project because it will be the part that users interact with. Users will be able to request data and control which regions they want to monitor through the web application. Our website will be using Mapbox AI because of how easy it is to implement our data with it. We will be showing historical temperature data and current data on the website on a 2D/3D map that has an overlay for the temperatures. We are currently in this phase, the whole team has been working on it together. Zachary has put in extra work to ensure we finish on time.

Figure 3: Schedule



FUTURE WORK

As the team worked on this project it became clear that this product had a lot of potential. Unfortunately, since we had a smaller team we were unable to expand it to its full potential and make it a worldwide monitor. We were comforted because the clients said they wanted the tool to be open source so we are expecting others to add to it after we all graduate.

For version 2.0 of thermalwatch, we believe it would be beneficial to upgrade the system to included North America in its scope. To do this the bounds on AWS and monsoon would need to be altered to include the new coordinates and testing would need to be done to ensure the map overlay is working as intended. Luckily this system is not complicated to understand it is just a lot of connecting parts to get it to function properly.

In addition, another feature that could be implemented is group monitoring. This feature would allow you to add users to your group via the website and be able to see what they are monitoring in addition to your saved locations. This could be especially useful for research teams who are focusing on separate areas.

CONCLUSION

With the increase in wildfires and rising coral reef temperatures, the Hawaiian islands are vulnerable to the threat of climate change. Team Satellite Fire Patrol is working on a project with clients Dr. Camille Gaillard, PhD student Benjamin Wiebe, and Dr. Christopher Doughty to create a software that helps identify warning signs in the Hawaii area from real-time satellite thermal data. Our web application, thermalwatch.org was built using AWS and monsoon and implements features in relation to these points:

- Historical data
- Real time data
- User authentication
- Custom alerts
- Overlaying thermal data

We believe that this product will help our client and other researchers to save time and easily access thermal data for their research. If our clients were to expand the project the product could be used worldwide to prevent wildfires and save sensitive ecological zones, potentially mitigating some of the effects of climate change.

Overall, capstone has been the most stressful project that all of us have experienced in our undergraduate degree. We wish we had more members from the start or had less member drop out of the project so we could have had a better spread of the workload and potentially have more features in the finished product.

This class has taught us what to expect from group projects in the future and how to deal with tough situations.

GLOSSARY

AWS - Amazon Web Services
Monsoon - Northern Arizona University's super computer.

APPENDIX: DEVELOPMENT ENVIRONMENT TOOLCHAIN

Basic Setup for development is as follows:

Hardware:

Our team mainly developed in windows and Mac. We do not believe there are any requirements for this project other than a computer that can run python and node. If you want to be able to test a lot of data files you will also need a lot of storage on this machine to ensure you can store all geojson files locally.

Toolchain:

The team relied on various tools in this project. We used code editor Visual Studio Code for our coding and git/github for maintaining our code. Here is a list of some of the frameworks and libraries we used:

- NumPy
- Vue.js
- Selenium
- VueCalendar
- Mapbox
- Google Accounts API

Setup:

For setup locally you must have vue.js installed in the same directory or one directory apart from your GEOJson data. To test the python scripts you can run them using “python filename.py” for the frontend testing you must have Vue.js installed on your machine and the accompanying packages for mapbox.
To run your basic website you will need to use the command “npm run build” to build your vue.js project and “npm run serve” after you make changes to your project to refresh the build. We have included documentation on how to run everything on github.

Production Cycle:

To run the data scripts that get the data downloaded on to your machine you will need to run “python ProcessDataManager.py” this will download all the requested information specified in the file to get all the thermal data files.

For testing out the front end you will need to traverse to the “thermalwatch” folder in the vue.js project. There there will be views which include the information needed for each separate page of the website. In the views folder you will be able to change the html and elements of the website for each page. Once you have made your changes you will need to traverse back to the “thermalwatch” folder and run “npm run serve” to refresh the changes you have made. If nothing changes you have made really large changes and must rebuild the website by running “npm run build” then “npm run serve” to see the newest version of the site.