Test Plan

01. Scope of work

1.1 Components and Functions to be Tested

We will test all front-end UI, routing between web pages, back-end Python and JavaScript code, and imports and exports from the database.

1.2. Components and Functions Not to be Tested

We got our IoT data from a Kaggle dataset and are assuming the data are accurate. Because we are not actually using IoT in our project, there will not be any testing involved for that component.

1.3. Third-Party Components

We used Leaflet.js for the US maps in the Maps Page to display air quality data collected in various locations. Leaflet is used for displaying a pin in each of the specified coordinates. There are no tests planned at this point because we can visually see the pin locations and any bugs would be obvious.

02. Quality and acceptance criteria

2.1. Release quality acceptance criteria

The baseline for each page in our project is:

- Main (Dashboard) Page
    - Users have access to relevant bar charts, scatterplots, and pie charts using the data extracted from our database
    - Users can access a calendar that lets them switch between dashboards on different dates
- US Map Page
    - Users can see a US map with colored pins that show the AQI at that location
    - Users can click on a pin and jump to the Analysis Page of that location
- Analysis page
    - Users have access to a map showing the chosen location
    - Users have access to a Mean chart of each pollutant (CO, NO_2, O_3 and SO_2)
    - Users have access to a Time-series line graph of pollution in the location
    - Users who are logged in can post comments on the page and interact with other users
- Login Page
    - Users can log in with a username and password
    - Users can choose the Forgot password/username link to reset their login info

- New Account Page
    - Users can enter their username, email, and password to create a new account
- Account Setting Page
    - Users can change the username, password, and user information stored in the database

2.2. Sprint quality acceptance criteria

Each issue completed must be functional without any outstanding bugs before it is pushed to a branch. The pushed code must be approved by two other members of our group before it is merged into the main branch.

03. Critical success factors

- Visually inform users about the air quality of various locations around the country
- Allow users to leave comments and interact with other users

04. Resources

4.1. Testing Tools

Tool	Comment
Unit Tests	for testing python and js code
Py Auto Gui	for testing the user interface
Flask Tests	for routing
Selenium	for routing

05. Test documentation and deliverables

Title	Responsible person(s)	Frequency (delivery time)	Method of delivery
Testing before pushing code updates to a branch	The person who was assigned the issue and wrote the code	Once a portion of a function is complete	Virtual
Testing before approving a merge into the main branch	Any two people in the development group	Every time a pull request is made by a member	Virtual

06. Defect and documentation tracking

6.1. Bug Priority Definitions

We will be using the typical method of assessing bug priority levels

Priority	Description
Highest priority bugs	Bugs that would cause serious system malfunctions, data loss, breach of user privacy, and similar security and functional issues for users, administrators, clients, and/or developers.
2nd priority bugs	Bugs that cause partial system malfunctions that affect the website performance for users, administrators, clients, and/or developers.
3rd priority bugs	Bugs that cause less serious system malfunctions that affect the website performance for users, administrators, clients, and/or developers.
4th priority bugs	Bugs that cause slight system malfunctions that do not affect the main website functionality for users, administrators, clients, and/or developers.
Lowest priority bugs	Bugs with minor or insignificant effects on the website performance for users, administrators, clients, and/or developers.