Testing Overview

for

W A T E R R A

Version 1.0 approved

Prepared by

Group 15: Cynthia (Yingxue) Liu, Jennie (Jian) Li, Zoe (Huaijin) Ning

McMaster University

March 14, 2023

Revision History

Version	Name	Reason For Changes	Date
1.0	*Cynthia* *Jennie* *Zoe*	• Add more details to the description and rationale of each test case. • Be more specific in corrections resulting from the testing parts	*March 3*

1. Introduction

In this month, we have performed extensive testing on our project to ensure its quality and reliability. We believe that rigorous testing is an integral part of the development process, and it is essential to detect any issues or bugs before the project is released to the end users. The testing process will involve various types of tests, including unit tests, integration tests, system tests, and acceptance tests, to ensure that the project meets all the functional and non-functional requirements.

Apart from testing, we also plan to implement several new features to enhance the project's functionality and user experience. These features have been identified based on feedback from our supervisor. Our development team is working diligently to ensure that the new features are implemented correctly and integrated seamlessly with the existing codebase.

Overall, we are committed to delivering a high-quality and robust project that meets the needs and expectations of our users. We believe that testing and continuous improvement are critical to achieving this goal, and we are dedicated to putting in the necessary effort to make it happen.

2. Test plan

There are three key features that we perform testing on. In this section, the test plans of each feature and the rationale behind them will be discussed separately. All of the tests will use fake data we upload to the *influxDB* database through *Postman*. Tests will be mainly implemented as *requirement-based testing.*

2.1 Feature 1: Data Visualization

The feature we will test is data visualization, which allows users to view and analyze data in various formats, such as graphs, charts, and tables. The test plan for this feature includes the following tests:

- Data display: Test if the system correctly displays data in the chosen format (e.g., line chart and table).
- Data filtering: Test if the system allows users to filter data based on specific criteria, such as date ranges (e.g., time period) or categories (e.g. node, probe, duration).
- Responsive design: Test if the data visualization components are responsive and adapt to different screen sizes and devices.

Rationale: Ensuring that the data visualization feature works correctly is essential for providing users with valuable insights and helping them make informed decisions. By testing different aspects of this feature, we can confirm that users can effectively visualize and analyze data using our system.

The tests are displayed in the following three aspects.

2.1.1 Colour indicating data level

The website shows the data level of a selected probe type on the website's initial map page.

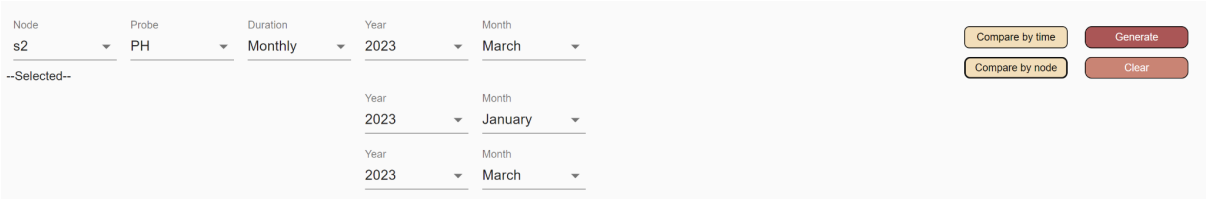
The testing needs to verify the following features, and they are tested by selecting multiple different probes and seeing the color change of map markers.

- Verify that the nodes change marker color based on the data level of the selected probe type.
    - Test 1: Change the selected probe to see if the markers change colors correctly.
    - Test 2: Set different data level intervals for a probe, and see if the markers change colors correctly according to the change of the intervals.
    - *Rationale*: It is more intuitive to distinguish the different levels by marker colors. Data level intervals applied to different probes should be easily changeable to adapt to waters in different regions, so, it is also necessary to make the correct change after changing the interval boundary values of each level.
- Verify that the color is generated according to the latest data.
    - Test: See if the markers change colors correctly after the new data is integrated.
    - *Rationale*: The marker colors should update based on the latest data. This is a practical application of our real-time database.
- Verify that nodes without the selected probe type are grayed out.
    - Test: See whether the markers of nodes without the selected probe type are grayed out correctly after the new data is integrated.
    - *Rationale*: The corresponding markers should be grayed out indicating there is no data for the selected probe type and avoiding ambiguity.
- Verify that the pop-up shows the correct value for each node.
    - Test: See if the pop-ups of markers display the correct data after the new data is integrated.

- *Rationale*: The pop-ups should update based on the latest data. This is a practical application of our real-time database.

2.1.2 Compare different times for fixed nodes and probes

By selecting a node, a probe type, and multiple time periods, users will be able to compare yearly/monthly/weekly data on an overlapping line chart. The user interface would be like below:



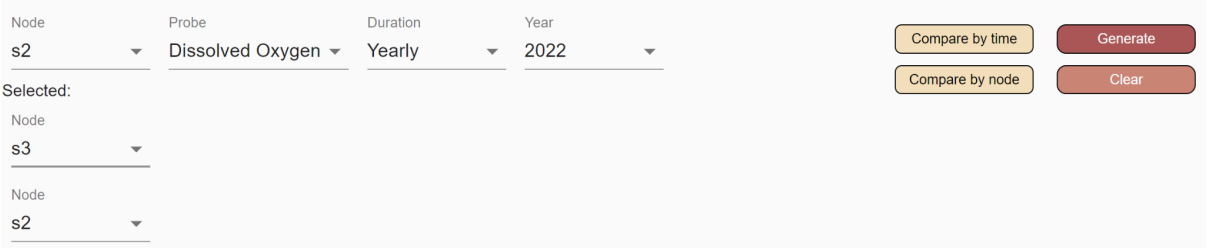
The testing needs to verify the following features, all of the testing need to be successful to ensure the function works properly:

- Verify that the “Compare by the time” button clones the time-selection dropdown(i.e. Dropdown on the right side of Duration).
  - Test 1: Select “Duration” of “Yearly”, and see if the clone dropdown is generated correctly in format.
  - Test 2: Select “Duration” of “Monthly”, and see if the clone dropdown is generated correctly in format.
  - Test 3: Select “Duration” of “Weekly”, and see if the clone dropdown is generated correctly in format.
  - *Rationale*: The clone dropdown gives a clear restriction on selection.
- Verify that clicking a drop-down other than time selection will start a new comparison after clicking the “Compare by the time” button.
  - Test 1: Select “Duration” of “Yearly”, and see if the clone dropdown is generated correctly in format.
  - Test 2: Select “Duration” of “Monthly”, and see if the clone dropdown is generated correctly in format.
  - Test 3: Select “Duration” of “Weekly”, and see if the clone dropdown is generated correctly in format.
  - *Rationale*: The clone dropdown gives a clear restriction on selection.
- Verify that the same data would not be populated twice(i.e. There is no duplicated time period)
  - Test: Select the same time period twice, and see if it deals with duplicated data through the console log.
  - *Rationale*: The exclusion of duplicates helps with efficiency by preventing calling the back-end API twice.
- Verify that the “Generate” button sends the data to the API and generates a line chart below the map.
  - Test 1: Select different data and generate the graphs multiple times, and see if the graphs are generated correctly.
  - Test 2: Select the same data and generate the graphs multiple times, and see if the same data generates the same graph correctly.

    - *Rationale*: A single line chart or an overlapping time series line chart needs to be generated based on the data.
- Verify the “Clear” button clears all input data.
    - Test 1: Clear all the data, and see if the UI successfully clears all the history selections.
    - Test 2: Clear all the data after generating the graphs, and see if the corresponding front-end variables change into the initial states through the console log.
    - *Rationale*: The “Clear” function needs to clear the user input and set all the variables to the initial states in order to deliver new data without any overlapping.
- Verify that the line chart shows the data for the selected node, probe type, and time periods.
    - Test 1: Select different data and generate the graphs multiple times, and see if the chart information is correct.
    - Test 2: Select data and generate the graphs, and see if the front-end records the correct variables through the console log.
    - Test 3: Select data and generate the graphs, and see if the variables sent to the back-end API are correct.
    - *Rationale*: The data has to be recorded by the front-end correctly and call the correct back-end API in order to return the correct result. The chart can do with most of the data graph.

2.1.3 Compare different nodes for fixed probe type and time

By selecting multiple nodes, and a probe type, users will be able to compare a specific probe type data of different nodes on an overlapping line chart.



The testing needs to verify the following features, different from the above method, the main comparison for this will be done via the “Compare by node” button.

- Verify that the “Compare by node” button clones the current node dropdown.
    - Test: Select different nodes, and see if the clone dropdown is generated correctly in format.
    - *Rationale*: The clone dropdown gives a clear restriction on selection.
- Verify that clicking drop-down other than node will start a new comparison after clicking the “Compare by node” button.
    - Test: Select different nodes, and see if the clone dropdown is generated correctly in format.
    - *Rationale*: The clone dropdown gives a clear restriction on selection.
- Verify that the same data would not be populated twice(i.e. There are no duplicated nodes)

    - Test: Select the same nodes twice, and see if it deals with duplicated data through the console log.
    - *Rationale*: The exclusion of duplicates helps with efficiency by preventing calling the back-end API twice.
- Verify that the “Generate” button sends the data to the API and generates a line chart below the map.
    - Test 1: Select different data and generate the graphs multiple times, and see if the graphs are generated correctly.
    - Test 2: Select the same data and generate the graphs multiple times, and see if the same data generates the same graph correctly.
    - *Rationale*: A single line chart or an overlapping time series line chart needs to be generated based on the data.
- Verify the “Clear” button clears all input data.
    - Test 1: Clear all the data, and see if the UI successfully clears all the history selections.
    - Test 2: Clear all the data after generating the graphs, and see if the corresponding front-end variables change into the initial states through the console log.
    - *Rationale*: The “Clear” function needs to clear the user input and set all the variables to the initial states in order to deliver new data without any overlapping.
- Verify that the line chart shows the data for the selected node, probe type, and time periods.
    - Test 1: Select different data and generate the graphs multiple times, and see if the chart information is correct.
    - Test 2: Select data and generate the graphs, and see if the front-end records the correct variables through the console log.
    - Test 3: Select data and generate the graphs, and see if the variables sent to the back-end API are correct.
    - *Rationale*: The data must be recorded by the front-end correctly and call the correct back-end API to return the correct result. The chart can do with most of the data graph.

3. Test case

3.1 Colour indicating data level

The test case for color indicating data level involves selecting a probe and observing the color of the data points on the map. For example, in a specific case, the user can select probes “PH” and “Air Temperature” to test if the system can generate colors for the data points based on their value. These two probes are chosen because they have the most data and a variety of colors, which means that every node has data for them, and even some of them have multiple data. This allows the user to test if the system generates the color according to the latest value.

Moreover, “Air Temperature” is also selected to test if the API fetches data correctly when there is a space in the name of the probes. This is important because some probes may have names with spaces or special characters, which could cause errors in data fetching and analysis.

During the testing, the system also undergoes random testing to check for any unforeseen errors or issues that may arise. This is an important step in testing as it helps to identify any potential bugs or flaws in the system.

Overall, testing for color indicating data level is a crucial aspect of data analysis as it helps users to quickly identify and interpret the data points on the chart. Ensuring that the system can generate colors based on the latest value and handle probes with spaces or special characters is essential for providing accurate and reliable data analysis tools to users.

3.2 Compare different time for fixed node and probe

The test case for different time periods for fixed nodes and probes involves selecting different dates to observe the data for fixed nodes and probes. For example, in a specific case, the user can select node “s2”, probe “Dissolved Oxygen”, duration “Weekly”, and date “20220801”. The purpose of this test is to ensure that the system can generate a chart that displays the data for the selected node and probe under a specific time period.

Furthermore, the test also checks the non-functional feature that allows the user to compare data of the same node and probe in different time periods. Once the user selects a time period, clicking on the “Compared by time” button allows the user to select the other time period. Following the example above, the user selects the date “20220801” first. Then, click on the “Compared by time” button to choose a second date under the same weekly duration “20221012”. Both of the selected time periods are saved for data fetching and displayed below the selection menu.

Tests like continuously clicking on the “Compared by time” button twice are also conducted to see if duplicate time-selected is produced. This is important because the system should not allow duplicate time-selected as it could cause confusion and errors in the data analysis.

Additionally, the test also checks if changing the duration when there is already time-selected clears out the selected time. This is an important feature because changing the duration without clearing out the selected time could produce incorrect or misleading data.

Overall, testing for different times for fixed nodes and probes is a crucial aspect of data analysis as it allows users to observe the trends and patterns in the data over time. Ensuring that the system can generate accurate charts and handle non-functional features appropriately is essential for providing reliable data analysis tools to users.

3.3 Compare different nodes for fixed probe type and time

The testing for different nodes for fixed probe type and time involves selecting different nodes to observe the data for a fixed probe type and time. This test is similar to the testing described in section 3.2, where the only difference is that we are changing the node drop-down menu for this test.

In this test, the user selects probe “PH”, node “s1”, “s2”, and date “20221012”. The system then generates a chart with two lines below to display the data for the selected nodes. The purpose of this test is to demonstrate that the system can generate a multi-line graph even if one node lacks some portion of data.

For instance, if node s1 has complete data for the selected probe and date, while node s2 has missing data for a certain period, the system can still generate a multi-line graph that shows all the available data within the time period for both nodes. This helps to visualize the trends and patterns in the data for each node, even if they do not have the same amount of data available.

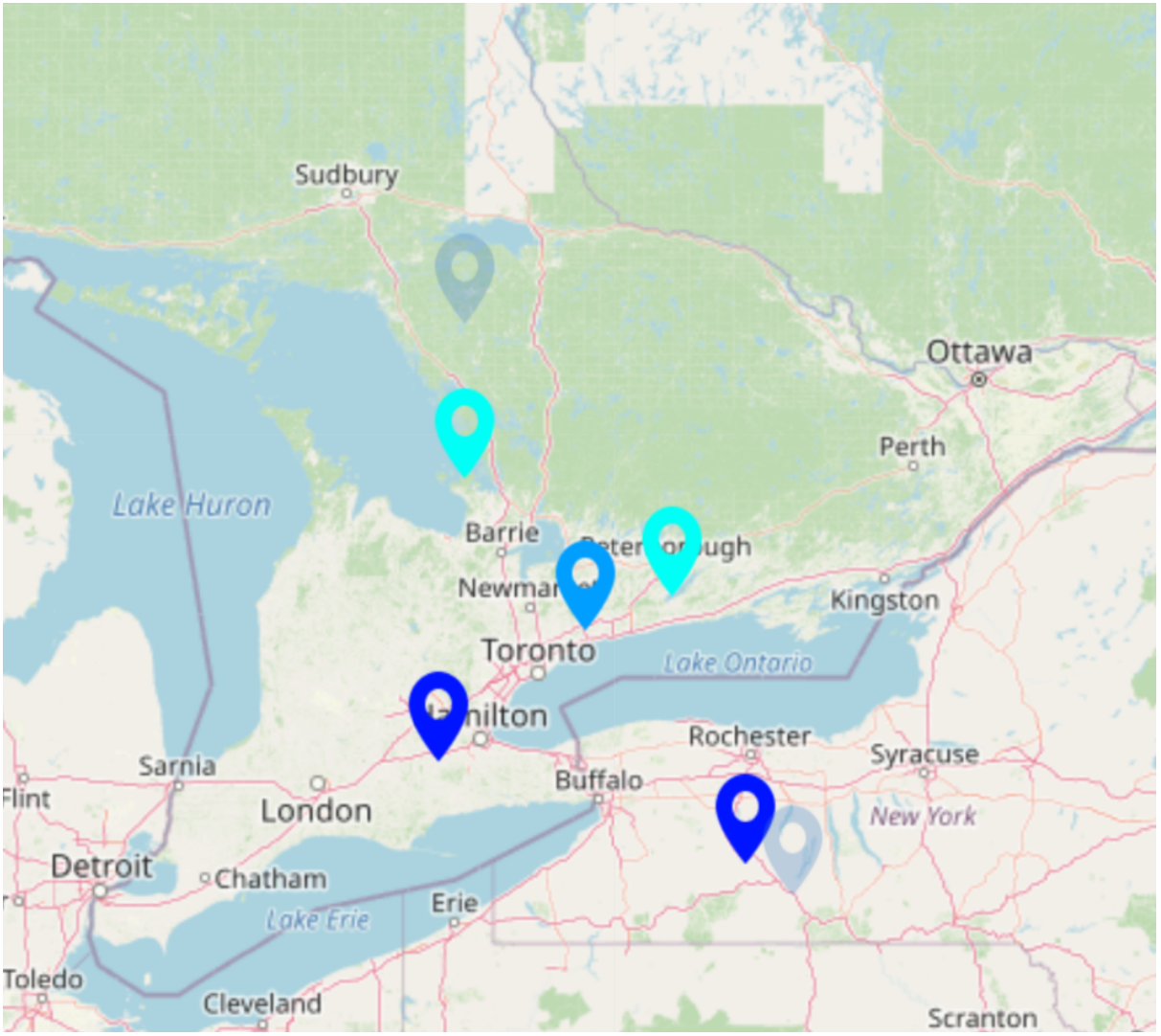
In summary, testing for different nodes for fixed probe type and time is a useful feature that allows users to explore the data from different nodes and observe the trends and patterns in the data. The ability to generate line graphs even with missing data from one or more nodes is an important aspect of this feature, as it helps to provide a more complete picture of the data.

4. Test result

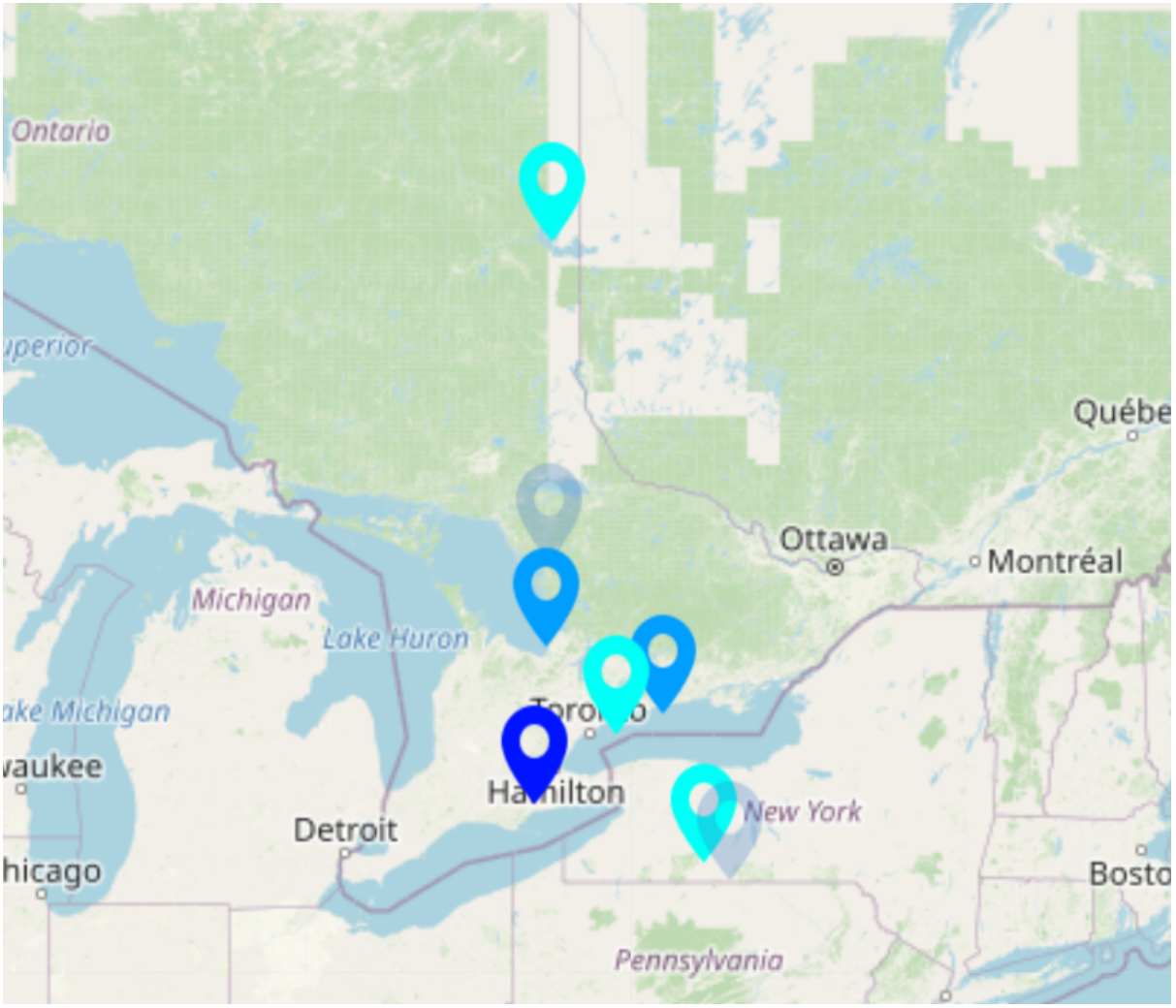
4.1 Colour indicating data level

To intuitively distinguish different levels of data of a probe on the map, node markers are displayed in different colors. A deep blue marker means that the latest data of the selected probe of this node is at high level. Similarly, a normal blue marker represents middle level, a light blue one represents low level, and a faded gray marker represents the absence of data.

Test: Select the probe “PH” to check if markers on the map are in correct colors.
According to the data we have in the database and the interval boundary values we set, the output for selecting probe “PH” is:



Test: Select the probe “Air Temperature” to check if markers on the map are in correct colors.
According to the data we have in the database and the interval boundary values we set, the output for selecting probe “Air Temperature” is:

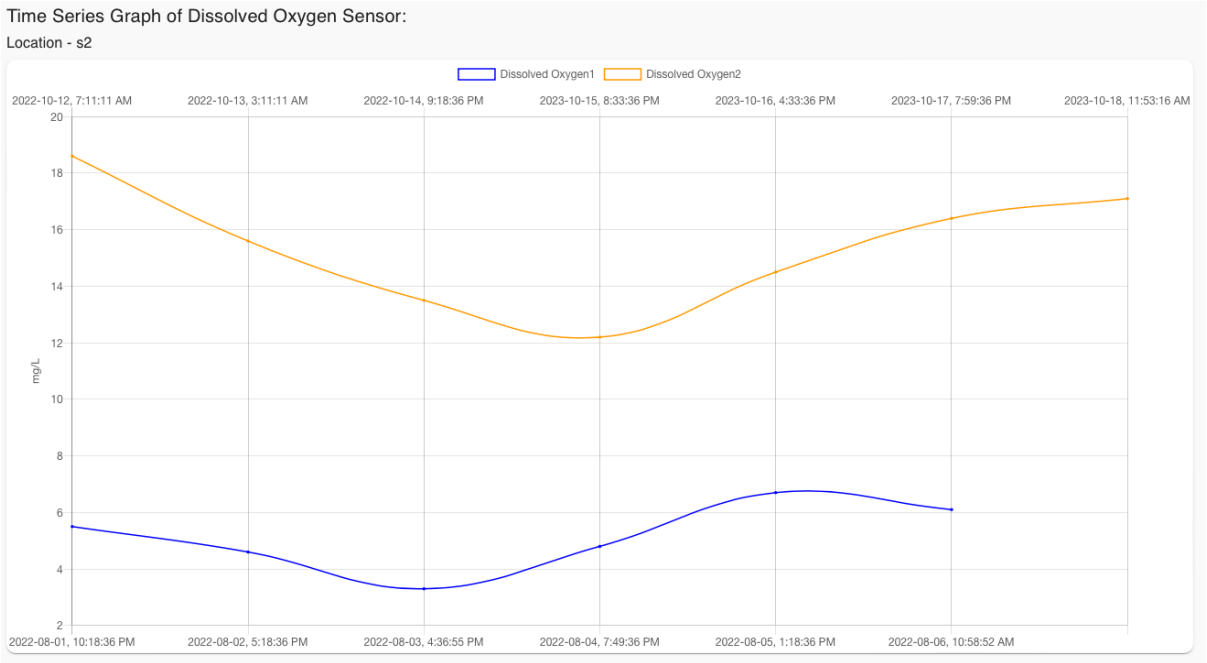


Corrections resulting from the testing:
We made some changes in *Map.tsx* and *MapDetail.tsx* to pass all test cases. In *Map.tsx*, The function that examines data levels and classifies nodes into high, middle, low, and gray kinds based on it has been improved through changing the logic of the for loop and conditions of the if statements. Then, the classified nodes are sent to *MapDetail.tsx* as arrays of an array, and elements on the map are created and rendered according to it. Each object in the sub arrays are mapped to generate correct markers and pop-ups. We also modified the function that creates markers to adjust marker size, marker source images, and pop-up content.

4.2 Compare different time for fixed node and probe

The chart is a multi-axis line chart generated by *Chart.js*. The probe value is plotted on the vertical y-axis and the time is plotted on the horizontal x-axes. To show the multi-line chart directly, we use different colors to draw the two lines. The lines share the same y-axis, but each corresponds to an independent x-axis. All points on both of the lines have a pop-up that can be displayed by hovering the cursor right on the point.

Test: Select node “s2”, probe “Dissolved Oxygen”, duration “Weekly”, date “20220801” and date “20221012” to generate a multi-line graph.
According to the data we have in the database and all the selected values, the graph looks like this:

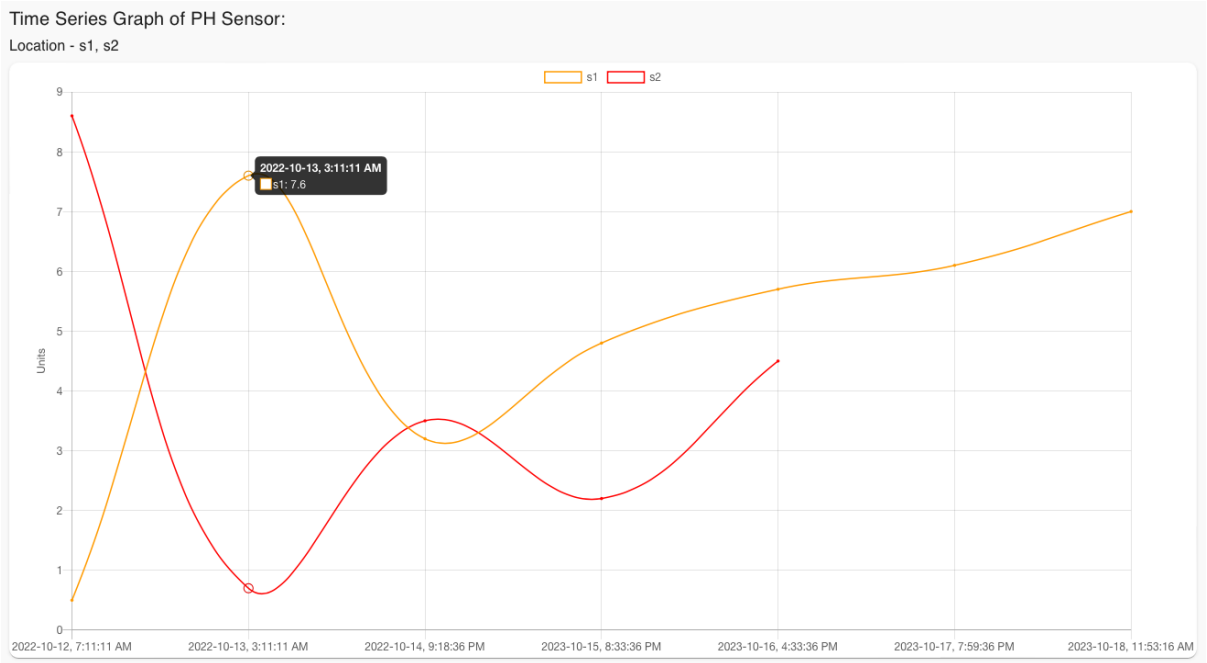


Corrections resulting from the testing:
We made some changes in *Menu.tsx*, *SensorTable.tsx*, and *Chart.tsx* to pass all test cases. In *Menu.tsx*, we have modified the click handler of the “Compared by time” button to change the format of saving selected time periods, and display the selected time below the menu. This modification is important and necessary to satisfy the need of drawing multi-line charts. Then, the selected time periods are passed to *SensorTable.tsx* and used to fetch data. The process of parsing fetched data is also adjusted to adapt the new format. In the render part of *SensorTable.tsx*, class Chart is called with parsed data to draw the chart. *Chart.tsx* undertakes the entire drawing function. Changes are made to set up the axes and datasets for multi-line charts.

4.3 Compare different nodes for fixed probe type and time

Charts generated here follow similar settings mentioned in section 4.2. The only difference is that both x and y axes are shared. The probe value is plotted on the vertical y-axis and the time is plotted on the horizontal x-axis. All points on both of the lines have a pop-up that can be displayed by hovering the cursor right on the point.

Test: Select probe "PH", node "s1", "s2", and date "20221012" to generate a multi-line graph.
According to the data we have in the database and the all the selected values, the graph looks like this:



Corrections resulting from the testing:
Corrections made here are similar to what mentioned in section 4.2. Main difference happened in *Menu.tsx* about the handler of the “Compared by node” button. Without changing the format of saving selected values, we simply added a push command to add the second selected node to an existing variable in our menu props. Small adjustments in *SensorTable.tsx* and *Chart.tsx* are made for chart visualization optimization as well.

5. Conclusion

Based on the test cases and results, we can conclude that the system is capable of providing reliable and accurate data analysis tools to the users. The system demonstrates its ability to generate color-coded node points based on the data level, compare data for fixed nodes and probes under different time periods, and compare different nodes for fixed probe type and time.

The corrections made during the testing process improved the system's performance and usability. Adjustments in *Map.tsx* and *MapDetail.tsx* ensured that markers and pop-ups were generated correctly according to data levels. Changes in *Menu.tsx*, *SensorTable.tsx*, and *Chart.tsx* facilitated the generation of multi-line graphs and provided better data visualization.

In conclusion, the system is well-designed and effective in providing valuable data analysis tools to users. By identifying and resolving potential issues during the testing process, the system's reliability and performance have been enhanced, making it a useful tool for users to explore and understand water quality data sets.