WATERRA

Ohneganos

2023 Computer Science
Department of Computer and Software

CYNTHIA LIU, JENNIE LI, ZOE NING
SUPERVISOR: DR. EMIL SEKERINSKI

KEYWORDS: Water Quality, Water Sensor, Indigenous Communities, Secondary Education Communities, Mac Water, Real-time Data, Digital Indigenous Map, Six Nations of the Grand River Territory

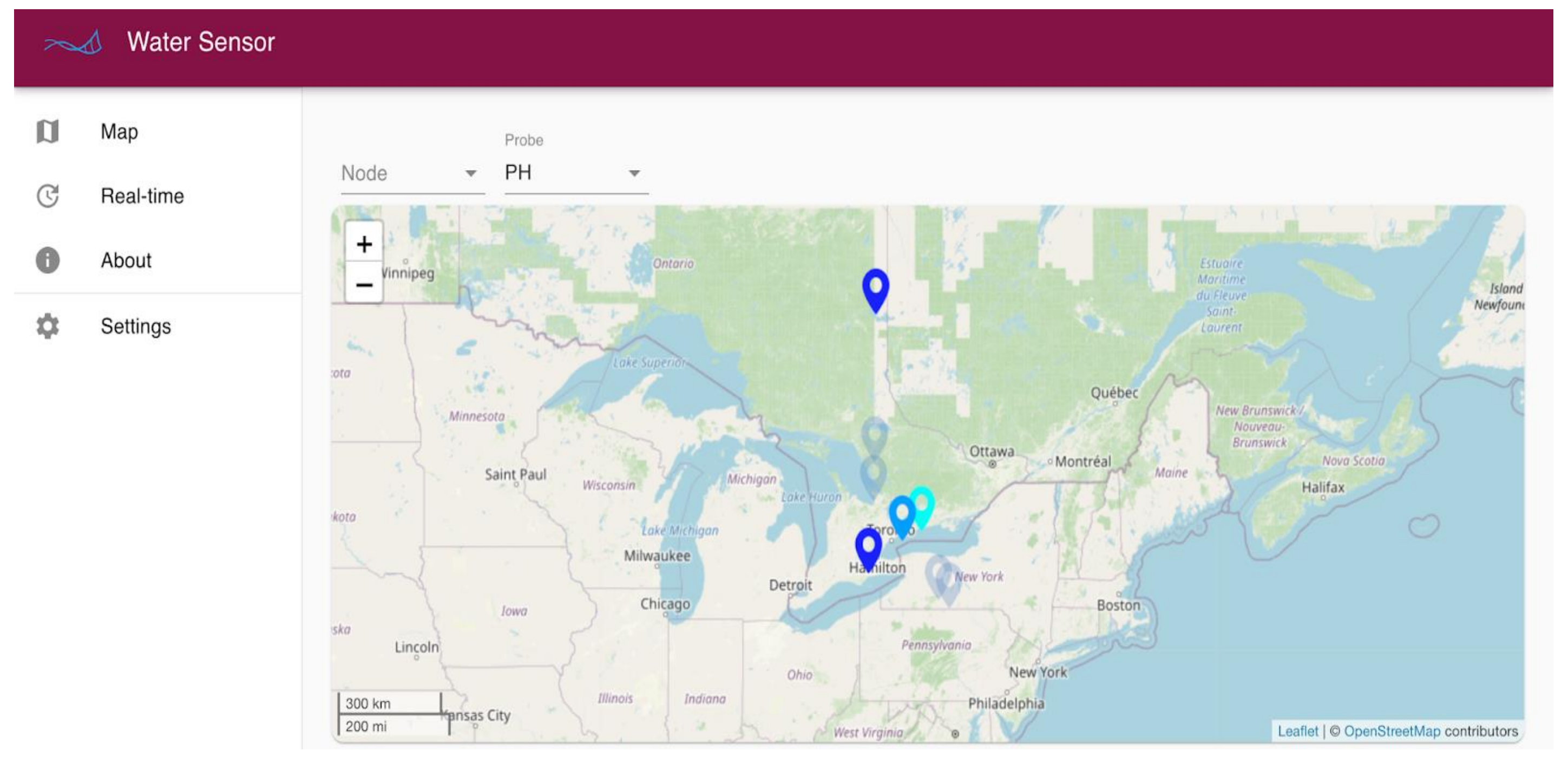
Introduction

Water quality is a critical issue that affects everyone, yet it can be overwhelming to understand the complexities of this topic. Our project is a sub-project of the *Ohneganos* program, which is an indigenous water research program focused on indigenous and other local communities to help them easily access and understand water quality data.
By integrating and analyzing data from a time-series database and using the tools *InfluxDB*, *Postman*, *FileZilla*, we have created a website that is user-friendly and informative. Our target audience includes both professional users, who may want to perform analyses on these data, and casual users who may have limited knowledge in biology and chemistry. Through easy access to this information, we hope to increase awareness about the importance of water quality and empower individuals to take action in protecting and preserving our water resources.

Design

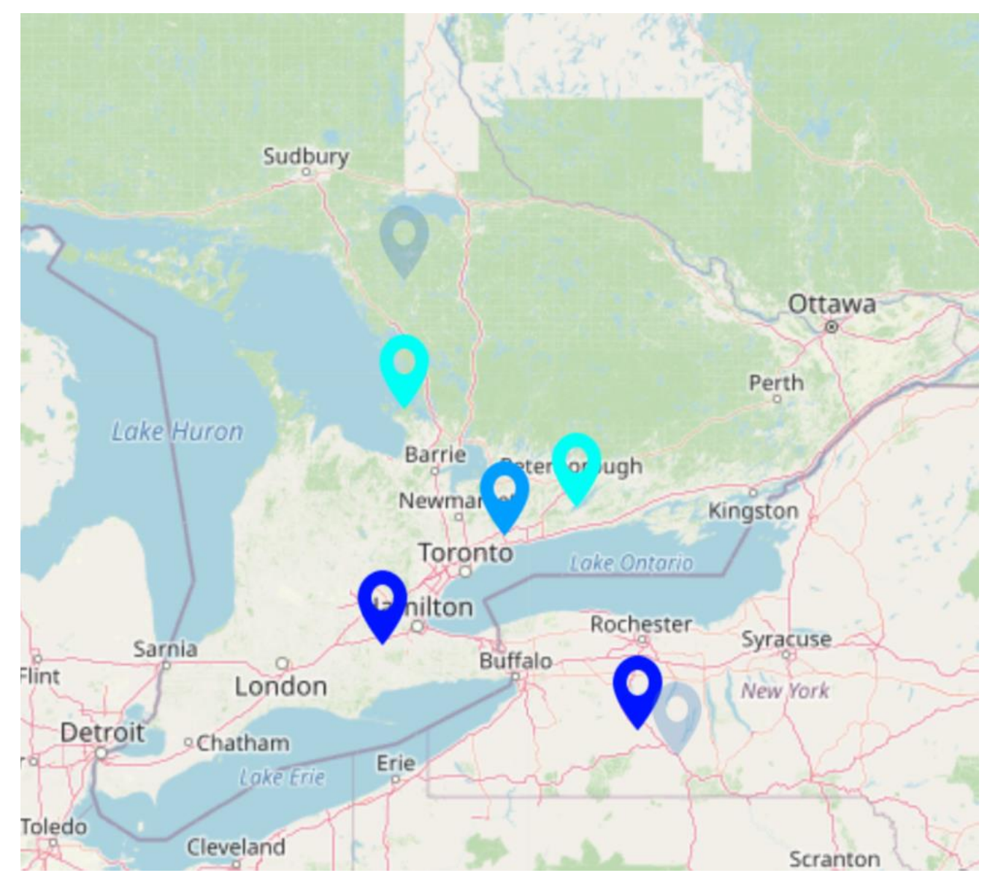
The website has a simple yet elegant design, with a title bar that displays the website's name and program logo. The sidebar contains four buttons that correspond to four different pages, Map, Real-time, About, and Settings, allowing users to switch between them with ease. During application, we will be focusing specifically on the map page.
The main section of the map page contains a map and a data analysis area. The map is fully functional using *OpenStreetMap* provided by *Leaflet*, with markers at all sensor locations, two zoom in and zoom out buttons, and a scale in the corner. The data analysis menu contains several drop-down lists and buttons that allow users to manually select nodes, probes, and time periods.

The name of our project, *Waterra*, combines the morphemes of both the words ‘water’ and ‘terra’, which can be interpreted as the water of the earth.

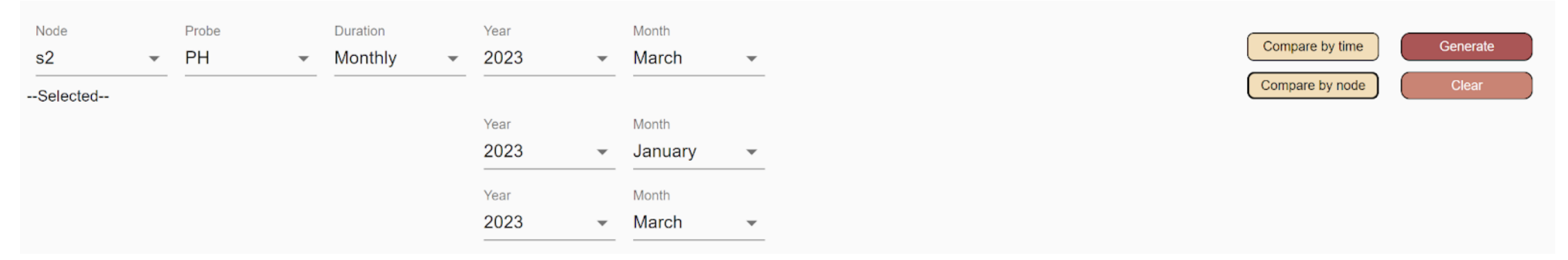


Functions

- ➢ There are three main functions that we implemented.
- ➢ The first one is color code markers. After selecting a probe, the color of each marker on the map will change according to the latest updated data of the node. Here we use three different shades of blue to differentiate data levels. For example, a dark blue marker means that the latest data of the node is at a 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.

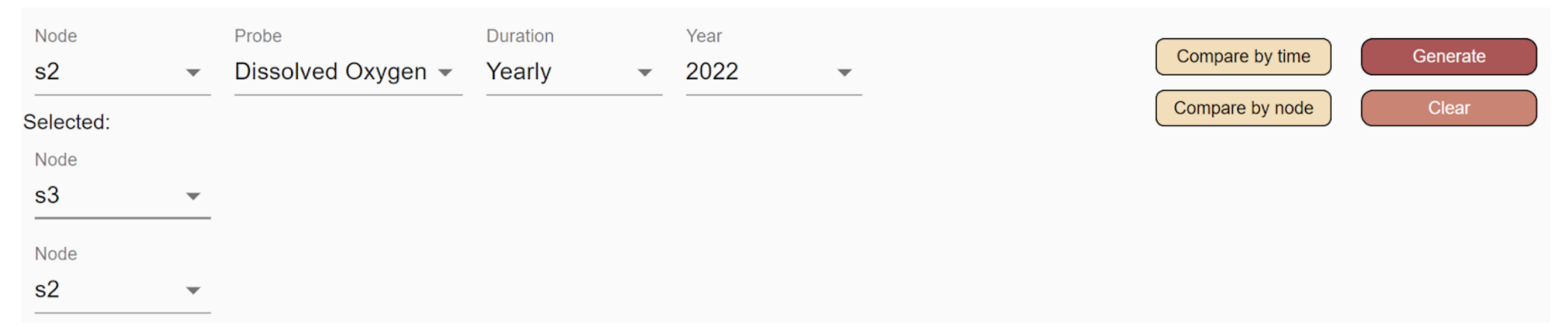


- ➢ The second one is to generate overlapping graphs for comparison of data trends of different time periods. Users can select a specific time period in yearly, monthly, or weekly duration, and then generate a line chart of all selections. The website can also generate overlapping graphs for comparison. Users can select two different time intervals for the same node and probe, however, note that all time intervals must be of the same size (i.e., Both yearly/monthly/weekly), and the node and probe selected must also be the same.



Node: s2 · Probe: PH · Duration: Monthly · Year: 2023 · Month: March
--Selected--
Year: 2023 · Month: January
Year: 2023 · Month: March

- ➢ The third function is similar to the second one, which is to generate overlapping graphs for comparison of data trends of different nodes on the same probe, same time period. The probe and time selected must also remain unchanged.



Node: s2 · Probe: Dissolved Oxygen · Duration: Yearly · Year: 2022
Selected:
Node: s3
Node: s2

Implementation

Architecture

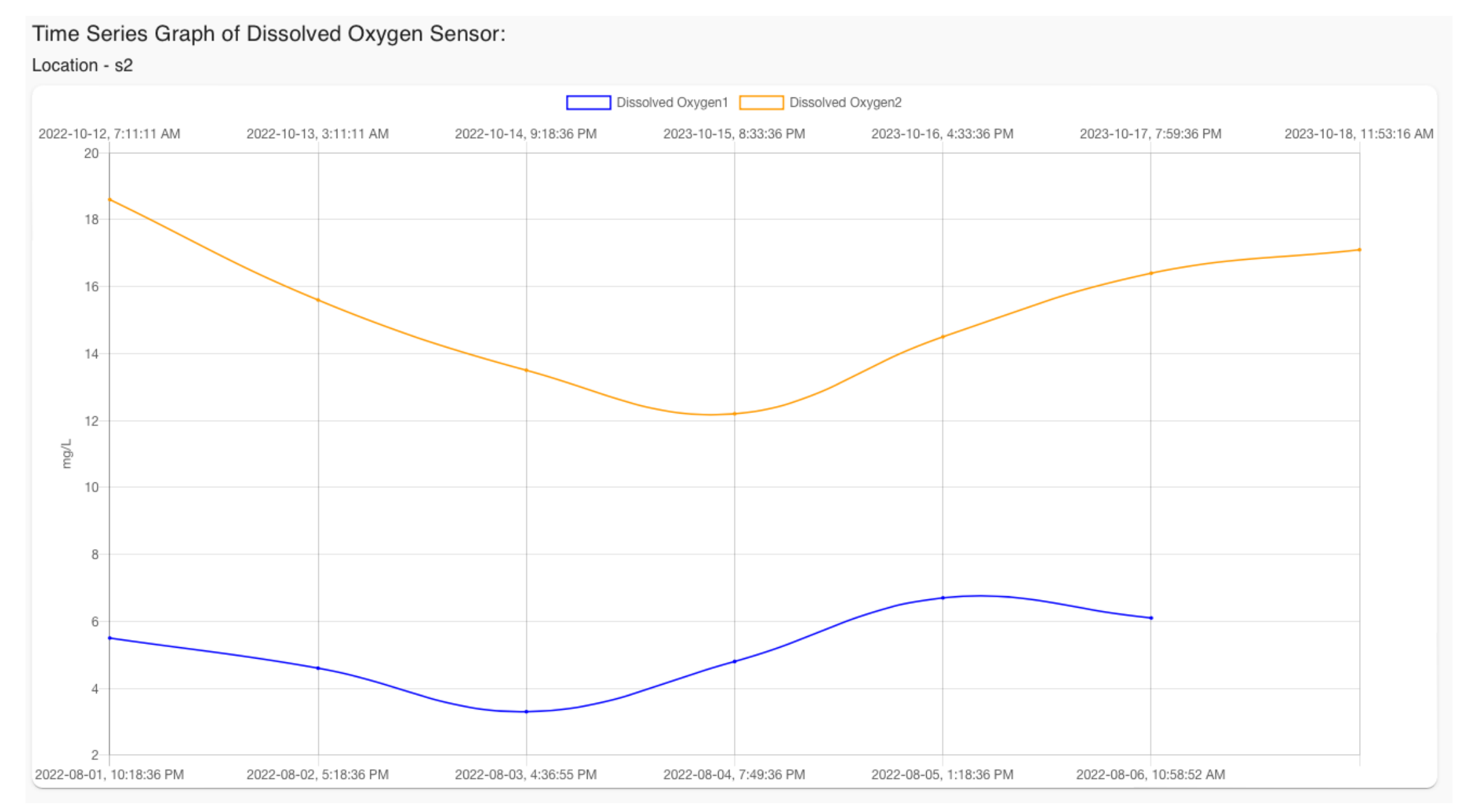
- ➢ The entire project is composed of three essential components, the frontend, the backend, and the database.
- ➢ The frontend uses *React* to build user interfaces in *TypeScript*. It is responsible for presenting all visual components, lke HTML elements and allowing the user to interact with the application and receive real-time updates.
- ➢ The backend is designed to manage the data and serve it to the frontend. It implements APIs that interact with the database using *GO* language.
- ➢ The database is the central repository where all the data sent by deposited sensors is received, stored, and updated. It consists of two tables in *InfluxDB*, where the sensor information is recorded by time, id, name/sensor type, and value.
- ➢ When users act at the front end, the website generates a URL accordingly and fetches necessary data from the database using APIs for the functionality that drives the behavior of the application. The database is secured and maintained by the backend, ensuring the data is safe and consistent across the entire system.

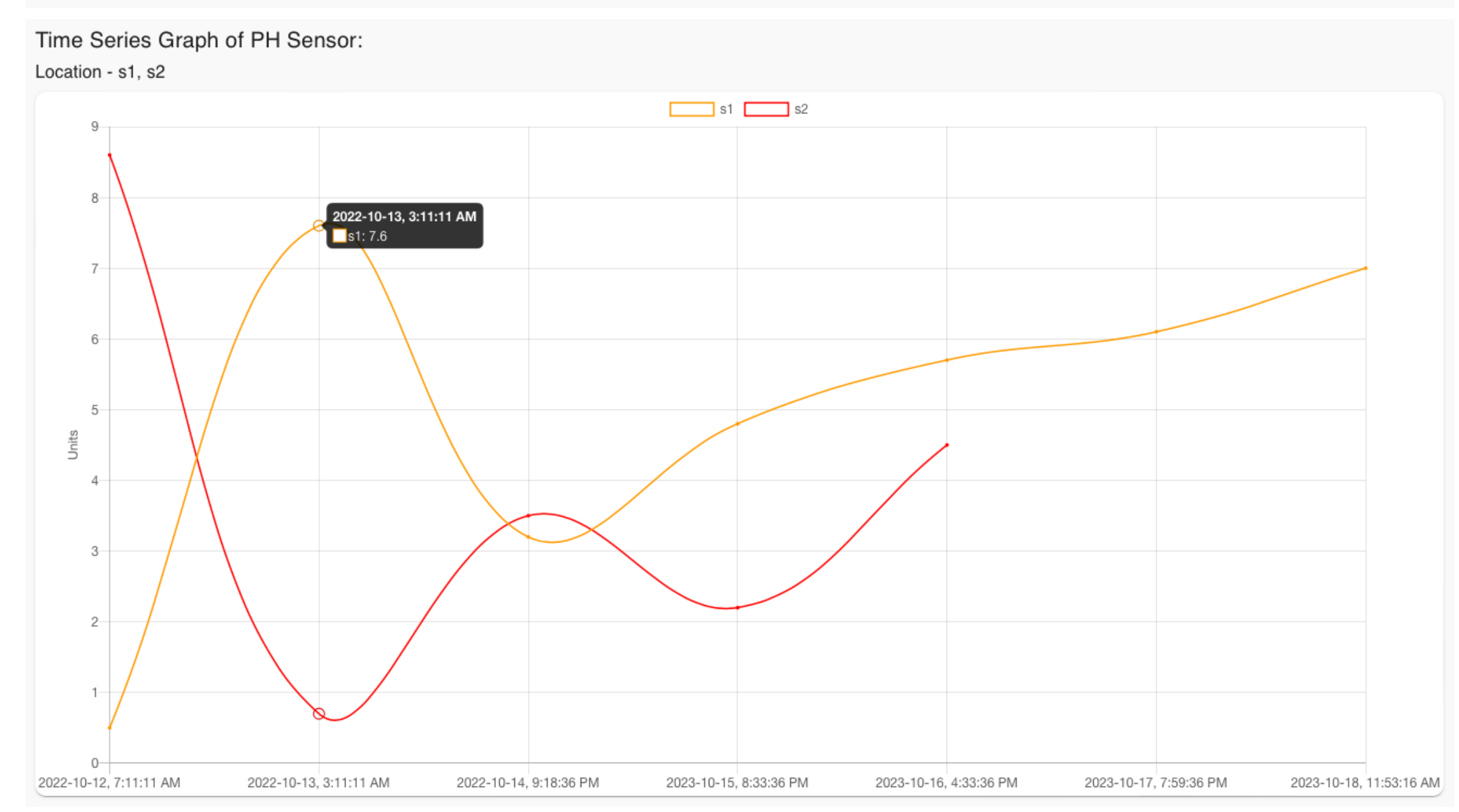
Database

- ➢ An *InfluxDB* time-series database is used to save both the node and the probe data. The database contains two tables, the *sensorInfo* table, and the *sensorMeasurements* table.
- ➢ The *sensorInfo* table has four columns: time, id, name, and value. The value represents the coordinates of the node at the time. The id, name, and value are all stored as strings.
- ➢ The *sensorMeasurements* table also has four columns: time, id, sensortype, and value. The value is the measurement of the probe from the node at the time. The id, sensortype, and value are stored as strings. Time in both tables are using the Unix timestamp method.
- ➢ The database is initialized and set up using a helper tool of *InfluxDB* and *Postman*. All fetching URLs from the frontend are converted into SQL commands in backend APIs. Different actions from users form different SQL commands to manipulate and query the database.

Code Structure

- ➢ All backend code is integrated into one file named *main.go*. It contains several functions that implement the APIs, constructors of six types that are used in API functions, and a main function to boot up everything.
- ➢ All frontend code is in the directory */frontend/src/components*. There are nine *TypeScript* files in total, and each of them corresponds to one component in the user interface.
- **Map.tsx**: the functional map in the Map section. It is also in charge of handling State variable changes and fetching and parsing some sensor data.
- **MapDetail.tsx**: all elements on the map, such as markers and marker pop-ups. Markers of different colors and pop-ups containing different information are also generated here.
- **Menu.tsx**: the menu bar and buttons on the right. Handlers of buttons and Props variable changes are included.
- **CustomLayout.tsx**: define a customized button layout for buttons in the menu.
- **SensorTable.tsx**: fetch and prepare sensor data for charts and data tables. Display chart titles with selected nodes and probes.
- **Chart.tsx**: draw line charts using data from *SensorTable.tsx*.
- **DataCard.tsx**: display all data from *SensorTable.tsx* in table format.
- **Info.tsx**: display introduction, news, and social media posts of the project.
- **Settings.tsx**: website language selection and sensor configuration toolkit setup.





Testing

- ➢ 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 the tests will use fake data we upload to the *InfluxDB* database through *Postman*. Tests will be mainly implemented as *requirement-based testing.*
- ➢ 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.

Conclusion

The test results demonstrate that the Water Sensor website data visualization features can provide reliable and accurate data analysis tools to users. The testing process identified and resolved potential issues, improving the system's reliability and performance.

Yingxue (Cynthia) Liu (liuy363@mcmaster.ca)
Jian (Jennie) Li (lij416@mcmaster.ca)
Huaijin (Zoe) Ning (ningh4@mcmaster.ca)

BRIGHTER WORLD