

**FOMUM PRISTEN**

**00093053**

## PROJECT PORTFOLIO

### First Features

Using chart.js to display the temperature on a line chart on the dashboard

### Analysis

Use the lab manager when I open the dashboard I should be able to see the temperature of a room when I select a day or a room

#### **Happy path**

Given: The lab manager is on the home page

When: he selected a room number or day

Then: the temperature chart should be displayed on the dashboard, showing the temperature readings for the selected room or day.

#### **Unhappy Path:**

Given: that the lab manager is on the home page,

When: they select a room or day from the dropdown menu,

And: no temperature data is available for the selected room or day,

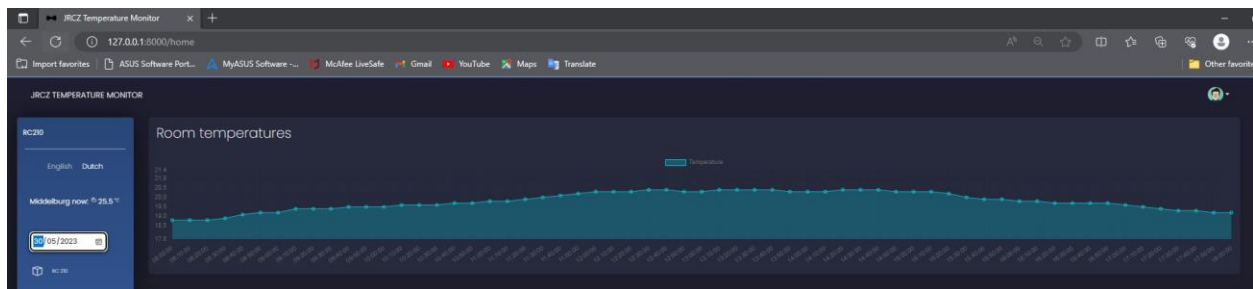
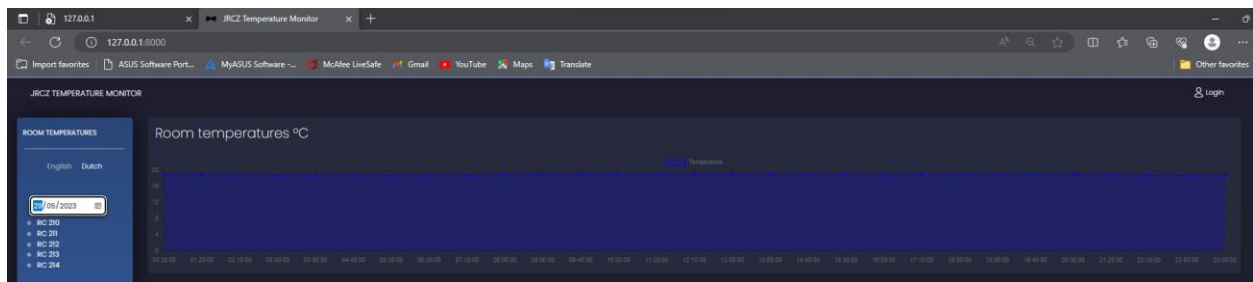
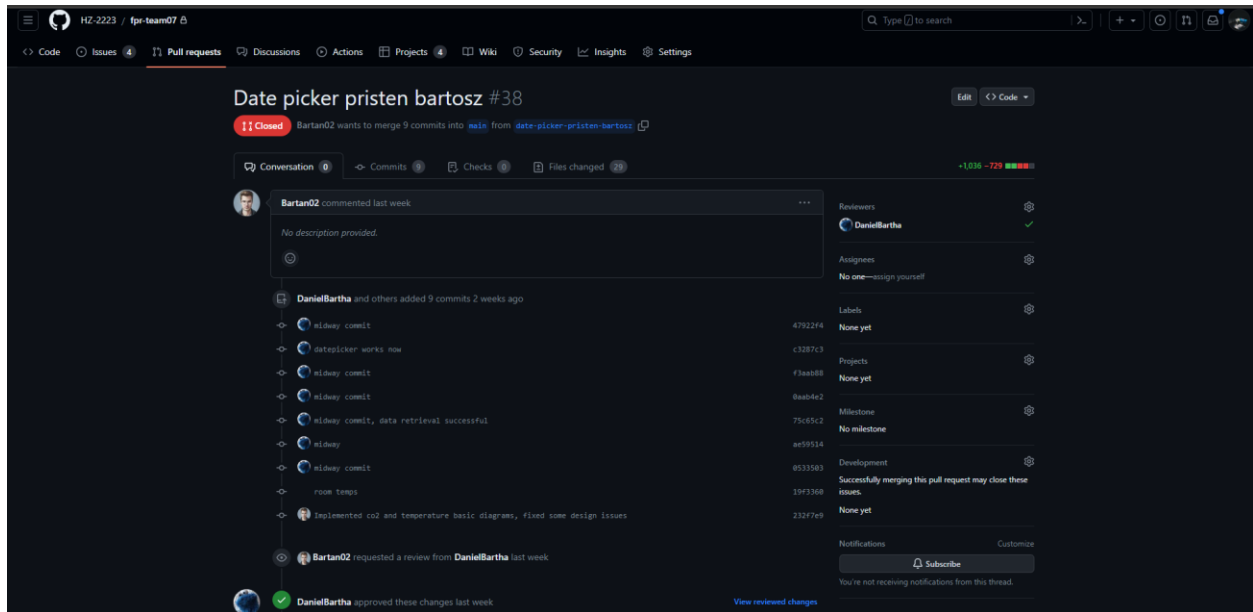
Then: lab manager will be unhappy since no information will be shown on the line chart

## Design

The feature goes beyond a basic CRUD model typically used for data manipulation. It incorporates the following elements:

- **Asynchronous Data Retrieval:** The code utilizes asynchronous operations to fetch temperature data from a server. This ensures smooth interaction with the server without blocking the user interface.
- **Chart Visualization:** The feature employs a charting library (such as Chart.js) to create a line chart that visualizes the temperature data. The chart provides a clear representation of temperature trends over time.
- **Date Selection:** Users can select a specific date using an input field. The feature captures the selected date and dynamically updates the chart to display the temperature readings for that day.
- **Data Filtering:** The code filters the temperature data based on the selected date, extracting the relevant readings, and updating the chart accordingly. This allows users to focus on the temperature data for a specific day.
- **Error Handling:** The code includes error handling to catch and log any potential errors that may occur during the data retrieval or chart rendering process. This ensures a smooth user experience and provides feedback in case of issues.

# Implementation



Using the piece of code below as an example. My code is internally consistent with my project in the sense that has

## Meaningful Naming Conventions:

The code demonstrates consistent and meaningful naming conventions for variables and functions.

Descriptive names have been used throughout the code, enhancing readability and understanding.

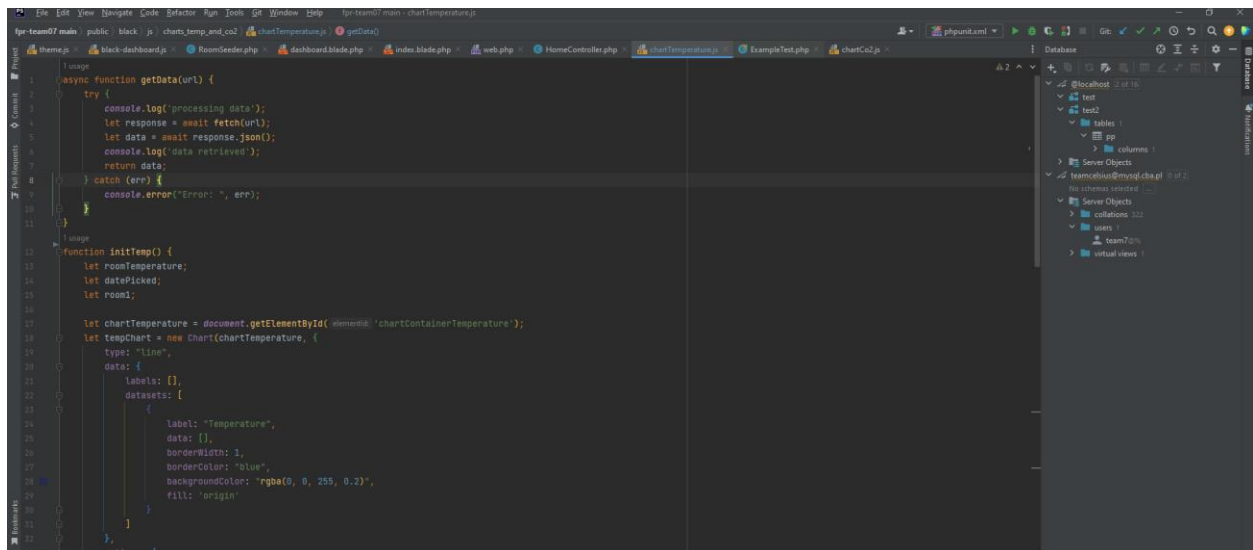
Well-chosen names accurately represent the purpose and functionality of the variables and functions.

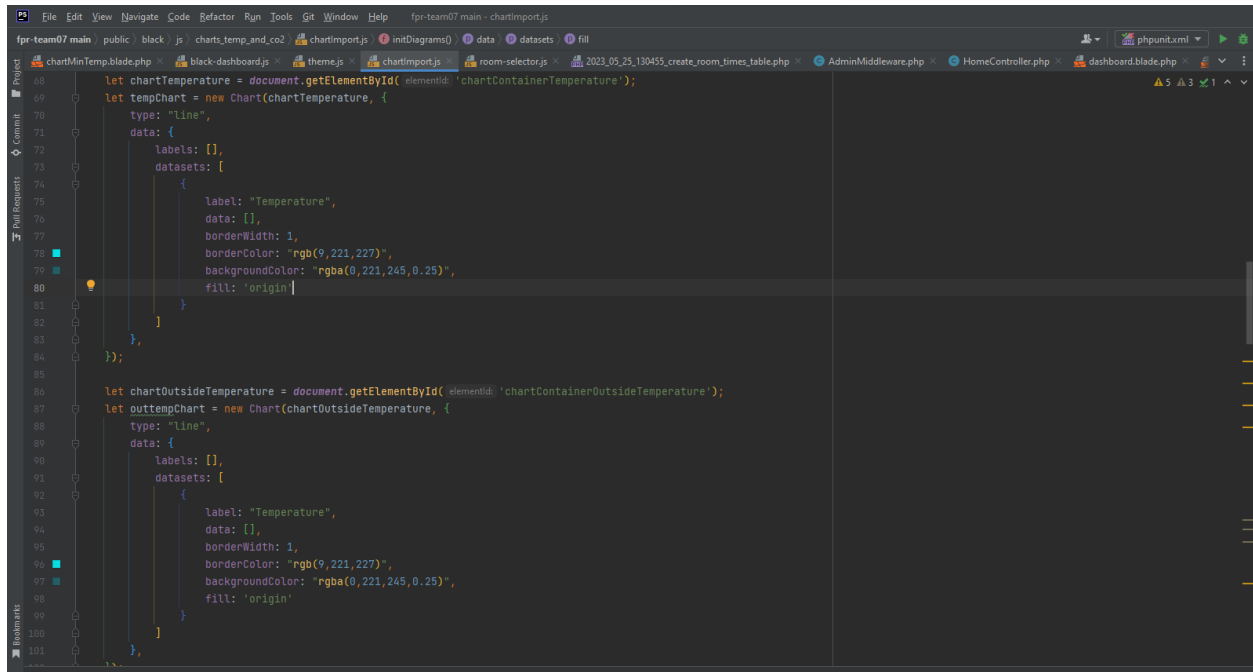
## Logical Code Structure:

The code exhibits a logical structure, making it easier to follow and comprehend.

Related functions or code blocks have been grouped together, promoting organization and clarity.

Adequate indentation and formatting have been applied, enhancing code readability and visual alignment.





```
68 let chartTemperature = document.getElementById('chartContainerTemperature');
69 let tempChart = new Chart(chartTemperature, {
70   type: "line",
71   data: {
72     labels: [],
73     datasets: [
74       {
75         label: "Temperature",
76         data: [],
77         borderWidth: 1,
78         borderColor: "rgb(9,221,227)",
79         backgroundColor: "rgba(0,221,245,0.25)",
80         fill: 'origin'
81       }
82     ],
83   },
84 });
85
86 let chartOutsideTemperature = document.getElementById('chartContainerOutsideTemperature');
87 let outtempChart = new Chart(chartOutsideTemperature, {
88   type: "line",
89   data: {
90     labels: [],
91     datasets: [
92       {
93         label: "Temperature",
94         data: [],
95         borderWidth: 1,
96         borderColor: "rgb(9,221,227)",
97         backgroundColor: "rgba(0,221,245,0.25)",
98         fill: 'origin'
99       }
100     ],
101   },
102 });
```

## Testing:

### Test Plan 1: Happy Path - Viewing Temperature on the Dashboard

#### Objective:

Ensure the temperature chart is displayed correctly on the dashboard when a room or date is selected.

#### Test Steps:

- Open the dashboard.
- Select a room or date from the dropdown menu.
- Observe the chart displayed on the top-right section of the dashboard.
- Verify the chart accurately represents the temperature readings for the selected room or date.

- Ensure proper labeling of the chart with axis titles, data labels, and a legend.
- Confirm the chart's visual appeal and ease of interpretation.

**Expected Results:**

- The chart is displayed correctly on the dashboard.
- The chart accurately represents the temperature readings for the selected room or date.
- The chart is properly labeled and visually appealing.

**Test Plan 2: Unhappy Path - No Temperature Chart Displayed****Objective:**

Verify appropriate feedback or error messaging when no temperature chart is displayed on the dashboard.

**Test Steps:**

- Open the dashboard.
- Select a room or date from the dropdown menu.
- Verify no chart is displayed in the top-right section of the dashboard.
- Ensure a user-friendly message or indication notifies the user about the absence of a temperature chart.
- Validate the message clearly conveys the absence of temperature data for the selected room or date.
- Check that the message does not disrupt the user experience or impede other dashboard features.

**Expected Results:**

- No temperature chart is displayed on the dashboard.
- A user-friendly message indicates the absence of temperature data for the selected room or date.
- The message is clear, unobtrusive, and does not hinder other dashboard features.

Executing these test plans will ensure thorough testing of the temperature functionality on the dashboard, covering both happy and unhappy path scenarios. It will verify accurate display of temperature data and handle cases where no data is available or no chart is displayed appropriately.

## Conclusion

In conclusion, the temperature display feature on the dashboard provides valuable benefits to lab managers for monitoring energy usage. This feature effectively addresses their needs by offering a dynamic temperature chart based on the selected room or day. Thorough testing and feedback from team members have ensured its functionality and usability. In the happy path scenario, the system accurately displays the temperature chart, enabling lab managers to analyze energy usage. In the unhappy path scenario, appropriate feedback or error messaging is provided when no temperature data is available. This prevents confusion or frustration. By validating the feature through acceptance tests and incorporating feedback, we have ensured its reliability and usefulness. The implementation of this feature enhances the user experience and supports data-driven decision making.

## Second feature:

### Feature

Using chart.js to display the CO2 level on a line chart on the dashboard

### Analysis

#### **Happy path**

As a lab manager, I want to be able to visualize and track the CO2 levels on the dashboard using a line chart.

**Given:** The lab manager is on the home page

**When:** he selected a room number or day

**Then:** the CO2 level chart should be displayed on the dashboard, showing the CO2 level readings for the selected room or day.

#### **Unhappy Path:**

**Given:** that the lab manager is on the home page,

**When:** they select a room or day from the dropdown menu,

**And:** no CO2 data is available for the selected room or day,

**Then:** the lab manager wont be able to see the chart data for the CO2 levels



## Design

The complexity of the CO2 chart code lies in its functionality beyond simple CRUD operations (Create, Read, Update, Delete). Here's an explanation of the code and its complexity:

**Chart Configuration:** The code initializes and configures the CO2 chart with various options such as type, data, labels, datasets, and styling settings. It defines two datasets: one for CO2 levels and another for an occupancy threshold.

**Event Listeners:** The code sets up event listeners for different room selections and a date picker. These listeners trigger data loading and chart update functions when the user interacts with the UI elements.

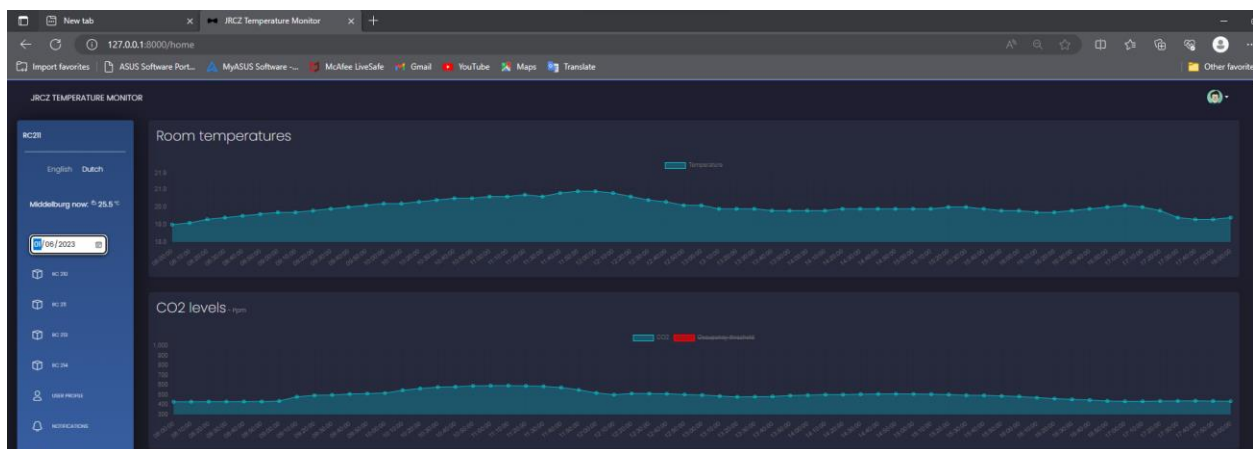
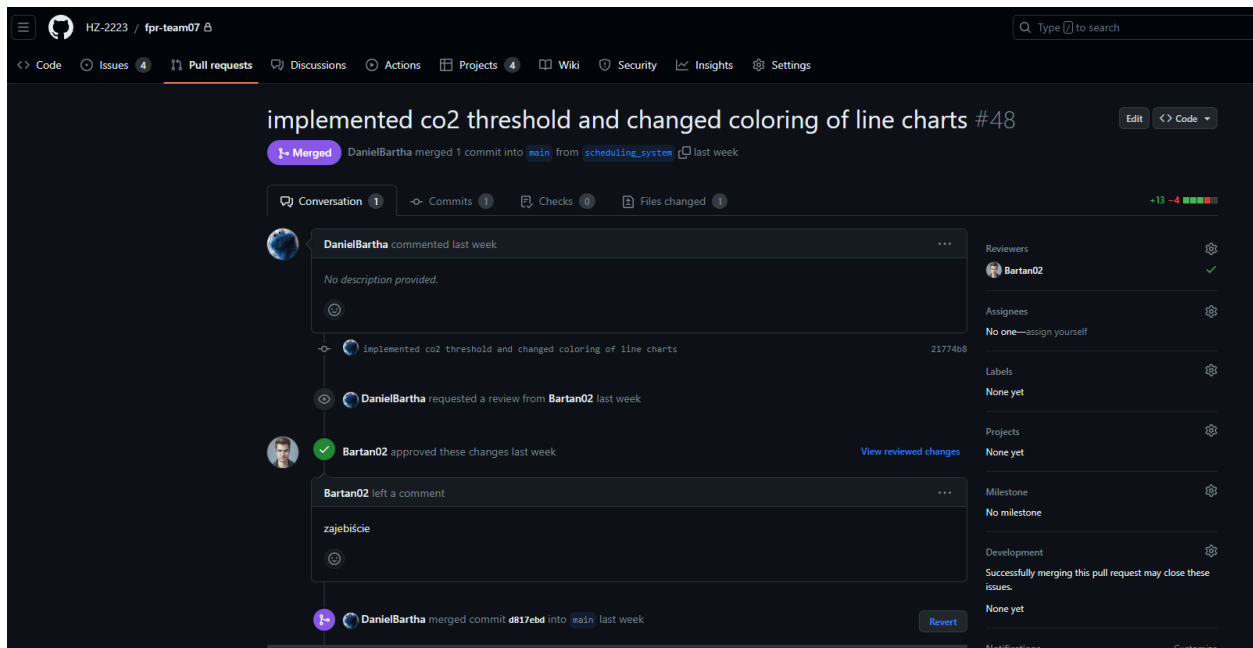
**Asynchronous Data Loading:** The `loadData` function is an asynchronous function that fetches CO2 and temperature data from a specific endpoint using the `getData` function. It awaits the response and then assigns the data to variables for further processing.

**Data Filtering and Transformation:** The `updateCharts` function filters and transforms the loaded data based on the selected date and specific conditions. It extracts CO2 and temperature data points that meet certain criteria (e.g., every 10 minutes and during working hours) and prepares them for chart visualization.

**Dynamic Chart Updates:** The `updateCharts` function updates the CO2 chart's data and labels based on the filtered data points. It also adjusts the chart's y-axis scale dynamically based on the temperature data range. Additionally, it updates the visibility and content of a "no data" message based on the availability of data for the selected date.

The complexity in this code arises from the combination of asynchronous data loading, data filtering and transformation, dynamic chart updates, and event-driven user interactions. The code handles data retrieval, manipulation, and chart rendering based on user selections and real-time data availability.

## Implementation:



Using the piece of code below as an example. My code is internally consistent with my project in the sense that:

- The code utilizes meaningful naming conventions for variables and functions, enhancing clarity and understanding.
- Descriptive and intuitive names are used throughout the code, accurately representing the purpose and functionality of the elements.
- The code follows a logical structure, making it easier to follow and comprehend.
- Related functions or code blocks are grouped together, promoting organization and clarity.
- Adequate indentation and formatting are applied, improving code readability and visual alignment.

```
function initDiagrams() {
  roomPicked = 0;
  roomNumberDisplay.innerHTML = 'Loading RC210...';
  let chartC02 = document.getElementById( 'elementId: 'chartContainerCo2');
  let co2Chart = new Chart(chartC02, {
    type: "line",
    data: {
      labels: [],
      datasets: [
        {
          label: "CO2",
          data: [],
          borderWidth: 1,
          borderColor: "rgb(9,221,227)",
          backgroundColor: "rgba(9,221,245,0.25)",
          fill: 'origin'
        },
      ],
    },
  });
}
```

```
3+ usages
function updateCharts() {
  datePicked = document.getElementById( 'elementId: 'start').value;
  let co2Data = [];
  let co2Labels = [];
  let temperatureData = [];
  let temperatureLabels = [];
  for (let i = 0; i < room1.length; i++) {
    console.log(room1[i]['dateAndTime']['date']);
    const roomDate = room1[i]['dateAndTime']['date'];
    if (roomDate === datePicked) {
      if (getMinuteFromTime(room1[i]['dateAndTime']['time']) % 10 === 0 && checkWorkingTime(room1[i]['dateAndTime']['time'])) { // Display data every 10 minutes
        co2Data.push(room1[i]['co2']);
        co2Labels.push(room1[i]['dateAndTime']['time']);
        temperatureData.push(room1[i]['temperature']);
        temperatureLabels.push(room1[i]['dateAndTime']['time']);
      }
    }
  }
}
```

## Testing

### **Test Plan: Happy Path - Displaying CO2 Levels**

#### **Objective:**

Ensure that the CO2 levels are displayed accurately on the dashboard for the selected date or time period.

#### **Test Steps:**

- Open the dashboard.
- Locate the date picker or input field for selecting the date or time period.
- Select a specific date or time period for analysis.
- Observe the CO2 levels displayed on the dashboard.
- Ensure that the visualization of the CO2 levels is clear

#### **Expected Results:**

- The CO2 levels are displayed accurately on the dashboard for the selected date or time period.
- The displayed CO2 levels correspond to the selected date or time period.

### **Test Plan: Unhappy Path - No CO2 Data Available**

#### **Objective:**

Verify that appropriate feedback is provided when no CO2 data is available for the selected date or time period.

**Test Steps:**

Open the dashboard.

- Locate the date picker or input field for selecting the date or time period.
- Select a specific date or time period for which no CO2 data is available.
- Observe the dashboard to determine the feedback provided.
- Ensure that a user-friendly message or indication is displayed, indicating the absence of CO2 data.
- Validate that the message clearly conveys the unavailability of CO2 data for the selected date or time period.
- Confirm that the absence of CO2 data and the corresponding message do not hinder the functionality or user experience of other dashboard features.

**Expected Results:**

- No CO2 levels are displayed on the dashboard for the selected date or time period.
- A user-friendly message indicates the absence of CO2 data.
- The message clearly conveys the unavailability of CO2 data for the selected date or time period.
- The absence of CO2 data and the corresponding message do not hinder the functionality or user experience of other dashboard features.
- By executing these test plans, we can ensure that the CO2 level display feature on the dashboard is thoroughly tested under both happy and unhappy path scenarios. These tests will validate the accurate display of CO2 levels for the selected date or time period and appropriate handling when no data is available.

## Conclusion

In conclusion, the implementation of the CO2 level feature on the dashboard serves as a valuable tool for lab managers to assess occupancy in a room based on CO2 levels. By monitoring and displaying real-time CO2 readings, this feature enables lab managers to determine if someone is present in the room, providing insights for effective space management and safety protocols.

Through thorough validation and testing, we have ensured the accuracy and reliability of the CO2 level feature. In the happy path scenario, the dashboard displays real-time CO2 readings, allowing lab managers to easily identify occupancy status. In the unhappy path scenario, where no CO2 data is available or the levels indicate no occupancy, the dashboard appropriately conveys this information, preventing false assumptions.

Overall, the CO2 level feature enhances the functionality and efficiency of room occupancy monitoring, empowering lab managers to make informed decisions regarding resource allocation and safety measures.