Web Application Developer Guide

Report Management for Supervisors

This guide is meant to serve as a reference and provide helpful information related to the development process for BIMS web application Report Management for supervisors, including the Yearly Inspection Report Tool and the Longitudinal Analysis Tool

NOTE: The BIMS web application is an ongoing project still in development.

Contribution of the Fall 2021 Capstone Team

The Fall 2021 Capstone team contribution to the BIMS web application includes database integration, creation of distinct user roles(admin, inspector, and supervisor), implementation of the login process for each role, and implementation of Report Management features for the supervisor role, including the Yearly Inspection Report Tool and the Longitudinal Analysis Tool.

Report Management Features

The Report Management features in this web application allow supervisors to view compiled bridge inspection report data. There are two main Report Management features: the ***Yearly Inspection Report Tool*** and the ***Longitudinal Analysis Tool***.

Yearly Inspection Report Tool

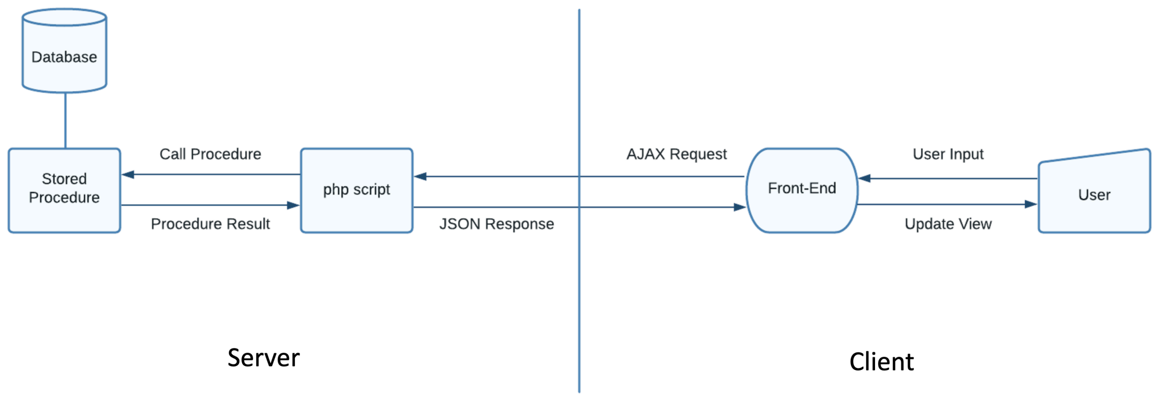
This tool provides a summary of all bridge inspection data for the selected year with visual pie chart representations of inspection ratings and statuses.

Longitudinal Analysis Tool

This tool provides a breakdown of bridge inspection data over a given timeframe of up to 10 years, with visual line chart representations of inspection ratings and the ability to compare up to three bridges simultaneously.

Data Flow

In general, the flow of data within the web application can be seen in the diagram below:



Longitudinal Analysis Parameters
supervisor-search-params-longitudinal-analysis.php

Handling the Addition of Up To Three Bridges

It is worth explaining here how the addition of bridge search elements is handled for the bridge selection part of the page.

Upon visiting the page for the first time (in a session), there is a single bridge search HTML element. This element is referred to as "Bridge 1" throughout the code and comments. The search functionality and onclick functions are already defined for Bridge 1 upon first visit. This is in contrast to additional bridges that the user chooses to add, each of which must be dynamically created, including all HTML and associated JavaScript for search functionlaity and onclick functions for that particular bridge element.

To accomplish this, the *buildBridgeElement* function is called when the user clicks the "Add Another Bridge" Button. The *buildBridgeElement* function takes care of dynamically creating the HTML for the new bridge element, as well as defining the JavaScript onclick functions and search functionality for the new bridge element. Therefore, each bridge search element will have its own function definitions for the search functionality itself, "Confirm Selection" button onclick, and remove onclick.

Longitudinal Analysis - Line Chart

Dynamically Building the Line Chart

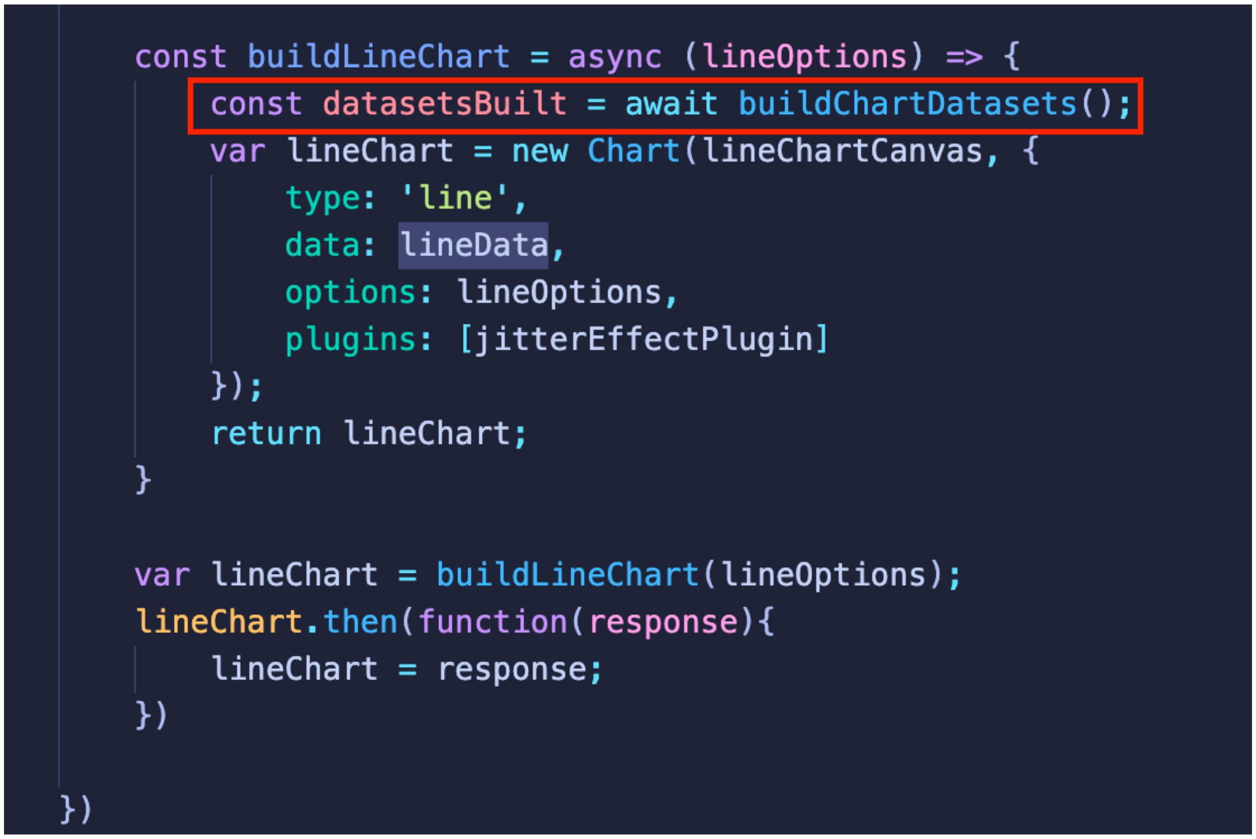
Dynamically building the line chart that is generated in the Longitudinal Analysis Report requires that all chart datasets (including inspection ratings, label/name, point color, etc.) also be dynamically generated using data from the database. To accomplish this, three functions are defined:

- **fetchAllChartInspections**: fetches all chart inspections for each of the selected bridges
- **buildChartDataSets**: parses inspection data returned by *fetchAllChartInspections* and creates the chart dataset for each selected bridge, according to the chart.js requirements, and pushes it to an array inside a JavaScript object, *lineData.datasets*

- **buildLineChart**: Creates the new Line Chart using *lineData*

Because the execution of *buildLineChart* is dependent on the execution of *buildChartDatasets*, and *buildChartDatasets* is dependent on the the execution of *fetchAllChartInspections*, it must be ensured that these functions execute in sequence. To accomplish this, nesting of *async/ await* function calls is used.

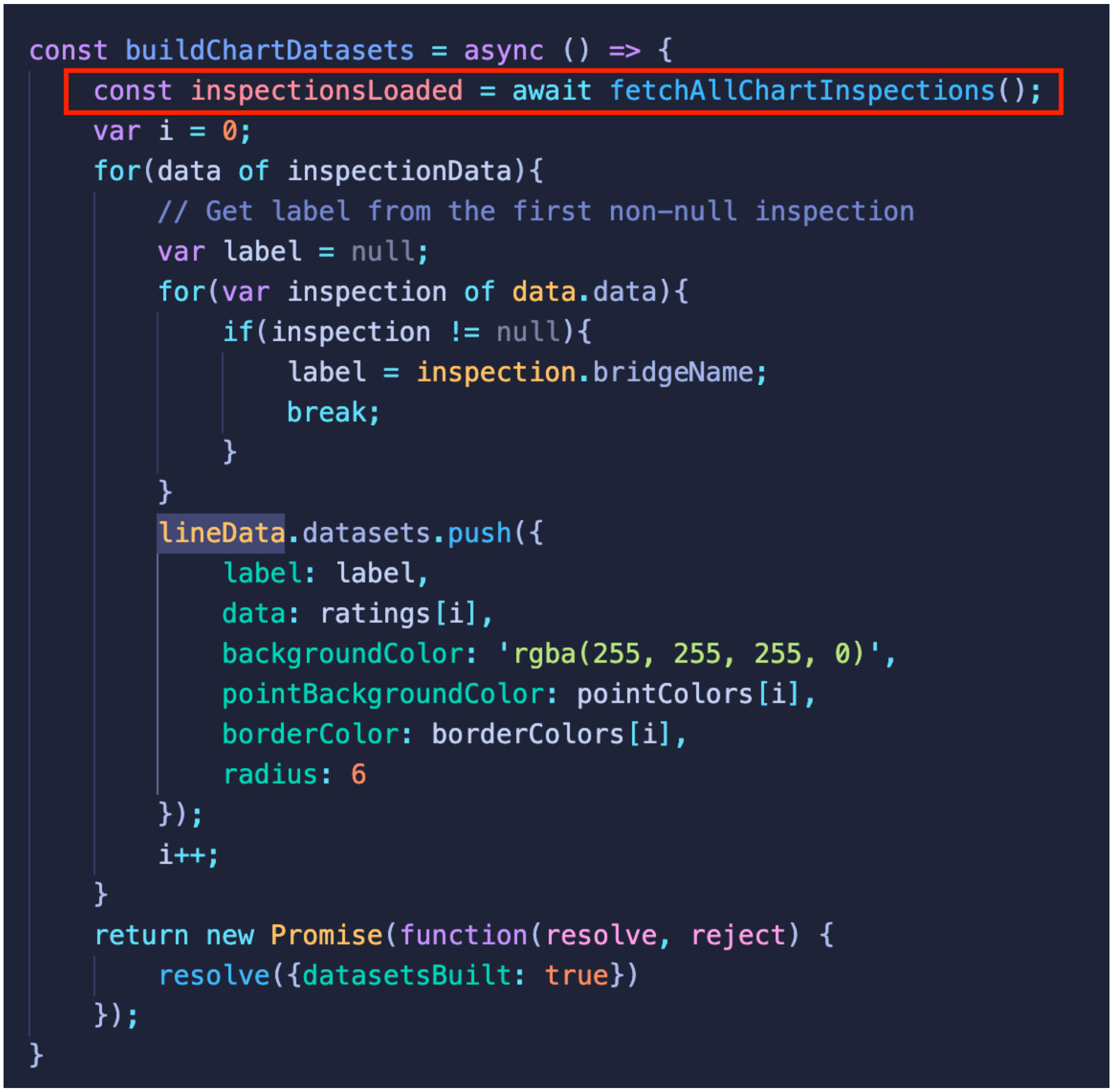
Specifically, only *buildLineChart* needs to be explicitly called. From there, *buildLineChart* calls and awaits the execution of *buildChartDatasets*. In turn, *buildChartDatasets* calls and awaits the execution of *fetchAllChartInspections*. After *fetchAllChartInspections* finishes executing, then *buildChartDatasets* can continue on and finish executing, after which *buildLineChart* will finally continue on and finish executing. The end result is a dynamically generated line chart representation of bridge inspection ratings from the database.



```
    const buildLineChart = async (lineOptions) => {
        const datasetsBuilt = await buildChartDatasets();
        var lineChart = new Chart(lineChartCanvas, {
            type: 'line',
            data: lineData,
            options: lineOptions,
            plugins: [jitterEffectPlugin]
        });
        return lineChart;
    }

    var lineChart = buildLineChart(lineOptions);
    lineChart.then(function(response){
        lineChart = response;
    })

})
```



```
const buildChartDatasets = async () => {
    const inspectionsLoaded = await fetchAllChartInspections();
    var i = 0;
    for(data of inspectionData){
        // Get label from the first non-null inspection
        var label = null;
        for(var inspection of data.data){
            if(inspection != null){
                label = inspection.bridgeName;
                break;
            }
        }
        lineData.datasets.push({
            label: label,
            data: ratings[i],
            backgroundColor: 'rgba(255, 255, 255, 0)',
            pointBackgroundColor: pointColors[i],
            borderColor: borderColors[i],
            radius: 6
        });
        i++;
    }
    return new Promise(function(resolve, reject) {
        resolve({datasetsBuilt: true})
    });
}
```

A Note About the Use of PHP Session Variables in Report Management Features

For the YIRT and LAT, php session variables are used to store current user input parameters for generating reports. These session variables are used to maintain the state of report screens throughout the session, and are updated as the user makes new selections or enters new report parameters. For example, selecting a new year in the YIRT or selecting new bridges or a new timeframe in the LAT.

Longitudinal Analysis Tool Session Variables

- **selectedBridgeNames***
- **selectedBridgeNumbers***
- **selectedBridgeCounties***
- **yearBegin**
- **yearEnd**

***Bridge names, numbers, and counties map to one another by index. This means that the bridge name at selectedBridgeNames[i] corresponds to the bridge number at selectedBridgeNumbers[i] and the county at selectedBridgeCounties[i].**

Yearly Inspection Report Tool Session Variable

- **YIR_SelectedYear**

Third-Party Libraries and Plugins

Chart.js Library (v2.9.4)

Both the Yearly Inspection Report Tool (YIRT) and the Longitudinal Analysis Tool (LAT) make use of the Chart.js library to render interactive charts for displaying bridge inspection report data. The YIRT makes use of the Chart.js Pie Chart and the LAT makes use of the Chart.js Line Chart.

Helpul Links:

- Chart.js: https://www.chartjs.org/
- Line Chart Documentation: https://www.chartjs.org/docs/2.9.4/charts/line.html
- Pie Chart Documentation: https://www.chartjs.org/docs/2.9.4/charts/doughnut.html

DataTables Plugin (v1.10.25)

Both the Yearly Inspection Report Tool (YIRT) and the Longitudinal Analysis Tool (LAT) make use of the DataTables plugin for jQuery to render interactive data tables for displaying bridge inspection report records.

Helpful Links:

- DataTables Documentation: https://datatables.net/manual/index
- Initializing DataTables in the LAT (see function *loadTable* in *supervisor/LA-functions.js*): https://datatables.net/reference/option/data
- To rebuild or modify DataTables with the latest versions: https://datatables.net/download/#dt/dt-1.10.25

Other libraries used throughout the project

***Some of these libraries were already in use by previous capstone teams in other areas of the BIMS project**

jQuery(v3.3.1): https://jquery.com/

OwlCarousel (v2.3.4) (jQuery plugin): https://owlcarousel2.github.io/OwlCarousel2/

Bootstrap (v4.5.3): https://getbootstrap.com/

AdminLTE (v3.1.0) (built on top of Bootstrap): https://adminlte.io/

- **AdminLTE CardWidget:** https://adminlte.io/docs/3.1/javascript/card-widget.html