Frontend Documentation

Index.js:

Main file that renders all the components of the dashboard.
Uses axios for get requests.
React Bootstrap tabs was imported for the tab functionality that calls each component used.
This file is used as the highest hierarchical component that calls every other component within it. In this file, we do all the processing needed for Homepage and Formpage. Additionally It calls all the tab components such as HomePage/SensorPage/FormPage/About/Documentation.

TabSection Class:

- CompontentDidMount()
    - Sets isLoadingForm,isLoadingFarm,formsub,farmsInfo states.
    - isLoading states are for the loading screen component
    - Formsub uses axios get request of all formSubmissions
    - farmsInfo uses axios get request to get all farms
- percent(Soil/Biomass/Bag/Yield)
    - Given an array and iterates through the array to find the amount of nulls total.
    - Takes ratio of Nulls vs total possible values (Expected # barcode * # of col)
    - The resulting percentage is 1 - Ratio of nulls. Giving us a ratio of non nulls.
    - Null values were checking for can be found here https://docs.google.com/document/d/1sYBG9xoswHbxLFAQpfREsxxS8oN-RGzsH5pu2iKiu-A/edit?usp=sharing
- render()
    - Two main variables temp tempPercent.
    - Temp holds `{"state":"","year":"","soil":[],"biomass":[],"yield":[],"bag":[]}'` with the farmcode as its key. All formsubmissions related toward the farmcode will be appended to its respective group (soil/biomass/yield/bag).
    - tempPercent holds `'{"state":"","year":"","soil":0,"biomass":0,"yield":0,"bag":0}'` , storing the percentages for each farmcode that was computed by percent(Soil/Biomass/Bag/Yield)

HTML:
The html returned for this page is a single div that contains our tab container. Each of our tabs calls a component used for that specific tabs. Component FormPage and HomePage take in a parameter that holds a list of key value pairs that hold farm code as keys and json of percentages as values.

Homepage.js:

Represents the homepage of the dashboard that shows the leaflet map, Error logs, and farmcode list. Home page uses the react-leaflet api and react-input groups for the farmcode input text. Leaflet relies on a height style defined for it's css to work. Code after imports is defaults for using leaflet.

- setFarmCode()
    - Function that is called whenever anything in the input box is changed. Readjusts the leaflet latitude and longitude when a correct farm code is inputted.
- Percent(Soil/Biomass/Yield/Bag)
    - Function just adds a sting % to the end of the number so it can be viewed better in the marker popup.

FormPage.js:

Parameters - this.props.percents, this.props.fullInfo

- Percent is for array of percents for each farmcode
- Fullinfo is farmcode standard information

Form page is the datavisualization for each individual farmcode. The top row holds three input groups for Farm code filter, state filter, and year filter. Second row holds the major table of every farmcode recorded and the percentages they hold. Additionally each farmcode is a button that will turn the second row display to the doughnut display for that specific farmcode.

When the farmcode input field is changed, the second row will change to the doughnut specification. When year or state is changed, the table will filter to show only farmcodes that fit their filters. When the doughnuts are clicked, the lower row of the doughnuts will show a table of each barcode corresponding to the doughnut variable clicked.

- handleChangeCode()
    - Function that handles whenever onChange of Farm code text field is changed.
    - Uses this.props.percent to create new doughnut data so doughnuts update
    - When input is empty, flip state back to 0 so we no longer show doughnut graph and show table instead
- botRightTable()
    - Based on specShow, it will show the checkerboard for either soil/biomass/bag/yield. This is called whenever one of the doughnuts is clicked
    - Calls the components that show the checkerboards
- queryPercents()
    - Functions used for filtering the overall table based state and year.
    - Returns a table <tr> element

        - Variables initialized as a red cross until farmcode variable equal to 100. Then changes to be a green check.
        - Multiple if statements to handle logic of filter
- rightDisplay()
    - Doughnut display for a farmcode. Data for doughnuts is in the form shown by variable “dataDefault”
    - When clicked, changes specShow state to a number of its order, 1 for soil, 2 for biomass, 3 for bag, and 4 for yield.
    - When specShow changes, it uses one of the (First/Second/Third/Fourth)graph.js components.

FirstGraph/SecondGraph/ThirdGraph/FourthGraph.js

Parameters - this.props.farmCode, this.props.fullInfo

- farmCode just contains the string of the farmCode selected from form page
- Fullinfo is farmcode standard information

These are the components used whenever a doughnut is clicked. Depending on what state “specShow”, either first/second/third/fourthGraph.js will be used. Returns a bootstrap table with header columns based on https://docs.google.com/document/d/1sYBG9xoswHbxLFAQpfREsxxS8oN-RGzsH5pu2iKiu-A/edit?usp=sharing.
Contents of the table are shown from using displayTable().

- DisplayTable()
    - Returns a map function in order to list a table of all the barcodes
    - Map function calls eachEntry() that takes in soil/biomass/bag/yield form submission data.
    - ForEach function used to get each index of the array that holds each individual form entry for respective farmCode.
- EachEntry()
    - Returns a <tr> element with # of columns respective to the header
    - Col variables initially assigned to red crosses that state that the element was null.
    - If variable for the farmcode wasn’t null then it is changed to a green cross.