Celyna Su (@celyna)
Jesse Garcia (@jgarc269)
Cristina Lawson (@cristina95138)
Enrique Munoz (@codeMasterHacker)
Luccas Chang (@LukeTheChang)

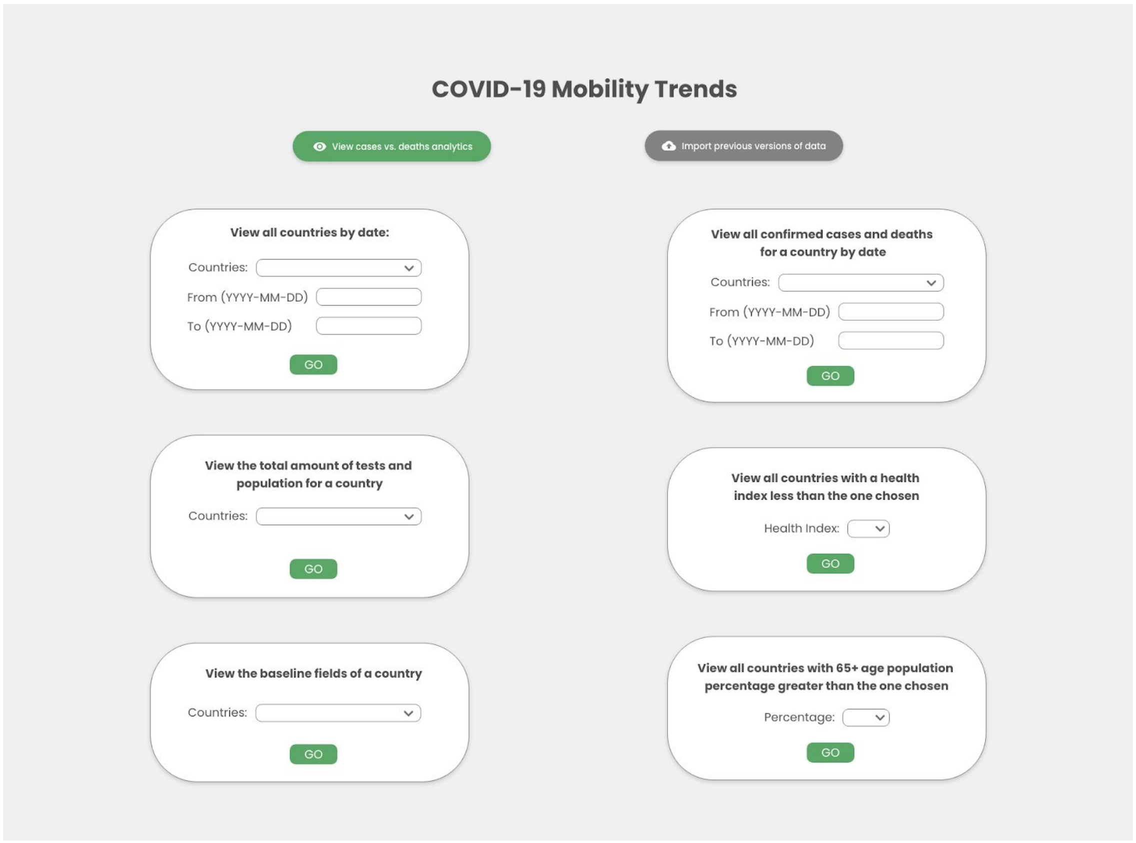
Lab 7 - It's Corona Time

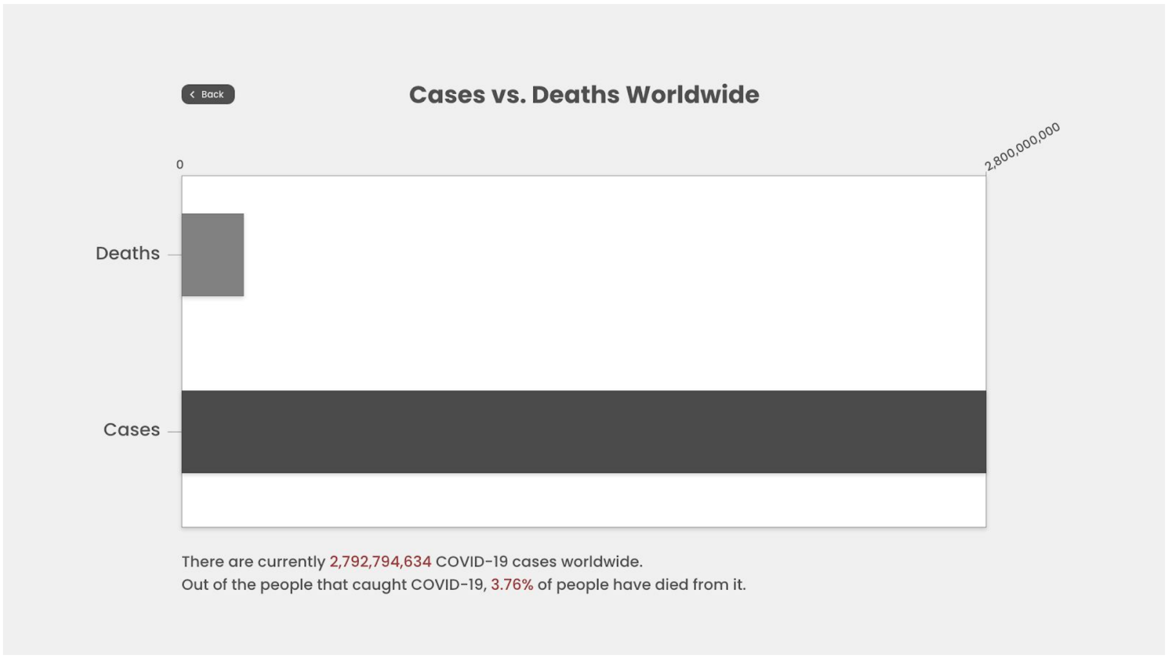
Data Set (for bookmarking)

Features (analytics to implement) to Implement in Next Sprint:

- **World Data: (COVID-19.csv)**
  - **Feature 1:** as a user, I want the worldwide cases vs. deaths analytic to load faster.
  - **Feature 2:** as a user, I want the country cases vs. deaths analytic to load faster.
  - **Feature 3:** as a user, I want the worldwide mobility trends analytic to load faster.
  - **Feature 4:** as a user, I want the country mobility trends analytic to load faster.
  - **Feature 5:** as a user, I want the cases vs. mobility analytic to load faster.
  - **Feature 6:** as a user, I want the workplaces vs. transportation analytic to load faster.
  - **Feature 7:** as a user, I want the rich vs. poor countries testing analytic to have a faster runtime

GUI design:


COVID-19 Mobility Trends
View cases vs. deaths analytics
Import previous versions of data
View all countries by date:
Countries:
From (YYYY-MM-DD)
To (YYYY-MM-DD)
GO
View all confirmed cases and deaths for a country by date
Countries:
From (YYYY-MM-DD)
To (YYYY-MM-DD)
GO
View the total amount of tests and population for a country
Countries:
GO
View all countries with a health index less than the one chosen
Health Index:
GO
View the baseline fields of a country
Countries:
GO
View all countries with 65+ age population percentage greater than the one chosen
Percentage:
GO



There are currently 2,792,794,634 COVID-19 cases worldwide.
Out of the people that caught COVID-19, 3.76% of people have died from it.

User test cases:

- **Feature 1 Test Cases:** as a user, I want the worldwide cases vs. deaths analytic to load faster.
  - **Test Case 1:** The user presses the worldwide cases vs Deaths button and gets displayed a chart with the amount of time it took to calculate.
    - Correct Output: The analytic will do a full calculation, display the results in the graph and the amount of time it took for the calculation.
  - **Test Case 2:** The user presses the return to search operations page and press the worldwide cases vs Deaths button again.
    - Correct Output: The analytic uses the previous calculation, checks for changes and displays the graph with the faster time.
- **Feature 2 Test Cases:** as a user, I want the country cases vs. deaths analytic to load faster.
  - **Test Case 1:** the analytic loads the first time with O(nlogn) runtime
    - Correct Output: the analytic, but results found via slower search.
  - **Test Case 2:** the analytic loads after the first time with O(n) runtime
    - Correct Output: the analytic, but results found via faster search.
- **Feature 3 Test Cases:** as a user, I want the worldwide mobility trends analytic to load faster.
  - **est Case 1:** the analytic loads the first time with O(nlogn) runtime
    - Correct Output: the analytic, but results found via slower search.
  - **Test Case 2:** the analytic loads after the first time with O(n) runtime
    - Correct Output: the analytic, but results found via faster search.
- **Feature 4 Test Cases:** as a user, I want the country mobility trends analytic to load faster.
  - **Test Case 1:** The user presses the cases vs. mobility trends and gets displayed a chart with the amount of time it took to calculate.
    - Correct Output: he analytic will do a full calculation, display the results in the graph and the amount of time it took for the calculation

  - **Test Case 2:** The user presses the return to search operations page and press the cases vs. mobility trends button again.
    - Correct Output: The user presses the return to search operations page and press the cases vs. mobility trends.
- **Feature 5 Test Cases:** as a user, I want the cases vs. mobility analytic to load faster.
  - **Test Case 1:** The user presses the cases vs. mobility button and gets displayed a chart with the amount of time it took to calculate.
    - Correct Output: The analytic will do a full calculation, display the results in the graph and the amount of time it took for the calculation.
  - **Test Case 2:** The user presses the return to search operations page and press the cases vs. mobility button again.
    - Correct Output: The analytic uses the previous calculation, checks for changes and displays the graph with the faster time.
- **Feature 6 Test Cases:** as a user, I want the workplaces vs. transportation analytic to load faster.
  - **Test Case 1:** as a user, I want to see the monthly workplace and transit trends of a specific country in a specific month.
    - Correct Output: Two drop-down lists should be available for the user to select the desired country and a desired month.
  - **Test Case 2:** as a user, I want to see the graph representing the workplace and transit mobility trends of the specified country and graph. So I select the submit button.
    - Correct Output: Upon selecting the submit button, the webapp will take the user to the page showing the desired graph. Being the first instance of this specified search, the process should have an approximate runtime of O(n), involving initial traversals of the .csv file.
  - **Test Case 3:** as a user, I want to revisit the graph representing the workplace and transit mobility trends of the specified country and graph. So I select the desired country and month and then select the submit button.
    - Correct Output: Upon selecting the submit button, the webapp will take the user to the page showing the desired graph. Being the second instance of the search made earlier, the process has a faster runtime by saving the initial search in a data member in the doGet class/function.
- **Feature 7 Test Cases:** as a user, I want the rich vs. poor countries testing analytic.
  - **Test Case 1:** the analytic loads the first time with O(n^2) runtime
    - Correct Output: the analytic, a pie chart, but results found via slower search, more time elapsed, which will be printed in the console
  - **Test Case 2:** the analytic loads after the first time with O(n) runtime
    - Correct Output: the analytic, a pie chart, but results found via faster search, less time elapsed, which will be printed in the console

Taskboard:

Done list of last sprint:

- Worldwide cases vs deaths analytic
  - Implemented by Jesse Garcia

        - Updated SearchOperationPage.jps
            - Created form that displays a message and a submit button to the user
            - Created connection to send the user to the new page using the submit button
        - Updated CovidFile.java
            - Implemented getCases function to get the total number of cases from all the countries in all months.
            - Implemented getDeaths function to get the total number of deaths from all the countries in all months.
        - Updated SearchOperations.java
            - Created connection between the front end to the server using the submit button
            - Submit button checks off a boolean value to calculate the analytic
            - Use getCases and getDeaths to calculate the analytic.
            - Calculate the percentage of people that die from the data
            - Send the user to the new page after calculation.
        - Created caseVsDeathsPage.jsp
            - Created Connection to receive the data from the server to implement into the graph.
            - Implement jsp to display the graph to the user.
- Worldwide change from baseline average of various mobility analytic
    - Implemented by Cristina Lawson
        - Created worldwideMobilityTrendsPage.jsp
            - Implemented ChartJS chart for the frontend
        - Updated CovidFile.java
            - Implemented a getAllMobilityAvg function for grocery and Pharmacy, park, residential, retail, transit stations and workplace
        - Updated SearchOperations.java
            - Implemented frontend way for the user to press submit
                - Communicates with backend to bring user to the analytics page and show the respective graph
- specific mobility comparison to cases per month
    - Implemented by Jesse Garcia
        - Updated searchOperationPage.jsp to allow the user to choose a month and a specific mobility
            - Implemented a submit button to grab the data and send it to the server
            - Implemented a way to send the user to the next page

        - Updated the searchOperationPage.java to retrieve the data sent from the front end.
            - Implemented getCasesPerMonth function
            - Implemented function to send the data back to a jsp file
        - Created new jsp file for new analytic
            - Implement function to retrieve the data sent from the server
            - implement data into the graph
            - display the graph to the user.
- Change from baseline mobility analytic
    - Implemented by Jesse Garcia
        - Updated SearchOperationPage.jps
            - Created form that displays a message, drop down box and a submit button to the user
            - Created connection to send the user to the new page using the submit button
            - Created connection to send the data from the drop down box to the server after the submit button
        - Updated CovidFile.java
            - Implemented a getMobilityAvg function for grocery and Pharmacy, park, residential, retail, transit stations and workplace
                - Function takes in the data from the drop down month to filter the data by the user choice of month.
        - Updated SearchOperations.java
            - Created connection between the front end to the server using the submit button and to receive the data from the dropbox front end
            - Submit button checks off a boolean value to calculate the analytic
            - Implement function to send the data to the new page by mobility
            - Implemented a function to send the user to the new page after calculation.
        - Created allMobilityPage.jsp
            - Receive the data from the server to implement into the graph.
            - Added 6 different sections to the graph to display all the data at once.
            - Implement jsp to display the graph to the user
- Comparison between the number of tests done by the top 5 richests countries, those with the top 5 highest gdp, and the number of tests done by the top 5 poorest countries, those with 5 lowest gdp.
    - Implemented by Enrique Munoz
        - Added 3 functions to CovidFile.java that collects all 133 countries' names, total tests, and gdp

        - Implemented bubble sort function to sort the countries based on their respective gdp
        - Created a pie chart using JFreeChart by getting the first 5 countries and the last 5 countries from the sorted countries array and gave the the pie chart the key value pairs: (country name, number of tests)
        - Created a new .jsp file that displays the JFreeCharts, in this case a pie chart
- Comparison between mobility trends of public transit and workplaces
    - Implemented by Luccas
        - Added and revamped two functions in covidfile.java that collects the mobility trends in workplace and public transit on a daily basis in a month
        - Added a .jsp file that generates a graph based on canvasjs two show the differences between the two cases of mobility trends

To Do task list for the next sprint:

- Improve Analytic Speed
    - Jesse
        - Worldwide cases vs Deaths
            - Implement an additional data structure to store the changes made
            - Use the already calculated data and update the changes
        - Average mobility per month
            - Implement an additional data structure to store the changes made
            - Implement a way to use the data to improve the speed
        - Specific mobility Avg vs Cases
            - Implement an additional data structure to store the changes made
            - Implement a way to keep track of both avg and cases and use them to make the calculations faster.
    - Cristina
        - Implement faster way for analytics to load
            - Update country cases vs. deaths analytic
            - Update worldwide mobility trends analytic
        - Update UI
            - Implement a new interface that is more in-line with our GUI designs
                - Update searchOperationsPage.jsp
                - Update displayResultsPage.jsp
                - Update editFilesPage.jsp
                - Update index.jsp
    - Luccas
        - revamp analytic for comparing mobility trends between public transit and workplaces

            - Consider adding a data member array that stores the previous result so that the program no longer needs to traverse the csv file again to find the specific trends in the same month
            - Consider also adding a flag to indicate whether the necessary covidfile function needs to be called again if the csv has been altered in some way since the user's previous calling; if not then the webapp calls upon the data member array to create the graph, if so, then the covidfile function will be called to generate a more up-to-date array to use in the resulting graph
    - Celyna
        - Implement front-end to look like current GUI
    - Enrique
        - Rich vs. poor countries testing analytic
            - Give each country a ranking, a number between 1 and 133 inclusive, where 1 is the richest and 133 is the poorest, based on the column GDP
                - This will be a new column in the csv file called gdpRanking
            - Then run a linear search on the csv file where we get 10 rows where a country's gdp ranking of 1, 2, 3, 4, 5, 129, 130, 131, 132, 133
                - This will give us the top 5 richest countries and the top 5 poorest countries and their respective other information, like gdp, total cases, total deaths, mobility trends, ect, since we're retrieving an entire row
                - This faster than bubble sort since bubble sort is not linear