Administrative

Team Name:

Rona Team

Team Members:

- Caleb Cazacu
- Kevin Cheddar
- Dilimulati Diliyaer

Github URL:

Link to Video:

Extended and Refined Proposal

Problem: What problem are we trying to solve?

People need accurate data about Covid-19 testing cases.
(We are trying to solve the problem of getting accurate Covid-19 testing/vaccination data to the users)

Motivation: Why is this a problem?

Covid-19 has greatly impacted the whole world. There is a lot of misinformation and this can help people get a better understanding of the accurate state of Covid-19 and wherever they live within the United States.

Features: When do we know that we have solved the problem?

- Users can type in a specific state or whole nation and date to get the data for confirmed cases, deaths, incident rate, and fatality ratio
- Users will be able to choose to see data for a specific day, set of days, or the whole month

Data: (Public data set we will be using and the link to the public data set)

CSSEGISandData/COVID-19: Novel Coronavirus (COVID-19) Cases, provided by JHU CSSE

Data will be read from .csv files into our program. Each day of data has roughly 3300 lines for each county within the United States. Each line contains an FIPS code, county name, state, country, last updated date, latitude and longitude, number of confirmed cases, number of deaths, number of people who recovered, number of active cases, incident rate, and case fatality ratio. Our time frame for our data will range from June 1 to June 30, 2021 which will be 30 csv files.

Tools: Programming languages or any tools/frameworks we will be using:
C++

Data Structures Implemented : Preliminary Data Structures/Algorithms we implemented.

Tree (BST): Name of the State paired with a state object of data including confirmed cases, death counts, incidence rates, and fatality rates.

Tree data structure also including functions like

insert();	insert a node into the tree.
search();	search a node by its state name and return that node.
printAll();	print all nodes from the tree through inorder.

MinHeap: An array based minimal heap data structure that stores the data as an node object which includes the state's name, confirmed cases, death counts, incidence rate, and fatality rates

insert();	insert a node into the tree.
search();	search a node by its state name (key) and return that node (value).
printAll();	print all nodes from the heap by order from min to max.
minHeapify();	functions that make sures the properties of a minimal heap is valid at all times.

Node:
Each data structure above is going to contain a Node that stores the covid-19 data.

```
Struct Node {
    long confirmed;
    long deaths;
    double incidenceRate;
    double fatalityRate;

    (Only for Tree)
    Node* left;
    Node* right:
}
```

Algorithms Implemented:

We created a **Menu** class in order to display the console commands based Menu in an object oriented manner.

Menu class contain also have multiple functions to help us navigate through the menu easily.

1. mainMenu(); the main function for the Menu tree that requests input from the user and calls another function within it accordingly.
2. allData(); if the user choose to print all data for the whole month, this function would be called.
3. state(); if the user choose to access data for certain state or the whole nation, this function would be called, and get input from the user (state).
4. timeFrame(); after user enter a valid state name this function would be called in order to ask the user which date's data they would love to access.
5. wholeMonth(); if the user choose they would like the data for whole month, this function would be called.
6. segment(); if the user choose they would like the data for a set of days, this function would be called and get users input dates.
7. oneDay(); if the user choose they would like the data for one day, this function would be called and get users input date..
8.treeOrHeap(); after the state and the time frame are established, this function would be call in order to provide user the data from Tree or Heap accordingly.

Distribution of Responsibility and Roles: Who is responsible for what?

1. Data Structure Implementation (MinHeap) - Caleb
2. Data Structure Implementation (BST) - Kevin
3. Gathering Data, creating interface (Menu class), implementing program flow - Dilimulati

Analysis

Changes:

Since the first proposal to the final project submission, we made some small changes to our Covid-19 Analyser. First of all, we changed the public data we are going to use for the project, previously our idea was to provide user the covid-19 data for each state within USA but after days of research, we found that any public covid data that provide states data is either not completely accurate or has too little of data for us the reach the requirement. Therefore we changed our approach and pursued a more detailed public data which provided us covid data for each county for every state in the US. Such a change gaved us more detailed and accurate covid-19 data and also helped us meet the requirement for reading 100,000 lines of data. Due to the fact that the public data that we used for the project is different from what we initially planned for, the data that we are able to provide to the user(Feature) also changed a little bit.

Second major change we made to our project is the data structure that we were implementing **(After getting aprovement from the professor)**. Originally we were going to implement BST and ordered map, but after learning the map from lectures we realized implementing the ordered map with BST once again is a redundant move. Therefore we switched to the array based min-heap instead of the map. Such an approach gave us more understanding of the different data structures and we think it would benefit us in the future.

Complexity:

Tree:

- insert() worst: O(logn)
- search() worst: O(logn)
- printAll() worst: O(n)

Heap:

- insert() worst: O(logn)
- search() worst: O(n)
- printAll() worst: O(n)
- minHeapify() worst: O(logn)

Reflection

Overall Experience:

As a group, our overall experience was great. One of the biggest reasons that we work together greatly is because although we distributed our individual responsibility and each person is in charge of one area of the project, we still helped each other out and none of us felt like we were just doing 3 separate projects. By utilizing Discord as our primary communication platform, we enabled easy accessible voice chat and video chat so we could solve any problem in a timely manner.

Challenges:

Although we completed the project fairly easily, we did overcome some challenges along the way. One of the hardest part for a group programming project is not on individual's work in our opinion, all 3 of us completed our own responsibility with not many obstacles. However, when we assemble our functions or program files together and trying to make them work properly, that was the biggest challenge for us. Due to each of us having different programming styles, it took time for us to understand each other's code and fix our own accordingly. Fixing return types, changing variable types, changing parameters within each function - all of these actions helped us overcome the difficulties we encountered and helped us complete this project.

If we start over:

If we were going to start the project all over again as a group, the first thing we would love to change is to have a agreement on specific programming styles and stick to it. We realized the importance of programming style and all of our code should look like a single person created it, no matter how many people contributed. We would also inspect each other code on daily bases and leave comments or suggestions. Anything that would help us understand each others point of view will help us very much by the time we put all our code together to make them work.

What we learned:

Dilimulati: since I was in charge of making the interface(Menu) and assembling all our code together, I learned the importance of object-oriented programming and the importance of keeping the main.cpp as clean as possbile. I created our project's menu as a class, as an object, and connected its functionalities with many different implemented functions. Such an approach helped our group bug fixing or making minor changes to the program without going out of hands.

Kevin: I learned more about collaborating with people who think and code differently. It was actually great to hear different perspectives and come up with our approaches together while bouncing ideas off each other. I learned more about the importance of .h and .cpp files as we had to separate our code into each type of file for proper implementation. I was reminded of the benefit of github as we didn't use it until the end so we shared files directly which led to a few errors on my end.

Caleb: In this project I learned about the frameworks behind different data structures especially the map/minheap data structure as we were unable to use the STL library for the data structure so we were forced to create our own also learned of the importance of teamwork when available for projects, my group was very helpful in coming up with ideas, helping fix bugs, and keeping the project on track. An approach that probably would have been beneficial would have been to become more familiarized with github as it would have helped in the collaborative process as we could share solutions at the same time to make testing the code with each other's parts more efficient.