Andrey Yamkovoy, Arshad Mohammed, Joe Hand, Zemyna Kaite

# CSU11013 Group 9
# Programming Project Report

## Introduction

Our project presents Covid-19 data in an easy-to-understand and transparent manner, with a strong emphasis on user engagement, ensuring that whoever is viewing the software has full control over what data they want to see and how it is presented.

We started by determining what tasks needed to be completed and how we should divide them among ourselves. After that, we started focusing on our individual assignments. Each section below explains what each of us contributed to the project.

**Note: Before loading the program for the first time, complete the "SQLite Installation Instructions" near the bottom of the report.**

## Andrey

In the project I worked on the acquisition and formatting of the data to make it more accessible for my group members. I also worked on making several visualizations of the data in forms of tables in order to be able to see the data in a readable format in the application.

Using the SQuelized processing library and an SQL database, we were able to send custom queries that would return any data we wanted in the processing Table format which we could then manipulate and create visualisations from. I then wrote the class DB_import which imports all the data into the database in an effective manner using stringBuilders and indexes.

I made the TextWidget class to be able to have user input through text boxes. I then began work on implementing on table visualization of the data which is variable with the amount of data it receives such that it changes row/column size and automatically changes column width depending on the longest value in a set of rows.

These tables are made by using the Grid and Cell classes that I believe are also impressive due to the fact that I implemented them in such a way that elements of the table are selectable by up/down keys and the tables scroll up and down when the current selected element goes offscreen.

The Cumulative Cases Screen shows 2 tables which show the total cases for all counties/states, amount of cases, the population and the cases per million  in a specific county and area for a specific date.  The Timeline cases Screen shows 3 tables which allow you to select a specific county and or area and look at the cases and cases per million amounts over a range of dates.

The tables are fully interactive and display which is currently selected and when a county or area is selected the tables update when you hit the update button.

## Arshad

I worked on setting up communications between the team and setting up a decent workflow between the using discord. The discord  proved to be incredibly helpful, especially when debugging and committing.

I wanted to do a graphical representation of the covid data, and decided on a tree map visualisation. I found a helpful library called Squarify, however it wasn't effective enough to use because it was difficult to store information within the classes provided in the library. I wrote up the library in processing to work with, but it utilised recursive constructors, which were difficult to use and did not work in processing, so I scrapped that and imported the library normally.

I started by making the tree map function with static values. Then I made it so that when I clicked on the screen it changed the values at random, and after that I made it move the x and y coordinates to the final location. At first, this was a challenging task, and I spent a week trying to find out how to do it. Once this was done, I implemented the same system for the x and y coordinates but for the length and width of each rectangle in the tree map. I now had an animated tree map.

Next I had to use the data set parsed by Andrey and map that on. To do this I used another class and object system to run parallel object arrays with the object arrays for the treeMap class.
I then used a heat map for the tree map and added button widgets for moving different dates.
I made two versions; One was fifty states, and the other was the top ten states. We ultimately decided on the tree map with the top ten states.

## Joe

I worked on graphical representation of the data, specifically creating bar charts to show the amount of cases specific to areas in different states at different dates. I also worked on line charts, which can be used to show the total number of cases in different states over time. Both charts use queries to acquire the data that can be changed by the user using buttons on the screen, allowing the user to change the states and dates of the data they are viewing.

Classes were made for the bars, points as well as the lines to connect the points. For the Bar class an ArrayList of Bar objects is made when creating the bar chart on screen. When created, the y-position of each bar is decreased, with the height of each increased at the same time, allowing for an animated movement of the bar. A similar method is used for the points, though only a y-position variable for the points is used. The line class simply draws a line between two points, taking in those points and a colour as parameters.

When a user presses a button to change the query, the ArrayLists are emptied and then remade with the new values. When the mouse is hovering over a point/bar, more specific information is given regarding the data.

My main issue as I worked on the charts was making sure that the bars/points animated correctly and presented themselves on the graph correctly. The map() method is used to make sure that the size of the bars/points is kept within the graph. The speed at which points/bars move along the screen slowly decreases, giving a smoother effect while also limiting how far the objects can move along the screen. I made sure that the speed variable did not go below 0, to prevent the charts' animation from going in the wrong direction.

## Zemyna

I began by making a simple default backdrop for all of our screens in PhotoShop, and by creating a home screen with a title and buttons that brought you to all of our different screens. This was accomplished by using the "Screen" and "Button" classes. I established a simple colour theme for our project and wrote the case statements that displayed each different screen.

I decided to work on a map image that would display the total cases per each state (up to 28/04/2020) with a simple colour key. First, I found an image of the United States and uploaded that to our project, drawing it on the designated "Map" screen. I then made a "State" class, which allowed me to make each state it's own object. Each state had its own cropped out image, as well as a rectangle region within the state (or on the label) with a width and height, a title, a population and the total cases in that state.

The state would be drawn directly over the original map image if the mouse was hovered over the rectangle region allocated to each state, and the tint of the state was determined by the total cases in that state, and corresponded to a key drawn on the left side of the map.

I also introduced a small feature whereby if the user clicked on any of the states, a pop-up on the right would appear with some key information such as the name of the state, its population, the number of cases in that state, the total number of cases in the United States, and finally the percentage of cases in that state compared to the total number of cases in the United States. The population and the total cases per state as well as the percentage were gotten using methods and/or queries from our Covid database.

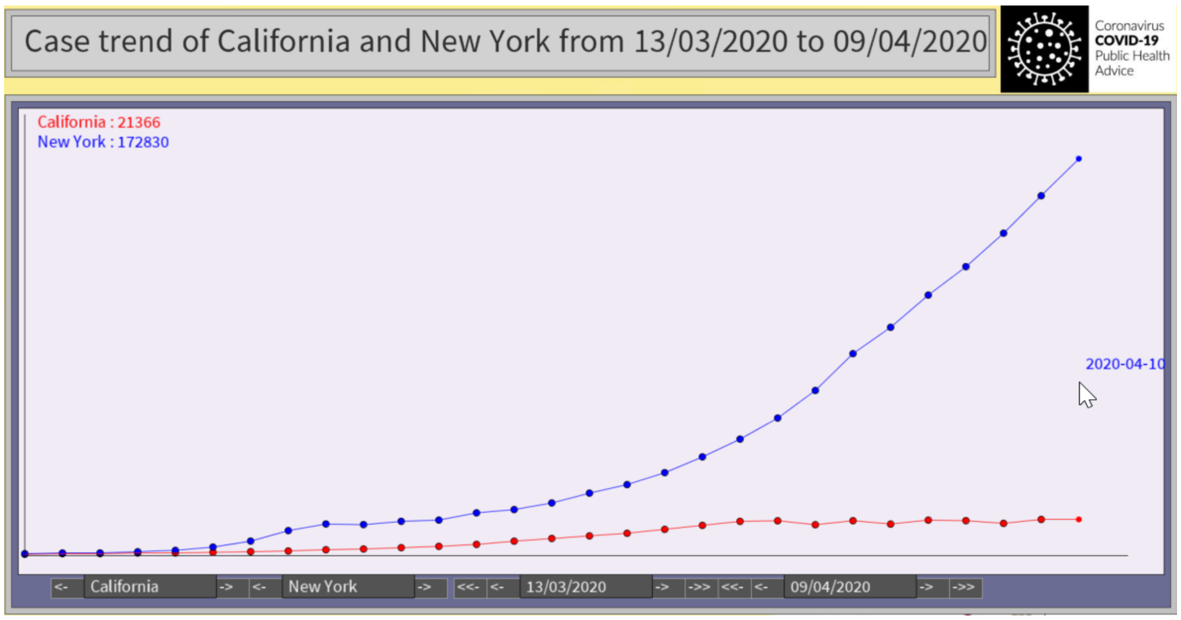
Then, I moved on and worked on the "Headline Figures" Screen. This screen displays three simple Covid figures: the total number of confirmed cases in the United States, the total number of confirmed cases in the state of California, and the total number of confirmed cases in Washington, DC, the nation's capital, as well as a date showing the last date on our data set.

After that, I worked on all of the other screens, adding headers and backgrounds to each one to keep our project's theme consistent. To make it look as clean and uniform as possible, each screen has identical borders, headers, and unique sections to separate each "part" of a screen.

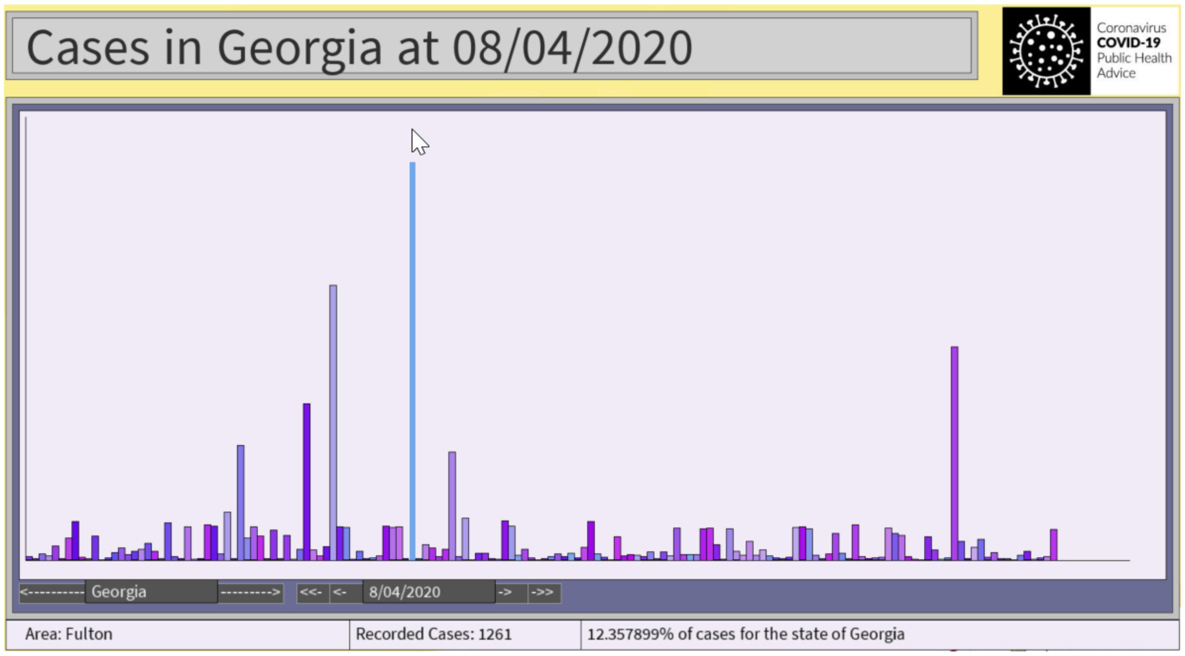
## Conclusion

When completed, this project included a variety of visualisations and displays of the Covid-19 dataset we were given. The use of data tables, charts, and graphs, as well as more broad data representations like the US map and tree map, brought the project together and resulted in a fantastic data viewer.

Sample output of Line Chart:



Sample output of the Bar Chart:

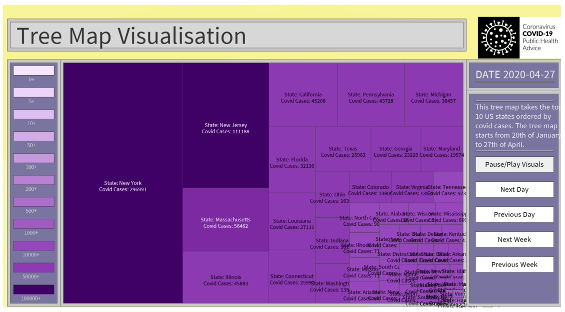


Area: Fulton | Recorded Cases: 1261 | 12.357899% of cases for the state of Georgia

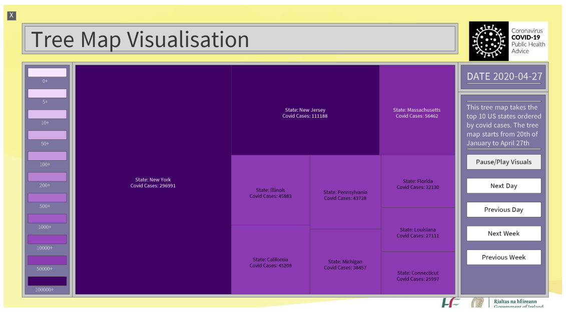
The Tree Map Data class:

```
class TreeMapData
  {
    String[] stateNames;
    String date;
    TreeMapData (String theArray[], String date)
      {
        this.date = date;
        stateNames = new String[SIZE_OF_TREE_MAP];
        for (int i = 0; i < theArray.length; i++)
        {
          stateNames[i] = theArray[i];
        }
```

Tree Map Visualisations, 50 States vs Top 10 States.

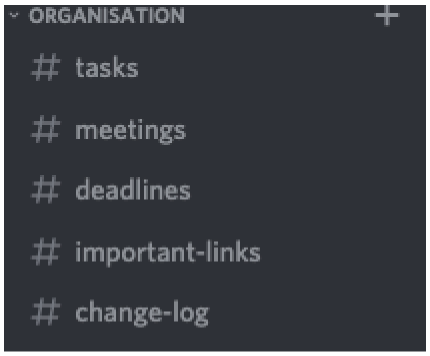


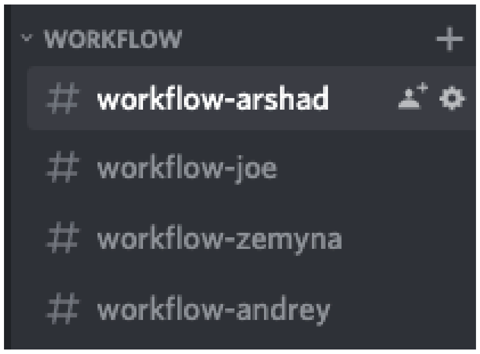
This tree map takes the to 10 US states ordered by covid cases. The tree map starts from 20th of Januar to 27th of April.



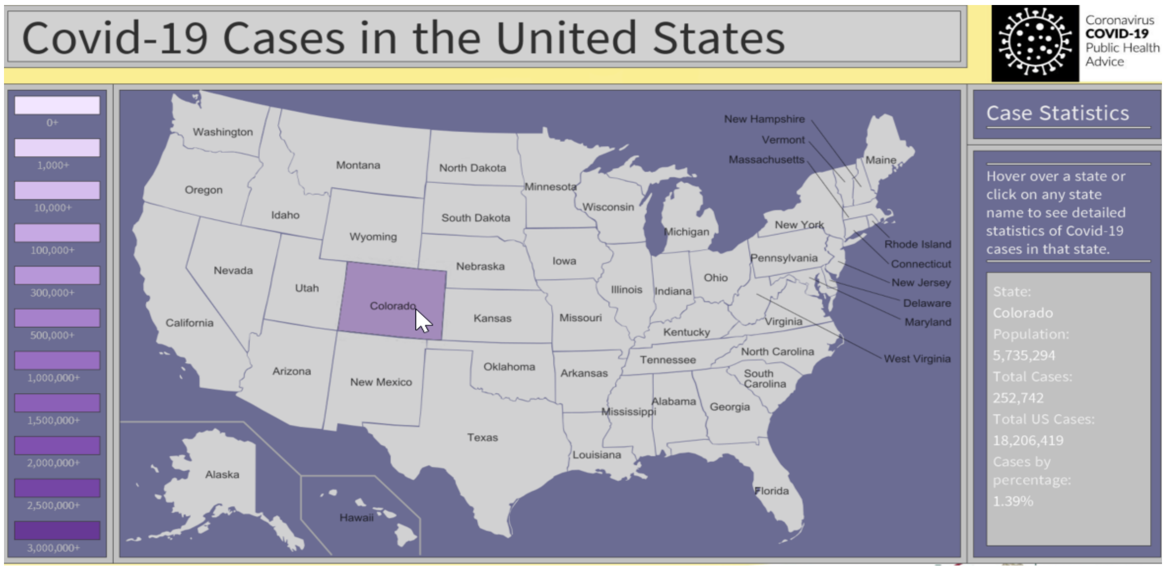
This tree map takes the top 10 US states ordered by covid cases. The tree map starts from 20th of January to April 27th

Discord Channels that were used for communication and sharing of workflow:





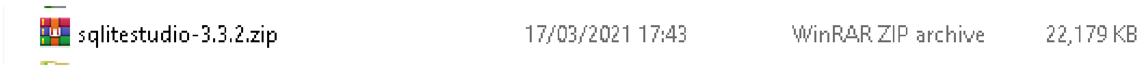
Sample output of the US map:


Covid-19 Cases in the United States
Coronavirus COVID-19 Public Health Advice
Case Statistics
Hover over a state or click on any state name to see detailed statistics of Covid-19 cases in that state.
State: Colorado
Population: 5,735,294
Total Cases: 252,742
Total US Cases: 18,206,419
Cases by percentage: 1.39%
0+
1,000+
10,000+
100,000+
300,000+
500,000+
1,000,000+
1,500,000+
2,000,000+
2,500,000+
3,000,000+

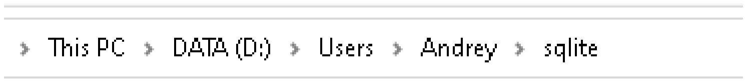
## SQLite Installation Instructions

As we are using an SQL database in order to run the program first one must download sqlite and set the connection path  in the setup in the Main tab in order to connect sqlite and the program.

The download for sqlite is in the repository,


sqlitestudio-3.3.2.zip 17/03/2021 17:43 WinRAR ZIP archive 22,179 KB

In order to install it you have to unzip it and place it into a folder called sqlite or alternatively change the folder name in the setup of the main.

For example, this is how I did it, 
This PC › DATA (D:) › Users › Andrey › sqlite

and thus the location of the sqlite file in the connection path is

```
//myConnection = new SQLiteConnection("jdbc:sqlite:/
```

For the connection path in the setup() of the main, you then need to put the path of where you put the files from the repository so that it finds the database.

For example

```
/D:\\Users\\Andrey\\Desktop\\Programming project repoistory\\CS1013-2021-9.\\covid_data.db");
```

So in the end this is the format that it is in

myConnection = newSQLiteConnection(“jdbc:sqlite:/Replace with Letter of disk space where program is placed:\\Replace with path of where you put the folder containing the program”);
Full example: myConnection = new SQLiteConnection("jdbc:sqlite:/D:\\Users\\Andrey\\Desktop\\Programming projectrepoistory\\CS1013-2021-9.\\covid_data.db");