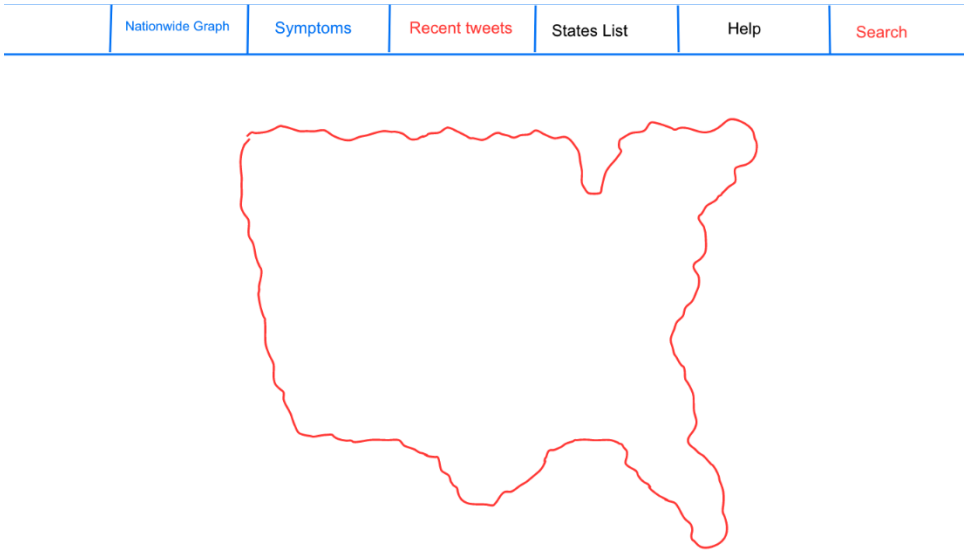
Group 3 CSU11013 Project Report

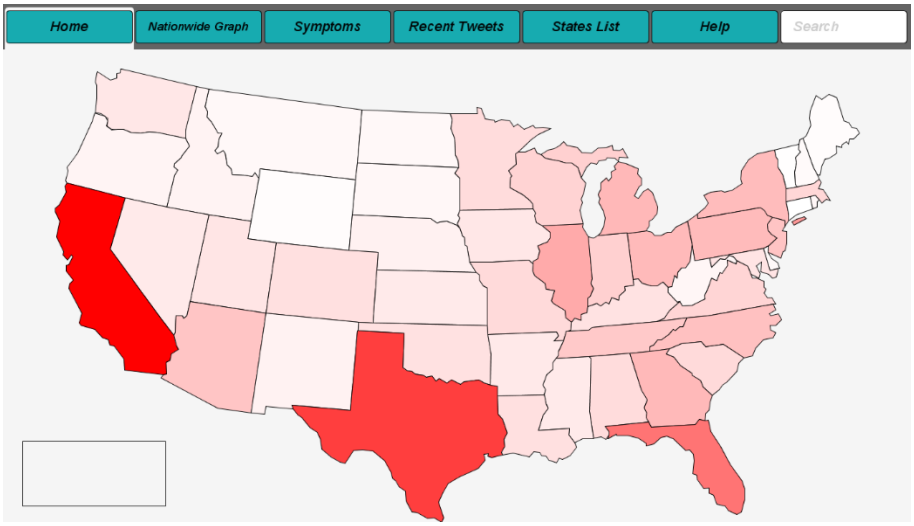
Outline of Design:

- To start off, we first created a class called datapoint that would store information from each line in the excel file. We then created a method that would take in the data from the excel file and pass all the information to a new datapoint object, and then adding each new datapoint to an array list of datapoints. This would make processing and manipulating the data much easier.
- After creating this array list, we created a design plan which would be able to incorporate everything we wanted to display from the data such as a heat map, cases per date in individual states, covid-19 symptoms etc. We decided on having a program with multiple tabs that would bring you to different screens. Each screen would serve its own purpose. We also wanted a search bar somewhere in the program that would make it easier to navigate to certain states without having to scroll through all of them.
- In the end we settled for a design that would have 6 tabs/buttons which would be constantly displayed along the top of the screen, and a search bar to the right of those tabs.

The original draft we came up with on our first meeting.



The final project



Splitting Up of Work and Organisation of Team:

We created a Discord server for our communication. This allowed us to be in constant communication with any menial issues we needed help with. We also met every week on a Tuesday afternoon to prepare the needed code for our Thursday labs.

We decided to split up the work as follows:

Anton:

- Created feedback box which saves text to a file.
- Created scrollbar class and implemented on multiple screens.
- Implemented first scrolling feature (which was replaced by scrollbar).
- Coded states class.

Laura:

- Coded the bar charts.
- Coded the widgets (except the scrollbar).
- Coded the switching of screens and both screens that displayed the bar charts (National Graph, and States List screen).
- Implemented the search feature.

Aisling:

- Coded the class for the Tweets.
- Implemented the Twitter4J library to allow us to fetch Tweets to display.
- Implemented the GeoMap library to create our map.
- Created functions to determine the colour of each state in relation to the highest nubmer of cases nationwide.

Liam:

- Created the initial version of the datapoints class.
- Wrote the code in main to fetch data from the excel file and pass it into data points
- Coded the symptoms screen with symptoms, buttons that link the HSE website and a scrollable table.
- Coded the county class.

Features Implemented:

Twitter

- We utilised the library Twitter4J to implement a Twitter aspect to our project. This involves fetching up- to-date Tweets related to the COVID-19 situation. The Tweets are stored as an array list and a suitable design for outputting each Tweet was created.
- We created a button to refresh Tweets. When this is pressed, the array list is overwritten, and the new Tweets were stored in their place. These Tweets are drawn to the screen while the user is on the Tweets screen.

Heat Map

- Our first page is a map of the United States. We used the GeoMap library to do this. Each state has its own ID number which allowed us to draw each state in a different colour. We drew each state with a colour that was calculated on the state's number of cases in comparison to the highest number of cases in the whole country. This part of the project also allows the user to see the name of the state on which their mouse is currently hovering over.

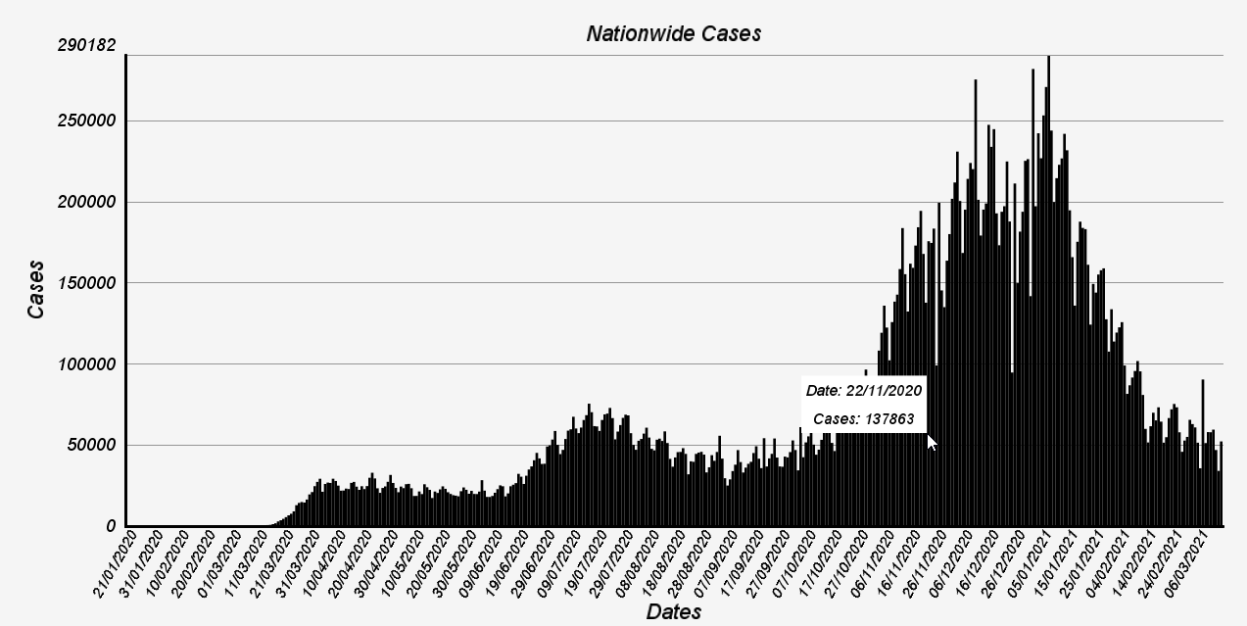
Search Box

- We implemented a search feature that can be used by the user to search up a state or a county by their name.

Graph and Textual Output of Data

- We coded the graphical output of data. This was done by creating a BarChart class that had basic methods such as a constructor, an empty draw method, and a hoverOver() method that is utilised by all the subclasses that sort the data by date rather than by area. Then, I wrote 5 subclasses that handled several types of data. This was done so that they handled different areas (nationwide-level, statewide-level, and countywide-level) and whether they displayed data based on date or subsets of the area. Each of these several types of graphs are accessed via buttons that can be pressed on different screen or through a search function that takes both area names and geoid. Before the program is displayed to the user it loads up all the graphs so that they can be called instantly when user input is detected. They automatically customise the intervals on the y-axis based on the maximum y-value in the graph.

An example of a nationwide graph at 1M cases



Screens and Widgets

- As mentioned in the above paragraph, these graphs can be accessed on different screens and through widgets. These widgets were implemented early in the project, and we did not meet any issues with them as we had previously coded widgets for an assignment and so these were heavily inspired by them. They are used in a variety of ways in the program; however, the main use is for switching screens and to select a graph. The screens were designed around the idea of material design. These were also inspired by a screen class that we wrote for an earlier assignment and so we did not meet many issues with their implementation.

User Feedback Box

- Feedback plays a very vital role in developing applications. Software is fundamentally made for other people to use for different reasons, meaning that you want to make sure that all of the users are happy while using the application. If the users aren't happy with the application, then they will just stop using it.
- To make sure that the users are as satisfied as possible when using our application, we have developed a feedback text box which lets the user type out any concerns or wishes they have in regard to the application, which then is saved as a text file. The developers might not always encounter all the bugs that are necessary to be fixed, which is why we are giving the users an option to let us know of anything that needs fixing.
- The user is limited to 650 characters as to reduce file size and to improve program efficiency. As of right now, the file is only saved locally, however if further implemented, it could be possible to store these files in a database, and then accessed through a network.
- This feedback can be read by the developers and then be used to improve our program further by fixing any bugs the users have reported or implementing any suggestions.

Scrollbar

- Scrolling is necessary to navigate through the data as since there is a lot of data, it does not all fit on the screen. Initially, we had first implemented a scrolling feature, which would function by the use of a mouse wheel or mousepad, however we witnessed some cross compatibility issues, so we decided to switch to a scrollbar instead.
- The scrollbar operates the same way as any scrollbar on a webpage or software. When the mouse hovers over the bar, the scrollbar will light up a yellow colour to let the user know that they can now click and drag the scrollbar to navigate the data. When the user stops hovering over the bar, then the scrollbar will switch back to the red colour. There is a bit of weight on the scrollbar to make it function smoother.

Problems Encountered:

Scrolling Issues

As previously mentioned, we witnessed some cross compatibility issues with the scrolling feature. For example, on Windows the scrolling would work perfectly on the "States List" page, however on a Mac there would be some whitespace created at the top and bottom of the page. This was tested in our group as two group members had Mac and two members had Windows, and running the exact same versions this had happened. On the "Symptoms" page, the Windows users also had problems with scrolling as it would come out very glitchy, however the Mac users did not witness this problem.

We couldn't find a good fix for this compatibility problem, so that's why we decided to implement a scroll bar.

The scrollbar worked well, however we later noticed that on the "States List" page the scrollbar caused the whole page to lag, specifically when using the 10k+ files. It was later found out that the cause of this lag was the drawing of the vertical rectangle within the draw() function of the scrollbar class. A fix that was implemented is instead of drawing the rectangle within the scrollbar class, it is instead being drawn inside of the screen class.

Fetching Tweets

A minor issue encountered when using the Twitter4J library is that you cannot fetch Tweets from Twitter constantly, otherwise the Twitter Developer account you have the program linked to can get blocked from accessing the Twitter API. To counteract this, we only fetch 15 Tweets at a time. The function to fetch Tweets is called once when the user clicks onto the "Tweets" screen and is only called again when the user clicks the "Refresh Tweets" button.

Bar Charts

One of the main issues we encountered while implementing the bar graphs was that the labels were too small to showcase, this was overcome by only displaying a certain amount of labels and being able to see the specific label that belongs to a bar by hovering over it.

Home Screen

Initially our plan was to have a list of scrollable texts that represented each of the data points, however this proved to be very inefficient for larger data sizes. This has since been replaced by a heat map on the home screen.