PROGRAMMING PROJECT GROUP 19 REPORT

Design:

Our main goal was to create a program that somewhat resembles a website. We wanted to implement multiple screens with buttons that allowed you to seamlessly and intuitively travel through them. The buttons add a menu-like interface allowing the user to interact with various aspects of the program. We chose a more visual approach over a textual one to allow the user to view data easily and clearly, rather than having to search through vast amounts of text. We also wanted to add a few more elements than just data visualization, that's why we chose to add a "Quiz", "Charity", and "About Us" section. Adding these elements really brought our entire design together and we felt created a really great, well rounded experience for the user.

Home Screen – Fig.1:

As in you see in Fig.1, we wanted to keep our design website-like. For us, the obvious choice was to design a home screen that laid out options for the user. The Statue of Liberty image is the background we chose for this screen, mainly because it's an iconic monument that will instantly be recognized by the user as something to do with the US. At the very top of the home screen the user will see "Welcome to the US COVID-19 Data and Statistics App". Underneath the title in white writing, we have a live case and live death count for the US which updates every time you run the program. To do this we used a New York Times 'Github' repository, URL of which can be found linked underneath the text. On the right hand-side of the home screen six yellow buttons are displayed - "Monthly Statistics", "State Statistics", "Heat Map", "Quiz", "Charity", and "About Us". By hovering over any of these buttons, the outline will change from black to white to add an interactive feel for the user. When a button is clicked on, the program takes you to another screen and displays the query you chose, i.e. clicking on the "Monthly Statistics" button takes you to a visual display of the monthly COVID-19 statistics for the US.

Monthly & Daily screen – Fig.2:

Fig.2a and Fig.2b shows our monthly and daily screens. We roughly stuck to a yellow-orange colour scheme for the graphs and a light grey-blue for accents as we thought the two colours look nice together. Y-axes are labelled with the case amounts and X-axes are labelled with months, days or states depending on the graph. Each graph has a title related to the data it's displaying, e.g. "Months". There is a text box displayed the first time a user views the monthly and daily graphs; prompting them to click on a bar to display more data. This was done to aid the user's dynamic experience, as above. Once a bar is clicked on in the monthly screen, it brings you to a graph displaying the daily data for the month corresponding to the bar the user clicked on, for example, clicking on the bar that represents November 2020 brings you to the daily screen that shows the COVID-19 cases on dates 01/11/2020 through to 30/11/2020. When in the daily screen, clicking on a bar will bring the user to the date breakdown screen.

Date Breakdown Screen – Fig.3:

This screen is accessed through the daily screen as above. The screen can be seen in Fig.3. We chose an image of the white house as the background for a nice contrast with the relatively simple podium. The date corresponding to the data is displayed at the top of the screen in black text as "Top cases on the xx/xx/xx". The podium is occupied by the three states that had the most cases that day. It also provides the percentage of total national cases the state had. For example (in Fig 3), Michigan occupied the top pedestal on 16/11/2020. It had 12417 cases which accounted for 7% of the nation's cases that day. This data is displayed above the top pedestal in white as "Michigan, 12417, 7%". The "back" and "home" buttons (Fig.9a and Fig.9b) at the top left of the screen allow the user to return back to the daily graph that they just came from or back to the home screen.

State Screen – Fig.4:

The state screens can be seen in Fig.4a and Fig.4b. It looks very similar to the graphs mentioned above. This is because they're all constructed using the same class. There are two screens displaying the state graphs because there wasn't enough space to output all the areas onto one graph. The user doesn't suffer here, though. An arrow pointing to the "next" button (Fig.9c) in the top right corner of the first graph invites the user to see the remaining states and the "back" button allows a sequential and smooth transition between the two. The user is able to return home on both pages using the "home" button.

Heat Map – Fig.5:

The heatmap (Fig.5a) is an interactive map of America that lets you hover over the different states and see the state name and total cases. We chose red as the map colour which shows up darker the higher the cases in a given state are. The colour scale above Central America highlights this idea. We learned to use 'Hashmaps' where we were able to store two different types. This is so that we could easily access them together and break them into separate arrays where necessary and manipulate the data as required. Text prompting the user to hover their mouse over a state is displayed, doing so will change the state to a blue colour and a pie chart will display the top 10 administrative areas most affected by COVID-19 in that state on the left-hand side of the screen. We developed a colour scheme that went from red to yellow to green so that the user could easily see the different slices. We chose to display the top 10 admin areas so that the pie chart would be readable and make sense to the user. A key containing the list of areas and their corresponding colour within the pie chart is also shown with the percentage breakdown. You can see an example of this in Fig.5b, where the user is hovering over California.

Quiz Fig.6:

The "Quiz" (Fig.6a) is just a simple COVID-19 symptom test. Here, we once again stuck to a yellow theme with light-blue accents. A user clicks the "Begin" button (Fig.9d) and starts the test. Here (Fig.6b) they will be asked whether they have experienced any common COVID-19 symptoms. Should the user answer "Yes" (Fig.9e) to any of the questions, they will be shown a message recommending them to self-isolate and contact their GP if needs be (Fig.6c). Otherwise, if only the "No" button (Fig. 9f), they will be advised to continue to stay safe and follow government guidelines (Fig.6d). The user can navigate between the questions with ease using the "back" buttons provided, and return home at the end of the test. There are also a number COVID-19 cells floating around the screen, which we felt were an appealing touch and made the screen less bare.

Charity Screen – Fig.7:

The charity screen (Fig.7) was an idea that we came up with to help spread awareness as well as help those who are in dire need of medical supplies to combat the pandemic. We chose an impactful background for the screen, to truly capture the user's attention and invoke an emotional response. We decided to use a different set of colours to our main screen in order for it to stand out. We implemented an informational box to let users understand the harsh situation of COVID-19 worldwide, especially in third-world countries. We then utilized a "Link to Donate!" button (Fig.9g) on the side that, once clicked, opens your default browser and takes you to the 'Unicef' donation page for COVID-19 (Fig.7b). We also included a back button to take the user back to the home screen.

About us - Fig.8:

The about us page (Fig.8) was another simple idea that we decided to add. We used a blue table style format to keep it nice and clean. The page contains important information about the developers behind the program, such as names and email addresses. It also includes little fun bits of information about us, for instance our hobbies.

How we split up the work:

The most effective and efficient work method we came up with was each member working on their own part of the project (screens, classes, methods, overall design, etc.). On Mondays we amalgamated everyone's work into one weekly file, discussed what the next step in the project should be and agreed on any necessary changes or fixes we should implement before demonstrating on Thursdays. If anyone was having trouble with their part of the program throughout the week, they'd text into our group chat so we could help each other to try and come up with a solution. We set our work for the days to follow, polished our code and tried to come up with new ideas to extend the program at the labs. As well as setting short-term, weekly goals we kept focused on what we wanted the end product to be. We found that working on our own tasks and then showing each other the result was a better approach than everyone working on the same thing at the same time. This way, problems that one person couldn't fix were solved quickly and with relative ease with some collaborative brainstorming.

How we went above and beyond:

Graphs:

The backbone of our program is the graphs which allow us to visualise the imported data. The graph class has been perfected to display user queries passed in as parameters to the constructor. Whether you want to view COVID-19 cases by state, month or date, the class can take in specific data and create a graph from it. Having all these graphs as one class greatly benefited efficiency and the user's enjoyment. One can easily fly between graphs and back home using the buttons in the top left. A lot of time went into modifying the code to handle all cases, potentially more time than actually displaying the graphs themselves. We feel it was worth it though to allow a more fluid, website-like experience that we have echoed throughout the last four weeks.

About us, Charity, and Quiz screens:

We thought telling the users some information about the members of our team would be a good addition to the program. We are proud of what we were able to build together and having our names on the project was important to us. With regards to the charity screen, sometimes we forget that other less well-off countries are suffering during the pandemic too. Our charity section gave us a chance to inform our users about such countries and gives them an easy way to donate if they want to. The quiz screen gives the program more functionality - they can easily test themselves for COVID-19 symptoms and get a quick recommendation on what they should do in either case. It also adds a bit of interactivity to the program which otherwise would largely lack this aspect (other than clicking on the graphs).

What we learned:

Inheritance:

As we got further along in our project and continued to learn more in our programming class, we found ourselves learning techniques that would have benefitted us at an earlier stage. Inheritance definitely fell under that category. The graph class we have now could have been done a lot quicker and more efficiently, had we known what we do now. However, since we spent so much time and effort on what we have presented to you, we decided changing our approach would cause more problems than it would've resolved.

Variable names and use of constants:

As time went on, we found ourselves running into increasingly more problems regarding variable names. Knowing what we know now, we would've spent more time naming these variables at the beginning. None of us have ever spent this long on one program, using this number of variables, but in future we will definitely put more care into such a task. Another point of improvement we found was through the use of constants. We didn't fully appreciate how much the use of constants would help us, in the beginning. As the weeks went on, we found ourselves constantly adding to our constants tab, to

aid the debugging and appearance of our program. Had we had to do the program again, this is an aspect we would look to improve upon.

Importance of communication and comments:

Another facet of experience we gained from this project, was that of communication skills. The majority of these skills were done through our Discord meetings and our WhatsApp group. However, another vital avenue of communication, that we hadn't explored in the past, was in our program's comments. As the weeks went on, we all noticed an improvement in our comments. This was because we realised how difficult it could be to understand someone else's code without them. This is something we will all look to incorporate in all our code, as we progress in our programming careers.

Problem solving:

A couple of times we found that an original approach we had to a problem was ineffective or too complicated. For example, we wanted to read 2 numbers from a website (https://www.nytimes.com/interactive/2021/us/covid-cases.html, "cases" and "deaths"). We figured that importing the table containing these numbers to Excel, linking to the website, converting to a CSV file and then reading it into our program would do the job. However, after many attempts to make this work, we decided to leave Excel out of the equation because once the file was saved as a CSV the link to the website was removed and updates weren't able to happen. We ended up finding the "New York Times for Developers" on our search for an API to try do something with JSON or XML, but instead we found a series of data files uploaded onto GitHub by the NYT. This was perfect because we could convert these files to CSV and read them into our program with ease. This is only a single, brief example of a problem we had to solve, but there were many others that we were able to overcome.

Conclusion:

We thoroughly enjoyed working as a team. We surprised ourselves how much we could achieve in a few weeks and gained an abundance of knowledge and experience, as detailed above. We hope you enjoyed viewing our program, as much as we did making it.

Fig 1:

Fig 2a:

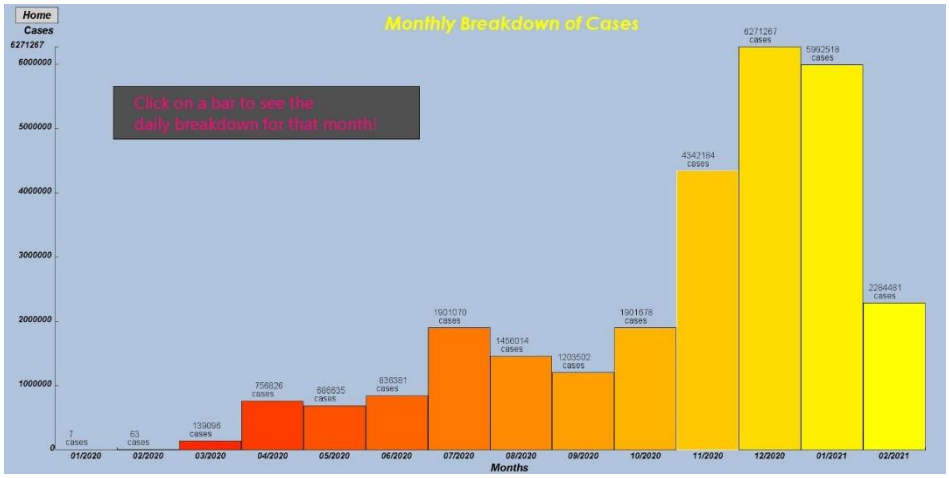

Fig 2b:

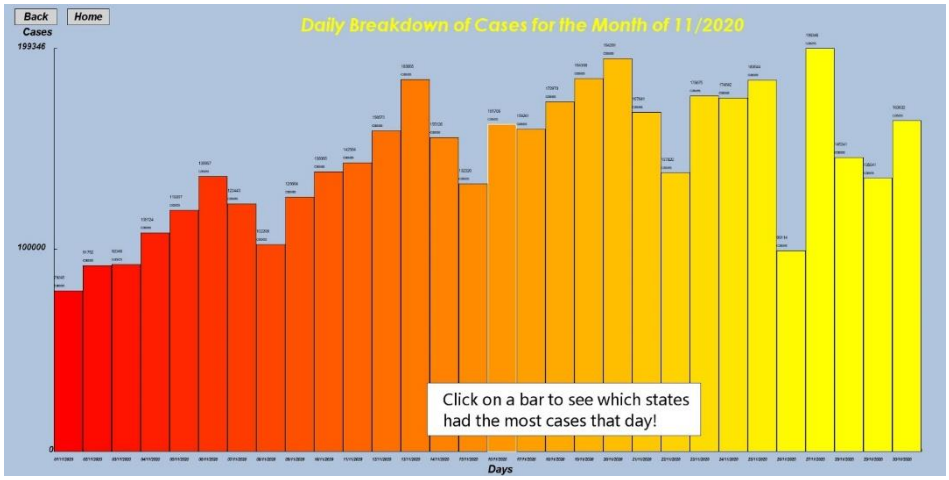

Fig 3:

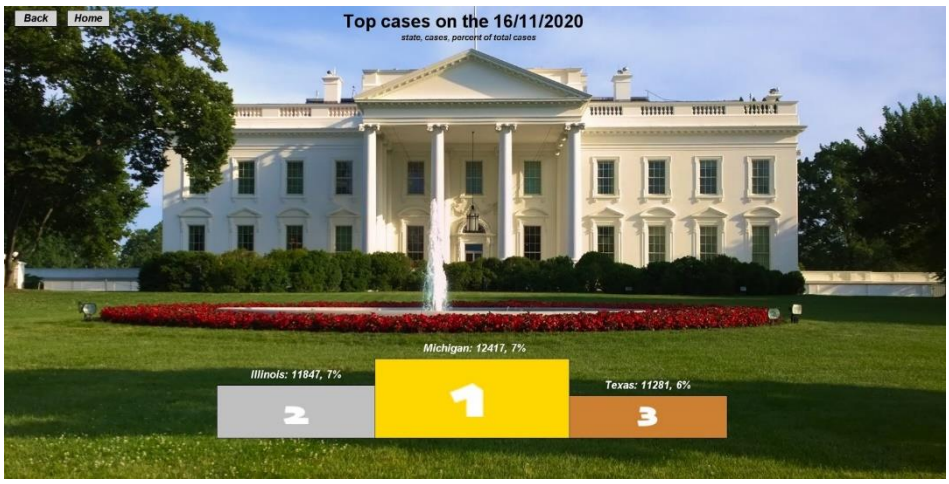

Fig 4a:

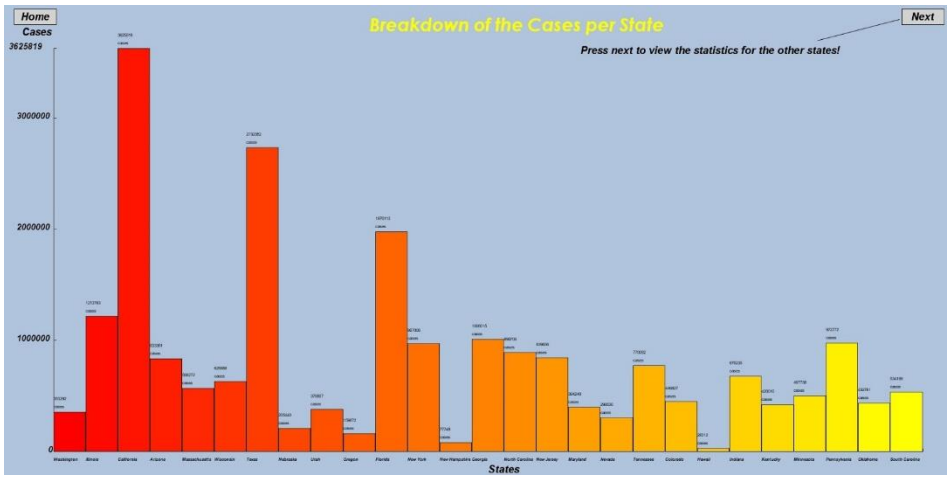

Fig 4b:

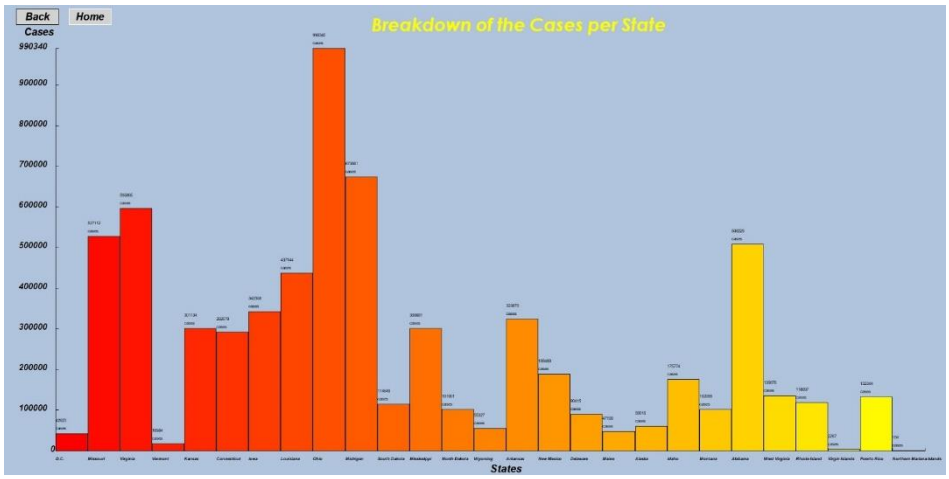

Fig 5a:

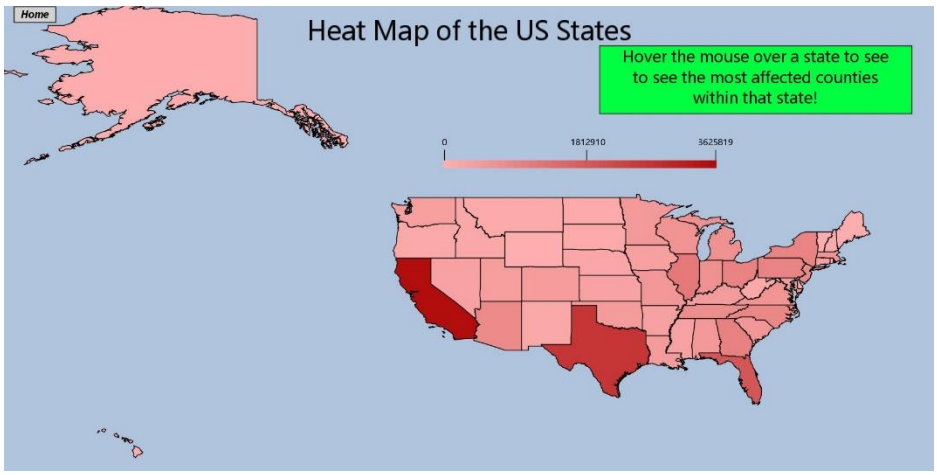

Fig 5b:

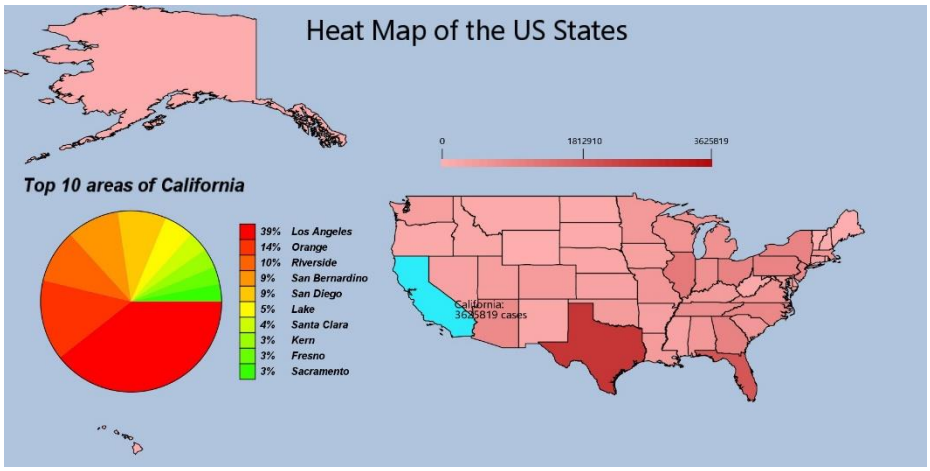

Fig 6a:

Fig 6b:

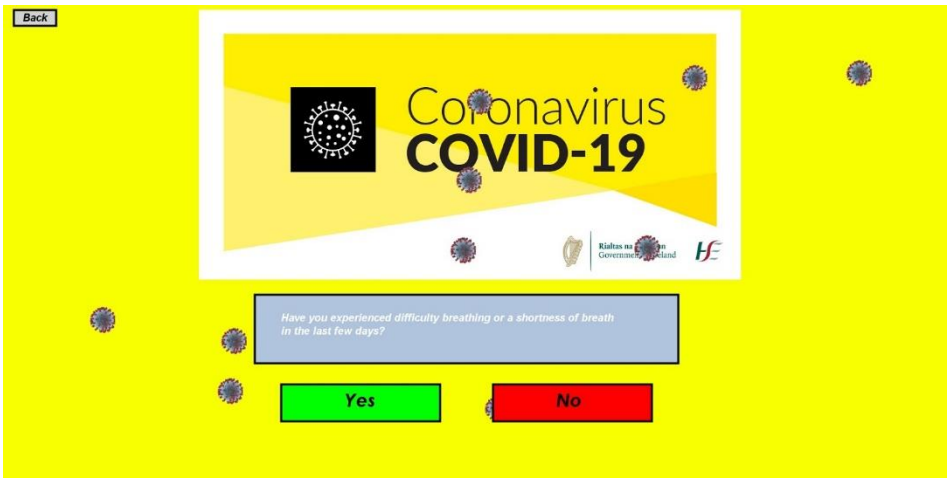

Fig 6c:

Fig 6d:

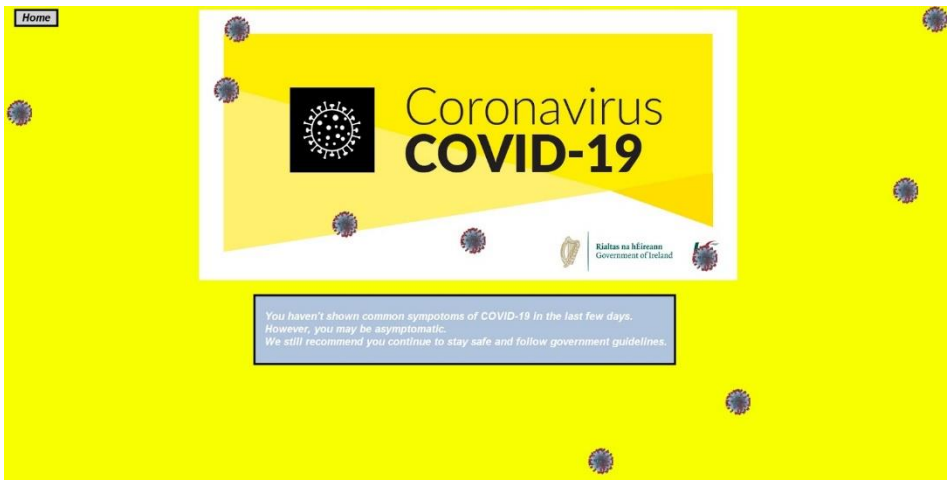

Fig 7a:

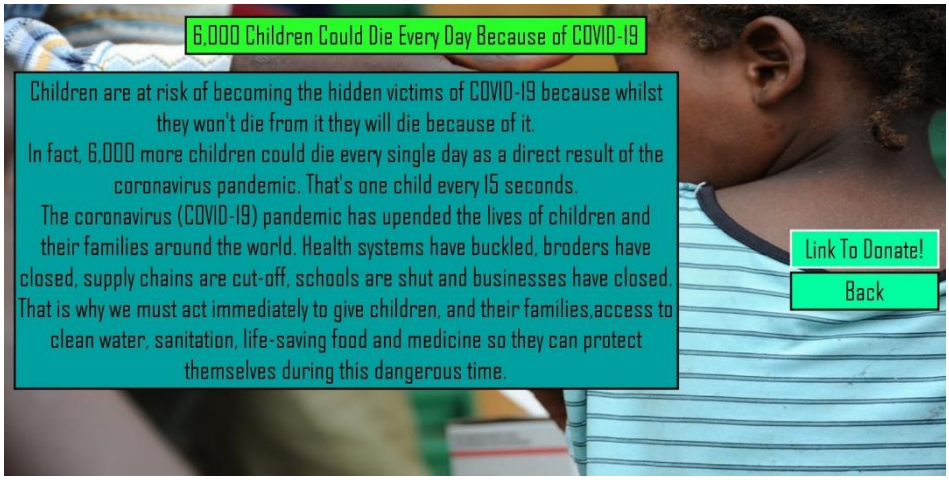

Fig 7b:

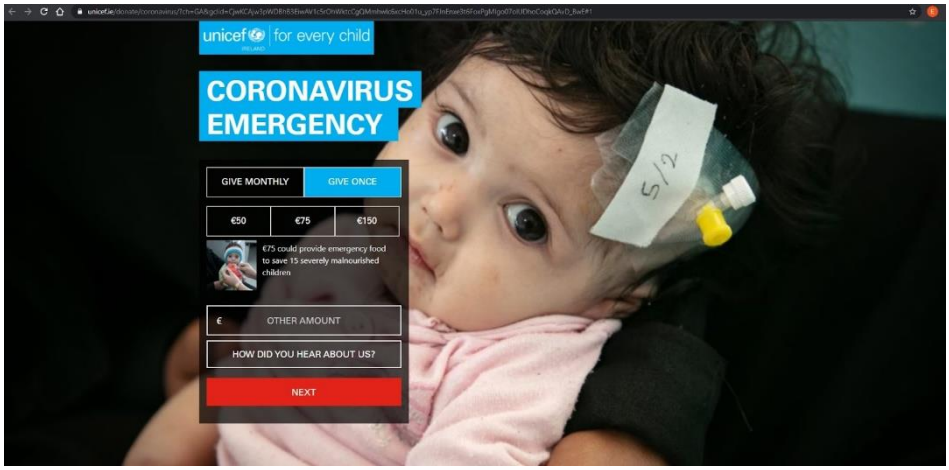

Fig 8:

Fig 9a:

Fig 9b:

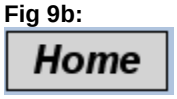

Fig 9c:

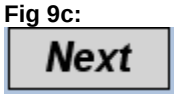

Fig 9d:

Fig 9e:

Fig 9f:

Fig 9g: