

Coronavirus: The Whole Picture

Data Consumption and why bigger is better

Muhammad H. Iqbal

Robert Gordon's College

A paper

presented to

The British Science Association

In fulfillment of the requirements of the

CREST Gold Award

Dec. 2020

Table of Contents

Table of Contents	2
Disclaimer	4
Introduction	5
Chapter 1: Analysis	6
1.1 Description of the problem:	6
1.2 Scope:	8
1.3 Boundaries:	8
1.4 Constraints:	9
1.5 Project Requirements:	10
1.5.1 End user requirements:	10
1.5.2 Functional requirements:	11
1.6 Resources required:	12
1.7 UML use case diagram:	13
Chapter 2: Design	14
2.1 Design Philosophy:	14
2.2 Site Wireframes:	15
2.3 UML class diagram for SIR program:	17
2.4 Generic Pseudocode for the GET function:	18
2.5 Generic Pseudocode for charting operations	18
2.6 Pseudo Media Queries:	20
Chapter 3: Implementation	22
Chapter 4: Testing	23
4.1 Log of ongoing testing:	23
4.2 Usability testing:	25
Usefulness Chart:	27
Navigation Chart:	28
Content Chart:	29
Total ratings Chart:	31
4.3 Compatibility Testing:	32
Chapter 5: Evaluation	33
5.1 Advantages and Disadvantages of JS frameworks:	33
5.2 Program Maintainability and Robustness:	35
5.3 Overall fitness for purpose:	36
5.4 Additional notes:	37
Conclusion	38

Resources	39
Acknowledgements	40

Disclaimer

The final product of this CREST Award is not officially licenced software, and I am the sole proprietor of it. I also state that I have no intent to monetise this application now or at any other point in the future.

The only information of the testers I have chosen to disclose are age and gender. I have chosen not to share anything else about them for the sake of their own personal privacy, as well as many other obvious reasons.

My mentor, Dr Bruce Scharlau of the University of Aberdeen, helped guide me on the overall look and design of the app as well as providing ways to evaluate its effectiveness as a solution, and there is no one else that I consulted when designing and developing it. All code written and used is a result of my own work. I, Hashim Iqbal, take full responsibility for the development and future maintenance of this application.

Introduction

I would like to start off by expressing my thanks to my mentor, Dr Bruce Scharlau. Bruce provided some really insightful guidance for the duration of my project, and it helped me understand my true goals during development.

Displaying lots of data while simultaneously maintaining a cohesive UI was the biggest challenge of this project, and that was mainly what my me and Bruce worked on to ensure a positive user experience. As well as this, there were a number of server-side processes that would mean getting the app onto the web that Bruce supported with.

Be sure to check out the Github repository I have set up for the project, which contains all the code I have worked on, as well as the shared GDrive folder, which contains more resources. The links to these will be found in the 'Resources' section of the report.

This project was really fun to work on. While there have been so many obstacles and roadblocks to overcome during development, I learned a lot of new practical skills, like deploying applications to a cloud service or using different libraries within JS, and some more of the general theory behind server-side algorithms and processes.

With the current climate, it is important to consider the full picture that data tries to show, instead of picking and choosing facts as one pleases. The general aim of this project was to present a more accurate depiction of how COVID-19 affected the world, and I would say that, for the most part, it has succeeded in that goal. Everybody is talking about data privacy, but I think data consumption matters now more than ever, and that statistical apps should be providing users with as much data as possible, rather than optimising for a better UI.



Chapter 1: Analysis

1.1 Description of the problem:

The importance of data management and processing has always been apparent, in any kind of industry. Nowadays, with the Coronavirus spreading all across the globe, the interest in consumer apps that display all kinds of information relating to a pandemic has never been higher. At the time of writing this, these apps and websites are already widely available, from a number of different developers. During the ‘research’ phase of this project, a discovery I made was that a lot of these available apps focus more on creating a fancy UI instead of processing other important data that users may want for convenience.

The most important information everyone wants to see, whether the statistics represent a particular country or the whole world, are three numbers: confirmed cases, recoveries and deaths. However, this doesn’t automatically negate the usefulness of other important data. Seeing these numbers alongside additional data points like population density, average age and testing stats helps elevate the project from an app that simply shows the virus’ impact to an app that helps the user understand how it affects civilisations. It is still possible to maintain good UI design without sacrificing important info.

With all this in mind, the main goal of the project was to develop a web solution that would display comprehensive global and country-by-country info relating to the virus, as well as an interactive map displaying hotspots for disease spread, through the use of various API calls in conjunction with libraries that would enable the implementation of UI elements like maps and charts (see ‘Resources’ section of the report for APIs used and library links). Separately, I spent some time during the research phase learning about the SIR model, and the calculus behind it, to aid in developing a Python algorithm that would model and graph the spread of any viral outbreak, specifying infection and removal rates.

A secondary research goal was to examine the advantages and disadvantages of using multiple different frameworks when implementing the solution. This allowed me to branch out and explore the syntax and theory of different frameworks in JS, while cataloguing what my preferred framework would be for the final product. Creating small prototypes to test the ease of API calls and adding UI elements helped make this decision. Eventually, I found

developing with no frameworks at all would be easiest, as the syntax was easiest to digest and understand for me. I will discuss more about the advantages and disadvantages of some frameworks in Chapter 5.

The third goal was to deploy the app to a cloud platform, so that test data could be acquired easily, and to learn how to prepare server-side operations for deployment to cloud platforms using Node.js.

1.2 Scope:

By the end of the project, the following had been delivered:

1. A full project plan, including details of tasks needing to be done, as well as timescales.
2. A completed design of the scripts with pseudocode, and wireframes to show what pages were intended to look like.
3. Fully functioning prototypes that made use of API calls and UI elements to display stats relating to the COVID-19 pandemic.
4. A fully functioning final product deployed to Heroku, making use of dependencies like chart.js and mapbox as well as API calls to display COVID-19 stats.
5. A fully functioning separate Python algorithm that would plot and display the change in the population for each bucket: S, I and R.
6. A comprehensive test plan, indicating the processes to test as well as details of the test persona.
7. The final results of testing.
8. An evaluative report.

1.3 Boundaries:

The project would include:

1. A system that would make API calls and fetch the data from them, formatting them into tables or assigning IDs to them.
2. A system that would use the Chart.js library to take the fetched data and draw up any kind of chart/graph (line, bar, pie etc.) with this data and appropriate headings.
3. A system that would use Mapbox to implement a digital map of the effect of the virus on the world, highlighting hotspots of activity.
4. A system that would create a server that would return the static files and be hosted on Heroku's server ports.
5. A separate system developed in Python that would use differential equations to plot and show the change of three lines, each representing one bucket of the SIR model.

1.4 Constraints:

1. There were no real time constraints, as I could spend as long as was needed to complete the project.
 - a. I needed to exceed 70 total hours of work to meet the requirements for a CREST Gold Award.
2. There was a list of languages used in development. These included HTML5, CSS3, JavaScript 1.8.5 and Python 3.8.0.
3. I chose to develop mainly using vanilla JavaScript because this would provide the most concise and easiest to digest and understand syntax.
4. Microsoft Visual Studio Code 1.5.1 was used as the editor for all development. This ran on my operating system of choice, and was completely free.
5. I needed to ensure that this project, all the resources in it, and the final report covered all the criteria specified in the profile form issued by CREST Awards.

1.5 Project Requirements:

1.5.1 End user requirements:

1. Users should start off at a home page, which details a little bit about what the purpose of the site is, and how to navigate it using the hamburger menu. Links to other pages will also be provided on the home page itself.
2. Users should be able to access the ‘global’ page, which shows global COVID-19 data, including total cases, total deaths, total recoveries, mortality rate and currently active cases. Alongside this, they should be able to see a pie chart displaying this data.
3. Users should be able to access the ‘countries’ page, which forms a breakdown of the statistics per country, which users can select using a drop down menu. A line chart showing the total cases over time should also be automatically generated to show the total cases in this country over time.
4. Users should be able to access the ‘interactive map’ page, which plots points on a digital map which, when clicked on, will display a pop-up that shows the data for that specific country. .
5. Users should be able to access the ‘SIR’ page, which gives information about SIR modelling, with associated images and an embedded Python script I authored to demonstrate the power of the predictive model.
6. Users should be able to access the ‘resources’ page, which lists all the resources used in the project, like APIs and libraries, with links to take them to the relevant documentation.

1.5.2 Functional requirements:

1. The 'global' page will make use of an API call through JavaScript to fetch the data, and then the mortality rate and currently active cases will be worked out using numerical operations. This info will then be displayed. Chart.js will be used to draw the pie chart.
2. The 'countries' page will make use of some JQuery (breaks the rules) and other vanilla JavaScript to fetch the data once a new country that doesn't have a value of 'default' is selected, and then generate a new line graph using this data and Chart.js. This is where other comprehensive data per country will also be displayed.
3. The 'maps' page will make use of 'mapbox' to initialise the digital map, and a marker will be placed at every country using their latitude and longitude. Code will be written to display a popup when these maps are clicked.
4. The 'SIR' page will contain images and a textual description of the model as well as how it works. A repl.it will also be embedded into the site, which contains my Python script for anybody to edit and run.

1.6 Resources required:

Analysis	<ul style="list-style-type: none">● Gantt Project● Google Chrome● Google Docs
Design	<ul style="list-style-type: none">● Lucidchart
Implementation	<ul style="list-style-type: none">● API resources● Necessary libraries● Visual Studio Code 1.5.1
Testing	<ul style="list-style-type: none">● API resources● Google Chrome● JavaScript testing sites● Necessary libraries● Visual Studio Code 1.5.1
Evaluation	<ul style="list-style-type: none">● Google Docs

1.7 UML use case diagram:

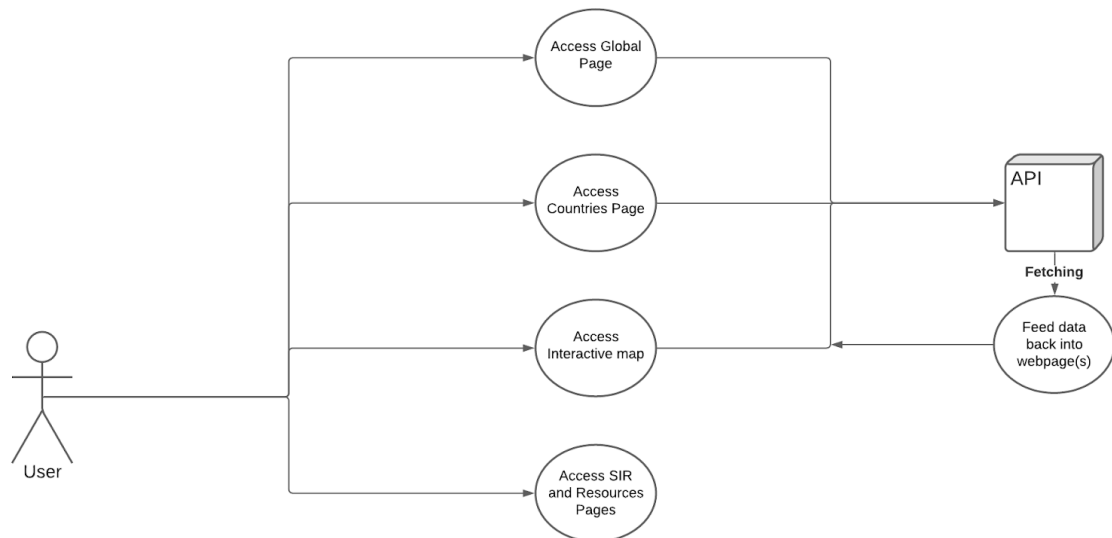


Figure 1

As can be seen, figure 1 represents the processes the user should be able to reasonably test and access once the website is finished, and the various integrations and communications with outside sources, like the APIs.

Each of these processes will be assessed and evaluated fully in the ‘Testing’ section of the report, which’ll help conclude whether the project overall is fit for purpose or not.

Chapter 2: Design

2.1 Design Philosophy:

In order to effectively convey a compact and friendly UI, I had to think of how I was going to display a significant amount of coronavirus data onto each webpage. The design would follow a kind of nested structure, where, starting off on the global page, the user would see a small amount of digestible data that hasn't been refined for a particular country of the user's choosing. Then, if the user wants more information, they descend further into the next by accessing the countries page, which breaks this data down by Country, and also provides non-coronavirus information, like population density, cardiovascular death rates and diabetes prevalence.

As mentioned earlier, the point of including this data is because some users may be looking to understand why the statistics are as high as they are for a particular country. A common use case for this larger amount of data is a user trying to find out why Italy was hit by the Coronavirus harder than a country like Pakistan was. From what we know about COVID-19, it affects the elderly very badly. According to the application, 16.24% of the Italian population is aged 70 or older. Only 2.78% of the Pakistani population is aged 70 or older. Therefore, the application has given us a reasonable argument as to why Italy has more deaths and infections as a result of COVID-19. Simultaneously, this still benefits the people quickly wanting to grab Coronavirus numbers and go.

This was just one example, so the potential benefit this could have for researchers and people trying to find out more information as to why the numbers are the way they are, a design philosophy that favours more data over less is important. A COVID-19 statistics app doesn't just need COVID-19 related numbers. If that were the case, we'd never be able to determine what kinds of people the virus affects, and other valuable information.

2.2 Site Wireframes:

The wireframes were generated using ‘Lucidchart’, a free online application that allows for drawing flowcharts, wireframe designs for websites and UML case diagrams. UI elements, like charts and maps, will be denoted by shapes like circles and squares. The ‘x’s denote fetched information from the API to be displayed. Other appropriate annotations are made to demonstrate what each element is responsible for. Wireframes for ‘global’, ‘countries’ and ‘map’ have been made, as the ‘resources’ and ‘sir’ pages had simple designs that didn’t include UI elements or fetched data, just text and images.

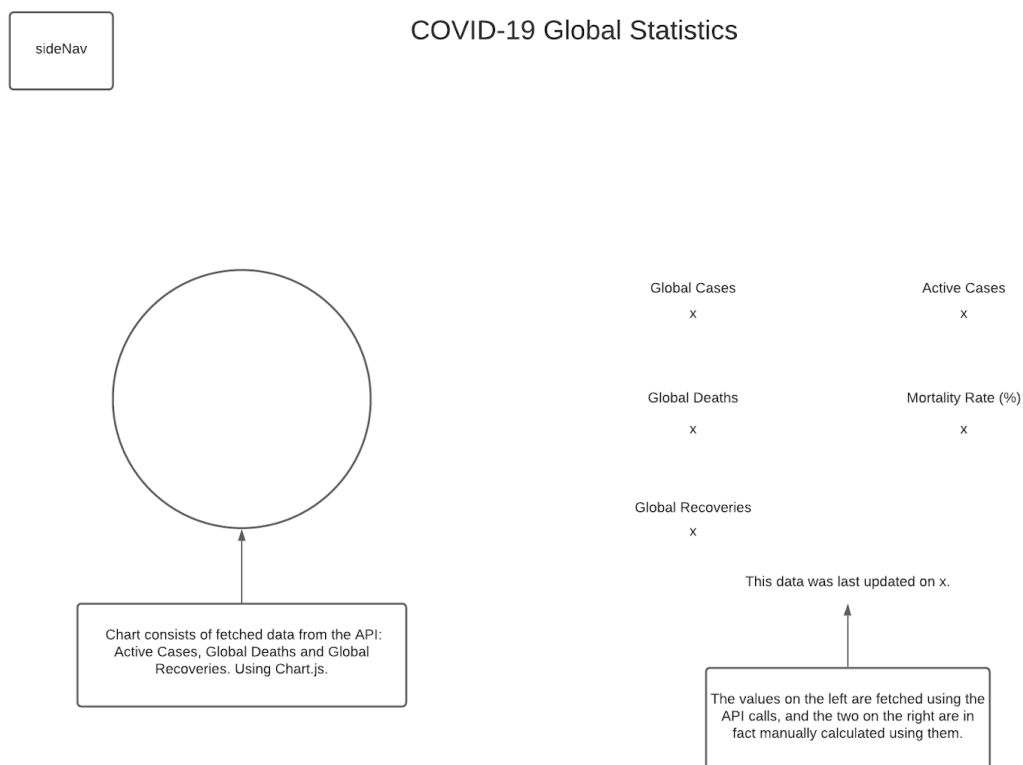


Figure 2a

COVID-19 Countries Information

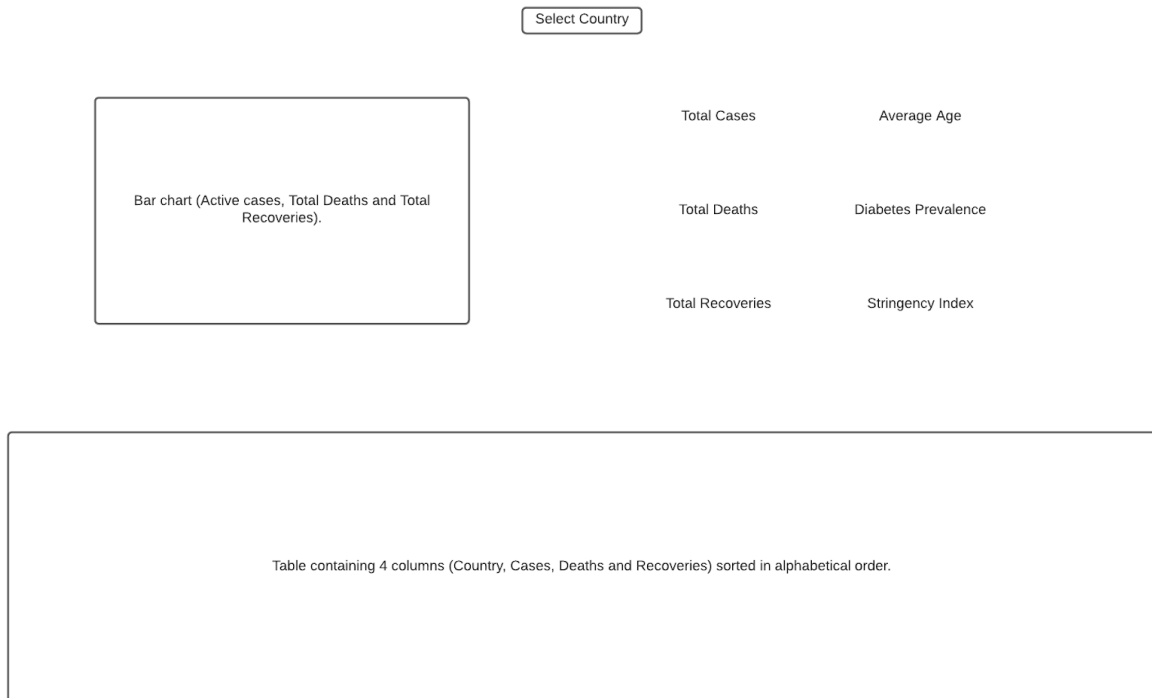


Figure 2b

COVID-19 Interactive hotspot map

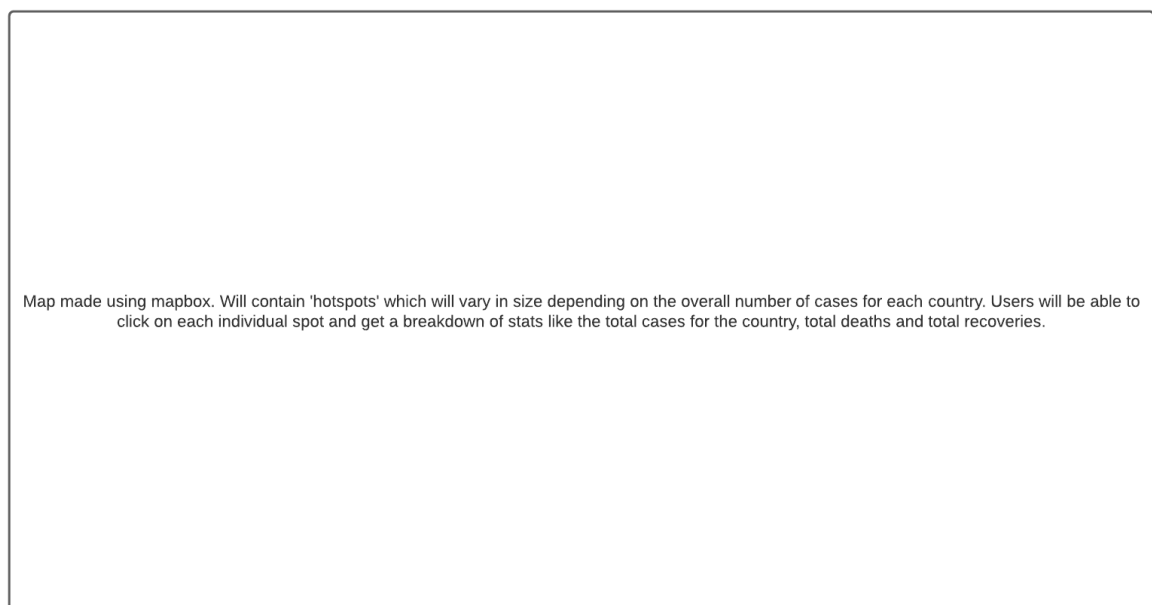


Figure 2c

2.3 UML class diagram for SIR program:

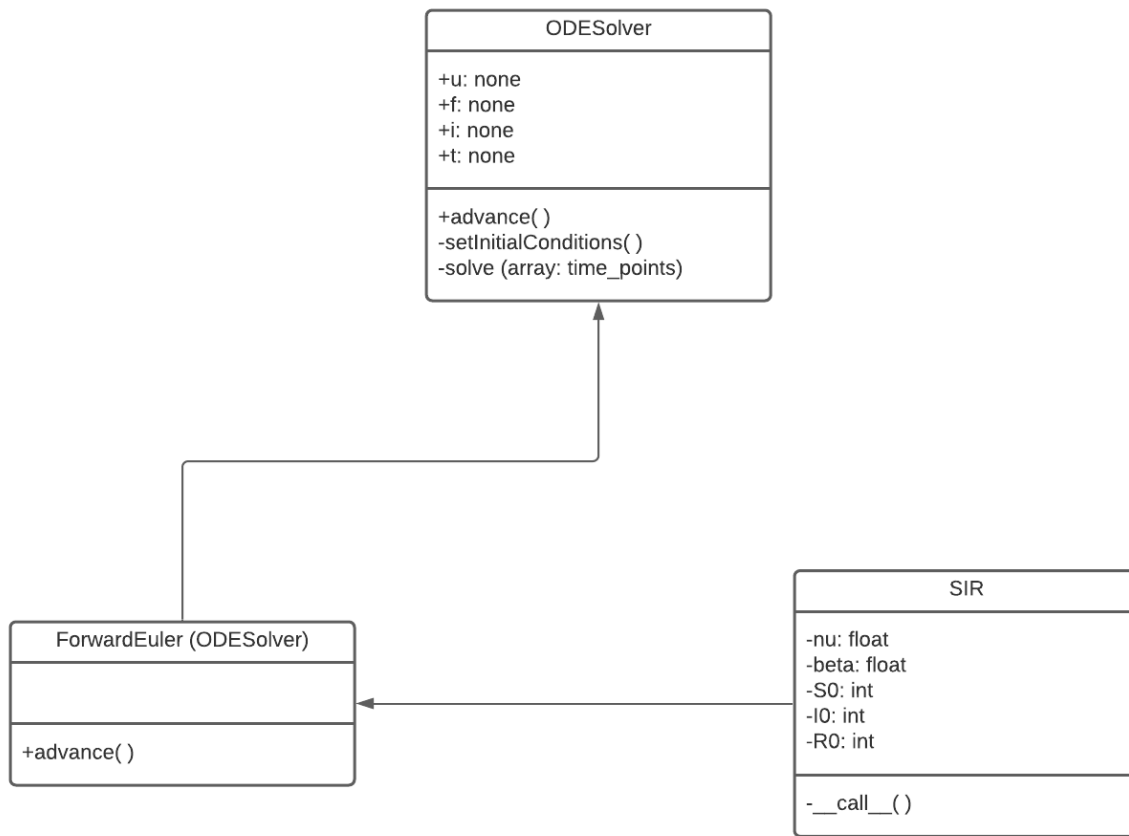


Figure 3

This UML class diagram was also made using lucidchart.

The arrows denote inheritance i.e. if there is an arrow going from ForwardEuler to ODESolver, that means that ForwardEuler inherits attributes and operations from it. Since ForwardEuler is a class imported into the 'sir' program, the class SIR will have an arrow pointed to ForwardEuler. The pluses and minuses denote public and private functions/attributes. A private function or attribute will exist only in one class, and can only be altered within the context of that class. A public function or attribute can exist in many classes, and can easily be altered in this way.

2.4 Generic Pseudocode for the GET function:

```
async function fetchData() {  
  let variable url = 'https://api.covid19api.com/summary'  
  fetch response from url  
  let variable data = response  
  
  return data to console  
  
end function  
}
```

Figure 4a

In Figure 4a, the purpose of the ‘async’ next to the function means that it will be asynchronous with all other functions written in the same script. What this means is that the order in which functions are executed can be manually defined. Simply, the function instructs to fetch a response from the URL, and store this response in a variable called ‘data’ where further operations like separating statistics by ID can be performed.

2.5 Generic Pseudocode for charting operations

```
async function chartInit() {  
  await fetchData()  
  
  let variable chart = new chart {  
    type: 'line/bar/pie/doughnut/radar/polarArea'  
    data: data  
    labels: [  
      ...  
      ...  
      ...  
    ]  
  }  
}
```

Figure 4b

In Figure 4b, the function draws a chart of any type as specified at line 15. This function is also executed asynchronously, and line 12 tells it that it can only execute the rest of the function if `fetchData()` has finished executing, allowing for it to plot the data received from that function.

```
<script src = '../JavaScript/x.js'>
<script src =
'https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.min.js'>

<script>
fetchData()
chartInit()
</script>
```

Figure 4c

It should also be noted that, as well as linking to the external file that holds these functions, it would also be imperative to link to the `chart.js` CDN as well, if the `chartInit()` function is to work. After that, the `<script>` tags can be used in the HTML file to call the functions, as shown above. Since the order of execution has already been defined in the external file, it doesn't matter what order they are called inside of the `<script>` tags.

2.6 Pseudo Media Queries:

The purpose of a media query is to change the outline of how a webpage looks depending on two factors: the size of the screen, or if the webpage is being printed out. As an example, if a webpage was being printed out for whatever reason, you may want to hide particular elements like navbars or background graphics. Alternatively, if a webpage was being viewed on a smaller device, like a phone or tablet, the elements on a page would need to be shrunk down to accommodate for the limited space.

```
/* Generally accepted screen size for mobile phones portrait or  
landscape */  
@media screen and (max-device-width: 640px){  
    {...}  
}
```

Figure 5a

```
/* Generally accepted screen size for tablets portrait or  
landscape */  
@media screen and (min-device-width: 768px) and (max-device-width:  
1024px) {  
    {...}  
}
```

Figure 5b

Figure 5a would contain the styles for phones either in portrait or landscape mode, shrinking down divs and other elements in order for them to fit comfortably on the screen. Figure 5b would do the exact same, but for tablets in portrait or landscape modes.

```
@media print {  
    {...}  
}
```

Figure 5c

Additionally, a media query for printers was also implemented in the unlikely event that the webpage would be printed out. The elements and divs were shrunk down to account for A4 page size, and the navbar element was hidden, as it would be unnecessary and meaningless on the page.

Chapter 3: Implementation

The website was fully implemented and deployed to Heroku. Evidence of the completed scripts and files can be found at the Github repository that was set up for this particular project (<https://github.com/Haz-ctrl/COVID-19-Info-Tracker>).

The website is live. It can be found here: <https://coronavirus-stats-app.herokuapp.com/>

Chapter 4: Testing

4.1 Log of ongoing testing:

As the data on the attached spreadsheet shows (can also be found in the ‘Resources’ section), there were various errors I encountered during individually testing the application. A few screenshots of particular tests where I had to squash some bugs are shown below.

Problem Description	Resolved?	Solution
Country drop down list doesn't return weather information.	<input checked="" type="checkbox"/>	Resorting to using JQuery to change the page properties based on the drop down selection. While this breaks my own rules, it's the only way to implement the site correctly.

Figure 6

As can be seen with this particular error, I had to use JQuery to resolve it. At the beginning of this project, I chose to develop my final product without using frameworks like JQuery, as their syntax and code was harder to understand and implement personally. However, this proves me wrong. I will go into further detail in chapter 5, but the solution to one of the biggest bugs in this project turned out to be the one I was running from. From this, I was able to implement a working drop down that refreshed everything on the page each time a different option was selected, all with the power of this framework.


The sidenav doesn't feel like a clickable image. This can confuse a lot of people trying to figure out how they should go about navigating the page.		Changed the element from to <button> which has a clickable feeling. The background colour is white so that it pops out more and is obvious to the user.
--	---	--

Figure 7

This error probably wouldn't seem to be as game-breaking as the previous one, but this error goes to show how smaller changes also help users. When starting with usability testing, I observed that a lot of users I was overlooking didn't immediately know to click on the hamburger menu, and assumed it was an image, because the mouse didn't show a clickable feeling. Using a very easy solution to change this, every user from then on knew where to click in order to access the other pages in the site. Every small change, even if it's just for convenience rather than operation, matters.

For example, I plan to implement a caching system for the website in the future. This caching system will cache the HTTP requests I'm making, and

4.2 Usability testing:

Apart from testing the website myself, in order to evaluate the friendliness of the UI, I needed to get others to test it for me, so that their experiences could determine how the website functioned as a user-driven experience. With little input from myself, other than telling them what I wanted them to do on the site, they were able to work out how the navigation system worked, and were able to access and view the different content in the pages easily. Below is a small sample of the testers details, to show how varied they are in demographic:

User ID	Age	Gender
001	17	Male
002	17	Male
003	61	Male
004	35	Male
005	20	Female
006	22	Female
007	18	Male
008	47	Female

Figure 8

This shows the vast difference between my testers, as they're of all ages and backgrounds. This will help to illustrate how the app doesn't appeal to one audience, but appeals to any user trying to find coronavirus information, unlike apps today that range from bare in terms of data to overcomplicated and difficult to figure out.

Below are a few samples of tests run with users, and the ideas put forward by them that then influenced my redesign of the website, to fit their needs:

Comments	Potential Bugs or issues resolved?
Could make numbers on the global page larger.	<input checked="" type="checkbox"/>
Mercator projection could be a problem. Make sure to consult the mapbox documentation.	<input checked="" type="checkbox"/>
Could make numbers on the countries page larger.	<input checked="" type="checkbox"/>
Maybe fix the graphs to show time in real units rather than something arbitrary.	<input checked="" type="checkbox"/>
N/A	<input checked="" type="checkbox"/>
Could make the red dots on the maps page a little smaller. Could also add different coloured points that are clickable at the turning points of each graph.	<input checked="" type="checkbox"/>

Figure 9

After these issues or suggestions were identified by users, I redesigned parts of the website to fit these needs. Testing through users was really helpful as I was able to see through the perspective of someone who was seeing this website for the very first time, and hadn't spent hours developing it before. The suggestions brought forward by users were understandable and contributed to the overall experience of the website.

So, overall, what did the users think of the website? I posed 4 questions to them at the end of every test run:

1. Out of 5, how useful do you think a website with this design philosophy of more detailed data is to you?
2. Out of 5, how easy did you find it to navigate around the website's different pages?
3. Out of 5, how effectively would you say that the website's content is laid out?
4. Out of 5, how would you rate this website overall?

After obtaining these ratings from each user, I created pie charts for each of the 4 sections which would show how effective the website was in conveying an effective and friendly UI. The pie charts are on the following page:

Usefulness Chart:

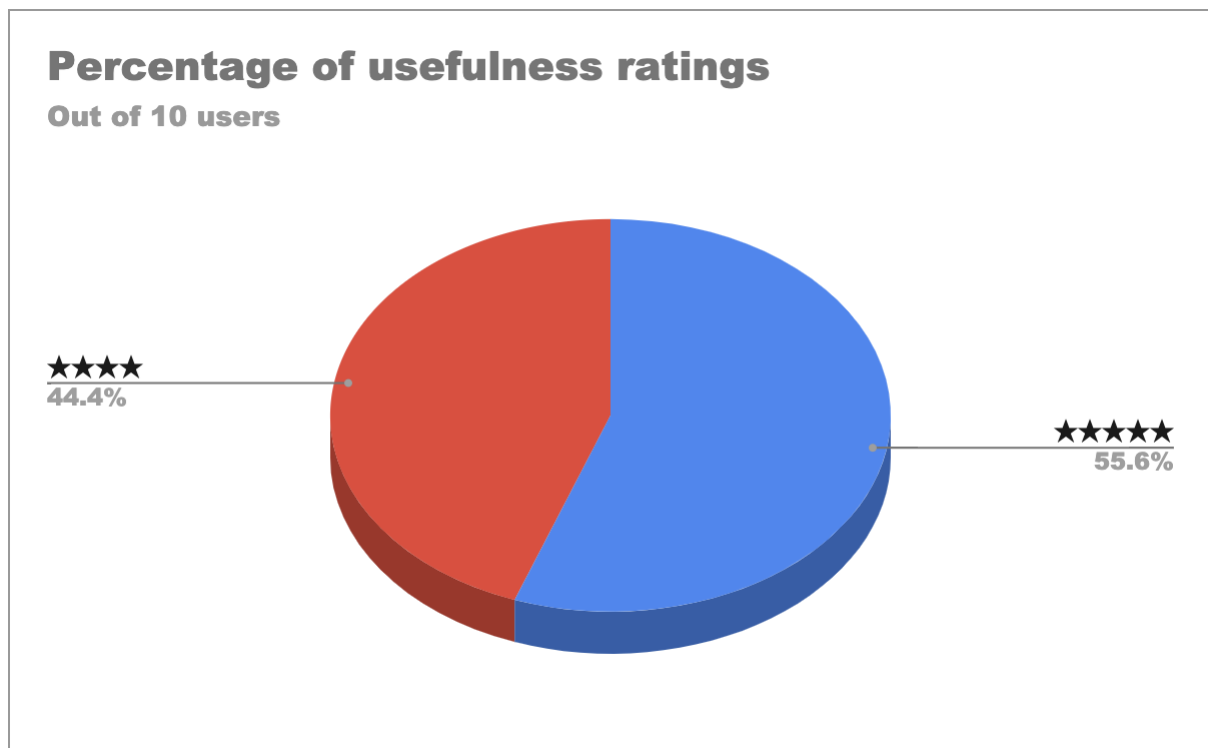


Figure 10

Overall, no user gave 3/5 or below for this section, which shows that users appreciate the greater emphasis on comprehensive data with this application. A majority of users in the study thought this approach was excellent and conveyed the gravity of the pandemic quite effectively. The minority of users still thought that this approach was good, but could use some tuning, as there were some complaints regarding the surprisingly sparse number of countries to choose from on the countries page. However, the arrangement of the data set is what caused faults such as these.

Navigation Chart:

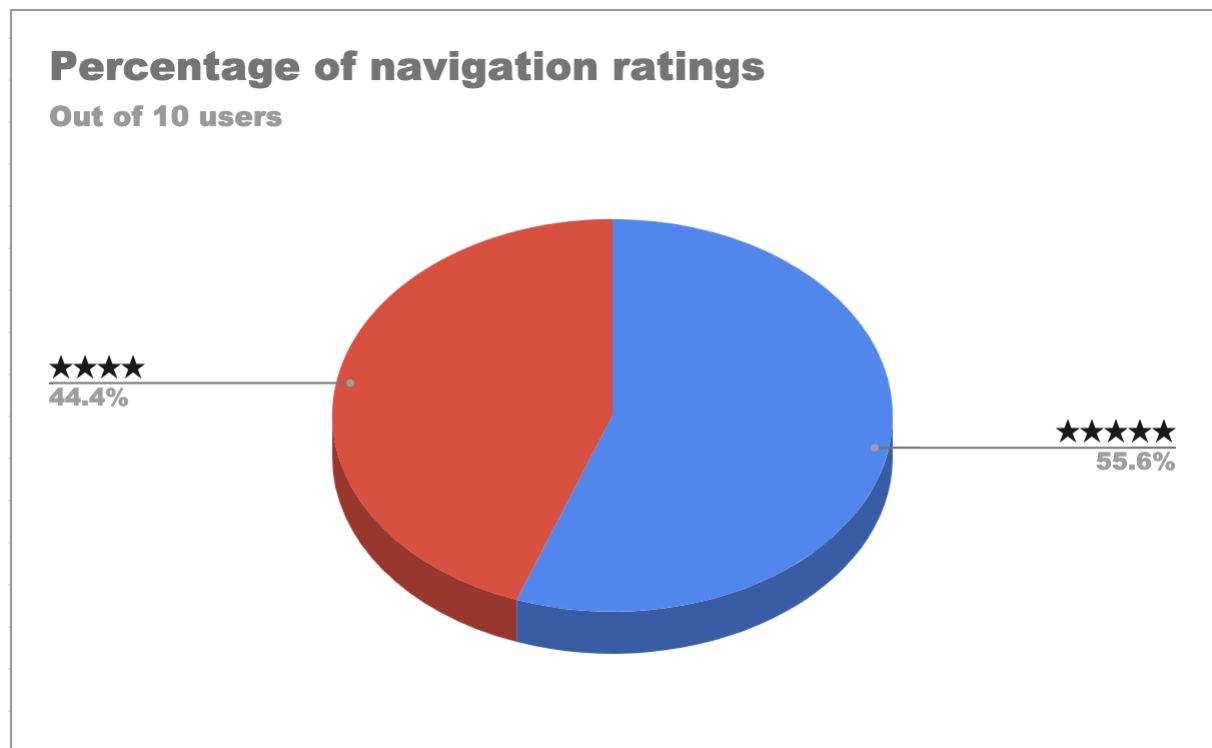


Figure 11

Most users really liked the setup of the navigation structure of the website. Others gave lower ratings due to there only being a hamburger menu, and not much else to literally tell users what to click. This was a very good point, so I chose to create a homepage which is always reached first using the website link. The homepage tells the user very briefly what the website is about, and how they can navigate it using the hamburger menu. It also included direct links to other pages as well, and other pages had links back to the home page.

Content Chart:

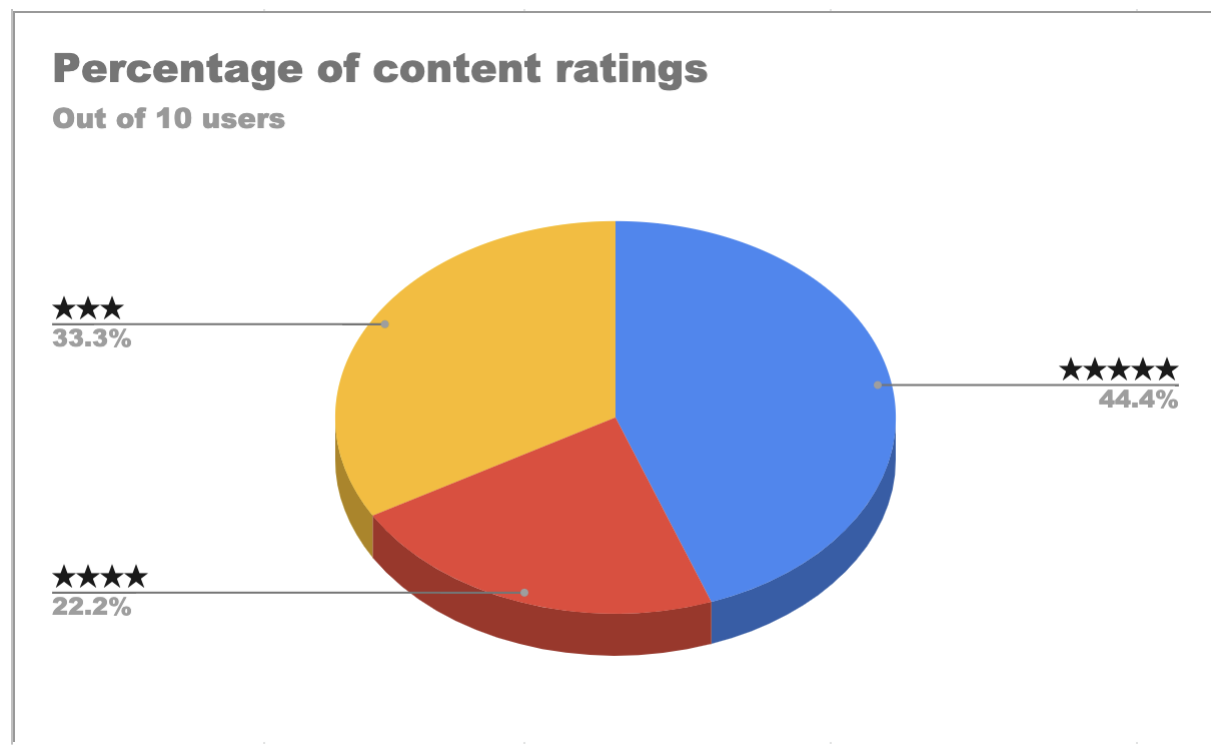


Figure 12

The content ratings were a little more divided among users, and understandably so. Users giving lower scores commented that the content being displayed was fine, it's just that the format it was displayed in probably needed to be changed. This is also an understandable result, as users tend to have their own preferences at the end of the day, and it's not actually a problem between keyboard and chair. This also shows the many different demographics devs have to appeal to, which can be frustrating and leaves others alienated from your app.

Perhaps the introduction of more frameworks can help this app achieve a state that appeals to most users. More on this in the Evaluation and Conclusion, but it looks like the way I lay my content out could be improved in future.

Total ratings Chart:

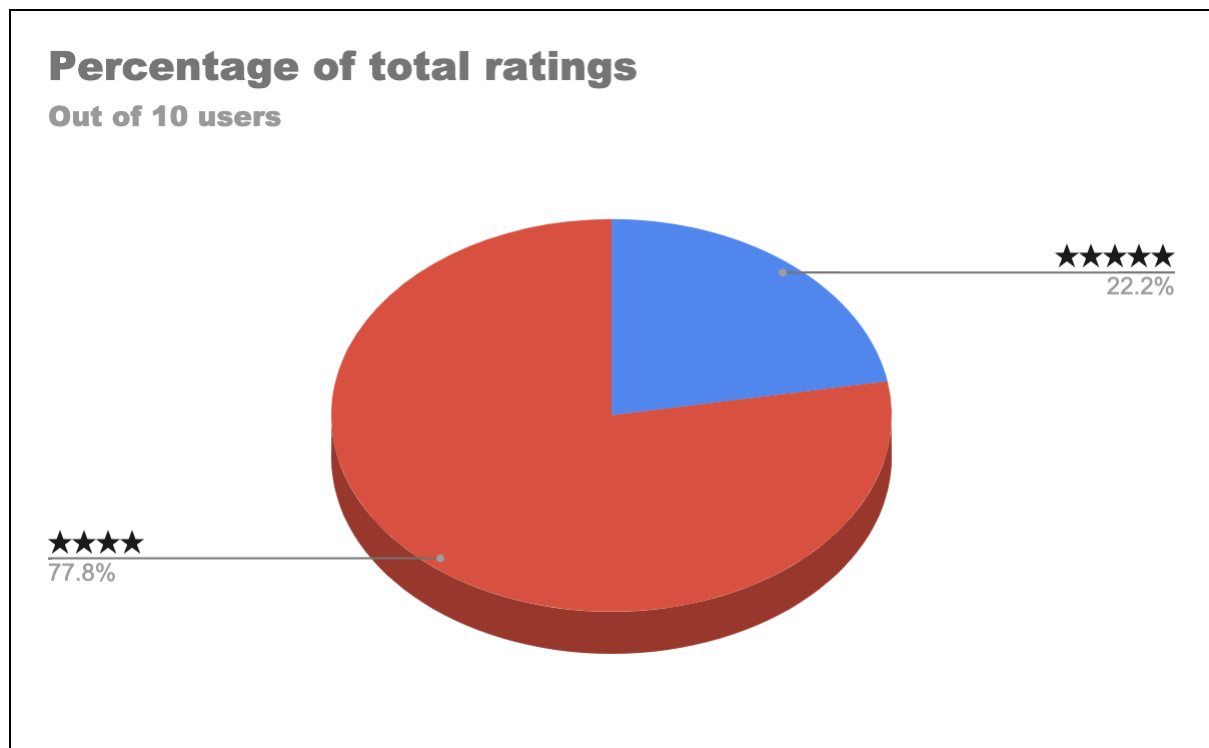


Figure 13

As can be seen, all users rated the app a 4-5 overall. This is a very good result, as it shows that the application has been an enjoyable experience for most users, and they resonate with it a lot. There are definitely a few things that can and should be changed, which was gathered from users during testing, and from a few things I noticed myself from individual testing. There will definitely be a longer term maintenance cycle for this project in the future, but, for now, these results are very good, and have proven me in my aim to develop a comfortable user experience while using a lot more data.

4.3 Compatibility Testing:

In order to determine what devices and what browsers the website was able to run on, I tested how it functioned on each. Below are the final results of my compatibility testing:

Operating System	Google Chrome V80.0.3987	FireFox V73.0	Safari 14	Opera 72	Internet Explorer 11	Edge V87.0.664.47
MacOS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
iOS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Windows	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Android	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

As can be seen, the website runs on most devices and most browsers. The browser that provides least support is, understandably, the discontinued Internet Explorer. It is also worth noting that, on Windows and Android machines, the font of the web-page isn't as it normally appears on Apple systems. This is because the font for the whole project was set to 'Apple System', which isn't a recognised font on Windows or Android systems, so the font is defaulted to be the next in the chain of fonts specified in the CSS.

Chapter 5: Evaluation

5.1 Advantages and Disadvantages of JS frameworks:

One of the goals of this project was to evaluate a number of different frameworks used during the course of development time in the project. In order to assess the effectiveness of each one, and making a final decision on what methodology was used for the final product, a series of ‘prototypes’ were made using the different frameworks, and advantages and disadvantages were noted.

The prototypes weren’t deployed, and as a result, only the source code can be seen through the github repository. Many of the points made can be evidenced by looking through this source code, and my experiences with writing them. It should be noted that these are my brief, yet well documented experiences with these frameworks. All points made are based on my understanding and experience as a newcomer to these frameworks with a fresh perspective.

There is no such thing as a ‘perfect development framework’, as it purely depends on what goals and specifications you are aiming to meet. For certain

The following table explains the differences, and unique pros and cons for each of the three development frameworks: regular JS, React.js and JQuery:

Framework:	Advantages:	Disadvantages:
React	<ul style="list-style-type: none"> • The best framework to use when implementing UI elements into a website. • Writing code in separate .jsx files for each UI element means a more agile approach to development, so that components can be implemented individually. • The favourable option when building complex stacked applications, which make use of server-side processes and databases. 	<ul style="list-style-type: none"> • An app using the React framework is usually built using the terminal, not a CDN. • This can then mean that a lot of materials are already pre-provisioned by the time you start implementation. • The syntax and overall nature of writing a React app can be seen as overly complicated, and not very beginner-friendly.
JQuery	<ul style="list-style-type: none"> • Very easy to reference. The CDN can be linked to very easily inside of the 'script' tags. • Best framework for handling HTTP requests. The syntax is greatly simplified. While React is UI oriented, JQuery is requests oriented. 	<ul style="list-style-type: none"> • Very old framework that is losing relevance. This would mean that devs won't want to maintain it anymore. • Arguments that requests in JQuery actually involve stricter rate-limiting, which can be a problem. This is a very controversial, long-running argument.
Vanilla JS	<ul style="list-style-type: none"> • The syntax for HTTP requests is more extensive, but can be broken down and understood easier. • No reliance on CDNs or other links that may likely fail. 	<ul style="list-style-type: none"> • Implementing UI elements and requests is still possible, but requires more code. • Developers like framework oriented implementation because it's more efficient for certain tasks. Vue.js, Angular.js, Express.js and React.js are currently popular.

5.2 Program Maintainability and Robustness:

There are a number of factors to consider when the program's maintainability comes into consideration, and the list of potential weak points goes as follows:

1. CDNs such as Chart.js and Mapbox are used to externally link to these libraries and unlock their features. Linking to libraries versus downloading libraries is a major discussion within the development community. However improbable it may be, there is a chance that links can fail; the library's latest update causes some problems or the entire thing just becomes obsolete.
 - a. In this case, the link fails and the elements provided by the library fail to load.
 - b. The method of linking to libraries is the most efficient, and takes up less storage rather than downloading the entire library, but puts a lot of faith in the people who manage said library.
 - c. However, downloading a file onto a home media server or hard drive would not only mean that there are strenuous storage requirements, but future updates to the library never manifest into the app, as the file has become an outdated copy.
2. The API calls cause some issues in the future.
 - a. With something like the Coronavirus, APIs will eventually cease all maintenance and updates. Data for cases and deaths will hopefully dissipate eventually into meaningless data that no one is interested in.
 - b. From a health aspect, this is a good thing, but in terms of programming, this app will become worthless at some point in the future.
 - c. To that end, my plan for this app's future is to make it an application showing the historic effect that Coronavirus had on the world, with elements like line charts and timescales becoming more important.
 - d. Users will be able to appreciate the comprehensive overview of statistics as well as the growth and effect of the virus over time.
 - e. There are still some aspects of this project that are useful in any sense, like the ODESolver and SIR modeller written in Python, which help show the modelling and mathematics behind any disease, not just COVID.

5.3 Overall fitness for purpose:

Overall, the application designed was fit for purpose, as the test data acquired from the persona showed that the UI was friendly and intuitive to users of all kinds. The choice to include a more comprehensive analysis of data also resonated with researchers and other professionals who wanted to see more detailed information for a particular country quickly.

However, a major point to note was that I broke my own rules. In order to get the value from the drop down list selected on the countries page, I resorted to using JQuery, because the solution was easier to implement.

What this shows is that not every requirement in the world will be satisfied without using a module, library or framework of some kind on top of your code. Even this project, where the aim was just to use vanilla javascript, still required the assistance of one. For all the advantages and disadvantages of all the various JavaScript frameworks, there will come a time in any application where the help of at least one is required. Later down the line, the introduction of frameworks could make this app so much more than it is now.

The third aim was fulfilled, as the website was deployed to Heroku. However, there are some problems encountered regarding this that will be covered in 5.4.

Finally, how did this app stack up compared to others? The test data speaks for itself: users don't want an app that gives brief data only, they want an app that has something for everyone. People who want quick data can easily get it, and people who want a detailed breakdown can also have that too. Personally, I think a design philosophy such as the one adopted for this project would mean that the world, and developers specifically, can start to understand what users truly want, and that they can appeal to all users instead of just one demographic.

5.4 Additional notes:

Firstly, the deployment to Heroku causes some lag when the website loads. According to the website, the ‘dynos’ that run the site occasionally hibernate during periods of inactivity. When this happens, it can take some time waking them up again, explaining the noticeable load. Unfortunately, not much can be done regarding this, since an add on is required to schedule when dynos fall asleep, and potential costs arise from this.

Another thing worth mentioning is the lag that the ‘Countries’ page will sometimes experience, and I predict this is probably to do with accessing the JSON data source defined. Since it is ‘comprehensive’ data for each country, it is a fairly large file of ~50MB. Accessing and crawling through a file this substantial will sometimes cause a minor delay from when the user submits their input to receiving the output. A cache for the HTTP request could be set up, but more dependencies, knowledge and time would be required to do this. While this cannot be done currently, I’ve decided that I may implement a cache system for this app in the future.

Conclusion

In conclusion, was this project successful? Yes. Was it fully implemented the way I originally intended? No. Six months ago, I started with three numbers on a white screen and nothing else, just because I needed something to alleviate my lockdown boredom. After branching out to discover new libraries and APIs, I knew the potential for something like this was far bigger than I knew.

During the testing phase, a user told me that he wouldn't give me 5 stars for anything, because one app doesn't fit every user. I realise now that he was right, because every user has their own preference, which makes a developer's job impossible. Some users want a widget on their home screen that will just display global cases, deaths and recoveries. Other users want a fully fledged app that gives a detailed analysis and breakdown of the virus for each country.

The original reason I started this project was to provide an application that gives users more freedom and possibility with data. Any user can be happy with this app, because it appeals to both sides who want different things. Those who want quick, easy data and those who want something bigger will find something to enjoy in this.

So, to round off this paper, I want to say that the world needs to start thinking about optimising for everyone, not just someone. Thank you for reading, I hope you enjoyed it!

-Hashim

Resources

The website is live. See it here - <https://coronavirus-stats-app.herokuapp.com/>

GDrive Folder (Full of project resources) - <https://bit.ly/3pnKvmx>

Youtube Playlist (Code and Concept walkthroughs) - <https://bit.ly/35udSLl>

Acknowledgements

Heroku Dev Center (Plenty of Documentation to get started) - <https://devcenter.heroku.com/>

Chart.js Documentation - <https://www.chartjs.org/docs/latest/>

Chart.js Github (Contribute your ideas!) - <https://github.com/chartjs/Chart.js>

Mapbox Documentation - <https://docs.mapbox.com>

Mapbox Github (Contribute your ideas!) - <https://github.com/mapbox/mapbox-gl-js>

JQuery Documentation - <https://api.jquery.com/>

JQuery Github (Contribute your ideas!) - <https://github.com/jquery/jquery>

SIR modelling source:

https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology#The_SIR_model

Global Page API - <https://api.covid19api.com/summary>

Countries Page API - <https://covid.ourworldindata.org/data/owid-covid-data.json>

Maps Page API - <https://www.trackcorona.live/api/countries>

Johns Hopkins University for sourcing all data:

<https://github.com/CSSEGISandData/COVID-19>