Project Plan

Team Member Responsibilities

Our intention with this project is to work on all aspects of the project concurrently. We did this to prevent falling behind on later deliverables, especially since they appear to have more work in them. Below is a table that describes how we have initially split up the project.

Scraper (section 1.1)	Carlin
API (section 1.2)	Lucas/Sean
Swagger (section 1.3)	Lucas/Sean
Website (section 1.4)	Marcus/Adam

However, we understand that as the project progresses the areas that need the most work will change. Hence the above separation is flexible, and will change to cater to the needs of the project.

Meeting Plan

We plan to have two meetings a week, one in person meeting on Mondays before our mentoring meeting, and one over discord on fridays.

We have decided with this schedule as it will emulate the standup meetings implemented in agile development. As we are all university students, we cannot work full time on this project. So instead of the traditional daily standup structure we have stretched it out to a meeting every 3 days on average, thus holding the meeting 2 times a week.

Also, the monday meeting is positioned before the mentor meeting. This timing is good for multiple reasons. It guarantees that we will all be present, as we are all required to go to the meeting. Also, it allows us to debrief before the meeting about our past work, and come up with any questions we want to ask our mentor, making sure to get the most out of our mentors time.

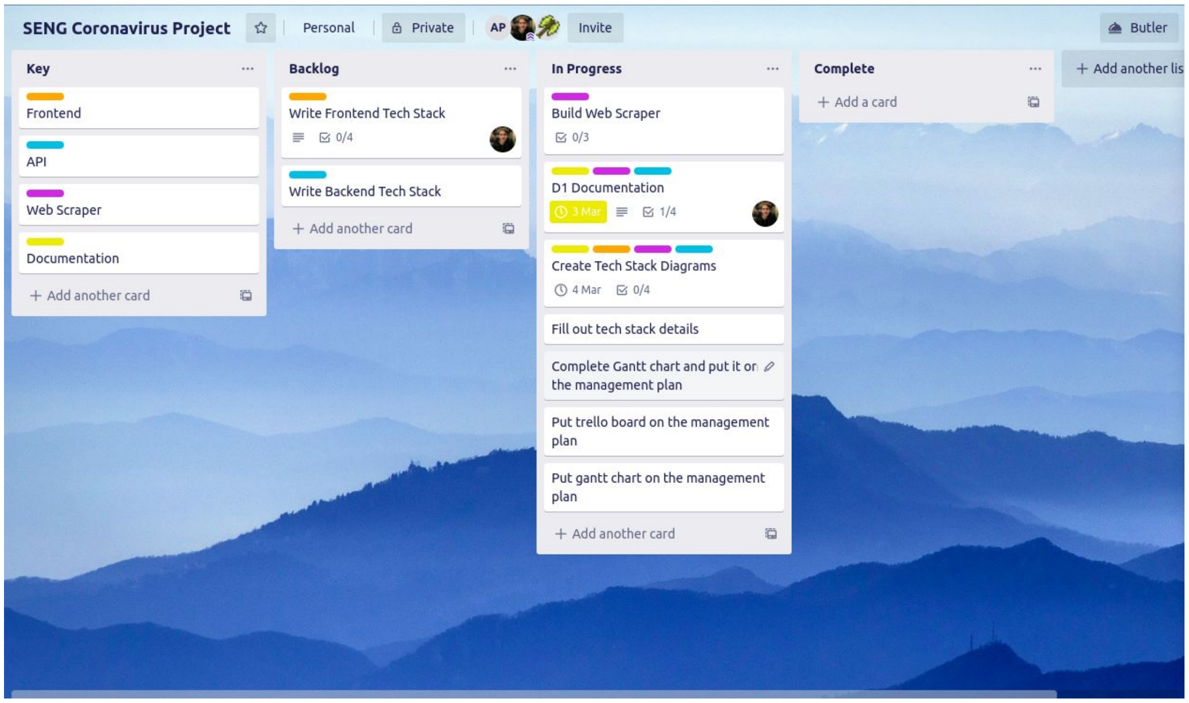
Management Tools

Trello

Trello allows us to break down the project into smaller chunks. We can then categorise those chunks based upon what area of work they are in. It also allows us to assign tasks to certain people, and to keep track of what work is done and what work needs to be done.

We will use Trello to help manage our project in the above way, helping to make working on the project easy, as you just pick up new work from the to do section. Using Trello also removes the need for a project manager, freeing up one of our team members to work on the project.

Link: https://trello.com/b/Cq9fvAmt/seng-coronavirus-project



SENG Coronavirus Project | Personal | Private | AP | Invite | Butler

Key
- Frontend
- API
- Web Scraper
- Documentation
- + Add another card

Backlog
- Write Frontend Tech Stack — 0/4
- Write Backend Tech Stack
- + Add another card

In Progress
- Build Web Scraper — 0/3
- D1 Documentation — 3 Mar — 1/4
- Create Tech Stack Diagrams — 4 Mar — 0/4
- Fill out tech stack details
- Complete Gantt chart and put it on the management plan
- Put trello board on the management plan
- Put gantt chart on the management plan
- + Add another card

Complete
- + Add a card

+ Add another lis

Shared Calendar

A shared calendar will be used so that all members of the team understand when meetings will be held, and when deliverables are due. It also allows us to create our own deadlines internally for organisation and project management.

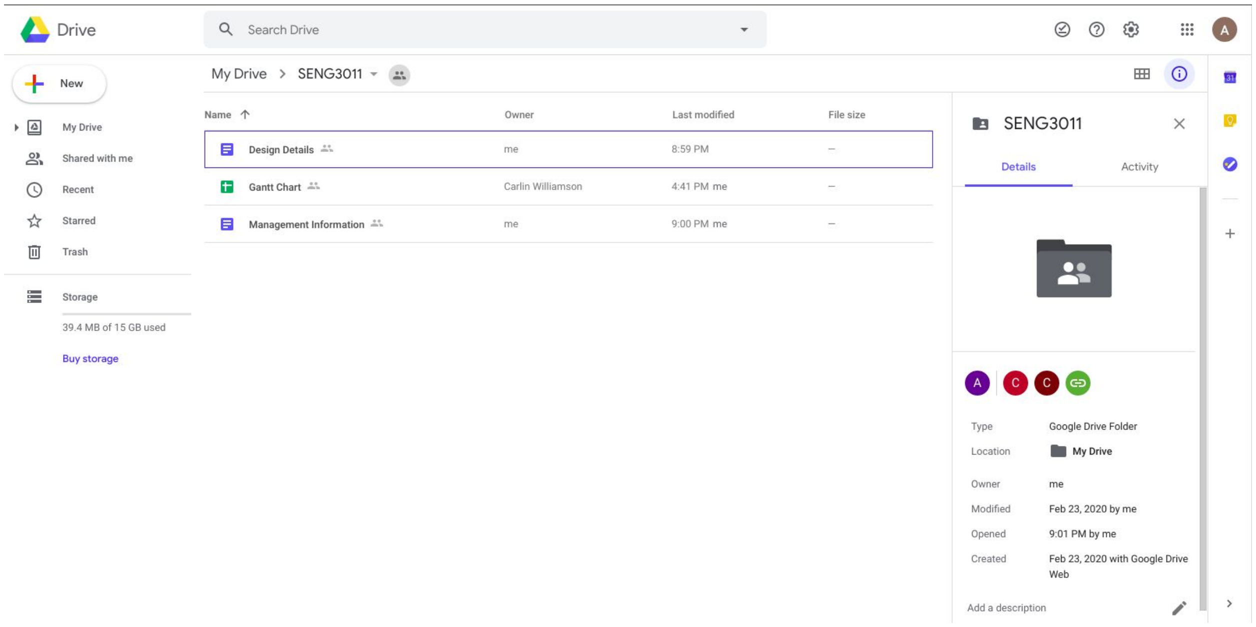
Discord

Using a discord server, we are able to create multiple channels for various discussions. These channels facilitate organisation and reduce confusion in our communication channels. We are also able to hold meetings on discord remotely. By holding the meetings remotely, we will be able to be flexible with the times and fit them into our chaotic university schedules.

Google Drive/Docs

Google drive will serve as our shared repository for documents. As some of us are developing on linux environments, we do not have access to Microsoft Word, hence we would not be able to work on a word document that is stored within the GitHub repository. Google Drive and Docs solves this issue due to its cross platform deployment, being on the web.

It also allows for collaborative work concurrently, meaning there is never any conflicts in the reports. The only issue here is that we are unable to include the google doc in the repository, but to solve this we will export the reports as PDFs and upload them into the repository.



Name	Owner	Last modified	File size
Design Details	me	8:59 PM	–
Gantt Chart	Carlin Williamson	4:41 PM me	–
Management Information	me	9:00 PM me	–

Type	Google Drive Folder
Location	My Drive
Owner	me
Modified	Feb 23, 2020 by me
Opened	9:01 PM by me
Created	Feb 23, 2020 with Google Drive Web

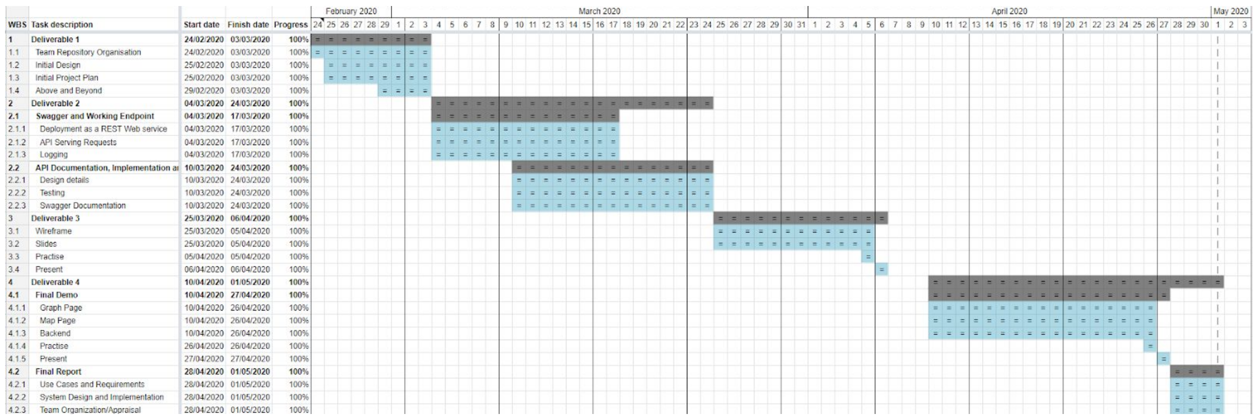
link: https://drive.google.com/open?id=17CZm5GeVSLflXhvoYO0LegdQZjU5O20Y

GitHub

We created a github repository to allow us to work simultaneously on the project. This allows us to all work individually, increasing overall productivity. GitHub, through git, also allows for version control, allowing us to roll back to a stable version should we make a mistake.
link: https://github.com/z5122506/SENG3011-calmClams

Gantt Chart

To keep track of the general progression of the project, helping us to understand both what needs to be done but also how we are going with progress on the project.



WBS	Task description	Start date	Finish date	Progress
1	**Deliverable 1**	**24/02/2020**	**03/03/2020**	**100%**
1.1	Team Repository Organisation	24/02/2020	03/03/2020	100%
1.2	Initial Design	25/02/2020	03/03/2020	100%
1.3	Initial Project Plan	25/02/2020	03/03/2020	100%
1.4	Above and Beyond	29/02/2020	03/03/2020	100%
2	**Deliverable 2**	**04/03/2020**	**24/03/2020**	**100%**
2.1	**Swagger and Working Endpoint**	**04/03/2020**	**17/03/2020**	**100%**
2.1.1	Deployment as a REST Web service	04/03/2020	17/03/2020	100%
2.1.2	API Serving Requests	04/03/2020	17/03/2020	100%
2.1.3	Logging	04/03/2020	17/03/2020	100%
2.2	**API Documentation, Implementation a**	**10/03/2020**	**24/03/2020**	**100%**
2.2.1	Design details	10/03/2020	24/03/2020	100%
2.2.2	Testing	10/03/2020	24/03/2020	100%
2.2.3	Swagger Documentation	10/03/2020	24/03/2020	100%
3	**Deliverable 3**	**25/03/2020**	**06/04/2020**	**100%**
3.1	Wireframe	25/03/2020	05/04/2020	100%
3.2	Slides	25/03/2020	05/04/2020	100%
3.3	Practise	05/04/2020	05/04/2020	100%
3.4	Present	06/04/2020	06/04/2020	100%
4	**Deliverable 4**	**10/04/2020**	**01/05/2020**	**100%**
4.1	**Final Demo**	**10/04/2020**	**27/04/2020**	**100%**
4.1.1	Graph Page	10/04/2020	26/04/2020	100%
4.1.2	Map Page	10/04/2020	26/04/2020	100%
4.1.3	Backend	10/04/2020	26/04/2020	100%
4.1.4	Practise	26/04/2020	26/04/2020	100%
4.1.5	Present	27/04/2020	27/04/2020	100%
4.2	**Final Report**	**28/04/2020**	**01/05/2020**	**100%**
4.2.1	Use Cases and Requirements	28/04/2020	01/05/2020	100%
4.2.2	System Design and Implementation	28/04/2020	01/05/2020	100%
4.2.3	Team Organization/Appraisal	28/04/2020	01/05/2020	100%

link: Gantt Chart

Deliverable 4

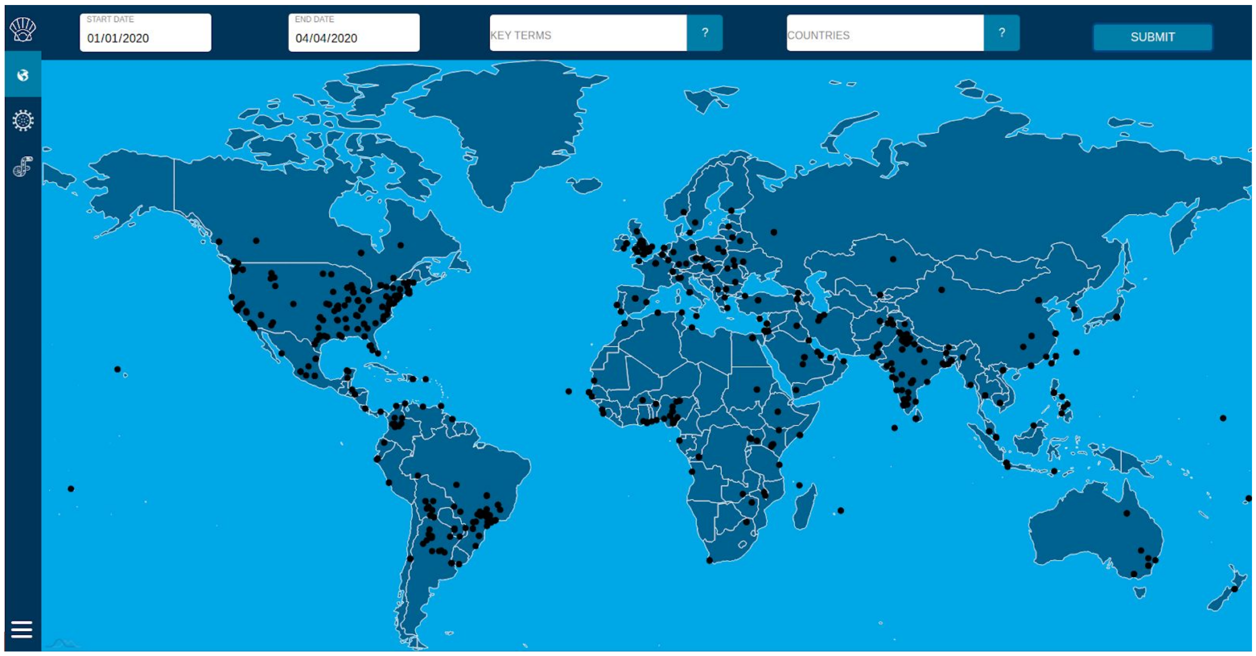
1 Responsibilities of the Team

Team Member	Responsibilities
Carlin	Graph section of the frontend, web scraper
Lucas	API, error handling, frontend, ebola backend
Adam	Map section of frontend, web application backend
Marcus	API, orchestrated frontend
Sean	Graph prediction engine, business requirements

2 How Did the Project Go?

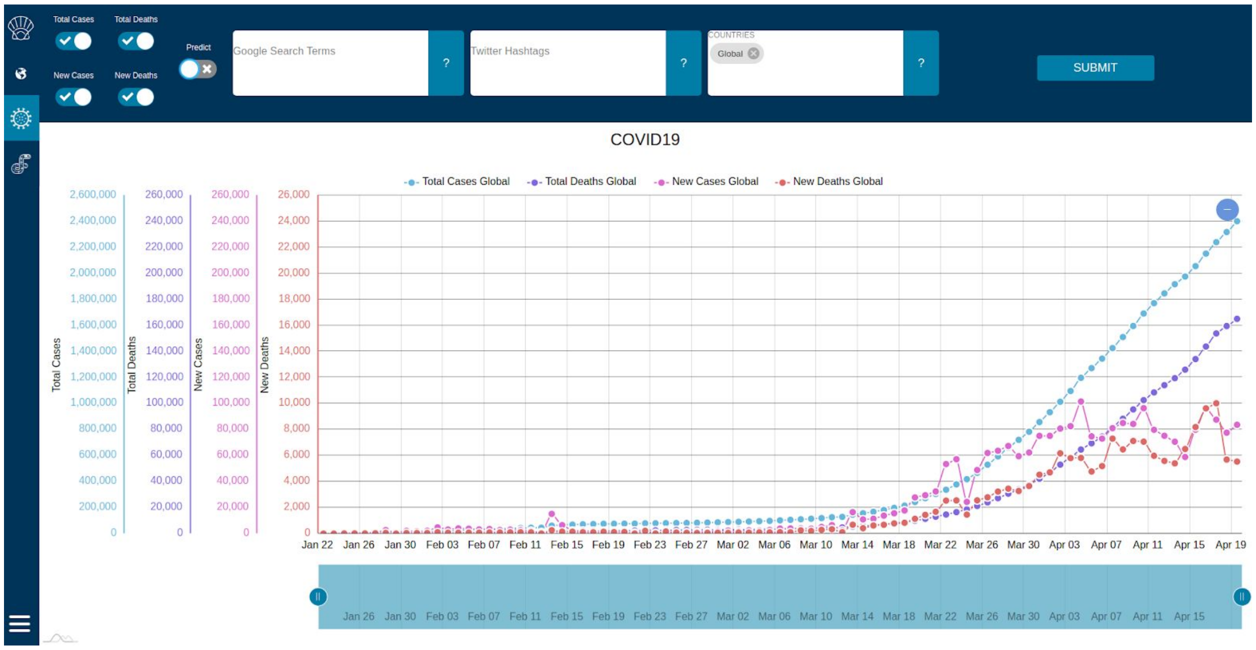
2.1 Major Achievements

One of the major achievements of our project was the map that we integrated into the frontend. Our API produced a large amount of data, with our database storing over 30,000 records. So we needed a compact way to view that information that did not overload the user. The map allowed us to display all the data at once, and display information about the data such as its location without cluttering the website or overloading the user with data. This achievement allowed for effective use of our API, and other teams APIs that we integrated into the map.



Above: The map displaying a range of reports from the 1st of January, 2020, to the 4th of April, 2020

Another one of our achievements, and arguably more impressive, was the graphing engine we created. This, much like the map, was able to display a large volume of data easily, but it also was able to facilitate detailed comparisons at a glance. This meant that we were able to continuously integrate more data into the system, and through selective filtering a user could make meaningful conclusions from the data. This, coupled with our prediction engine, helped produce an impressive platform for analysing outbreaks and developing an understanding of the potential future of an outbreak.



Above: The graph showing current data relating to the current COVID-19 pandemic

These two achievements culminate in the final achievement of producing a working prototype of the outbreak analysis platform that we had planned to create at the beginning of this course. It is encouraging to see these plans come into fruition.

2.2 Issues/Problems Encountered

Although we managed to deliver a final product that we were happy with, we faced several potential issues that could have threatened this outcome. One major problem was that our API could not give us global data cleanly. At worst, this meant that our app could not interpret whatever API we were using, and at best we would still be left with a map that was less accurate, and less visually appealing, than we were hoping for. Our team managed to write a sanitiser that verifies whatever data we received, allowing our app to interpret it without breaking.Through trial and error, we also managed to find an API which seemed to have the least amount of inaccuracies in global data. Combined with our interpreter, we managed to mostly nullify this issue.

However, there were some goals we did not manage to achieve because of unforeseen circumstances. The most notable of these was implementing pagination for the reports. Originally, we had not anticipated we would need this feature, as nobody in the team had visualised the sheer number of reports that may be returned by any one query. After our initial prototype, we noticed this flaw, and so immediately resolved to implement pagination for the reports. However, after many hours of work we realised that this would be a lot more complicated than originally anticipated, due to the interaction with the map-to-report link. Due to this realisation, we pivoted our work towards more important features, leaving this time-intensive change in our backlog. In future projects, more planning and research would allow us to quickly identify this problem, and plan for it with the design of our app's architecture.

2.3 What Kind of Skills Would Have Been Useful?

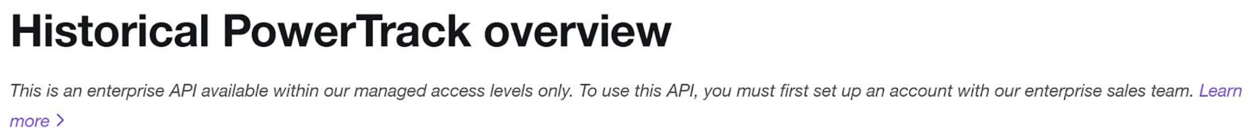
To complete this project, a wide variety of soft, and technical skills are needed, that aren't necessarily provided by the uni. While technical skills allowed us to create a versatile and functional product, the development of our soft skills allowed us to accurately convey the value of what we have actually made to others and communicate well with each other.

None of our team had taken a course in web development, so we had to draw on our previous industry experience so that we could create the app we were aiming for. A few members of our team had a solid understanding of Web App development with React, which enabled our team to both quickly create a quality frontend interface, and provide assistance to the less experienced members of the team to bring them up to speed with any potential problems and bug fixing. Similarly, API development and deployment were skills that nobody in our team had. Part way through the course, we had to research what this skill was, and then plan out the most time efficient and high quality way of implementing this skill. If everyone in our team understood API's, and how to utilise them, before we had started our project, we would have spent significantly less time researching and learning through trial and error.

Another area of skills that would have been useful in the creation of our application would have been some more business focused skills. Whilst most of our degrees have had a focus on technical skills, we found ourselves ill equipped when it came to making business decisions about what our application should do and what audience it should target. More importantly, we would have benefited from some skills relating to presentations and reporting, as we do not currently have the business perspective to know what comes out of these areas.

2.4 Things We Would Do Differently

Similar to the Google Trends Data, we were planning to use the Twitter API to retrieve historical Twitter trends. We had researched the Twitter API and knew that the functionality existed for us to achieve our goals, however it required acquiring an Enterprise Twitter Account. This involved filling out an application to be reviewed and approved by Twitter. We submitted an application and were approved but unfortunately, it took too long for Twitter to respond and so we did not have enough time to implement the API into our website. Next time we would understand that some API's require an application process and apply for them much earlier. Fortunately, we were able to implement the Google Trends API functionality so that we could demonstrate our feature as intended.


Historical PowerTrack overview

This is an enterprise API available within our managed access levels only. To use this API, you must first set up an account with our enterprise sales team. Learn more >

Above: Screenshot of the Twitter API feature we intended to use requiring an enterprise account

One front end design feature that we felt would be particularly useful was the addition of autocomplete. When implementing this feature however, it proved difficult to find a convenient tool to use that worked with our Javascript framework and one that did not require copying vast amounts of code. There was one solution that involved importing a Material-UI Autocomplete Component which can be found here: Autocomplete Component. Unfortunately, this component is one of Material-UI's Lab components which means it is still in an experimental stage and has not been finished. What this meant for us, was that it made it more difficult for us to customise what the component looked like in our website. Functionally it worked great and we were happy with the end result, but next time we would have definitely preferred to have conducted more research and perhaps trialled different implementations so that we could find a better one to suit our website.