UNSW SYDNEY

University of New South Wales

Software Engineering Workshop 3

SENG3011

Management Information

Thicc Peas

z5238611, Hao Cheong

z5238064, Brandon Green

z519527, Fanrui Li

z5257129, Ryan Vu Anh Nguyen

z5263663, Max Emerson Owen

Table of Contents

1.0 Project Plan

1.1 Team Member responsibilities

These are main roles for each team member during each sprint; each sprint corresponds to the 'stages' outlined in the project specification. In general, as there are many different services required for the final project (Web-Scraper, API, API documentation website, web application, etc.) we have opted to have certain team members specialise in certain fields they have experience in to maximise efficiency.

1.1.1 Sprint 1: Web Scraper, API, API documentation

Team Member	Role Name	Role Description
Hao Cheong	• Documentation Manager • Webscraper Manager • Google Cloud Developer	• Will coordinate documentation and report writing • Manages design and development of the web-scraper • Manages the Google Cloud deployment
Brandon Green	• Taskmaster (Project Manager) • API Test writer	• Breaks down bigger tasks, assigns them to teammates and manages the Trello board • Creates ideas and concepts for the project • Designs the test needed to test the API endpoints
Fanrui Li	• API Developer	• Develops the API
Ryan Vu Anh Nguyen	• Team Coordinator • API Manager • Database Developer	• Makes meeting notes and organises meetings • Manages design and development of the API • Develops the database
Max Owen	• Database Manager • Webscraper Developer	• Manages design and development of the database • Develops web-scrapers for report data

1.1.2 Sprint 2: Website

Team Member	Role Name	Role Description
Hao Cheong	• Documentation Manager • Web Developer • Google Cloud Developer	• Will coordinate documentation and report writing • Develops the website • Manages the Google Cloud deployment
Brandon Green	• Taskmaster (Project Manager) • Web Developer	• Breaks town bigger tasks, assigns them to teammates and manages the Trello board • Develops the website
Fanrui Li	• Website Manager	• Manages the development of the website
Ryan Vu Anh Nguyen	• Team Coordinator • Google Cloud Developer	• Makes meeting notes • Organises meetings • Manages the Google Cloud deployment
Max Owen	• Web Developer	• Develops the website

1.2 Workflow Management

The following section describes how our team will communicate, meet and operate during the term.

Our team is anticipating a weekly workload of 6 - 7 hours per person for this project. This is subject to change, as we are unsure of the specifics of future deliverables and have not assigned tasks for these unknowns. We have a tentative Gantt chart to outline our future deadlines.

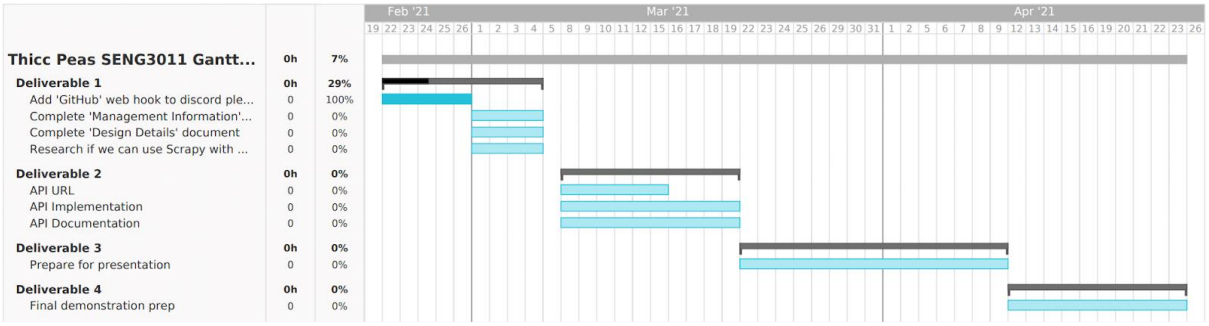

Figure 1. Gantt Chart

To check in on this work, we will be setting tasks and deadlines in our trello board (which has been linked to this gantt chart online), explained below.

1.2.1 Project Management

Main team communication as well as communication with the mentor shall occur on Discord. Task-based project management will occur on a Trello board (below) where we shall utilise a Kanban style of task management. All reports, code and other deliverables shall exist on the GitHub repository.

On top of this, we also used ours Github's issue page to coordinate and distributed the required work and goals required to complete the assigned work. This allows us to better coordinate work and by using the github webhook, we are given constant updates to any changes made to the system

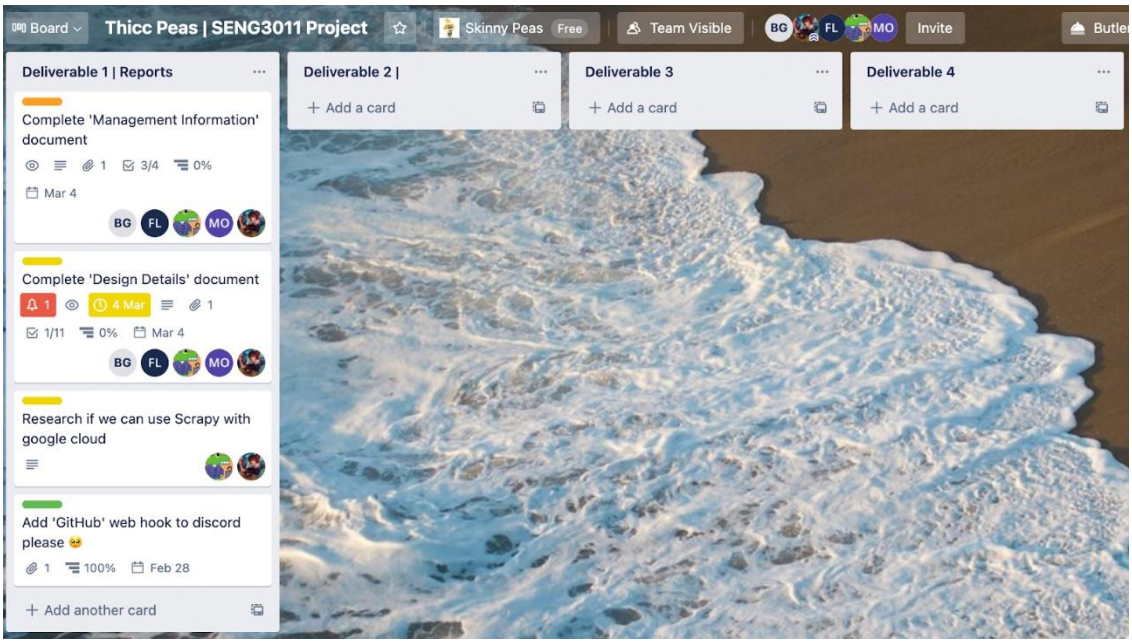

Figure 2. Trello Board

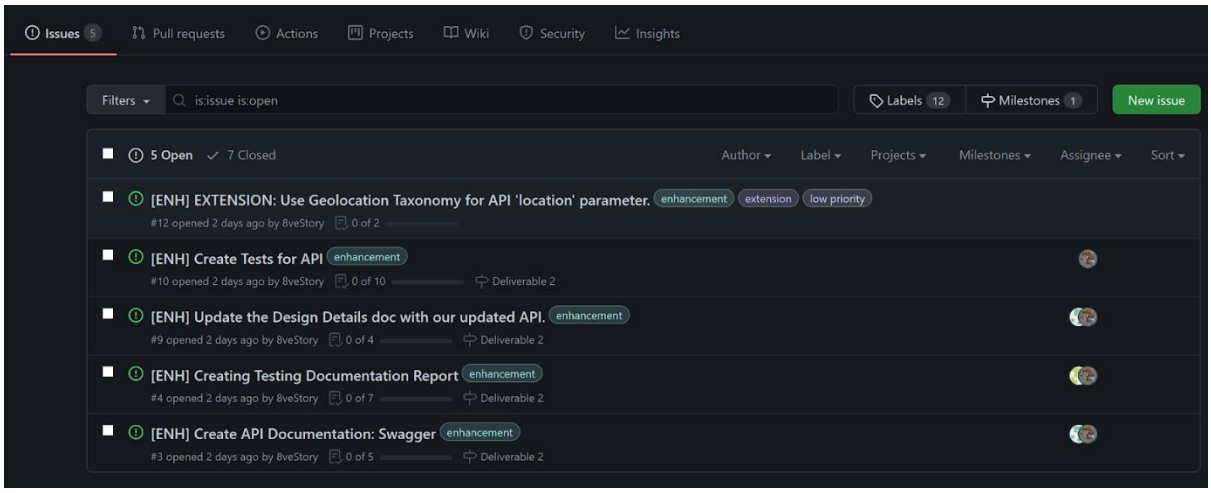

Figure 3. Github Issue Board

We shall perform branch-based development in the repository. Every new feature or task is allocated to a new branch. Once a feature is complete then a pull request (PR) can be raised to merge the feature branch into the main branch. After review by another team member the PR can be merged. We aim for feature branches to be active for a maximum of 7 days to minimise tedious merge conflicts with the main branch. Additionally, the main branch should always be in a stable state.

If time permits, unit tests and an automated build system will be implemented using GitHub Actions to ensure the stability of the main branch.

1.2.2 Meetings

The team shall meet every *Sunday at 1pm* as well as during the mentoring session on Thursday 7pm - 7:20pm. Additional meetings can freely be discussed and held on the Discord voice channels if required. All meeting notes shall be handled by 'Team Coordinator' who will publish the notes to Google Drive and the Discord (below). These meeting notes can be used to document progress and tracking important details which can be used to improve the quality of our final product. Any impromptu meetings will also have meeting notes made to ensure that every member is always updated to the progress of the project

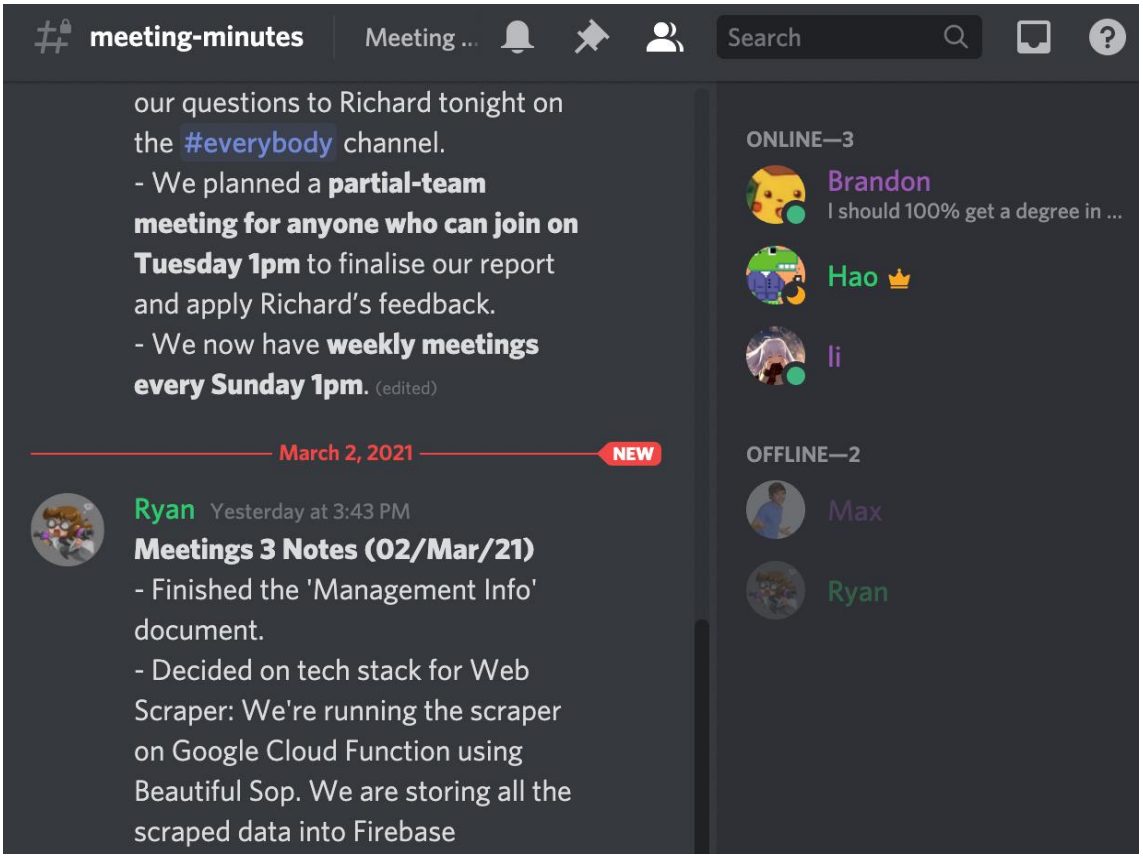

Figure 4. Discord meeting minutes

2.0 Team Appraisal

2.1 Responsibilities and Team Organisation

In terms of team organisation, there were numerous strengths among the team such that we decided it would be better to specialise the task rather than having to split each component of the project for each team member.

Some team members are more familiar with frontend development and using frameworks such as React.js, HTML and CSS and some are more familiar with backend frameworks such as NodeJS. Occasionally, team members would move around between different components not part of their specialty as they proved more time-consuming and challenging than initially predicted. The team had the prior advantage of having worked together on a project which means team familiarity was present and we were able to hit the ground running.

The weaknesses we had were the lack of API development knowledge. To rectify this, the team went and studied a number of popular API's such as Dropbox's and Spotify's API, and started early by experimenting with different testing suites such as Chai.js and incrementally built upon it ensuring stability for our API.

To minimise the crunch required for our applications, the team kept self-imposed soft deadlines and Agile-like weekly meetings to ensure the team was up to date with each other's progress. We would discuss what has been completed so far, what was left to be completed, and what problems we were encountering. Meetings minutes were kept on the team Discord and goals for the next soft deadline were set.

We used a number of tools to aid in the team's management. Discord was the primary source of communication between the team and Github issues as well as milestones were used to keep track of the team's progress. We also implemented a Github webhook to Discord so every Github update was sent to a Discord channel for every Github change made, keeping all members aware of the team's current development project.

API Development

During the development of the API, the team split up work into 3 components: API documentation, API development, and scraper development. Early on, the team quickly allocated the work and through self-imposed soft deadlines during the weekends to keep each other up to date with our individual progress. This was to minimize the crunch which we would need to do near the end of the application ensuring that all the components were stable for each other's development.

Web Application Development

The process was similar to the API. The team started with a wireframe of the application on Figma and distributed these pages to individual members for development. Due to scheduling conflicts, the soft deadline system we had for the API was not used, instead, the team kept each other updated through asynchronous stand-up meetings on Discord.

2.2 Problems Encountered

On the API side the biggest challenge which the experience was the deployment of the API. The team did not have any prior experience with cloud-based deployment platforms so everything had to be learned from scratch. The necessity of this research also meant that numerous other parts such as learning the testing frameworks were put on hold. Ultimately, the team, through rigorous research, managed to pin down a suitable provider (Google Cloud Platform and successfully launched the API.

The CDC data source also added a lot of problems to the development of our application. The odd structure of their web pages along with the inconsistency between web pages meant that the possible data sources which could be stored into the API were constantly changing. On top of this, the team members assigned to this also had to deal with a new scraping library which we were unfamiliar with which exacerbated many of our problems. Ultimately, we made a judgment call on the available scrapable information. This did however, limit the amount of data that we had in our API.

On the Web application side, there were technical and innovative challenges which the team needed to overcome. In the beginning, there was the innovative challenge of coming up with an idea. The team decided against building a simple report reading system as we agreed that the idea was fundamentally unnecessary. The team was unknowledgeable in the domain of immunology and health systems. Given that the team had some connections within the medical field including a pharmacist and a hospital CEO, we interviewed them to get a better understanding of the flaws of the current system and formulate an idea around those instead. The final idea was something that the team agreed was useful and had genuine business value to be added.

The team, while having some basic understanding of web frontend development, was still not familiar enough to actually create an application. The team resorted to splitting various pages across different members and learning the necessary framework from the scratch (React, HTML, CSS, etc). This prolonged the development cycle of the application considerably. To try to mitigate this, the team tried to work concurrently such that any new knowledge gained by one member could quickly be communicated to another and speeding up the learning process.

2.3 Possible Improvements

In hindsight, there are a number of things that could have been improved to make the final product more robust and complete.

The first improvement was the complexity and specificity which our API provides. When completing the final application, the limitation of the API held us back on what we could have come up with. Examples such as the ability to filter reports base on reported cases and to be able to filter disease via their symptoms. This filtering could be obtained by filtering the date client-side, but would have been more convenient if this was already available on the back-end. The team should have considered the fine-grained filtering possibly required of the final application during the development of this API.

The team also misjudged the difficulty of using the frontend frameworks ReactJS, MaterialUI and Bulma CSS. Given another chance, the team would have spent a little bit longer early on studying the frameworks we were planning on using. This would have given us enough time to experiment with the framework such that the final frontend component was more robust.

Given the team's inexperience with the frontend, we were also unfamiliar with UI/UX design principles. This again forced the team to study and learn, sometimes through trial and error, to finalise a design which we were satisfied with. If given more time, the team would have either studied further in UI/UX or consult a design expert to improve the visual design of our application.

2.4 Team Achievements

While there were many challenges and improvements, ultimately the team is very proud of the final application, its idea, and the API, and all the technology which the team learned from them.

The learning outcome which came from developing the application and the deployment of the API provided us with a better understanding of the technology behind them as well as other possible tech stacks which could be used for future projects. The team also gained the ability to work with DevOps technology such as Google Cloud Platform, AWS, and many other cloud providers

Altogether, the team believes that the final idea of the application had genuine usefulness and adds business value to the domain of vaccines and disease outbreaks. It provides a solution to a genuine problem in the current system as well as being backed up by professionals in the medical domain.