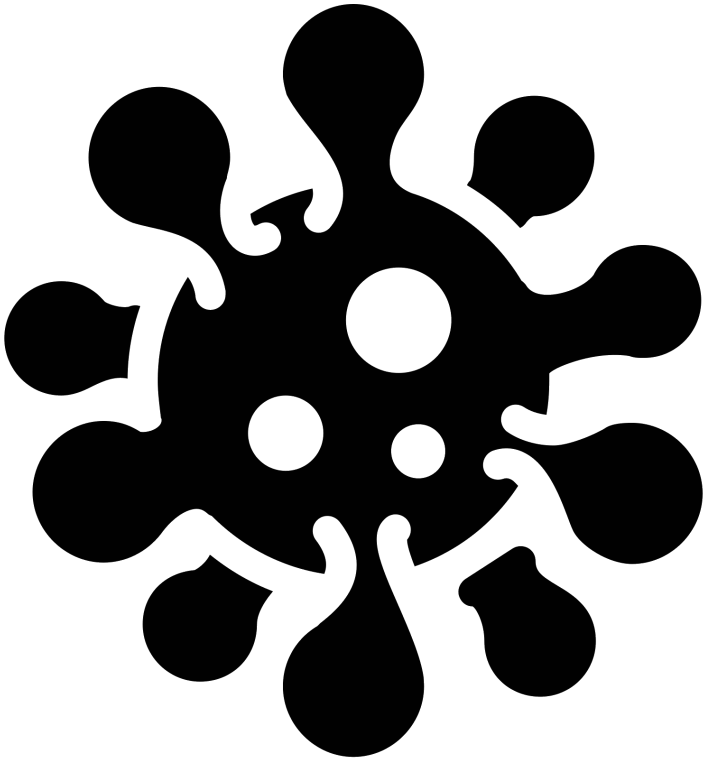
Project Management Report



Team Viral

Aimen Hamed - z53110143
Henry Ho - z5312521
Leila Barry - z5206290
Neelam Samachar - z5310063
Shannen Jakosalem - z5164261

Mentor: Nikhil Ahuja
Course: SENG3011

Table of Contents

Table of Contents **2**

Introduction **3**

Team Member Responsibilities **4**

Work Arrangements **7**

Remote Working 7

Meetings 7

Communication 9

Difficulties in Meeting Deadlines 9

Tools 10

Git Practices **13**

Version Control Management Practice 13

Commit Message Conventions 14

Commit Message Conventions - Issue Identification Tags 14

Branch Naming Conventions 15

Automated Testing Pipeline 16

Conclusion **17**

Introduction

The purpose of this report is to elaborate Team Viral's approach to project management with regards to the SENG3011 project. This document not only serves as a method to communicate our approach to our mentor for the purposes of deliverable 1, but also as a tool to foster a shared understanding of expectations and responsibilities amongst each team member. We believe it is important to have such a document as it will enable the team to have a structured and coordinated approach in the early stages of the project, thereby reducing the risk of losing time due to issues such as administrative errors and several team members mistakenly completing the same task.

As of writing this report, the specifications for deliverables 2 and 3 have just been released and as such the team has little understanding of how they will impact our approach. Therefore, the team notes that any part of our project management approach may change to address the requirements laid out by these later deliverables. If such a change is necessary, the team will revise our current approach and modify it where necessary. To provide some guidance on what these modifications should be like, the team has agreed that their impact should be minimal in order to reduce disruption to progress as we have identified that transitioning from one way of working to another will require time to adjust to.

This report is divided into three sections that touch on a particular aspect of project development. These sections, what they cover and why they are discussed are presented in the table below in order of when they will appear in the report.

Section Name	**Description**	**Purpose**
Team Member Responsibilities	An overview of each team member's role and responsibilities.	To ensure that each team member knows what tasks they are expected to complete and the role they will play in the team.
Work Arrangements	An overview of different aspects of project management such as how the team will communicate, what tools will be used, etc.	To ensure that the team has a shared understanding of how we're going to complete the project.
Git Practices	An overview of how the team will use Git. This tool will be discussed in its own section due to its complexity.	To define a standardised approach to Git so that each team member uses it in the same way, encouraging consistency.

Team Member Responsibilities

It is important to note that these role assignments are flexible and can change if a team member wishes to work on another part of the project or if a particular area needs more attention (e.g. if the backend component of the project is found to be more complex than anticipated, team members working on the frontend may temporarily shift to backend work).

The team has been divided into two sub-teams: frontend and backend. This division was done in accordance with each team member's interests and previous experience. See the table below for the assignment of roles and an explanation of what each role entails.

In terms of tasks and responsibilities, we intend to continually rotate the roles around, to allow each team member to get a feel of what each role entails. We also intend to keep updating the tasks and responsibilities of each team member as we progress with the project. Hence, team members will not be fixed to a specific role and have purposely made it flexible as a result. In the long term of the project, we intend:

Team Member	Agile Role	Technical role	Tasks/Responsibilities
Henry	Scrum Master, Developer, Developmental tester	Crawler	• Is involved in using the crawler to scrape the data from the WHO website. • Is responsible for organising future meetings and updating the meeting minutes document. • Responsible for updating individual issues/epics on the Jira taskboard. • In terms of the Deliverable 1 report, Henry will be working on the first question of the design details section and will be working in conjunction with Aimen for the third question. • Will be working in conjunction with Aimen to finalise the tech stack that will be used and the software architecture. • He will also be conducting further research on possible spider frameworks. • Will be working on testing as we progress through the implementation of the code.
Aimen	Product Owner, Developer, Developmental tester	Full Stack	• Intends to work on mostly the backend, but is willing to help in the implementation of the frontend if required. • Responsible for updating individual issues/epics on the Jira taskboard. • He will be working in conjunction with Henry and Leila in terms of implementing the backend. • Wrote/included the CI/CD pipeline for automated testing. • In terms of the deliverable 1 report, he will be working on the second question of the design details section and will be working with Henry on the third question and the software architecture. • Completed the swagger implementation for the API. • Completed the setup for the github repository in the structure required. • Will be working on testing as we progress through the implementation of the code.
Neelam	Project Manager,	Frontend	• Intends to work on the frontend and will be working in conjunction with

	Developer, Developmental Tester		Shannen to complete this. ● This will require familiarising and learning React and JavaScript. ● Responsible for updating individual issues/epics on the Jira taskboard. ● In terms of the deliverable 1 report, Neelam will be working on the project management section of the report with Leila and Shannen, specifically the tasks/responsibilities allocation between team members. ● Will be working on testing as we progress through the implementation of the code.
Leila	Developer, Developmental tester	Backend	● Intends to work with Aimen on the backend of the API, which requires learning JavaScript. ● Responsible for updating individual issues/epics on the Jira taskboard. ● In terms of the deliverable 1 report, she will be working in conjunction with Neelam and Shannen to work on the project management side of the report. ● This includes how the team will be coordinating with each other throughout the entirety of the project and adding information on how the group intends to use git. ● Will be working on testing as we progress through the implementation of the code.
Shannen	Developer, Developmental tester	Full Stack	● Intends to work on both the backend (if required) and frontend, but plans to work primarily on the frontend. ● Responsible for updating individual issues/epics on the Jira taskboard. ● In terms of the deliverable 1 report, she will be working in conjunction with Leila and Neelam on the project management section of the report, primarily focusing on the working arrangements of the team, including communication methods and the tools that will be used. ● Will be working on testing as we progress through the implementation of the code.

Work Arrangements

Remote Working

The team has and will continue to work remotely as this is the most convenient work arrangement for us all. Working like this will allow us to attend meetings scheduled on short notice and enable us to gain invaluable experience in remote working tools such as GitHub and Jira. Since the team is spread out across Sydney, remote working also means that we can use the time needed to travel to an in-person meeting for higher value tasks such as programming. This work arrangement also offers the safest way to progress through the assignment as it reduces the likelihood of potential exposure to COVID-19 as a result of travel and interacting with each other in-person.

Meetings

In order to ensure our team's individual and collective goals are aligned, we will be meeting regularly to coordinate our activities. Productive meetings are important as they provide an opportunity to discuss key details of the project, raise any issues or risks, increase team morale, and set achievable goals (Project Management Institute, 2008).

The team plans to meet twice a week: Saturday afternoons and after our mentoring session on Tuesdays, to discuss the progress we've made on our assigned tasks, important matters that require input from the whole team (e.g. system architecture) and any problems that we may have encountered. The team has agreed on keeping a flexible meeting schedule and will arrange additional meetings if the need arises (e.g. when the deadline of a deliverable is approaching and the team feels like we haven't done enough work). These team meetings will be held in addition to our weekly mentor meetings where we plan to seek advice from our mentor on matters the team may struggle to resolve on our own such as understanding the project specifications.

In order to ensure that we are using our meeting times effectively, we will be making use of meeting minutes. Our minutes are broken down into a number of sections including: a list of items we wish to discuss, details for each of the topics discussed, and a list of questions and/or statements we wish to clarify during our mentoring session. In order to kick-start our meetings in a time efficient manner, team members can update the list of items to be discussed prior to the meeting thus ensuring we have a clear set of goals for the meeting.

Our meeting minutes template is given below in Figure 1.

Viral - Minutes X

DATE: XX/XX	TIME: XX:XX A/PM	LOCATION: XXX

TYPE OF MEETING	General Meeting
MEETING CALLED BY	All
NOTETAKER	Henry

ITEM	TOPIC	DETAILS
1	Attendance	Everyone
2	Last week's actionables	
3		
4		
5		

This Week's Actionables:

WHO	TODO	DEADLINE
Aimen	-	
Henry	-	
Leila	-	
Neelam	-	
Shannen	-	

NEXT MEETING: XX/XX @ XX:XX A/PM
MEETING ADJOURNED AT: XX:XX A/PM

Last Week's Actionables:

WHO	TODO	DEADLINE
Aimen	-	
Henry	-	
Leila	-	
Neelam	-	
Shannen	-	

Figure 1: Meeting minutes template

Thus, our systematic approach to regular meetings ensures that we are using our time well, communicating effectively, and ensuring that our next steps are coordinated.

Communication

We have chosen to use two platforms for team communication: Messenger and Discord.

Messenger is used for text-based communication. So far, we have been using it to ask quick questions (e.g. when meetings are), give brief updates on our progress, bring up any problems we've encountered and discuss minor issues (e.g. how to arrange our Kanban board). This platform is reserved for communication on matters that aren't urgent and require little to no discussion.

Discord is used for more substantial communications and is the platform we use for team meetings. Some matters require a greater level of discussion that can't be effectively facilitated by text-based communication via Messenger and so the team has decided to use Discord to address this.

We acknowledge that having two platforms for team communication may seem needlessly complicated. However, we believe that each platform is better suited to a particular function. Messenger has limited functionality but supports quick communication well. Discord has a wealth of advanced features not relevant to the project but a handful address limitations found on Messenger. For example, channels on Discord allow us to store important links to resources that would otherwise be lost on Messenger. Moreover, Discord's voice chat feature allows for instant voice-based communication whilst the same feature on Messenger requires more tedious set up. As such, having two separate platforms (one for quick text-based communication and another for longer voice-based communication), although not optimal for other groups, is an arrangement that has been working well for our group to date.

It is also acknowledged that we could've used Microsoft Teams and the biggest drawback in doing so is having a centralised platform for communication. We view MS Teams as a platform for mentor-team communication and a forum on which to draw information and pose questions rather than a space for us to communicate as a team.

Difficulties in Meeting Deadlines

The team acknowledges that the SENG3011 project has been designed to be challenging and we expect it to be a time intensive endeavour. We also acknowledge that each team member has other responsibilities they have to uphold whether it be completing assignments for other courses or working. If a team member suspects that they will be unable to meet a deadline, they are expected to communicate this to the rest of the team as soon as possible so that their task can be reassigned to someone else and the deadline can still be met. Team members are also expected to be upfront with their current workloads so that tasks can be equitably assigned in light of what each team member is capable of doing at a particular point in time. The reassignment of tasks will be done in consideration of time availability and how willing a team member is to take on another task to ensure that they are not forced onto anybody and to keep a healthy team dynamic.

Tools

We will be using several tools to help us complete the project. These tools have been selected by first identifying aspects of the project that need some sort of tool to facilitate. These aspects and the reason why they're crucial to the project are as follows.

1. **Team communication** - to help us discuss the project
2. **Document management** - to help us create documents and share them with each other
3. **Project management** - to help us maintain a schedule in the completion of the project

The team then brainstormed tools that could address each of the aspects listed above. Although we could've explored other applications, we decided to limit our options to tools that we were familiar with, were comfortable using and/or could quickly pick up. Given we have a tight timeframe with which to complete the project, we decided it would be best to take this approach as exploring other applications would take up extra time that we could've used to progress the project.

See below the tools the team will be using and why we have selected them.

Team Communication		
Application	**Will the team be using it?**	**Justification**
Discord	Yes	See section on communication
Messenger	Yes	
Microsoft Teams	Only for mentor-team meetings	
Document Management		
Application	**Will the team be using it?**	**Justification**
Google Drive	Yes	Need a place to store documents such as meeting minutes, the project specification, etc. in an organised manner. We all have extensive experience in Google Drive through using it for other courses. Allows for concurrent collaboration.
Confluence	Yes	Serves a similar function to Google Drive but offers more features than is necessary for the project. We will be using it to submit our work as it is accessible to our tutors.
GitHub	Yes	Need a place to store code. Everyone has experience using GitHub. Our team also has more experience in pipelines using GitHub Actions. Allows for concurrent collaboration without impacting each other. Focus on infrastructure performance is relevant to our project. Is required by the course.
GitLab	No	Similar to GitHub however it is not required by the course and it places a greater focus on feature development which is not as relevant to our project.

Project Management		
Application	**Will the team be using it?**	**Justification**
Trello	No	Offers a host of advanced features beyond the scope of the project
Jira	Yes	Offers a subset of the project management features found on Trello that are relevant and useful to the project. Easier to use than Trello and requires no exploration of functionalities (i.e. we can get straight to using it)

Git Practices

Version Control Management Practice

Since we have decided to make use of GitHub to store, manage, and collaborate on our code it is vital that we follow a set of git practices to ensure the seamless operation of our GitHub codebase.

We considered two popular options for our version control management practice:

1. GitFlow, which makes use of larger feature branches which merge only upon completion, release branches for each release cycle, and hotfix branches used to patch production releases quickly. GitFlow is a largely release-based software workflow.
2. Trunk-based development, which makes use of minor, regular updates to a main branch.

Ultimately, we have chosen to go with trunk-based development over GitFlow as it streamlines the merging and integration phases, promotes CI/CD (continuous integration, continuous delivery, and continuous deployment), eases the code review process, and is generally at a much lower-risk of merge conflicts than the longer-living branches used in GitFlow.

The specific practices we will follow in our trunk-based development practice as recommended by Atlassian are as follows:

1. **Small commits** - Commit small changes and merge a couple of these small commits at a time to minimise cognitive overhead when reviewing changes and making decisions.
2. **Feature flags** - Making use of feature flags to build feature specifications piece by piece within the flag as opposed to having to wait to build the entire specification at once.
3. **Automated testing** - Implement automated testing through short-running unit and integration tests that are executed throughout different stages of the development process and upon merging code.
4. **Asynchronous code reviews** - Review the code prior to confirming a request to ensure that the new code meets the required specifications.
5. **Minimal branches** - Upon branch merges, delete the branch to maintain the organisation of the code-base.
6. **Regular merges** - We will be aiming to merge any ready branches to the trunk on a daily basis.

Commit Message Conventions

Concise and consistent commit messages are vital to communicating the context of changes and in turn, make reviewing changes significantly easier. An accurate and concise log of all changes through commits contributes greatly to the maintenance of the project, particularly the long-term.

In order to keep our changes log consistent amongst commits we will be using the following basic style guide:

1. The subject should contain an issue identification tag followed by a concise summary of the commit.
2. If required, follow the subject with a blank line, followed by a more in-depth explanation of the changes made and any important consequences of these changes.
3. Explain the problem that the commit solved.

An example commit message would look like:

```
BUGFIX-002 refactored <function> in <file>

I refactored <function> to use <method> as opposed to <previous method>.
Some important consequences of this change are: 1. Consequence A, 2.
Consequence B.

I made this change as the previous method was unintuitive / did not
behave as intended. This problem is now resolved.
```

Commit Message Conventions - Issue Identification Tags

We will make use of issue identification tags in commit messages to quickly perceive the reason for a commit and quickly identify any recurring issues. These tags will be of the form <ACTION>-<XXX> where action is one of: IMPL (implementation), TEST or BUGFIX and X is a digit 0-9. The following table of issue identification tags will be updated as issues arise in development.

Issue Identification Tag	Issue	Description	Example Subject
BUGFIX-001	Syntax error	Used when the commit fixed a simple syntax error in the code.	BUGFIX-001 corrected syntax error in <file>
BUGFIX-002	Refactoring	Used when the commit has refactored a portion of code.	BUGFIX-002 refactored <function> in <file>
TEST-003	Add unit test	Used when the commit has created new unit tests for the	TEST-003 added unit tests for

		code.	<function> in <file>
TEST-004	Add integration test	Used when the commit has created new integration tests for the code.	TEST-004 added integration tests for <file(s)/feature>
BUGFIX-005	Documentation	Used when the commit has added documentation to the code.	BUGFIX-005 improved documentation for <function> in <file>
BUGFIX-006	Fixed failing function	Used when the commit has fixed a function that was failing tests or not behaving as intended.	BUGFIX-006 fixed failing <function> in <file>
IMPL-007	Implement API route	Used when the commit has implemented a new API route.	IMPL-007 added <route> in <file>
IMPL-008	Implement helper function	Used when the commit has implemented a new helper function.	IMPL-008 added helper <function> in <file>
IMPL-009	Implement function	Used when the commit has implemented a new function.	IMPL-009 added <function> in <file>
TEST-010	Add full-stack test	Used when the commit has created a new full-stack test for the code.	TEST-010 added full-stack test
TEST-011	Add performance test	Used when the commit has created a new performance test.	TEST-011 added performance test`

Branch Naming Conventions

Since we are using feature branches, we will be following a simple naming convention to ensure the quick and intuitive differentiation between branches. Branches will be named in the format “feat/<feature><Jira ticket number>”. The following table of branches will be updated throughout the project - the branch status refers to whether the branch is active, inactive, or deleted.

Branch Name	**Feature**	**Jira Ticket Number**	**Description**	**Branch Status**
feat/search-bar12	Search bar	12	The functionality and presentation of a search	Currently non-existent.

			bar.	

Automated Testing Pipeline

An automated testing pipeline automates the process of reviewing and validating software. To do this we will need a testing framework and a tool to determine our test coverage. We will be using Jest for this as it has functionality for both automated testing and test coverage - essentially, two tools in one which removes the learning curve we would experience trying to integrate two separate tools on our own. Additionally, Jest is closely integrated with the React testing library, provides support for CI/CD pipelines as we are using, and is revered as a reliable framework by other developers. Thus, Jest was a clear choice for the implementation of automated testing in our CI/CD pipeline.

Other frameworks we considered using are:

1. Mocha JS - a JavaScript test framework running on Node.js for asynchronous testing.
2. Istanbul - a tool to measure test coverage.

We considered using Mocha JS because it integrates well with NodeJS, is popular amongst other developers, and is compatible with most browsers. The downsides to Mocha JS are that it runs all tests in the same process which means they share memory and are thus, not isolated from each other. Mocha JS also has no code coverage output and so would require another tool alongside it to ensure that our test suite is reliable.

Finally, to ensure that our software testing is successful we will be using a number of different tests on our project:

1. **Unit tests** - addressing each of the smallest possible units of behaviour to ensure they behave as expected.
2. **Integration tests** - addressing how multiple portions of our software interact with each other and ensuring that it behaves as expected.
3. **Full-stack tests** - addressing how the entire application works and ensuring it functions as expected. There should be only a few of these tests and they should be designed carefully to withstand changes in the user interface.
4. **Performance tests** - since our application has the potential to work with large quantities of data at any time it is important to ensure the application remains stable and acts at a reasonable speed. It will also be important to test how our application behaves under increased demand.

To implement these tests when certain actions are performed on our GitHub repository, we will be using GitHub Actions. GitHub Actions allows for workflow automation including triggering tests on certain actions such as pushes, issue creation, or new releases. We chose to use GitHub Actions over alternatives such as CircleCI as we have team members with experience using it, it supports Node.js, and makes use of live logs.

We will be using GitHub actions to test our code upon each push to ensure that it is behaving as intended and to allow our developers a quick insight into the functionality of our code. This minimises the time spent, ensuring the code behaves correctly so that we can

quickly find and fix bugs when they arise with a good indication of where to find them due to the test results.

Conclusion

In conclusion, we have seriously considered every aspect of our project management to ensure that we are working effectively towards an aligned goal. In doing so, we have highlighted a number of key strategies we will follow to do so regarding our meetings, communication, individual and team responsibilities, and git practices. Thus, this report is an important resource that we will be able to reference throughout our project to ensure cohesion and consistency throughout the development of our project.