SENG3011 - Software Engineering Workshop 3

Management Information

Team megAPIxels: Rubin Roy, Lachlan Fraser, Humza Saeed, Austin Walsh, Sam Thorley

Mentor: Aditya Kishore

Table of Contents

Team Member Responsibilities

The development cycle of the product will follow an agile methodology. As such, the team members will undertake roles following this methodology and be either a Product Owner, Scrum Master, Developer or Tester. These roles will determine each member's key skills and responsibilities needed for the development cycle.

Table 1: Team Member Roles and Responsibilities

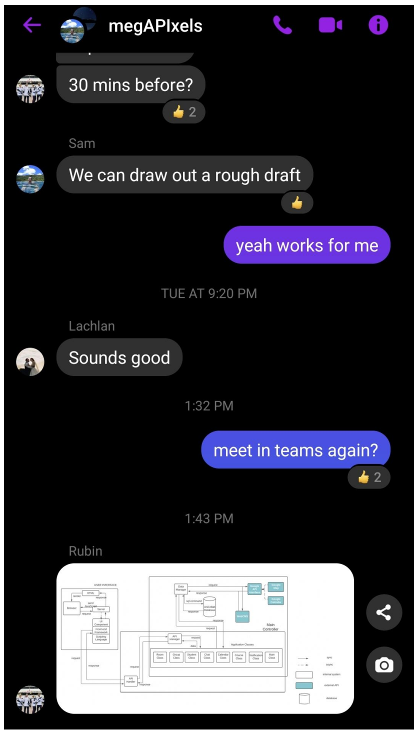
Role	Team Member	Responsibilities
Product Owner	Lachlan Fraser	● Define product specifications through the use of user stories and acceptance criteria ● Manage the scrum backlog ● Liaise with stakeholders ● Ensure product is market ready
Scrum Master	Austin Walsh	● Ensure sprint objectives are met ● Liaise with product owner on sprint plans and reviews ● Manage team's morale and environment ● Manage Confluence pages with ideas
Front-end Developer	Rubin Roy Austin Walsh	● Develop front-end using React ● Integrate API with front-end ● Deploy and maintain front-end
Back-end Developer	Sam Thorley Lachlan Fraser Humza Saeed	● Research API development process ● Setup PostgreSQL database and integrate it into Python ● Develop and deploy API and Python scraper ● Maintain API after deployment ● Generate API documentation
Development Tester	Humza Saeed Sam Thorley Rubin Roy	● Use blackbox, whitebox, unit and regression testing to ensure product works ● Undertake user validation testing ● Test all areas of product

Work Arrangements

To follow the agile methodologies the team decided to undertake some industry practices. The team will have daily meetings to check-in, review current code and understand what needs to be done in order to complete their current sprint. Daily meetings will be held every day at 9am, in which a stand-up will also be completed at that time. A tentative set of sprints have been created which the team will follow, however, the backlog of tasks will change and carry over between sprints depending on whether it was completed in the previous sprint. The teams will also ensure constant communication is happening between members, through the following communication outlets.

Communication

The team will mainly be communicating through a Facebook Messenger chat. This chat will be used by members to provide updates about their progress on assigned work, schedule or confirm upcoming meetings and for any other miscellaneous items that quickly need to reach the team. An example usage of the chat is seen in Figure 1 below.



megAPIxels

30 mins before?

👍 2

Sam

We can draw out a rough draft

👍

yeah works for me

TUE AT 9:20 PM

Lachlan

Sounds good

1:32 PM

meet in teams again?

👍 2

1:43 PM

Rubin

Figure 1: Team Messenger Chat

Alternatives to Facebook Messenger were considered, these included Discord, Slack and Teams. Discord provides a platform in which the team has a large control over the customisation, being able to create multiple text chats for different purposes and pin items to quickly be found again. It also allows people to voice/video chat in the same location. Messenger was chosen over Discord because of its ease of use, the team needed a simple chat to quickly contact each other which was better provided by Messenger.

Slack is a very similar program to Discord, allowing users to have control over most parts of their Slack. As such, for the same reasons as stated above, Messenger was chosen over Slack. Additionally, some members of the team had not used Slack before, while all members had used Messenger meaning it did not include a time cost for some members.

Teams was chosen to be used for meetings, but not for primary messaging of other members. It was decided that meetings would be held on teams for two main reasons, firstly, it would allow our mentor to easily access and see the meeting or any questions the team had about parts of the product, additionally, it had a higher quality and ease of use for video calls than Messenger, allowing members to have less trouble joining a meeting either on phone or computer.

Collaboration

Google Drive

A Google Drive will be used to hold collaborative documents that are required throughout the phases of development. Google Drive allows for documents to be created easily and can hold many different types of documents, such as pictures, editable diagrams and pdfs. Importantly, Google Drive has an excellent collaborative system, allowing members to edit the same document simultaneously with no issues, comment on sections or suggest edits that can be made.

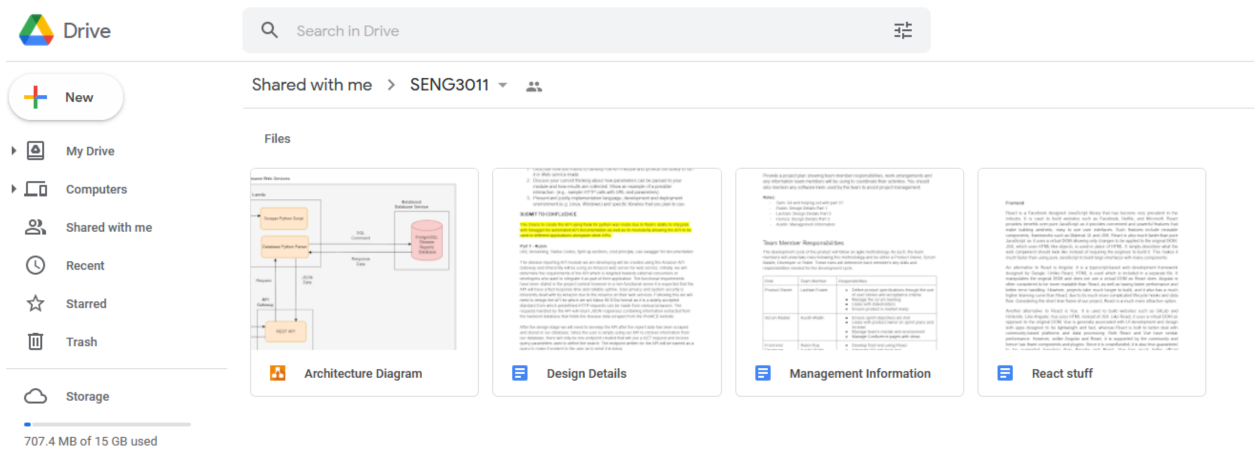


Figure 2: Team Google Drive

Confluence was another option to host documents, and while it is not used by the team to collaborate on documents, it will be used to submit final documents to the mentor. Confluence provides many similar features to that of Google Drive, but was not chosen due to the time cost as most of the members had not used Confluence before.

Github

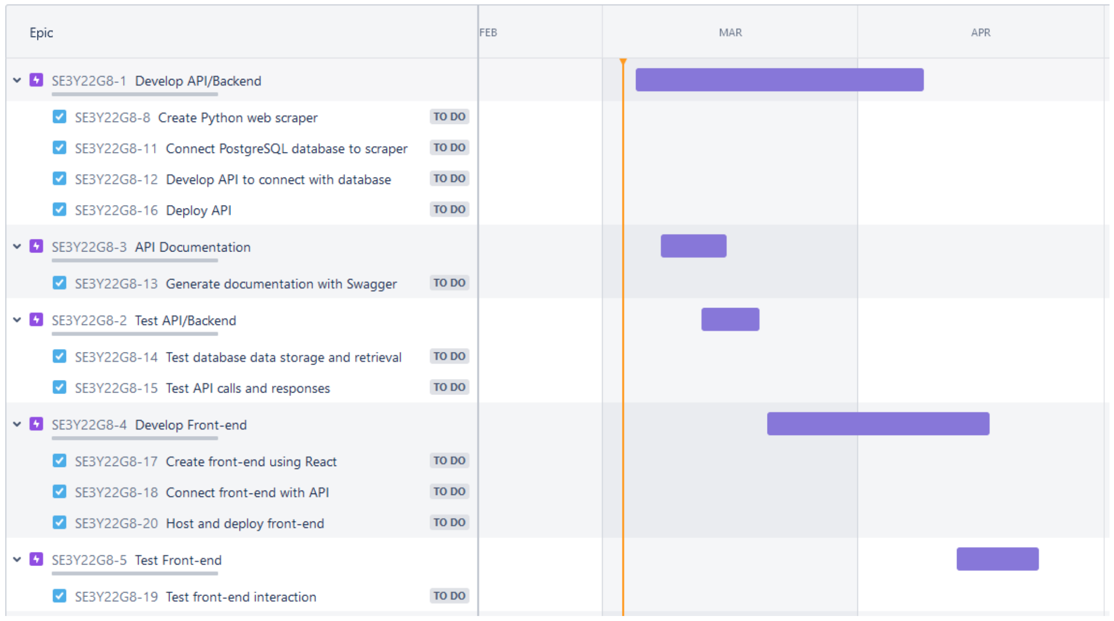
To develop the product the team will be using a Github repository where the code from each phase of the development will be worked on. Github allows for the team to easily collaborate on code during the development cycle of the product. As version control software, it means members will have no issues with working on the same code pages or losing code due to unforeseen circumstances. Github also provides settings ensuring the team will follow the agile principles. These settings include branch control, ensuring members have to pull the branch before merging, merge requests requiring review and stopping branches from being pushed to. The team will follow a rule for commit messages to ensure they are descriptive on their changes allowing all team members to understand what the commit is doing. This commit message rule will ensure the inclusion of completion time, overview of changes and which task it is from on the backlog. Github includes ways to continuously integrate and deploy the code, allowing for the maintenance of the product to be simple even after deployment, further following agile principles.

Some other version control software was considered, namely Gitlab and Azure. Gitlab is much the same as Github, providing a platform for members to collaboratively work on the development of the product. The main reason for the team’s choice of Github over Gitlab was due to past experience, most of the members have used Github more recently than Gitlab, reducing the time cost associated with relearning the Gitlab layout and specifics.

Azure is a Microsoft cloud platform that provides a platform for hosting, deploying and maintaining a product. It can also integrate existing products held on other software such as Github and Jenkins. Whilst it provides many useful features for the development cycle, the time cost associated with team members learning and setting up an account would be too high to justify its usage. The service is also subscription based and developed to be used by large companies so it is somewhat overcomplicated for what the team requires from a version control software.

Jira

Finally, a Jira board will allow the Product Owner and Scrum Master to create and assign sprint objectives which developers will complete. Jira provides the team with an easy way to see, manage and ensure members are working on their assigned sprint objectives. Specifically, this board will be updated throughout the project duration to help all team members to visualise and understand what has been completed, is being worked on, and still needs to be developed. This tool provides a wonderful visual interface to keep the team on task and provide information on project completion at a glance. Jira is also integrated with Confluence providing the team with easy access to both submit documents and view sprints within the one location.



Epic	
SE3Y22G8-1 Develop API/Backend	
SE3Y22G8-8 Create Python web scraper	TO DO
SE3Y22G8-11 Connect PostgreSQL database to scraper	TO DO
SE3Y22G8-12 Develop API to connect with database	TO DO
SE3Y22G8-16 Deploy API	TO DO
SE3Y22G8-3 API Documentation	
SE3Y22G8-13 Generate documentation with Swagger	TO DO
SE3Y22G8-2 Test API/Backend	
SE3Y22G8-14 Test database data storage and retrieval	TO DO
SE3Y22G8-15 Test API calls and responses	TO DO
SE3Y22G8-4 Develop Front-end	
SE3Y22G8-17 Create front-end using React	TO DO
SE3Y22G8-18 Connect front-end with API	TO DO
SE3Y22G8-20 Host and deploy front-end	TO DO
SE3Y22G8-5 Test Front-end	
SE3Y22G8-19 Test front-end interaction	TO DO

Figure 3: Initial Jira Roadmap

Github and Gitlab boards were another option to manage the sprint objectives and whilst Github might have been a good option given it was located within the product repository, the team determine Jira's additional integration with Confluence and easier management of epics would prove to be more useful for the team.

Application Ideas

As we look ahead to developing our own application in the second half of this project, we have brainstormed a few ideas and come up with a general outline of what we would like to achieve. These ideas are included here as a reference and explanation of where we would like to be at the end of this project.

Spec Requirements

The project specification for this section is purposely vague in order to give us freedom to choose a specific area of interest and focus on that. However, there are a few specific key points that we are required to follow.

Firstly, we must develop a web application that integrates disease reports from our API as well as at least one other API. We are then required to design an interface where users are able to search information about diseases, see visualised disease reports over time or based on location and get some predictions.

Additional Ideas

To build on what is mentioned in the spec, our team is thinking of constructing our application primarily around the use of a “heat-map”. This will take the form of a global map where each country is given a certain colour and/or pattern based on the number and type of disease reports from that location. This disease report data will be obtained from our own API as well as through using APIs developed by other teams. This will allow us to combine information from multiple data sources into the one application for users to absorb visually in an easy-to-use manner.

API Usage

In order to develop our application as described above, we will need to incorporate a number of APIs. Primarily, this will involve our own ProMED API that has been designed and developed in phase one of this project. Furthermore, we will incorporate APIs developed by other teams to broaden our field of disease reports for inclusion in our application. Finally, we may require the use of external APIs found online. Specifically, this will probably involve some sort of mapping API for us to create the global heat-map as well as a natural language processor or location API to accurately extract location information from an article.