Team Organisation & Appraisal

SENG3011

Team QQ

1 CONTENTS

2 PROJECT MANAGEMENT TOOLS

2.1 OVERVIEW OF PROJECT MANAGEMENT TOOLS

Throughout the term, our team utilised 3 main project management tools to help us keep track of the tasks to be completed, the deadlines for these tasks and the allocation of team members across these tasks. These tools (all Atlassian products) were the Jira Roadmap, Jira Board and Confluence.

The Jira tools in particular were chosen because they are highly effective in helping teams to organise future tasks and deadlines, they give visual depictions of what needs to be done and when. In Jira, projects can be broken down into epics. Epics are major milestones or bodies of work, which can then be broken down into issues which are smaller tasks that need to be completed. An example of one of our epics was “D4 Viralog Final Demonstration”, and an example of one of our issues within this epic was “Write Introduction part of script”. As our team utilised the Agile Methodology while developing our platform, Jira was the perfect software for helping us stick to Agile practices, since the epics and issues setup easily translated to sprints in the Agile Methodology.

2.1.1 Jira Roadmap

The Jira Roadmap shows a timeline of all epics that need to be completed within the project and their due dates. The Roadmap allowed our team to keep track of the project as a whole, it gave us the big picture of where we were at in relation to the entire course. It helped to coordinate the organization of our team, it kept us in sync and allowed us to keep track of and visualize the main deadlines throughout the course. As development was broken down into smaller teams, the Roadmap was our integrated platform which allowed us to keep track of what the other teams were up to and the goals that they were trying to achieve, as well as the deadlines for when these goals had to be completed.

The Roadmap focusses on work at the epic level. Team members could click on an individual epic to see a description about it, as well as which team members were responsible for it. Then the epic would also show the status of the individual issues within the epic, and was also capable of showing which epics were being “blocked” by other epics. An epic is blocked when it is dependent on the completion of another epic before it can be completed. We found this feature incredibly useful, it allowed us to visualize which tasks we needed to prioritize completing when there were team members who were experiencing blockages on the epics they were trying to work on.

2.1.2 Jira Board

While the Roadmap focusses on bodies of work at the epic level, the Jira Board allows teams to zero in on the work required within an epic at the issue (individual task) level. The Board shows the status of individual issues (e.g. to do, in progress, in need of review or done), the team member assigned to completing the issue, and the epic which the issue belongs to. There is also the option to add due dates to issues, but our team chose not to use this feature since we wanted more flexibility and instead stuck to just assigning deadlines to the epics using the Roadmap tool. Within the Board, issues can be filtered by epic, allowing team members to see which tasks are still to be completed (within that epic), and which team members are responsible for completing those tasks.

Our team found the Board to be a great way to visualize and keep track of the tasks that we needed to complete. When planning new epics, we found the board to be immensely helpful in allowing us to evenly allocate the work amongst team members. In the planning process we could allocate an issue to a certain team member, and they would receive an email about the issue that they had been given. Since development was broken down into smaller teams, the Board was where the individual teams

(e.g. the D4 presentation team which consisted of Bejai and Alex) could come together and collaborate. The board allowed the smaller teams to see only the tasks involved with completing that epic, and they could see the status of those tasks, for example, Bejai could move the issue "Write Introduction part of script" to the done column, which would let Alex (the only other team member working on that epic) know that Bejai had completed the introduction. The Board also allows issues to be filtered by team member, which we found useful when we wanted to see all of the tasks that we had individually been allocated. We found the Board to be super helpful in facilitating team collaboration, it united developmental teams around a single goal and promoted incremental development.

2.1.3 Confluence

Confluence is a wiki-styled team workspace which allows people and organisations to create spaces and pages. Confluence spaces help structure teams so that team members have access to the correct information. Team members within a certain space have access to the spaces' pages. When creating a new Confluence page, users have access to more than 70 templates which helps to make life easier.

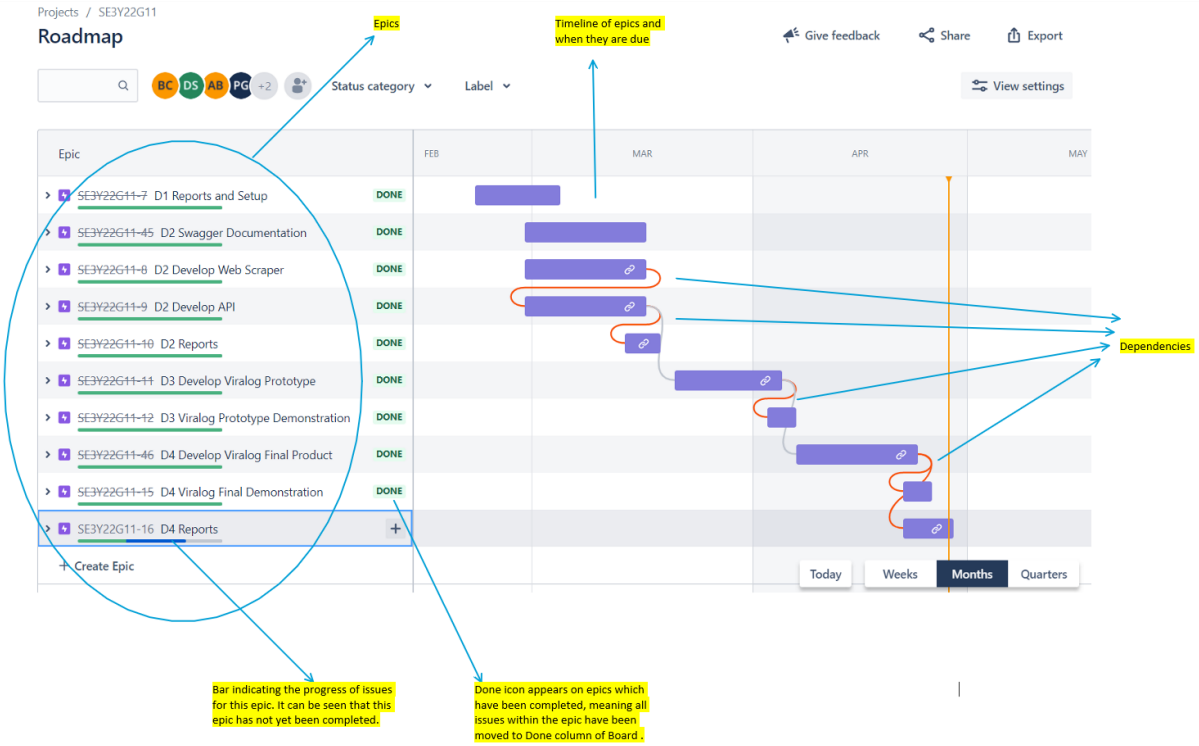
Our team chose to use Confluence since it was a simple and effective way to organise and plan our work. It allowed us to create our own pages which acted as 'folders' for other pages, and we could add whatever information we liked to our pages. We found the page templates to be extremely useful, for example we utilised the meeting minutes template to create separate pages for some of our meetings, which saved us valuable time when taking our meeting minutes (see section 3.1.1.1).

We decided to use Confluence as the central location for our documents throughout the project. We were originally planning to use Microsoft Teams, but once we started using Confluence, we realised that it would be much more useful for us. Confluence is far more flexible than teams, making it much easier to organise our documents. Confluence also gave us access to templates which we wouldn't have if we'd stuck to using Teams. Finally, although Confluence has a steeper learning curve, it is a tool that is widely used within industry, so it was a worthwhile investment of our time to learn it during the project since we will likely need to use it once we graduate.

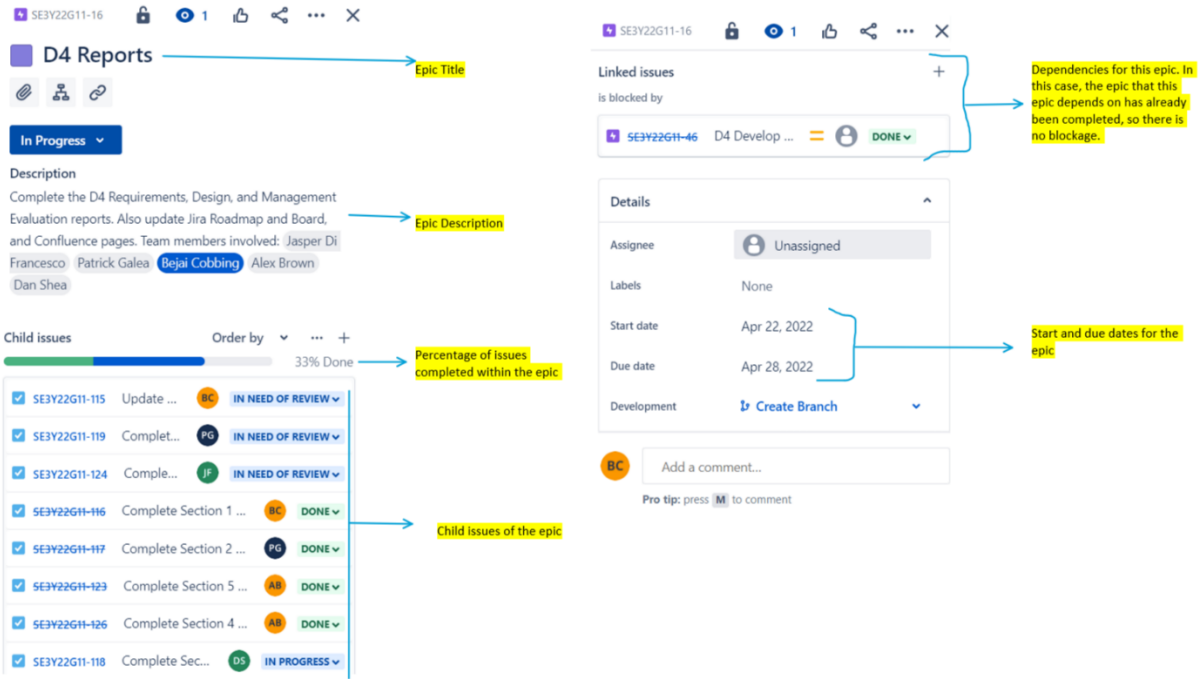
While we were originally planning on only using Confluence to keep track of our documents, some team members still had a personal preference to using Microsoft Teams. This was mostly due to Teams being the platform that they were most familiar with. For this reason, and because it didn't require much extra effort, we decided to also put all the documents on Teams (as well as Confluence). This meant that team members had the choice of where they wanted to access the documents from, but it also meant that documents needed to be posted to two separate locations.

2.2 Annotated Diagrams and Walkthrough of Management Tools

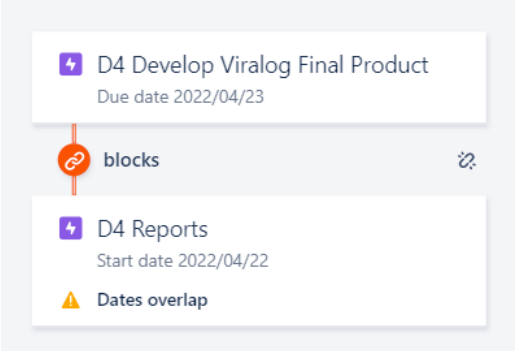
2.2.1 Project Roadmap



Above is the Jira Project Roadmap for our project. The Roadmap is a timeline of all the epics within the project (which can be seen to the left of the diagram). To the right of the diagram there is a timeline of each of the epics and when they are due. In this screenshot, most of the epics for our project have been completed, which is indicated by the full green progress bars underneath the epics to the left and the DONE icons beside them. However, the epic titled ”D4 Reports” does not have these, the progress bar indicates that there are still issues to be completed. If the individual epics are clicked on, Jira will display more information about the epic, including a description of the epic, the percentage of issues completed within the epic, the child issues which need to be completed, dependencies for this epic and the start and due date of the epic:

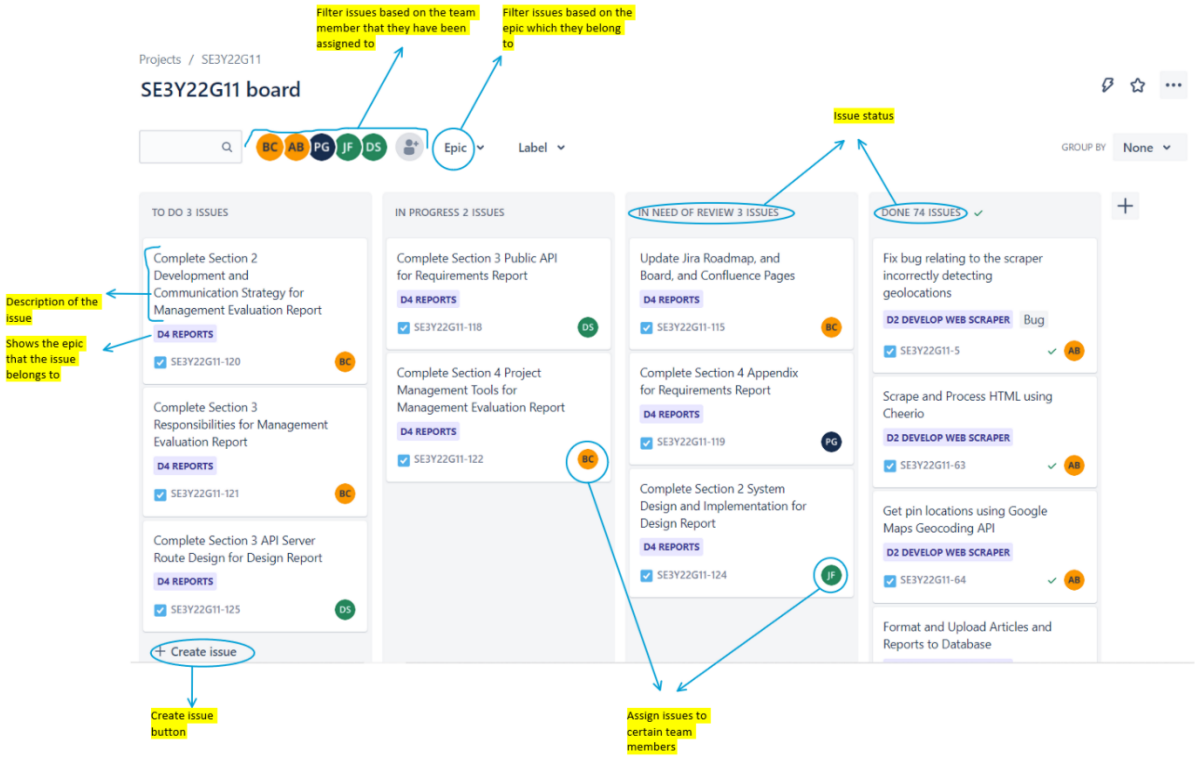


The lines between some of the epics (in the first picture) indicate dependencies. A dependency is when one epic depends upon another in order to be completed. Epics that are depended upon should be prioritised to be completed quickly so as to avoid blockages, which is when a task cannot be completed since it is waiting on the completion of another task. For example, in the first picture, “D2 Develop API” is dependent upon “D2 Develop Web Scraper” since the API could not be completed without the web scraper being completed first. In the second picture, we see that ”D4 Reports” is dependent upon another epic, but everything is all good since the other epic has already been completed. This can be seen in even further detail by clicking on a dependency from the roadmap:



Here we see that the issue that “D4 Reports” was dependent on was “D4 Develop Viralog Final Product”, which makes sense because as a group we needed to finish creating our platform before we could write a report about it.

2.2.2 Project Board



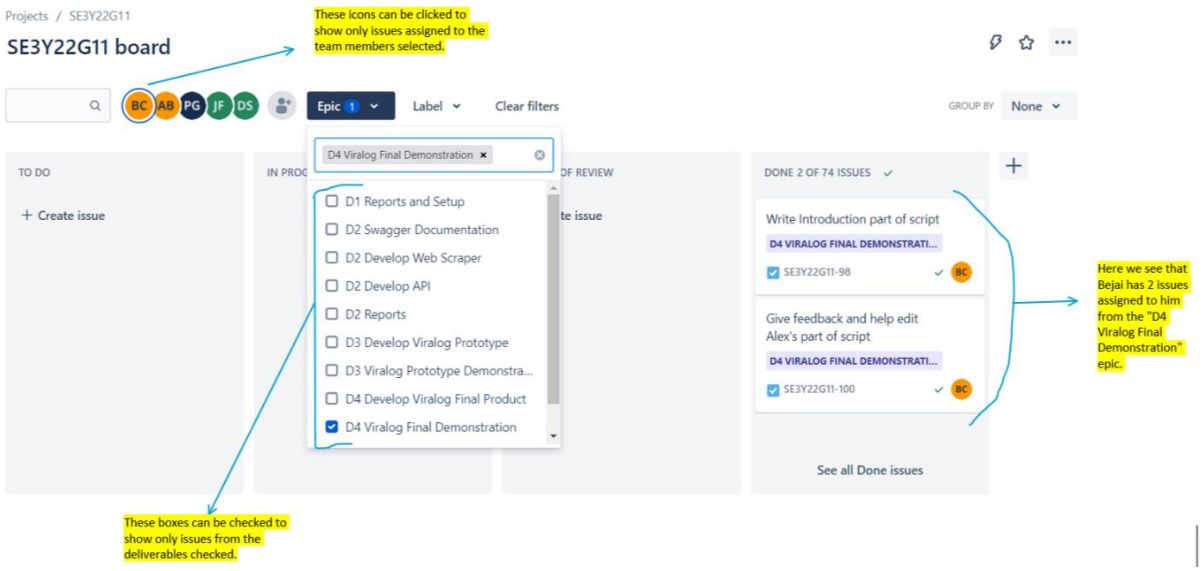
Above is the Jira Project Board for our project, which shows all of the individual issues throughout the project. The issues are split up into different columns dependent on their status. The 4 possible statuses for issues are the following:

- To do – Issues which need to be done.
- In progress – Issues which have currently being completed.
- In need of review – Issues which have been resolved (or features which have been implemented) and need to be reviewed before being pushed into the deployed version of the product. If the issue not approved, it would get moved back into the 'in progress' category where the team member who was assigned to that issue will need to make suitable changes.
- Done – Issues which have been resolved and reviewed.

The create issue button belongs in the to do column, which means that when issues are created they automatically start with the do status. As team members begin to complete the work involved with individual issues, they move the issue into the in progress column, and so on until it gets to the done status (which will trigger the Roadmap progress bar to automatically adjust for that epic).

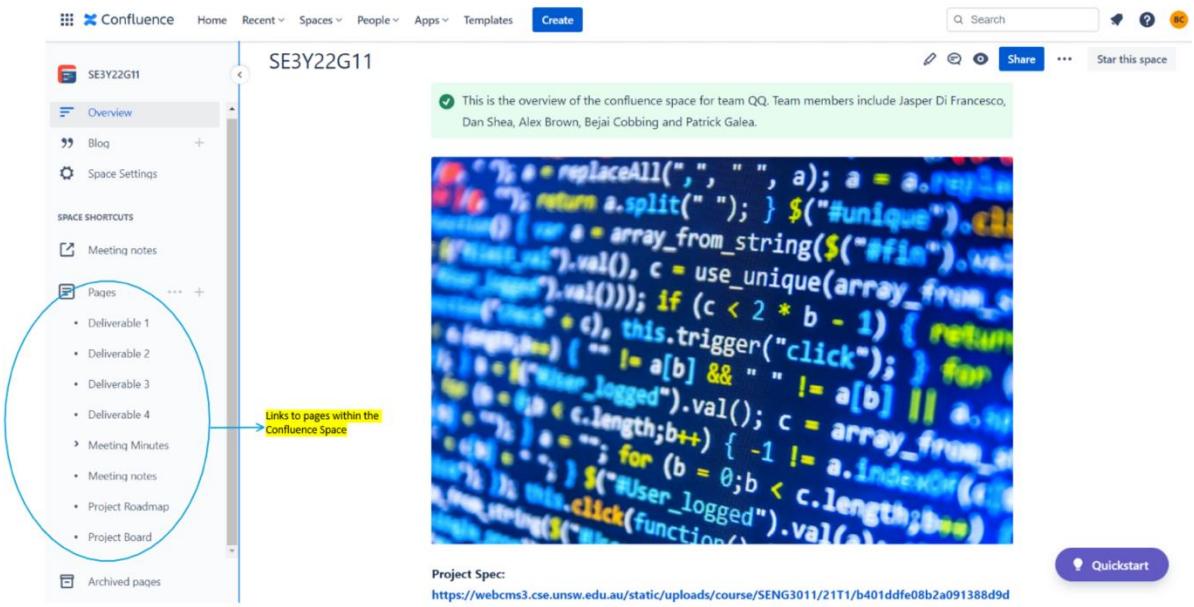
As well as a status, all issues also have a description and belong to a certain epic. Issues can optionally be assigned to certain teammates. For example, if one of our team members was to look at the Board right now, they would be able to see that Bejai has been assigned the issue of completing this section of this report, that this issue is a part of the "D4 Reports" epic, and that he is currently working on it.

The Board can also be filtered by epic or team member:



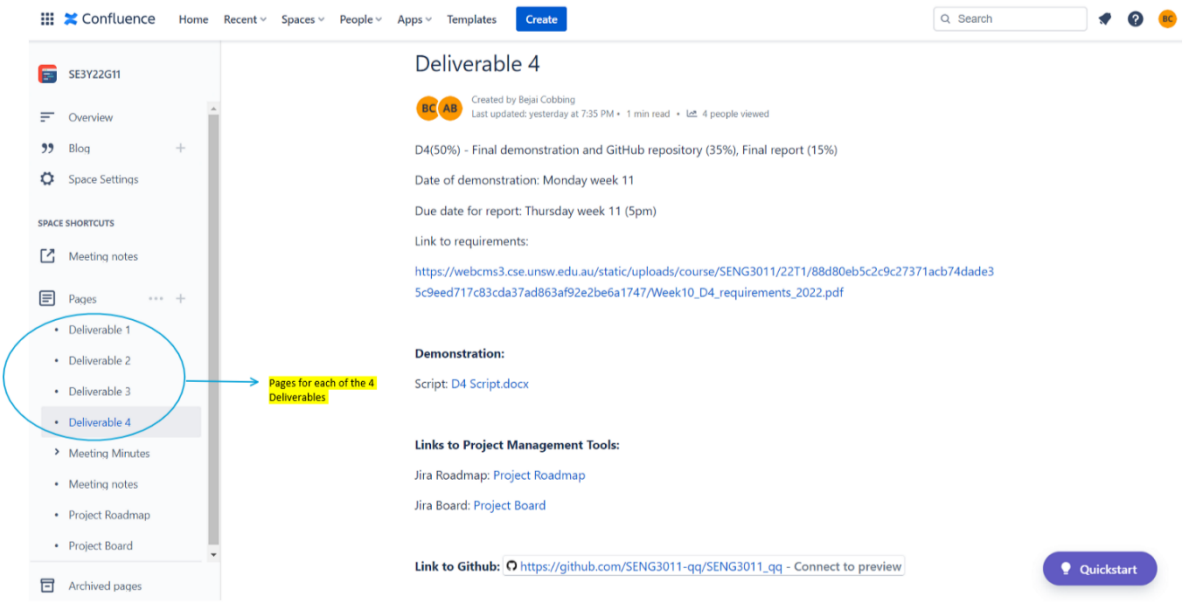
In the above diagram, Bejai has been selected as a team member and "D4 Viralog Final Demonstration" has been selected as an epic. There are only 2 issues returned, both of which have the done status.

2.2.3 Confluence Space Overview



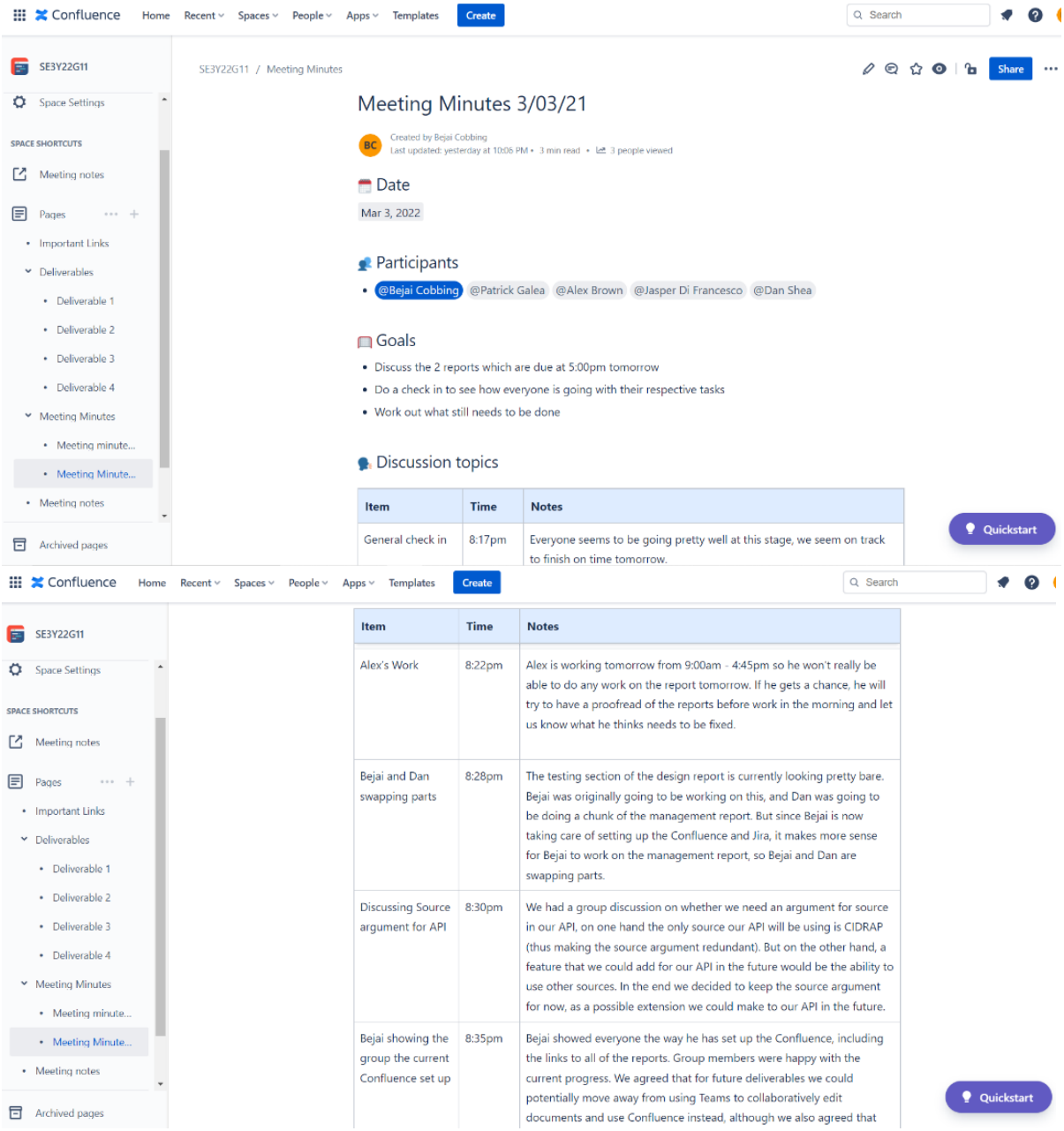
Above is the overview section of the “QQ” space on Confluence. The QQ space has each of our team members in it, so we all have access to each of the pages in the space. These pages can be seen on the toolbar to the left.

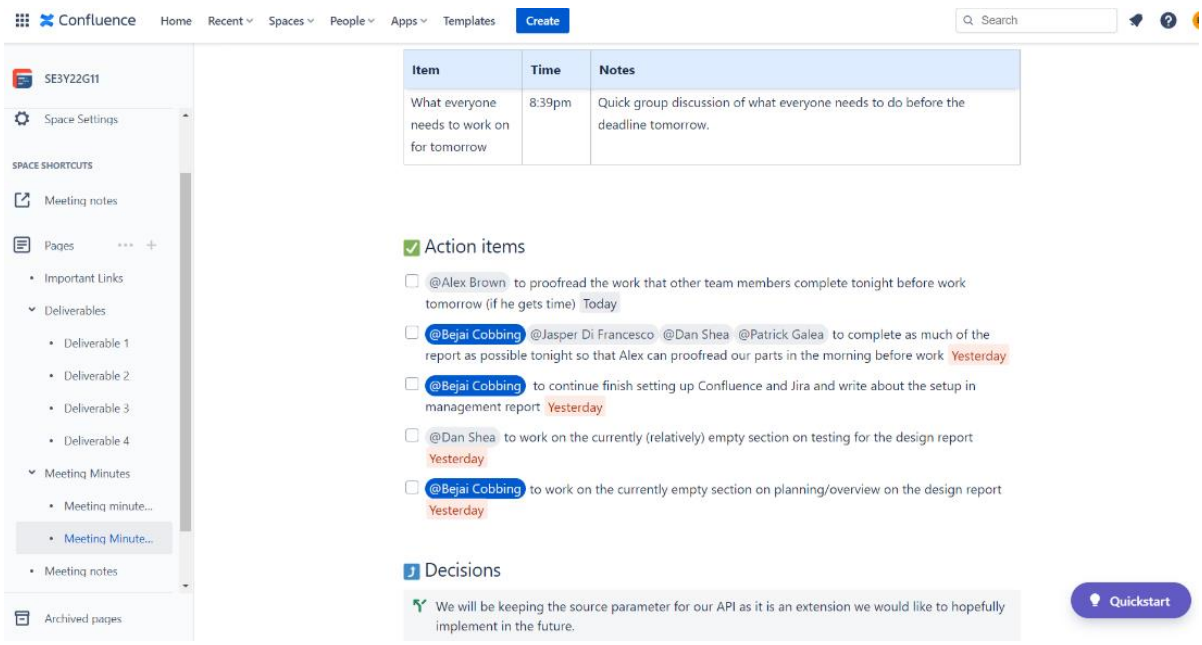
2.2.4 Deliverables Pages



The above diagram shows (part of) the Confluence page for Deliverable 4. Our team has created separate pages for each of the 4 deliverables (the other 3 deliverables pages are very similar to this one). This deliverable 4 page will later contain links to the PDFs for all of our reports for this deliverable.

2.2.5 Meeting Minutes Pages





The above pictures show an example of one of our Meeting Minutes Confluence pages. It was created using the meeting minutes Confluence template, which made our lives a lot easier. As discussed in section 3.1.1.1, we only used these meeting minutes templates for the first few weeks of the term, but for the time period that we did use them they were extremely effective in helping us to keep track of information from meetings.

3 Communication and Development Strategies

3.1 Communication Strategy

A key factor which helped drive our team's success throughout the term was our effective communication. Our main methods of communication were team meetings and our Messenger chat. We also utilised project management tools such as the Jira Roadmap and Board to help us communicate, these were discussed in detail in section 3.

3.1.1 Team Meetings

Our team generally had at least one team meeting per week, and often when we had deadlines approaching this number would increase by a great amount (sometimes up to 6 meetings per week). On top of these meetings, we would also have a weekly mentor meeting with Rue, who would fill us in on upcoming requirements for the project and give us feedback on past deliverables. All these meetings were facilitated through Microsoft Teams, since it is a reliable platform that we were all familiar with. Some epics involved only a subset of our team, and in those cases team meetings were conducted amongst just those team members involved.

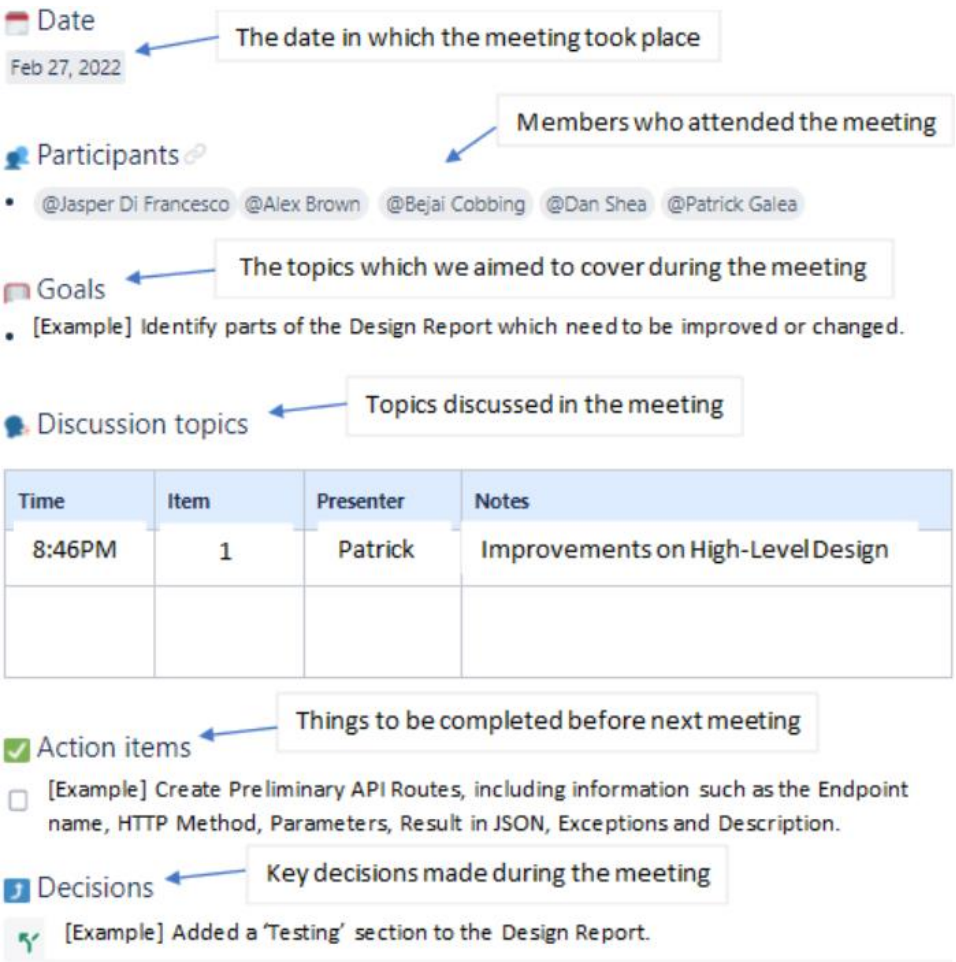
In team meetings, all team members would discuss what they had achieved or worked on since the last meeting, and what they aimed to accomplish before the next meeting. The purpose of this was to ensure that all team members were consistently up to date with the progress of the project. The team would also discuss future epics and would look at (and possibly update) the Jira Roadmap so that we could all get a general understanding of our long-term goals and where we were headed. We would then discuss the current epic/s that team members were working on, and look at the Jira Board and allocate issues to different members. This was so that everyone had a solid understanding of what they were supposed to be working on, so that the team could be structured efficiently and there

wouldn't be any overlap. Finally, during some team meetings we would conduct code reviews so that we could learn from one other and improve the quality of our code.

Originally, we planned to do some team meetings in person throughout the term, since we all live close to campus. However, we all had super busy schedules this term and we struggled to find times when everyone was available to meet, so we ended up sticking to just online Teams meetings as this gave us more flexibility.

3.1.1.1 Meeting Minutes

During the earlier parts of the term, for every meeting that we had, one designated team member would note down meeting minutes. These meeting minutes were kept track of using Confluence meeting minutes pages. The Confluence meeting minutes template can be seen below (see section 2.:



The benefits of recording meeting minutes included:

- Allowing team members to keep track of what happened during certain meetings. Team members could refer back to the meeting minutes from a particular meeting to refresh their memories on the decisions that were made during that meeting.
- Members who missed a meeting could keep up to date with what was discussed.
- A record of who completed what can serve as evidence for how well work was shared between group members, and roughly how much each team member completed.
- The process of recording what work has been done, what work needs to be completed and the timeframe to do so can serve as a beneficial reminder and motivation boost to get that work completed.

Unfortunately, our team only ended up recording meeting minutes for the first few weeks of the term. For the weeks when we did record meeting minutes, they took quite a while to write up, so we agreed that the extra time taken to do so wasn't worth the reward they posed. While the benefits of recording meeting minutes are quite pleasing, we had to make a tough management decision to prioritise writing our code. We decided that developing our web application and public API were better ways spend our time, and that our consistent communication within meetings was effective enough that we could get away without writing meeting minutes for every meeting. If we'd been given more time for the project then writing up minutes after every meeting is definitely something that we would have done.

3.1.2 Messenger Chat

The other main communication method used by our team was our Facebook Messenger chat, which we communicated on typically at least once per day. We chose Messenger for several reasons:

- It is very convenient for group chats.
- We were all very familiar with it, most of us have been using Facebook and Messenger for years.
- We all already had it installed on our phones.
- Messenger sends a push notification to members phones when another team member sends a message.

We used our Messenger chat to discuss day to day challenges which we were all facing. Generally, Messenger was used to discuss the individual tasks people were working on within epics. This is in contrast with our meetings in which we would discuss bigger picture plans and ideas. We also used our Messenger chat to plan out when we would be having future team meetings.

3.2 Development Strategy

Our development strategy was focused on a prototyping approach, where we would start by developing a limited working model to gain an understanding of how our ideas would translate to implementation. From there, we would continue to add features and continuously test our product. This was in line with Agile software development principles.

When integrating new features to our platform and API, we would use the Jira Roadmap tool to create new epics and set the deadlines for them, and then the Board tool to add issues to these epics and assign the issues to team members. When assigning tasks to team members, we tried to cater to the strengths of the different team members in order to increase the overall team efficiency.

Since we were faced with a great amount of time pressure to complete the project, and all of us were also completing other courses in parallel with this one, we needed a strategy to develop code quickly and efficiently. We did this by developing in short, coordinated sprints in which all team members prioritized this course. This method led to higher productivity as team members' attention was focused on a single objective instead of being spread over multiple priorities.

4 RESPONSIBILITIES

4.1 PROJECT MANAGEMENT RESPONSIBILITIES

Each member of the team was assigned to a specific role in the Agile development methodology. We found the most important roles that we would be utilising were product owner, scrum master, retrospective facilitator, head of testing and UX designer. Each of these roles was designated based on each member's previous experience, with each position and justification of choice for the role detailed below.

4.1.1 Product Owner

Bejai Cobbing took on the role of the product owner within the Agile team. His responsibilities included defining user stories and prioritising the team backlog so that product progress was maximised, and all developers were working efficiently. Within the stories responsibility, he ensured that the stories were meeting the user's needs and encompassed all the desired functionality of the system.

Bejai is skilled in writing accurate user stories and making sure they always link back to the objectives of the project. Additionally, Bejai is a good leader and can make well thought out decisions regarding the direction of the project, so he was best suited to this role.

4.1.2 Scrum Master

Alex Brown took on the role of scrum master within the Agile team. His responsibilities included facilitating the development of the team and ensuring they followed the agreed-upon processes. By way of facilitation, Alex worked on removing obstacles that would be impeding the team's achievement of goals.

Alex has experience building applications at scale and was on the architecture team so hence has the broadest knowledge of the system. He was effective in removing obstacles and ensuring all team members were on track.

4.1.3 Retrospective

Jasper Di Francesco took on the role of retrospective facilitator in the Agile team. His responsibilities were to facilitate the sprint retrospectives, which were designed to ensure that the team was building the habit of continuous improvement. By doing a retrospective at the end of a sprint, the team was able to discuss any issues they encountered and could learn from their mistakes to improve their abilities in the next sprint.

Jasper has experience leading retrospectives in different teams and helping people to find methods of improvement and given this was suited to the role.

4.1.4 Head of Testing

Dan Shea took the head of testing role in the Agile team. The role of the head of testing was to ensure that all the testing done on the system was thorough and that we had as high branch coverage as possible. Another responsibility was to enforce the process of writing tests as we would write a function to ensure that as much of the function was covered as possible. By having someone assigned to lead the charge by way of testing, we could ensure the system was as bug-free as possible and would function as expected in as many cases as we could anticipate. Given time constraints while building the front end platform and the pace with which the API was being changed to accommodate new features, the team was not able to write a formal testing suite for the API and front end. Despite

this, Dan was able to spend a significant amount of time conducting tests on the platform by using the site in various ways to try and find bugs.

Dan has extensive experience writing rigorous tests for software and is skilled in thinking of edge cases for functions. Given Dan's experience testing and knowledge of different testing frameworks, Dan was best suited to this role.

4.1.5 UX Designer

Patrick Galea took the role of UX designer in the team. UX designers are integral to the success of a website and the customer experience. They are specifically concerned with; the creative design of a website, the ease of searching for information and the links between pages. Patrick focused intently on making the user experience as smooth and as intuitive as possible. This is particularly important when having to mix and combine such a large amount of data and present it in a way that is logical.

Patrick has extensive experience in customer service and design and was able to effectively apply these skills in a development environment to ensure our system caters to the user in the most effective way possible. Given Patrick's experience, he was best suited to this role.

4.2 Development Responsibilities

Within the development team we split into multiple different teams. These are as follows: API server team, testing team, web scraping team, frontend team, and architecture team. We found that throughout the development process, depending on what was required, members had to switch between teams to match demand for development in that area. Fortunately, the agile style of development supports this team fluidity. Figure 1 shows our planned team structure and how it changed over the course of the development process. As you can see, the teams mostly went as planned and the actual members were usually consistent with the planned members.

Team	Planned Members	Actual Members
API Server team	Jasper Di Francesco Patrick Galea Dan Shea	Jasper Di Francesco Patrick Galea Dan Shea Alex Brown
Web scraping team	Alex Brown Bejai Cobbing	Alex Brown
Testing team	Bejai Cobbing Patrick Galea Dan Shea	Bejai Cobbing Patrick Galea Dan Shea
Frontend team	Alex Brown Jasper Di Francesco	Alex Brown Jasper Di Francesco Bejai Cobbing
Architecture team	Alex Brown	Alex Brown

Figure 1 - Development team planned members vs actual members

Below are details of the specific responsibilities of each team.

4.2.1 API Server Team

The API server team is concerned with managing the routes for the API, which involves collecting the data from the database and organising it into a JSON structure. This team also be managed connections to the database and the schema of the database, to ensure all the data is consistent, correct and only authorised parties can access the database. The API server team is only concerned with reading data from the database and not writing to the database, the main difficulty was writing efficient SQL queries which can give the user back as much information without repeating requests to the database. The API team had to be very dynamic as the features in the project were constantly updated and changed.

The team implemented far more advanced searches than simply finding a disease with a location and time period. This involved techniques such as fuzzy searching for disease names, disease aliases, and symptoms. The team also implemented features such as outbreak prediction, request logging and advanced searching.

4.2.2 Testing Team

The testing team has the job of ensuring the software is as bug-free as possible. This included writing black-box tests for each function to make sure we are getting the correct output based on input. Unfortunately, given time constraints, formal code tests could not be written for the final version of the API or the frontend. However, the platform, including all the data analysis tools and data fetching tools, was tested rigorously through usage by the testing team.

4.2.3 Scraper Team

The web scraping team had the task of collecting articles from the CIDRAP website, and sifting through them to find keywords like locations, diseases, and symptoms, then collecting the latitude and longitude values from the Google Geocoding API from the report location. They then write that data to the database, creating the article object and any report objects that exist within that article.

The scraper team was also tasked with integrating the data from the Team Viral API, which contained data on many reports found in articles on the World Health Organisation website. To remove the need to repeatedly make requests to the Team Viral API every time a user entered a page on the site, it was best to collect data from the Team Viral API during the scraping process.

4.2.4 Frontend Team

The frontend team was tasked with building the platform to display the data from the API in the most efficient and clear way possible. A challenging task that came with this was making the UI intuitive and easy to use so that any new user on the site will immediately be able to interact effectively with the data presented. The frontend team was also tasked with choosing the colour scheme, logo and all other UI design elements that come with producing a platform. Many of the features we wanted to implement on the frontend required some manipulation of our API, so the API and front end were often being developed concurrently with changes on the API being driven by features on the frontend. The frontend team was tasked with ensuring the API team knew what to develop.

5 APPRAISAL

Overall, we were very happy with how the project went and we felt like we did the best we could, given the constraints we had. The main constraint we faced during development was time. All team members had two other courses worth of work to complete in parallel along with other commitments such as work. This caused development to be more rushed than we would have liked, however the team still ensured deadlines were met to a high standard. Due to these commitments, development was conducted in short, coordinated bursts (sprints) in which team members prioritized the subject. When interleaved and balanced appropriately, this method led to higher productivity as team members' attention was focused on a single objective instead of spread over multiple priorities. These coordinated sprints happened regularly, usually once every week or two depending on deadlines, and we believe they played a key part in our success. The one downside to this approach is that it is very intense and sometimes leads to tasks being done last minute. If we were to do the project again, we would make a greater effort to start tasks earlier to avoid being swamped with work close to the deadline.

A key part in our management of the highly time constrained project was good communication. Our communication during the project was very strong through our regular team meetings – usually at least once per week – and our almost daily communication through Messenger. We made sure to meet more closer to deadlines to ensure they were met to a high standard.

Another strong point in our team was our allocation of tasks. Regarding implementation, we ensured members were allocated tasks based on their strengths and interests. This is important as it maximises the productivity of the team. Members who are interested in a particular area are more likely to invest time into it and complete it to a high standard.

Another key aspect which helped drive good communication was a constructive and collaborative learning environment. This meant that when any team member was unsure about how to perform a task, other members were keen and supportive in providing them assistance.

Another factor that contributed to our team's success was our use of project management tools such as Jira Roadmap and Board. These tools allowed team members to visualize the work which was required of them, as well as keep track of the deadlines of when they needed to have tasks completed by.

5.1 ACHIEVEMENTS

There were a few achievements we made in the project which we think are notable. First and foremost, would be the map page. Although it is far from optimal (as discussed in limitations below), the map page is a very useful and complex feature which relies on several systems. First, the data from CIDRAP and Team Viral's API is scraped and processed. In the processing stage, the scraper users the Google Maps Geocoding API to acquire latitude and longitude coordinates for the pins. This information is saved in the database to be queried by the API server. The Viralog frontend performs a request to get these reports, and it uses the *react-leaflet* library to render the interactive map. The reports are then grouped by location and a pin is created with the specified coordinates. Each pin (report group) has a popup element generated which contains the list of articles which contains reports at the specified location. This was a big achievement as it required a lot of coordination between systems transforming the data each step of the way.

Another big achievement was implementing the interactive graphs, in particular the report frequency graphs. The main challenge was converting a list of reports with different dates into a set of time periods with the corresponding report frequency. Jasper wrote an intuitive algorithm in the API server which calculated these frequencies, then used the *Charts.js* library to render these graphs. This was a big achievement which significantly improved the usefulness of the site as it allows users to visualise the activity of a specific disease over time.

Another notable achievement was our disease watching feature. This feature was also implemented by Jasper and allows users to save diseases they want to see summaries of on the dashboard page. The main notable aspect of this feature was that it saves the data persistently without the use of accounts. To do this, Jasper decided to save the preferences in cookies which saves the information persistently in the browser. This feature was an achievement as it allowed users to customise the site without having to go through the hassle of creating an account and remembering passwords.

The final notable achievement was the universal search bar. This feature tied together the site by allowing users to search for diseases based on disease names and/or symptoms. This universal search was notable for a few reasons. First and foremost, it uses the library *Fuze* to perform a fuzzy search, which means users do not need to spell the diseases or symptoms correctly. This is very useful for the average users as disease names are often new to people and hard to spell. Another important part of this feature which makes it an achievement is that the diseases can have multiple aliases which are matched against the search. For instance, “rubella” is an alias for measles, so if a user types the incorrectly spelt “rubela”, it still returns measles as a result. This is crucial to make the search useful as users cannot be expected to know the exact disease alias used by the site. The final reason why this search bar is notable is because, as described earlier, it is universal – users are able to type in both disease names and symptoms in the same input. This is important as separating the inputs adds unnecessary complexity which may confuse users. The combination of all these notable features makes the search bar an achievement we are proud of.

5.2 CHALLENGES

We encountered many challenges during development. One of the main challenges was successfully finding and integrating another team’s API. When deciding on which API to pick, none of the other team’s APIs fit our needs perfectly. Many didn’t function correctly, and out of the ones which did, many had missing information, or their routes didn’t match how we wanted to fetch the data.

We overcame this issue by picking the API which matched our requirements the closest and settled on Team Viral’s API. To overcome the missing information and missing route functionality, we decided to process the Team Viral API data on the scraper instead of the API server or frontend. The scraper fetches the articles from the Team Viral API, processes the data to the format which the system expects and saves it on the database. This approach had many benefits. First off, it significantly reduces the load put on our API server, the client’s web browser and Team Viral’s API. This is because the processing is done once per hour by the scraper and doesn’t have to be performed for each client request.

The main challenges we encountered while processing Team Viral API data were that it didn’t include pin coordinates, the dates were in an incorrect format and that some articles returned had dates in the future. To solve these issues, we used the Google Maps Geocoding API to get pin coordinates, fixed the date format and filtered out articles with dates in the future.

Another issue we experienced during development was that we had to keep rapidly evolving and fixing our tests as the output of our API evolved. We employed test-driven development by developing tests before we wrote the code. As we updated the API to add new features to the frontend, the old tests would fail, and we had to update the tests very frequently. This is an expected side effect of test-driven development and isn't necessarily a bad thing as it ensures your code is well tested throughout the whole development cycle. The main downside to this is that a lot of unnecessary time was spent updating tests which led to less work being performed on the actual functionality. Due to the importance of testing, we wouldn't do anything differently if we were to do the project again.

The next challenge we experienced during development was related to how we split up tasks between the backend and frontend. A few team members were either experienced or capable in only one of the two fields, which made implementing features more difficult. When implementing a feature, at least one backend and one frontend capable person must work on it. There was a bit of loss of information in the communication between these two parties, and often one implemented the feature slightly different to what the other was expecting. This slightly increased the time taken to complete features as multiple changes were often required on both the frontend and backend before they matched. If we were to do this project again, we would ensure the two parties had a solid understanding of the exact interface and functionality before implementation.

Another challenge we experienced was COVID-19. Alex got COVID in week 5, near when deliverable 2 was due, and Dan got COVID in week 10 when deliverable 4 was due. Luckily, we were granted extensions which helped us through the period where team members were out of action.

5.3 System Limitations

Although we are happy with the system we managed to implement, there are certainly a few things we could change or add to improve the user experience. Most of these limitations were brought about by the time constraints of the project, and we prioritised the features we believed were most important.

The first addition we would make to the project is that we would add comparison functionality between datasets. This would make the dashboard page much more useful as power users could use it to see if there is any correlation between datasets such as diseases in different countries.

This leads onto our second addition we would make. Currently there is no way to view country-specific disease information. This makes the site a lot less useful to people who are only interested in a particular country, such as government officials. This could be used with the comparison feature to compare the activity of diseases between countries.

Another limitation of our system is that it only takes data from two data sources: CIDRAP and the World Health Organization. Ideally, our system would integrate dozens of data sources to get a more comprehensive view on the activity of diseases around the world.

Another improvement we could make would be to improve the code quality. Although we believe we wrote pretty good code for the time we had, the code could definitely be structured better to improve readability and scalability.

The next limiting factor is that our project is spread over multiple hosting providers. The scraper and API server is on Google Cloud, the database is on AWS and the frontend is hosted on Vercel. The main constraint we had which led to these decisions was money. We initially planned to host the entire site on Google Cloud with the free trial, however we noticed that the database was using up almost the

whole budget. We decided to move the database onto AWS because it had a better free tier for databases. We decided to host the frontend on Vercel as it is a free hosting service for Next.js where you can deploy your site with a single command. In an ideal world, we would host the entire site on the Google Cloud Platform as it would significantly decrease management overhead, however we were unable to do this due to cost and time constraints.

If we were to do the project again, we would put more detailed thought into the purpose and target audience of our site before rushing into the design. This didn't end up being an issue as we were able to mould our design to fit a solid set of use cases during development.