**Graph Algorithm Tool Logbook**

**Sprint 1**

**Week 1 September 1st – September 7th**

Implement product backlog

In reflection, is the project at where it should be.

Plan

This week will be responsible for gathering high level requirements for the project I will undertake this year. (5 hours)

Action

Every great product needs to have greater planning to accomplish its goals. So, I want to dedicate significant time thinking about the technological challenges that I have to face and the best way to combat them to achieve my goals.

The first step is to get a list of initial requirements for my project. This proceeded me to look at the brief that was provided by Dr Sanderson on the project information page. This brief provided me with the general first step idea of the vision for the application I will be building. I made sure to note down any questions or concerns I had from reading this. From then on, I conversed with Dr Sanderson over email introducing myself and asking my questions and concerns. He responded clearing up my questions and gave more detail on what he expects in the application and from me. I also asked advice on general high-level architecture recommendations which will help my decision on what my application should be. Furthermore, I have emailed Dr Sanderson for setting up meetings in early October for an initial review and catch up for this project. This ensures that I am on the right track and the vision for this application is coherent with his. After clarifying requirements with him I started to plan my own personal requirements for this project. The brief that was assigned to me has a lot of room for ideas. It is broad enough that I have to make my own requirements that fit under the brief. So, the plan is to come up with my own personal requirements that I want to fulfil. There is a lot of freedom and flexibility for the technologies I can use for this project which is a huge advantage.

Reflection

This was a successful week in the initial investigation for this project. I have adequately looked into the first requirements posted on Moodle for my project. Established communication with my supervisor who answered all my questions. This allowed me to come up with my own requirements that I want to fulfil in this project too. This is a good basis to proceed to the next step of my investigation which is will be technological/planning requirements.

**Sprint 1**

**Week 2 September 8th – September 15th**

Plan

This week will be responsible for researching the most adequate project planning methodology to implement for this project.

Action

It is extremely difficult for a project to be successful without good project planning skills. So, what I aim to do this week is to research and use my anecdotal experiences to seek the best project planning methodology to utilise. This required me to read about various techniques online, the main being Waterfall and Agile. I read most into these two because I have the most experience with them and I know they work well. However, there are pros and cons for each. My task was mainly delving into both and researching which one will be most suitable. I read heavily into this using multiple sources that I added this as part of my background reading for my Initial report. Sources that I delved into were articles from companies, anecdotal blogs, YouTube videos. With all this research, it allowed me to make a good decision on the project planning methodology that will be most suitable for my project.

Review

Consolidating all the new information that I read online and gathering thoughts from past experiences I did come to a decision on what I will be using. This means I had a successful week as my goal I wanted has been achieved.

**Sprint 1**

**Week 3 September 18th – September 24th**

Plan

This week will be responsible for defining technological architecture. (5 hours)

Action

My first technological research was just simple high-level design decisions. Such as: Should it be a desktop or web application? Will the user need internet to operate the application? How many different algorithms should the applications start with? This is just a few high-level questions that I have to ponder on. With a couple of days of consolidating thoughts and doing appropriate research (Various, 2017) (Elizabeth, 2017), I made final decisions of what the higher-level design should be. I do acknowledge the fact that some of my decisions can change over the time of the project.

Naturally after this, I had to pick the programming languages I would like to use.  This investigation can be split into two; back-end and front-end. I want to architect this in the way that the back-end side of the application should handle all of the algorithm processing and server/client interaction. The front-end will be for user interaction and the aesthetics of the website. Using my experience during my placement year and considering my proficiency with the languages I made the executive decision on what to use, I will be aiming for a full JavaScript application with Node support.

Following this, As I am using NodeJS for backend and standard JavaScript for front end development I can then proceed on investigating various libraries and tools that will aid my project. I gathered many libraries and tools that would aid my project from graph representation and visualization to request/response MVC controllers.

I also sought advice from experience engineers that I worked with during my placement year. This was especially helpful in some technical decision I had to make. They reviewed my decisions and gave their opinion on it. I took their views into consideration. I have also added this research in my Initial Report as I spent many hours on this.

Reflection

This again was a successful a week, I made important decisions for the technical future of my application. I may have to clarify some points with my supervisor about some points but this can be done in the future.

**Sprint 1**

**Week 4 September 25th – September 30th**

Plan

I will be investigating the best graph libraries to use that will aid my project. (4 hours)

Action

It came to my realisation when doing some technical research that there are robust and reputable libraries that will aid my development for this application. This does not mean that there will be less work as I will still have to develop significant amounts of the teaching application. However, aspects of a graph like node/edge visualisation has been done many times and recreating this will be redundant in terms of time spent. So, my task for this week was to search the web for the best libraries I can use that is compatible with my goals and tech stack. What I looked for is a JavaScript library with good documentation that can provide node and edge visualisation. With good support of on-click functionality to display nodes and edges. It should also be scalable enough so I can modify the actually library itself and build on it. In turn, this should allow me to add my own graph search algorithms. A bonus will be that if there’s an active community I found many suitable candidates for this such as SigmaJS, VisJs and D3.js. I looked heavily into all of them (plus more) and narrowed it down to those three. With hours of research I did finally find a suitable library that will fit all my needs and does have an active community so if I have any concerns I can easily seek help. This search also came with many insightful information about what I want to do and add.

Reflection

This was a very successful week. I found a more than adequate library I can use for my project. This is a good start for the development of this project technical lifecycle. The next sprint is coming up so that is when I start testing and using the library to see if everything is suitable.

**Sprint 2**

**October 1st – October 13th**

Plan

The goal for this story is to test the Graph Library itself so it’s bug free and can do everything I want it to do. (6.5 hours)

Action

I want to dedicate significant time in actually testing library ‘SigmaJs’ before I start, this entail tasks: Investigating the library further to see if it supports all the features that I want to implement in my library, I also want to thoroughly do manual testing across the features that are described to see that there are not any production breaking bugs that can arise from the application.

From all the initial requirements I prescribed, I wanted to see if they were feasible with this libaray. My procedure for this was going through my requirements and reading the documentation and making a test with the library in JavaScript to see if it’s realistic. For example, the core requirement for my project is the able to visualize an algorithm running. So, the task that I set is to see if the library can successfully display nodes on the web page. To test this, I curated a quick template HTML page with the library and with nodes implemented. From this I could see that it’s possible to display the nodes with the library. The bonus is that nodes can be fully customized so I can change the colour, shape and size. Another core feature I wanted to test is the library supports click functionality. The library should be able to support adding a node on the screen on mouse click. So, another task was to test this too. On the documentation for this library there was a very basic click functionality that was provided. I implemented this into my test HTML too and it also proved successful that the library has this feature in place. I followed this principle with many other smaller features to see if the library was worthy enough. Features such as: Left/Right/Middle click functionality, Edge connection, Weights, Graph customization, UX enhancements. The other assignment I needed to complete is do a general bug check for the library. This is just a safety precaution to ensure that the library is fit for ‘production ready’ work. This involved me going through the documentation to see if they were any test implemented by the developers and running it and seeing the passes/failures. It also involved testing various functionalities manually.

Reflection

This story execution was successful! After careful investigation and testing it is concluded that this library will be a perfect fit for my project. All the core features that I strongly desire my application to have can be implemented with the library. Furthermore, my analysis of the documentation shows that there are tests implemented which has a high pass rate so this gives me the security that this is a robust library.

**Sprint 2**

**October 15th – October 21st**

Plan

Add Node to screen on mouse (left) click (5 hours)

Removal of Node from screen on mouse (right) click. (6 – 7 hours)

Action

Add Node to screen on mouse (left) click.

Now I have the confidence of the library, I can start implementing features that I have set out to do in my Agile plan. The first ordeal was node and edge click functionality. The basic idea being that with a standard left click, there should be a node displayed on were the click was positioned. The solution I did for this is make a basic JavaScript file clickFunctionality.js. The idea for this is that this file will be responsible for all behaviour regarding click interaction. So, for my task I started creating a few amounts of functions that will be responsible for executing this functionality which will display a node on the screen when left clicked. A safety check I added is that I also implemented a function were a user cannot put more than one node at the same position. This is implemented to stop confusion for the user as strange behaviours may occur. An additional feature that I applied on this was also making it so you can’t add a node while it is dragging. Overall, this whole feature was around 30 lines after refactoring and cleaning up. This stops confusion for a user if they attempt to add something and the window moves.

Removal of Node from screen on mouse (right) click.

As important as node creation is the ability to delete a node after it has been put on the display. So, the next task was to formulate a way to make the application remove a node. I began this feature with doing some brief research on other applications which share the same issue. From my findings, it was shown that the most initiative way would be a simple right click. I personally liked this idea so the next step was to see if I could technically incorporate this into the application. This required some investigation from the Sigma Js documentation. This did require some in-depth research as it wasn’t a standard feature that came with the library. So, solution for this was that I had to create a ‘left click captor’ myself. This was not overly challenging and I managed to come with an effective solution that suits my needs. So, there were two files overall that were amended. sigma.js and clickFunctionality.js. With the main changes being in sigma.js as this will cater for the new core behaviour and my clickFunctionality.js file will be the higher-level interaction with the application.

Reflection

I’m happy with the result of this feature. As a fundamental feature to the application, I managed to use around 30 lines of code. I thoroughly manual tested this to see the performance and fixed any bugs that may have risen while experimented. Overall, I am happy with the outcome of this feature.

This feature of node removal did take more than anticipated. This is because the default functionality was not supported by the library so I had to implement this myself. However, at the end I did manage to come up with an acceptable functionality to address this issue so I am happy about that. Supposed to be 3 hours took 10 hours.

**October 21st – October 30th**

Plan

The aim of this story is to implement the features:

Add Edge to screen when two nodes are connected. (3 hours)

Removal of Edge from screen on mouse (right click). (1-2 hours)

Action

Add Edge to screen when two nodes are connected.

This is another feature I branched out to do research on. With nodes on the canvas, there needs to

be an intuitive way to connect any of them together easily, creating edges. With research into other applications and personal brainstorming and experimenting, I came up with a design to execute this. The user will simply shift to click on the source node and do it again with the node the user intends to draw an edge to. This required me to firstly create a functionality to select the node. So, expanding clickFunctionality.js, I added code that on shift click, when the cursor is hovering a node, the node will be selected. This would result in the node turning a grey colour to highlight that is selected. I also implemented the code so that an already selected node will unselected if selected. Naturally after this, I implemented the feature that if two nodes are selected consecutively, there will be an edge drawn across them. This was tested and found out it works well.

Removal of Edge from screen on mouse (right click).

This was a smaller task compared to earlier. I just have to remove the edge that has been selected with right click. This a simpler task. So again, in clickFunctionality.js, I added a few lines of code that would support this. This was a simple task as the library comes with a remove feature and I just had to call and implement this in my application.

Reflection

I enjoyed the work that I completed for this. I spent the correctly estimated time on this. Improvements can be made in the future to make more emphasis when a node is selected however this is a minor detail and will most likely get added in Sprint 5. This was successful and there were major bugs that were discovered in the process. So currently the application now supports edge functionality which is a good milestone.

**October 31st**

Plan

Create text input in preparation for algorithm development. (2 hours)

Action

I will be making the back-end HTML/JavaScript for parsing user input. This will be important as this will get information such as which node the algorithm will start from. My focus for this is purely on the back-end and won’t be focusing on the aesthetics of the text box as this will be saved for a future time (more specifically Sprint 6). With that being said, this made the task simpler. I just had to parse the text inside. I also added some validation to it so the values are correct and true to what is being said. I tested this by creating a function that will print out any values entered in the text box. Which worked well.

Reflection

This was an important feature to implement. This will pave the way for Sprint 3 when I start developing the actual algorithms. This is paramount for the testing of the algorithms. So, overall, I am happy with this feature as it does do every I want it to do correctly. This was also a smaller task requiring just basic html and JavaScript so it was done swiftly.

**Sprint 3**

**November 3rd – November 8th**

Plan

Investigation of Dijkstra’s algorithm. (5 hours)

Action

In this story, I am responsible to research everything about the Dijkstra’s algorithm. This will require detailed investigation as this would be one of main algorithms for the application to run. As I last visited two years ago I needed a refresh on what it is and the overall technicalities of it. Firstly, I revisited Dr Sanderson’s material on the algorithm. This is important to do as my algorithm will have to closely follow this as this is what second year students will be most familiar with. So, going through this, I will closely take notes and put down ideas that will be useful for me in the future. Following this, emailed Mike Sanderson about this algorithm specifically asking questions on what I will require for my application. His responses were also noted down. This led me to go to a more general investigation. I was recommended the online course *Algorithms* from Princeton University as they educate about various algorithms, Dijkstra’s being one of them. So, I took the time to go over the section that covers thi8s algorithm which did further my understanding in topic. At this point I had good fundamental understanding of the algorithm but wanted to go further too. So actually, I did watch many YouTube videos from various tutors to get a wider understanding of the topic. On top of everything there were sites like *VisuAlgo* which give interactive examples of algorithms, so I experimented cases in this too.

Review

This is my first research task on the algorithm. This did prove very useful, as I did forget most of the details of this algorithm and this definitely re-jogged my memory about everything. I did learn a lot from various sources. With this fundamental knowledge learnt, I am more confident in how I will approach implementing this in my algorithm. I already have some ideas on ways on how to do this. This also paved the way for me on good sources that teaches algorithms such as course, YouTube guides and books. This contained information about other algorithms I will be using for this application too. So overall a good feature.

**Sprint 3**

**November 10th – November 14th**

Plan

Investigation of Kruskal’s Algorithm (6- 7 hours)

Action

After my Dijkstra’s investigation, I now have a flow on a way to revisit and relearn the algorithms. So again I started with Mike Sanderson’s notes. Going over the information that was given. Kruskal’s algorithm is fundamentally simpler than Dijkstra’s so it was easier to pick up. One idea I had was the ability to watch back lectures of Mike Sanderson’s current teaching of Kruskal’s. This will help me with further understanding of the topic. This again made me take more notes of the algorithm and what it entails. There are variations of Kruskal’s algorithm so I made sure that the main material I research from this is from Sanderson’s as this will be what the students will be studying for their main material for the Data Structures and Algorithms course. Following this I actually picked up watching an informative YouTube series about Kruskal’s algorithm from an individual called ‘Michael Sambol’. This provided a through description of the algorithm and also went through multiple examples showing the steps of algorithm. This was the main algorithm I watched about this but there were more that were informative such as ‘Yusef Shakeel’ and ‘University Academy’ series. I also did visit the Princeton course about this. However, this time, I referred to the book version. This book *Algorithms* written by Robert Sedgewick and Kevin Wayne (who also own the course). Something slightly different that I did from when I was learning Dijkstra’s algorithms was I curated draft pseudocode to help me learn this.

Reflection

Kruskal’s Algorithm concept proved challenging to me so it dedicated more time than Dijkstra’s to fully grasp the understanding. This required me to watch various example and guide from Princeton course and YouTube. Upon doing this, I successfully understood on the theory behind this algorithm. I was also happy with the book *Algorithms* that I discovered. This contain elaborate detail about ­many graph algorithms which proved useful to me.

**Sprint 3**

**November 15th – November 20th**

Plan

Investigation of Prim’s algorithm (5 hours)

Action

This is the last of the three main algorithms I will be investigating. With my previous investigation, I will be using similar sources as they will have information on Prim’s too. So, following the similar pattern, I started looking at Dr Sanderson’s notes for my initial starting point for this making my initial notes throughout the process. Most of the procedure is similar to the other algorithm process. What I did different here compared to the others is that I actually worked with a second-year student who was studying this in lab. So, going through this with another individual working with problems actually enforced my understanding for this. So, to finish of understanding this I once again went through Princeton course. Watched various YouTube and went through exercises educating me. Also with Prim’s Algorithm, Wikipedia has a good description and examples about it so that was a good resource.

Reflection

The learning of Prim’s algorithm was relatively smooth. As this is conceptually similar to Kruskal’s algorithm, understanding this algorithm was definitely easier. So, I feel this took me the shortest time to learn. However, I still spent the predicted time learning this algorithm as I wanted to get a very deep understanding of this algorithm.

**Sprint 3**

**November 20th – November 30th**

Plan

Implementation of Dijkstra’s Algorithm (4 hours)

Action

Now that I have the theoretical knowledge on how Dijkstra’s algorithm works, it is now time to implement it into my application. My first step for this is to write pseudocode for the algorithm without looking at external help (unless I get stuck). So, I went ahead and attempted this writing my code on paper gathering all the knowledge I learnt about the algorithm. After I was happy with my solution, I looked at Dr Sanderson lecture notes for his version. Our versions were different as he had implementations that were more efficient. What I did is I improved some of my pseudocode on the basis of his. After this, I naturally sought out more pseudo implementations online to see if I can make even more improvements. If there were any more adjustments I could make, I would revisit my pseudocode and amend them.

After I made all the changes I want for the pseudocode, It was time to slowly integrate this into the application. This algorithm had to be developed with the graph library in mind. There are core functions from this library that will be helpful to the algorithm. Functions such as sigma. *allNeighboursIndex* is an example as this will get all node and edges next to a specific node. So, I catered the algorithm to incorporate these helpful methods as this would be a good optimization. The procedure was always developing a bit and testing on localhost gradually. Which is made possible as a made a template from sprint 2.

Reflection

This was the first time implementing an algorithm in JavaScript, so this did take significant time as I was still learning the language and the sigma library but I am proud to say that I did manage to complete it. After some small manual testing I was shown that the desired Dijkstra’s behaviour is replicated in my application. Which is good news. I was also very happy that I managed to implement this with just fifty lines of code making it very user friendly. Something I can improve on later is the speed. Even though this would run very fast on all modern computers it is always a good skill to try and optimize as much as possible. This is the creation of Dijkstra.js

**Sprint 3**

**November 20th – November 30th**

Plan

Implementation of Kruskal’s Algorithm (4 hours)

Action

Following the success of the implementation of Dijkstra’s algorithm, I was more confident developing Kruskal’s. Following the same routine, I started to create pseudocode for Kruskal’s algorithm. Again, on paper, I went through this. I actually came up with two slightly different pseudocode solutions for this. One implementation involving a tree and the other just using arrays to solve it. What I desired was the fastest and most user-friendly version. So, I spent time comparing the two implementations to see which one was superior. To my conclusion, I found out the implementation containing the Tree was more efficient and I personally thought it looked cleaner so I decided to go for that. With that being confirmed, I then went to look at more Tree pseudocode online for Kruskal’s algorithm to see if there is anything I could improve. To my happiness, there were no major changes that had to be done using the Tree pseudocode. So, with that being said I started to implement it into my application. This definitely took more time than Dijkstra’s to implement. One of the reasons being that the data structure Tree is not part of core JavaScript so I had to use an external library. This required more investigation and time to get used to. However, in the end I did manage to be victorious and conquered the challenge.

Reflection

Even though learning this algorithm conceptually was difficult, implementing it was a smooth process. I liked the fact that I developed equally correct solutions and had to do further work to investigate which was the superior algorithm. I also had the chance to experiment with more external JavaScript libraries such as Tree.js, which is good experience. I am fully satisfied with the choices that were made and the results out of it. This is the creation of kruskal.js

**Sprint 3**

**November 20th – November 30th**

Plan

Implementation of Prim’s Algorithm (4 hours)

Action

From Kruskal’s which proved a challenged but was difficult I know that Prim’s algorithm would be similar in terms of implementation. So, it should not take as long. With my research earlier, it would user-friendlier to use arrays for this implementation. As I thought this was more straight forward as I am more confident with how to build algorithms and how to approach these problems. This was relatively smooth.

Reflection

As this is similar to Prim’s apart from some few differences this was also easier to implement and get working. So, I am happy with this solution. This is the creation of prims.js

**Sprint 3**

**December 2nd – December 10th**

Plan

Manual Dijkstra Testing (2 hours)

Action

So, after implementing Dijkstra’s the goal is to test it. What I will be doing is testing my implementation with various cases trying to find if there are scenarios that break the algorithm or incorrectly does the wrong path. What I did is that I mimicked some Dijkstra problems online and ported it to my application. I compared the final solution from my application to the problem online to see if path and final result are similar. I would know if they are right if my application follows the exact same path and gets the same result. This proved successful as I discovered few critical bugs under some specific conditions. This also made me list things I would have like to change. Fixes entailed modifying some HTML and JavaScript in index.html so it reflects on the initial inputs before the algorithm and less of Dijkstra.js.

Review

After implementing Dijkstra’s algorithm, I only briefly tested it to see if the main functionality was there. So, this was responsible for more in-depth testing. I was happy with this as it’s always good to spot bugs earlier than later. The list of bugs that were discovered: Application crash if too many symbols entered in text box. Wrong path caused if more than 10 edges on node. Negative values not correctly handled. All of these issues were successfully fixed.

**Sprint 3**

**December 2nd – December 10th**

Plan

Manuel Kruskal’s testing (2 hours)

Action

Similarly, to Dijkstra’s testing, I went ahead and tested Kruskal’s algorithm, following the same procedure of trying with multiple examples online and comparing results with already established answers. Just to make sure that my concept about this is correct, I made sure to have a quick recap on my understanding so its fully correct.

Review

Testing went well, list of bugs discovered: clicking on the Kruskal’s execute button many times will cause it to have delayed start. Start node appears in the wrong colour for Kruskal’s algorithm. These bugs were all fixed.

**Sprint 3**

**December 10th – December 20th**

Plan

Manuel Prim’s testing (2 hours)

Action

Similarly, to other testing, I went ahead and tested Kruskal’s algorithm, following the same procedure of trying with multiple examples online and comparing results with already established answers.

Review

Testing also went very well. There were actually no bugs that I discovered the broke the program. Partly because all the bugs discovered previously effected all the algorithms so upon fixing it prior that means it will fix it here too. However, I had a feature that I want to add in the future in relation to Prim’s algorithm. That is that I want the minimum spanning tree to have more emphasis. That will be added in the following sprints.

**Sprint 3**

**December 20th – December 31st**

Plan

First Review of the application looking back at the past four sprints. (5 hours)

Action

The purpose of this story is to go over everything I have done in the in the past four sprints and review it and see what I could change or review. I also opened up my application to other so they can have a first look of the application. I also had a meeting with my second assessor who also reviewed my application so far. She gave detailed constructive criticism on how my product is so far and what I can do to improve it. This is added to the list. I also went through some Computer Science final students and got some informal reviews about it. Again, these were noted down and shared and noted down.

Review

I am extremely happy on the result of the application from the past four months. Personally, I think the application has potential to be become a good product for student to use which will aid them on their graph algorithm studies. It is also promising that when given the application for other students to use they found it useful and the liked the idea. They also gave meaningful criticism for on features I should add. My second supervisor was also pleased with the progress I had so far. She also gave criticism for feature I should enhance or add in my application. All this feedback was appreciative as that means that I will have many good features to add in sprint 5 which will be implementing features that were advised from people who tested the application or just myself. Some of these features were: Lock node activation while dragging, disable double click zoom, more suitable validation for specific algorithms, emphasis on node select. More detail to what will be changed will be reported in the next sprint.

**Sprint 4**

**1st January - 7th January**

Plan

I plan to go through the list of features that need to be evaluated and see if they would be implemented in the application. (2 Hours)

Action

Over December, my plan was to consolidate all the reviews and opinions on the application. What my aim for this month is to see these reviews and evaluate them to see if they are feasible to implement this month or perhaps later. A small portion of the list of features or changes are those such as JSON Node/Edge input, instructional helper on first visit, skippable iterations. So, I got all these changes and started to plan out if I should do them all not. As I have limited time, I have to choose wisely on what features I have to implement as some of them are more ‘nice to have’ features. For example, the colour cosmetic feature where you can modify the colour of any node or edge is definitely a secondary feature which is does not have to be implemented and was not defined in the actual sprint plan.

Review

I came down with a list with what I want to implement for the rest of the sprint.

**10th January – 31st January**

Plan

Implement the features that have I decided. (20 hours)

Action

Before implementing anything, I made sure to meet up with my supervisor on the first week of term to update him on everything I had done over the Christmas holidays and what my plan for January were. I presented him with the list of features that I wanted to accomplish over this month and in return got feedback on if was suitable to do it all. Most of the features that I proposed were approved by him and agreed that there were good features to implement. The list I consolidated were a from a mixture of other students, myself and my supervisor. I am happy with the fact that I came up with this.

For the rest of the month I started to implement the following features:

**Add a button to reset all nodes and edges on graph (2 hours)**

This was a commonly requested feature that I had from users who were looking at the application. The problem was that if a user draws a whole graph and wants to eradicate it and start again, they have to manually remove every node and edge one by one. So a helpful feature would be to have a clear graph button which will erase everything in one action.

**Add a button to display a pre-determined graph on the screen. (2 hours)**

This feature was in fact thought from myself. If a user did not care about customising their own graph and just wanted to quickly load up a graph for the application, I will implement this feature which will enable to do this quickly. I thought this would be a good user-friendly idea.

**A way to pause and resume the graph algorithm traversal**. **(2 hours)**

Another small feature in which a user can optionally pause the algorithm teacher if they please. A user case for this is that if a user wants to pause to review what was happening in the algorithm so far, they have the ability to stop it in their own will and start again if they please.

**Implement a way to insert nodes and edges with JSON file. (10 hours)**

This feature was requested by my second supervisor. She recommended that there should also be an extra way to add nodes and edges into the graph. This would by JSON file of some format. The JSON file would have a specific schema which will the application will translate in order to create nodes and edges on the graph. Example of the schema:

{

nodes {

nodeId: 0

xCoordinate: 0

yCoordinate: 1

}, {

nodeId: 0

xCoordinate: 0

yCoordinate: 1

},

edges {

sourceNodeId: 0,

destNodeId: 1,

}

}

This required me to make multiple changes to the application. I had to first make a backend controller nodeJS file to accept the JSON input. This then also required me to make a mapper which changed the textual information from the input into node and edge objects that will be displayed on the screen. Lastly, and what took the longest time, was creating the front-end visualisation of capturing the JSON input and demonstrating an output of the graph.

**Implementation of a ‘tour’ guide so new users have an idea of how to interact with the application**. **(5 hours)**

An issue that I found existed is that brand new users of the algorithms sometimes had trouble knowing exactly what to do for this algorithm. So, this led me to set up a feature where users will get a quick tour of the website, demonstrating It’s features and how to operate it. Technically, this was not a major challenge. Bootstrap provides template ways of simulating a tour, so my job was just to plan and write down the most user-friendly steps that will be displayed on the tour.

Review

Overall I am happy with this sprint as I have implemented everything I wanted to do that was agree by myself and my supervisor. Most features did not cause issues to implement apart from ‘**Implement a way to insert nodes and edges with JSON file.’** This did actually take longer than expected as I had some issues trying to convert the textual information into objects. However, after reading more about various JavaScript topics such as pointers and objects, I sufficiently was able to accomplish this task. So I am pleased about that. Another issue that did take slightly longer than expected was the **A way to pause and resume the graph algorithm traversal**. There was a bug which made it so that the application will not properly resume operation after click. This was caused by the JavaScript code wrongfully creating two timer objects, however I am glad to say this has been resolved. All the other features were done smoothly. I also added bonus small features that I quickly added which progressed the application such as adding highlights on hover, keyboard shortcuts, saving node and edge progress and fixed a bug which zoomed in on double click.