Graph algorithm tool

Emmanuel Odame-Asante

1404940

University of Essex

Computer Science

Michael Sanderson

Maria Kyropoulou

1. **Background Reading**

This background chapter discusses and evaluates core topics integral to the application I am going to build from the technical aspects to the project planning methodology aspects. I will first be researching intensively about the graph algorithms I will implement which are taught in CE204. I will also be addressing other critical topics such as tech stack architecture and planning methodology (Waterfall vs Agile). This background reading is crucial in ensuring that I start this project correctly.

1. **Graph Terminology**

**Directed Graph**

An ordered pair G: = (V, E) with V is a set, whose elements are called vertices and E is a set of ordered pairs of vertices, called directed edges. [1] (Fig 1.0)

**Example of a Directed Graph (Fig 1.0)**

**Weighted Graph**

Graph in which a number (the weight) is assigned to each edge [2]. (Fig 2.0)

5

2

**Example of a Weighted Graph (Fig 2.0)**

**Greedy Algorithms**

Before delving into the details of the algorithms that will be used in this project, it is important to note a fundamental idea that will be applicable to them. All the algorithms discussed are in the Greedy Algorithm category. The formal definition being it’s a heuristic algorithm that at every step selects the best choice available at that step without regard to future consequences. [3] There are variants to this style of problem solving which exists in many solutions. Greedy algorithms are ubiquitous in graph problem solving, the reason being because they are simple, easy and very quick to program. However, the main disadvantage is that they are not intelligent at all and rarely does it find the most optimal solution.

In the example below, (Fig 3.0) the source is vertex A and the destination is vertex F. The edges highlight in red is the path taken using a Greedy algorithm approach. You can see with a typical Greedy algorithm the path will follow the red route which is (A – B – F) this results in a total cost of 20. However, the green path has a lower cost (A – C – F) which is the optimal path to destination F. This marks out the main disadvantage to Greedy algorithms, they are not heuristically intelligent. The example is simplified but this still shows

5

15

5

10

**Example of a Greedy Algorithm (Fig 3.0)**

**Dijkstra Algorithm**

Dijkstra's algorithm computes length of the shortest path from the source to each of the remaining vertices in the graph. [4] It is one of the most popular Greedy algorithm type in existence. This differs greatly from Prim’s and Kruskal’s because it does not create any minimum spanning trees. This algorithm was pioneered by Edsger W. Dijkstra in 1956 and published three years later. [5] However it has been revamped and improved numerous times to the standard we have today.

This algorithm is ubiquitous in the programming world and forms the foundation of optimization routing problems. Common use cases are GPS systems, Computing Network Routing systems, Social Networks and Telephone Networks. Many technological requires a type of shortest path finder and Dijkstra’s algorithm is integral to that. [6] A common example for Dijkstra is representing nodes as cities and edges as available paths between cities. If an individual is at City A and wants to travel to City F, what would be the shortest path from City A to F? Dijkstra Algorithm answers exactly this by working out the shortest distance. (Fig 4.0) I will go on to explain it with an example so it is easier to understand.

104 miles

100 miles

162 miles

150 miles

250 miles

138 miles

50 miles

138 miles

**Example of Dijkstra Algorithm (Fig 4.0)**

1. Supposed you want to find the shortest path from source (City A) to target (City F).
2. Mark the distance from City A to every other city on the map as infinity. This means they have not been visited yet.
3. The start vertex will be the source (City A).
4. After, go to the closest unvisited city to the starting point (City B).
5. From the current city (City B), update the distance to every unvisited City that is directly connected to it.
6. Make sure to mark cities as visited when visited.
7. Continue this process until you marked the target (City F).
8. Once you mark the target, you can find out the shortest path to it.

Disadvantage of Dijkstra’s Algorithm

Although Dijkstra’s Algorithm is paramount to the success of technology in this modern age, there are disadvantages with this algorithm. Dijkstra algorithm is a greedy algorithm, meaning, it performs a ‘blind’ search which wastes time. It also cannot handle negative edges which could lead to acyclic graphs which will produce the wrong result. [7]. Another small disadvantage is that the time complexity for this algorithm is O(n^2) where n is the number of nodes. However, the time complexity can be improved to O(E log V) if using a adjacency list with a binary heap.

**Minimum Spanning Tree**

What is a Spanning Tree? A Spanning Tree of a graph G is a subgraph which includes all of vertices of G but with the lowest amount of edges. Another definition, is a spanning tree of a graph on n vertex is a subset of n – 1 edges that form a tree [8] .

Then what is a Minimum Spanning Tree? A Minimum Spanning Tree of an edge weighted graph is a spanning tree whose weight is no larger than the weight of any other spanning tree. [9] There may be a case where there are multiple spanning trees however there can only be one minimum spanning tree. Kruskal’s and Prim’s are both minimum spanning tree algorithms meaning they both find a minimum spanning tree as their end result. From this, we can solve ‘shortest path’ type problems.

**Kruskal’s Algorithm**

Kruskal’s algorithm is a minimum spanning tree algorithm which finds an edge of the least possible weight that connects any two trees in a forest [10]. This algorithm was conceived by Joseph Kruskal in his book Proceedings of the American Mathematical Society in 1956. Similar to Dijkstra’s, it is a greedy algorithm. The difference being, it finds a Minimum Spanning Tree for a connected weighted graph adding increasing cost arcs at each step. [10]. Usages will be Network Design in fields such as telephone, electrical, computer and roads. My goal will be to implement a simple design of Kruskal’s algorithm. The time complexity for Kruskal’s Algorithm is O (E log V) where E is the number of edges and V is the number of vertices. I will demonstrate an example of a graph running Kruskal algorithm (Fig 4.0) and the final minimum spanning tree that it results in (Fig 5.0).

104 miles

100 miles

162 miles

150 miles

250 miles

138 miles

50 miles

138 miles

**Example of Kruskal Algorithm (Fig 4.0)**

104 miles

100 miles

162 miles

50 miles

138 miles

**Final Minimum Spanning Tree for Kruskal’s (Fig 5.0)**

1. Start with the lowest edge between two cities (E-D)
2. Iterate through the whole map picking the next lowest edge that does not form a cycle.
3. Iteration order: (E-D) (D-F) (B-D) (A-B) (B-C)
4. Edges (A-E) and (C-F) cause cycles so they will not be part of the minimum spanning tree.

**Prim’s Algorithm**

Prim’s algorithm is also a greedy algorithm which finds a minimum spanning tree for a weighted undirected graph. This algorithm was created by Czech mathematician Vojtěch Jarník [11] . This was then rediscovered and republished by Robert C Prim in 1957 [12]. This algorithm is very similar to Kruskal’s algorithm as they both result in a minimum spanning tree. There also have the same usages as Kruskal’s. Everything Kruskal’s can do, Prim’s can also do apart from very edge case conditions. However, there are a few differences between Kruskal’s and Prim’s algorithm. Firstly, Kruskal’s algorithm starts with an edge but Prim’s algorithm starts with a vertex. Secondly, Prim’s algorithm selects the edge to nodes that are connected to each other whereas Kruskal’s selects edges in a haphazard way that does not always have to be connected. There is also a difference in time complexity.

The time complexity from prims algorithm is O(n^2) where n is the amount of nodes present. An example of Prim’s algorithm (Fig 6.0) and the minimum spanning (Fig 7.0) created will be demonstrated.

104 miles

100 miles

162 miles

150 miles

250 miles

138 miles

50 miles

138 miles

**Example of Prim’s Algorithm (Fig 6.0)**

1. Arbitrarily pick a starting vertex to initialize the tree. Let’s pick C.
2. Grow the tree by finding the minimum weight edge that does not create a cycle.
3. Iteration order: (C-B) (B-D) (D-E) (D-F) (E-A)
4. Keep doing this until all possible edges are added.

104 miles

162 miles

150 miles

50 miles

138 miles

**Final Minimum Spanning Tree for Prim’s (Fig 7.0)**

1. **Web vs Desktop Application**

**Advantages of Desktop Application**

Desktop Applications are primarily internet connection independent. After download and installation, the computer can be disconnected from internet services and if designed properly and technically possible, the application will still be operational. An example will be if a user opened the application without internet it could still run whereas in a web application there would need to be internet to access the website. Another advantage is that desktop application will reduce hosting and maintenance costs. Although in 2017 the cost for domain name purchase and server power are minimal, it is still something to take into consideration. Finally, desktop applications are generally more robust and reliable compared to websites. Other features such as a singular centralized way to get to an application, easier to locate and it being ‘Alt-Tab’ compliant all increase user experience which makes a desktop application a strong software type to develop on.

**Advantages of Web Application**

Web Applications do not require any downloading or installations. This gives the user instant access to the application as they do not have to wait for a download. This could also help less technical users as they don’t have to worry about installing. This helps achieve one of my goals as I want to develop this application for non-technical users so they can install and run the application very easily. Leading on from this point, another advantage is a user does not have the manually update the application if there are changes. So, if there was a major bug in the application, I can deploy a hotfix that will instantaneously be applied to the live website. Platform independency is also a major advantage. With a Web application, you can design and develop for the standardised browsers. In contrast to a desktop application where I would have to develop on three different operating systems if I want full user accessibility. This in term means designing, developing and testing three different applications with various languages. Lastly, web applications can be adaptable to mobile phones.

1. **Waterfall vs Agile**

I am personally familiar with both techniques but I will be undertaking more in-depth research to see which methodology will be most suitable for my project. I will go over the advantages of both and relate them to how they affect my project.

Waterfall ideology is the more ‘traditional’ project planning approach. It has a linear approach to software development. [13] The sequence of events is normally: Gather Requirements, Design, Development, Testing, Fix issues and then Delivery. Anecdotally, Waterfall works best with projects with a fixed scope, time and budget. [14] The nature of my project is that I may not have a fixed time or scope as I will be simultaneously working on other projects so my time could vary dramatically. For example, for a single iteration, I may have designated an amount of time to complete it but due to unforeseeable events (other university projects or examinations) I may have to cut that time short. Scope is also a major factor too. I will be working closely with my supervisor on this project which suits Agile methodology because it is created for the direct interaction with clients. Waterfall makes it so that you lock down on a specific scope, removing the flexibility you have on adjusting requirements. Projects that have constant requirement changes are more suitable for Agile methodology. [15] I definitely value this flexibility of change of scope. If I interact directly with a supervisor or another user who recommends a feature, with Agile it is perfectly plausible to add it, unlike Waterfall. Also, if there is a sprint which has too many story points that I can handle for the current sprint, I can shift some stories into the next sprint. Agile also allows for regular testing. I feel in relation to my project that this is important. I will be implementing algorithms which will require rigorous testing to ensure its robustness. Unlike Agile, Waterfall has it’s testing at the end of the development (normally). If testing was at the end of my project and there were major issues with the algorithms it may require a significant revamp and that could be disastrous for my project. So, using Agile to test and review regularly will ensure in a better product. In addition to this, there could be an issue of not dedicating enough time testing with Waterfall which could in turn mean bugs will be present at the end of the development cycle. I aim to work closely with other students and my supervisor to achieve the best product. Agile allows me to also be flexible with my project. If I have a priority I want to bump up I could easily do that with Agile. Lastly, according to the CHAOS report, it is shown that Agile is generally more successful than Waterfall for various project sizes. [16] With the above findings considered, my conclusion is that I will be implementing the Agile project planning methodology.

1. **Testing**

A major lesson I learnt during my placement year is that testing is crucial for the success of a product. So, I aim to add full stack testing for this application to ensure its robustness. For my backend side of my application I aspire to get high Unit Testing coverage and adequate Functional testing. My research will look for the most suitable unit testing framework for my application. There are many frameworks that do the job of Unit Testing. Frameworks such as Jasmine, Mocha, AVA just to name a few do Unit testing excellently. Jasmine will most likely be my main choice as that is the most popular framework and comes with exquisite documentation. I also have worked with this during my placement year. The syntax also appears to be very intuitive and simple to pick up.

Below I will provide an example:

**function** helloWorld() {

**return** 'Hello world!';

}

describe('Hello world', **function**() {

it('says hello', **function**() {

expect(helloWorld()).toEqual('Hello world!');

});

});

As you can see, in a few lines a simple function can be successfully tested and it seems easy to read and use. This will be important for the meticulous algorithms I will be creating so I eradicate any bugs that may arise. I will be incorporating TDD practices into my application also with the help of Jasmine.

I will also be researching front-end browser based testing. This is essentially browser automation which will mock user’s interactions on the site, such as clicks, keystrokes etc. With browser automation testing it will be sure that all user interaction with the site will be bug free. There are many frameworks that can be used. I have also done research on various frameworks but I have good experience with Selenium so will be going for this. In my experience, this was a solid framework which provided all the functionality required for testing. I also researched on the best techniques on how to manual test.

1. **Project Goals**

The section below will illustrate the initial aims and objectives for this Graph Algorithm teaching tool. Overtime, my requirements may change in relation to the evolution of the product however this will be the crux of requirements. I will be working closely with my supervisor Dr Sanderson to ensure the validity of this.

Initial Brief provided:

*“The aim of this project is to implement a tool to assist with the teaching of graph algorithms such as those taught in CE204.*

*It should provide a graphical display of step-by-step implementation of the algorithms with accompanying descriptive text. This should be offered at various levels of abstraction (e.g. for Kruskal's algorithm a version showing the connection sets and a more abstract version that uses the concept of not forming a cycle). It should be able to work on arbitrary graphs (i.e. the descriptions should not be based on a particular graph).*

*Ideally there should also be a graphical user interface that allows the user to create graphs by adding nodes at points on the display and creating edges to connect them, allowing the user's chosen graph to be used to demonstrate the algorithm.*

*The tool could be written as a web application or a stand-alone Java program.”* [17]

1. **Web Application**

This application will be a Web Application. Users will be able to use the application regardless of the operating system they are on. This application will be workable on all major browsers such as Google Chrome, Internet Explorer, Safari and Opera. Adequate testing will take place for all browsers to ensure robustness. The main programming language for this application will be in JavaScript. I will be using JavaScript ES6/JQuery for the algorithm computation and website interaction. NodeJS will be purely for client/server management. All of this will be accompanied with various reputable libraries and frameworks such as Bootstrap for the visual aspects of the website, Express is the NodeJS framework I will be using which will be for client/server interaction and Sigma JS which is the main graph framework. Scripting languages such as HTML and CSS will be heavily used to manipulate the aesthetics of the web application making it as appealing to users as possible. I also plan on implementing a Restful design which is the best standard of providing communication between computer systems.

1. **Has the ability to run algorithms**

The application should be able to at least run the three main graph algorithms taught in CE201. They are as follows Dijkstra’s, Kruskal’s and Prim’s algorithm. The application will show algorithms at various levels of abstraction. They should operate in the way they are taught in CE201 so students are familiar with what they see. The application will also be able to other graph algorithms in the future such as A\*, BFS/DFS and sorting algorithms if there is time.

1. **Dijkstra’s Algorithm**

The user will have text fields to put in the source and destination node ids in which after they can execute the algorithm. The path that the algorithm calculates from the source to destination will be highlighted in a specific colour which can be modified by the user. The path not taken will be black by default. So, the user can clearly see the path Dijkstra’s algorithm takes. This will happen in step by step iterations (at a speed determined by the user) until the end of the path. There will also be text information of step-by-step instructions on what the algorithm is doing.

1. **Kruskal’s Algorithm**

This algorithm will simply get executed by a button click. This will show the paths from lowest edge to the highest until it explores the whole graph. The path that the algorithm calculates from the source to destination will be highlighted in a specific colour which can be modified by the user. The path not taken will be black by default. There will also be text information of step-by-step instructions on what the algorithm is doing. Highlight of what the Minimum Spanning Tree is will be highlighted too.

1. **Prim’s Algorithm**

Prim’s Algorithm will have a text field for the user to put in a designated node id for the algorithm to start executing from. After that, the algorithm will show the path from the start to the finish. The path that the algorithm calculates from the source to destination will be highlighted in a specific colour which can be modified by the user. The path not taken will be black by default. There will also be text information of step-by-step instructions on what the algorithm is doing. Highlight of what the Minimum Spanning Tree is will be highlighted too.

1. **Teaching Tool**

There should also be an intuitive helpful tool that will teach users to understand algorithms. Informative step-by-step instructions will be provided so the user gets a full understanding of the algorithms they are using. Option to pause and review the steps will be provided so the user can take time to process what has just been shown. This is included with an option to speed up or slow down the step-by-step process if the user sees fit. The application should also be fully animated and coloured, providing various stages of the algorithms. For example, Kruskal’s algorithm, there should be highlighted edges showing if the specific edge is in a spanning tree or not. Also alerts that say why it is not been added to the spanning tree.

1. **User Interaction and customization**

Users will be able to fully customize and create graphs they want to simulate in the application. There will be a ‘Free Mode’ where users will be able to plot their own nodes and edges having complete freedom on creating whatever graph they desire. Additionally, options where users can load up graph templates or even a random graph which will randomly assign nodes and edges. Users will have freedom of the aesthetics of nodes and edges. They can change the size and colour of nodes and edges and change the speed of algorithm iterations. There will also be able to change the background colour. Other features such as node/edge position edits and drag ability will be available too.

1. **Deliver a fully intuitive functional website**

The aim is to create a visually aesthetic intuitive website which is extremely easy to use. This will mean that the application will be very easy to use that students outside from Computer Science can easily run and understand the program. I will make sure I demo the application to non-Computer Science students to satisfy this requirement. This includes creating a simple yet catchy domain name and curating a user interface for non-technical individuals in mind. Constant interaction with individuals will be necessary to create the best product. The website will have a welcoming homepage with direct links to main algorithms immediately. It will also be designed in a RESTful approach so that each algorithm teacher will be easy to get to. For example, [www.example.com/kruskal](http://www.example.com/kruskal). The procedure will be to find a reliable Web Host such as Amazon Web Services. This will guarantee constant uptime and adequate bandwidth for my users. I will also have to purchase a domain name. The aim to get a domain name which is very easy to remember and understand. A feature will be implemented that if a user were to accidently refresh or even close the website, on resume their graph will be saved.

1. **Mobile Optimization**

I will be actively implementing mobile practises to ensure that the application is perfect for mobile devices. Google stated that in the USA, 94% of people with smartphones search for information on their phones. [18] This shows that I should pay serious consideration to mobile development. So, my goal will be to deliver an optimal product for mobile devices. This means making the layout and design as easy for mobile users as possible. Performance is also going to be a priority, I should make sure that the application is as smooth and responsive on mobile as it will be on desktop. I will also make sure that there is familiarity so the users can get a good one experience regardless of their platform. The aim of this is to be a responsible application. Meaning, that the look and feel would be similar to a desktop website however it will be modified to adapt to any screen size.

1. **High Testing Coverage**

A technical requirement I strive to achieve is the highest possible testing coverage I can achieve. This will mean working with various tools to provide this from the backend to the frontend. Manuel testing will be achieved by myself and also by others who will be using the application.

1. **Project Planning**

The project plan is an important tool for determining assumptions and decisions. It is also important for facilitating communications with various individuals such as my users and supervisors. In addition to this, it will be served as a schedule for high level features that should be completed in the application. This will closely be reviewed and monitored by my supervisor to ensure my goals are correct, realistic and valid. In this project, I will be adopting the Agile methodology. One of the significant reasons for this is that it will provide flexibility in my planning. So, if I wanted to push a story into the next sprint if busy because of other priorities, this will be a plausible action to do. More advantages of Agile methodology are stated in my Background Reading chapter.

I have created an Agile Sprint plan tool which will help with the planning of my project. The sprint plan includes the initial requirements that I have gathered from my supervisor and also my personal requirements that I want to achieve for this project. However, there is a strong chance this could change my plan could change. For example, there may be a feature that I would like to add to the application which means more stories will be added. I will be strictly following this plan trying to achieve all the goals that I have set out for myself. Creating this did require some planning. There were many different variations of the project sprint plan I could have taken but decided to go for this plan specifically. The reason being is that this plan allows me to have a good spread out of story workload over the project duration. I have a few intensive months but I have meticulously decided that those months were the right time to put many resources into the project. I tactically made certain sprints more intensive than others and at a specific time because during those times I know that I will have less priorities for other modules so I have more time to spend on this project.

The duration of the project will be eight sprints, with one sprint representing one month. In a sprint, I will have different type of stories that I aim to complete. This could be either planning, development or testing. I have also implemented the concept of story points. Story points is the criteria on how I estimate the intensity that the overall story may contain. This ranges from 1 to 5 with 1 being the least intensive type of story and 5 being the most. With this structural planning in place, it allows me to foresee the workload that I have anticipated. Which in turn will allow me to have full control and vision of what I can or cannot do during a sprint. In conclusion, I am very satisfied with this sprint plan. I believe that I would have a successful product if with a smooth development cycle if I don’t stray from this plan. My sprint plan is attached to at the end of the document.

**Trello Integration**

I will also be integrating Trello into my project planning. Trello is a web based project management application used for managing various projects. I had some experience with this during my placement year and found it extremely useful for my planning. So, the main advantage is that I have an application that will be crucial for good planning. My board can be accessed on any device so I can have constant access from anywhere. Another advantage is that anyone can easily access this board too with the public link supplied. This means that both my supervisors will have a constant view of the status of my project.

The Trello link is here: https://trello.com/b/Z1yjZmYK/graph-algorithm-tool-project

**Sprint 1 – September 2017**

At the start of my project, I aim to spend a sprint on just complete research and investigation.

I will consolidate major requirements to start envisioning the project. I also planned a meeting with Mike Sanderson to setup the vision he sees in the application.

This enabled me to get a finite set of confined requirements which I could start working on. I have acknowledged the fact that my requirements could potential change over time however, starting with this set will be adequate for the start of the project. So, I have the main points out that I can start to investigate. After this, I invested in investigating project planning ideas such as Waterfall, Agile etc. This led me to decide on what was the best project planning methodology approach to take for this project. Naturally afterwards, I can start investigation at a more technical level. I start by making a draft of my architecture defining what technologies I want to use and if they are feasible for this type of project. This is an important step as determining the tech stack will have a significant influence on the outcome of the project so it’s crucial that I research properly and make a wise decision. Advice from more experienced engineers such as my placement year colleagues will be sought out too. I will be investigating ideas such as programming languages to use, graph libraries, frameworks in testing and web application building.

**Sprint 2 – October 2017**

After investigating the technical details of the application to use the plan will to play around and test it to see if it is adequate enough to suit my needs. This may require me to alter the source code of any used libraries to cater to my specific needs which could take time. It would most likely also require implementing boilerplate code so I can start to implement the algorithms. I also plan to start some early stage manual testing for the designated graph library I will be using. As this is the start of the academic year I intentionally made the sprint less intensive so I can ease into the new year.

**Sprint 3 – November 2017**

This sprint will be highly intensive as it will be filled with major stories for this application. Having 15 story points in the sprint indicates to me I should prepare for this sprint. If necessary, I could also shift stories from Sprint 3 to Sprint 4 to reduce the workload. The main synopsis of this month will be researching the algorithms that was defined in the initial requirements for this project. This will involve revisiting Mike Sanderson’s lecture notes to get the theory around them. Accompany that with YouTube videos and also algorithms books will give me a good fundamental understanding of the algorithms. After I fully understand the theory, I plan to start programming those algorithms into my application. This will require time as I have to program the theory of the algorithm while modifying it to fit the graph library I am using and be coherent with my vision. It can be recognised that this month can potentially be a risk as there are many stories which could be a technical challenge. This is because I will be making a complete new algorithm that will have to be compatible for the application. I am prepared to seek aid from my supervisor if it gets too challenging.

**Sprint 4 – December 2017**

The aim for Sprint 4 will be manual testing everything that I have achieved in the previous sprints. Particularly Sprint 3 as that will be very intensive and filled with new features. If I pushed any stories from sprint 3 to this sprint I aim to complete them all. The plan will be to extensively look for bugs and glitches that may exist in the application. I will also open this application to the public as Beta so I can start getting first reviews from potential users. I will make the effort to demo my application to various individuals to see how they feel about it. It will also be a time when I sit down with my supervisor to see if he has any executive views on the application. This month will be very insightful as it will allow me to consolidate a list of features that I can work on next sprint.

**Sprint 5 – January 2018**

This sprint will just be purely dedicated to looking at reviews from users and supervisors and improving the application as much as possible. It is not clear the intensity of work that I will have to do for this sprint so I will if I find that I have more time then intended I will most likely start Sprint 6 early. I have also intentionally made this a light sprint just in case I have priorities for other modules.

**Sprint 6 – February 2018**

This will also a productive month in terms of development for this application. Before this point there was no real UX interface or design. It would just be a very bare template with the option to run algorithms. So, the focus for this month is to create an interface visually appealing and intuitive. I will be working with other students who have experience in UX that will advise me on what will be the best design that will attract many students to the application and more importantly, make them stay. This may be highly intensive because from experience user experience and user interface building can be highly difficult.

**Sprint 7 – March 2018**

By Sprint 7, I aim for the main crux of the application to be completed. The functionality of all my requirements should be completed. This will include the ability to run the three main algorithms and have an attractive web application. After that, I need to implement automation testing. This includes Unit and Functional tests for the back-end side of the application and Selenium for the front-end side of the application. I will be using the testing frameworks that I have researched.

**Sprint 8 - April 2018**

As the last sprint of the project, it will purely be for fixing last minute bugs and issues. I will also be taking in user reviews again due to the changes I have made over the last few sprints. Again, this will be time to listen to the people and implementing changes that they think will benefit the application.

Last reviews and check will take place from supervisor and various staff and if everything goes to plan I will release version 1.0 of the application for the public to use. This will require some admin work such as purchasing and activating domain names and server registration. This work should take no longer than a fortnight.

**Table (1). Project Sprint Plan**

Interim Oral Examination Deliverables

**By week 11 I aim to successfully complete all stories from sprint 1 to 3. This will form the basis of the interim oral examination deliverables.**

|  |  |  |
| --- | --- | --- |
| **Task Name** | **Description** | **Story Points** |
| **SPRINT 1 – September 2017** | | |
| Requirement Investigation | Consolidate initial requirements for this project with supervisor. | 1 |
| Architecture Investigation | Research the most adequate technologies to use that will help me accomplish my goals. | 2 |
| Planning Project methodology Investigation | Research the most suitable project methodology to implement in my project. | 2 |
| Graph Library Investigation | Research for the most suitable graph library that can visualise nodes and edges. | 2 |
| **SPRINT 2 - October 2017** | | |
| Graph Library Implementation Testing | Test the Graph Library itself so it’s bug free and can do everything I want it to do. | 2 |
| Node Edge click functionality  Browser initial input boxes | Implement mouse functionality that will add and remove nodes and edges. | 3 |
| **SPRINT 3 – November 2017** | | |
| Dijkstra Investigation | Learn about everything I need to know about Dijkstra’s algorithm and research the best way of programming this to my application. | 2 |
| Kruskal Investigation | Learn about everything I need to know about Kruskal’s algorithm and research the best way of programming this to my application. | 2 |
| Prims Investigation | Learn about everything I need to know about Prims algorithm and research the best way of programming this to my application. | 2 |
| Dijkstra Implementation | Successfully program Dijkstra algorithm into the application. | 3 |
| Kruskal Implementation | Successfully program Kruskal’s algorithm into the application. | 3 |
| Prims Implementation | Successfully program Prims algorithm into the application. | 3 |
| **SPRINT 4 – December 2017** | | |
| Manuel Dijkstra Testing | Intensively test by running many times under differences scenarios. | 2 |
| Manuel Kruskal Testing | Intensively test by running many times under differences scenarios. | 2 |
| Manuel Prims Testing | Intensively test by running many times under differences scenarios. | 2 |
| First Review | Open beta version of application to students and accept feedback and criticism. Also have an intensive meeting with supervisor discussing progress of application. | 1 |
| **SPRINT 5 – January 2018** | | |
| Implementing reviews. | Implement all the changes that needs to be done from feedback. | 2/3 |
| **SPRINT 6 – February 2018** | | |
| Enhance UX/UI experience | Enhance the aesthetics of the application. | 5 |
| **SPRINT 7 – March 2018** | | |
| Implement testing functionality | Program unit/functional tests and selenium into application. | 3 |
| **SPRINT 8 – April 2018** | | |
| Final Review | Get feedback for application. | 1 |
| Implementing reviews | Implement for application. | 1 |
| Launching web application | Register a domain and server host and launch application. | 1 |

1. **References**

[1] Bang-Jensen, Jørgen; Gutin, Gregory (2000) Digraphs: Theory, Algorithms and Applications

[2] Fletcher, Peter; Hoyle, Hughes; Patty, C. Wayne (1991) Foundations of Discrete Mathematics

[3] Gass, S.I. and Harris, C.M. (1996) Encyclopedia of Operations Research and management Science

[4] PUTHUPARAMPIL, MERIN REPORT DIJKSTRA’S ALGORITHM

[5] Dijkstra, Edsger W. An Interview with Edsger W. Dijkstra Frana, Phil

[6] What are the real life applications of Dijkstra's algorithm? Quora https://www.quora.com/What-are-the-real-life-applications-of-Dijkstras-algorithm

[7] PUTHUPARAMPIL, MERIN REPORT DIJKSTRA’S ALGORITHM PAGE 15

[8] Skiena (2003) Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica ®

[9] Wayne., Robert Sedgewick and Kevin (2011) Algorithms

[10] Cormen, Thomas; Charles E Leiserson, Ronald L Rivest, Clifford Stein (2013) Introduction To Algorithms (Third ed)

[11] Jarník, Vojtěch (1930) Práce Moravské Přírodovědecké Společnosti

[12] Prim (1957) Shortest connection networks And some generalizations

[13] Lotz, Mary Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?

[14] Haworth, Suzanna What Should You Use For Your Project? <http://www.thedigitalprojectmanager.com/agile-vs-waterfall/>

[15] Haworth, Suzanna What Should You Use For Your Project? <http://www.thedigitalprojectmanager.com/agile-vs-waterfall/>

[16] Wojewoda, Shane Hastie Stéphane Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch https://www.infoq.com/articles/standish-chaos-2015

[17] Sanderson, Michael Project Choices Database https://moodle.essex.ac.uk/mod/data/view.php?d=296&rid=26934

[18] Google, Google Developers Mobile Site

https://developers.google.com/search/mobile-sites/