Project Title: AI Search Visualiser
Functional Specification

Name	**Student ID**
James Farrelly	17396736
Emily Whyte	17405094

Project Supervisor	Charlie Daly

Date of document completion	02/12/2020

0. Table of contents

1. Introduction

1.1 Overview

AI Search Visualiser is a web application used for algorithm visualisation. Users will be able to visualise various search algorithms; Depth-First Search, Breadth-First Search, Iterative Deepening Depth-First Search and A* Search. Students can use this application to aid in their revision or understanding of search algorithms.

Users will be able to choose which algorithms they visualise and select heuristics where appropriate. The application will step through the algorithm and allow users to set their own pace or let it play automatically. Users can also choose to view the steps taken to complete the search in a text log. Search graph nodes will be colour coded to indicate which state (open/closed) they are in. This will help users to understand the different processes of each algorithm.

Users can select two algorithms and visualise both, comparing the different ways they search through the same graph. This can allow users to see which searches are more efficient.

Users will also be able to enter custom nodes and change the graphs that are being searched through in the algorithm visualisation.

1.2 Business Context

The AI Search Visualiser could be deployed alongside the teaching material used by the lecturers of the CA318 Advanced Algorithms and AI Search module in DCU, or courses covering the same content in other colleges. It would be helpful to offer students a way to revise the algorithms they are learning for assignments or exams.

1.3 Glossary

Term	Definition
Heuristic	Method to find a solution to a problem more quickly or to find an approximate solution.
Graph/Tree	A structure of nodes that are related to each other.
Canvas	The front-end component that users draw their graph on.
GUI	The graphical user interface that allows a user to interact with the web application with the use of graphics or audio.

Term	Definition
React	An open-source, front end javascript library used for developing user interface and UI components
Visualization	Any technique used to create images, diagrams, graphs that help communicate a message

2. General Description

2.1 Product/System Functions

2.1.1 Algorithm selection

When users open the web application, they will be shown a selection of search algorithms which can be visualised. These are Depth-First Search, Breadth-First Search, Iterative Deepening Depth-First Search and A* Search. For certain algorithms, users can choose which heuristics will be used in the visualisation. Once the algorithm (and heuristic) has been selected, the web application generates random node values for the algorithm to search through.

2.1.2 Algorithm visualisation

After selecting the search algorithm, the web application will generate a visual representation of a graph or tree of nodes containing different values. The user will be able to press play or use buttons to step through the algorithm at their own pace. As the algorithm progresses, the nodes will be colour coded to display which state (open/closed) they are in. The visualisation will be clear and should be easy for users to understand.

2.1.3 Algorithm steps text log

Users have the option to toggle whether or not they can see a text log of the steps taken by the algorithm. When enabled, this log can provide additional context for the moves made by the algorithm as it searches through the graph or tree.

2.1.4 Algorithm comparison

Users can select a second suitable algorithm from the remaining list of algorithms to compare to one they have already selected. The AI Search Visualiser will visualise the same graph or tree twice, side-by-side. The two selected algorithms will work through the graphs at the same pace and allow users to watch. This is particularly useful in understanding the different ways in which different algorithms process the same nodes.

2.1.5 Custom nodes

Users may choose to create a graph that contains custom node values, such as from an example found in lecture notes or homework. They can add multiple nodes, specifying which other nodes they connect to and what values they will contain. Users can then use this custom graph to visualise the search algorithms provided by the AI Search Visualiser.

2.2 User Characteristics and Objectives

Users of our web application will primarily be students who wish to learn more about how AI search algorithms operate such as students of the CA318 Advanced Algorithms and AI Search module in DCU. However, the web application could also be useful to self-taught programmers or those who simply want a refresh on the subject. Users are expected to know the basics of how each algorithm operates. As users are expected to be programmers, it can be presumed that they have a high level of computer literacy.

Still, users must be given an initial, brief tutorial on how to operate the web application. The application will have this, it will be skippable if the user wishes and can be reopened at any time during the use of the application. Upon selecting an algorithm, the default graph for the algorithm visualization will be a simple graph which shows off every feature of the specific algorithm to introduce the user to it.

Our aim is for the application to be as familiar and user friendly as possible. The user should be able to use the canvas drawing tools and step iteration controls without being told what to do. To do this we will make the UI similar to other web applications that the users are expected to have previously used such as google slides (specifically how to insert shapes). The step controller will be similar to the media player controller on web applications like youtube. There will be a significant focus on readability and simplicity, as this is where other algorithm visualization tools can confuse students.

Nodes will be colour coded to show changes in their state such as visited and queued nodes in Breadth-First Search. This will ‘show’ the user information rather than ‘telling’ them which will allow us to keep the UI as clear and simple as we can.

2.3 Operational Scenarios

2.3.1 First time using the web application

The user upon first entering the web application will be greeted with a quick tutorial which shall highlight the main features of the application as well as how to use it. The user might wish to go through the tutorial step by step, at their own pace. However, they will also be given the option to skip to the end if they wish.

2.3.2 Selecting algorithm

The user will then be able to select from a dropdown list the algorithm they wish to visualize on the graph. The list includes Depth-First Search, Breadth-First Search, Iterative Deepening Depth-First Search and A* Search. The chosen algorithm will affect how the graph is navigated as well as the text log message and rules available to the user.

2.3.3 Customising graph nodes

Next, the user will be given the option to create their own graph to visualize or choose with the default graph. Users who decide to create their graph will be able to choose the location of each vertical as well the link between each node.

2.3.4 Choosing custom options

If a certain algorithm is selected, the next step in the process for the user is to select the starting node, where the algorithm will begin. The user may also choose which heuristic the algorithm will use if applicable. Each option will appear in a dropdown list of the available options, which will vary.

2.3.5 Steps Iteration

Users can progress through the visualization either step by step or automatically using the set controls which will appear familiar to any generic media player users. Users will be able to backtrack steps as well as pause and continue. There will be a step counter letting the user know what step they are currently on. Nodes will begin to change colour to signify changes, for example, visited nodes being green.

2.3.6 Algorithm steps text log

The user may wish to know what exactly is happening at each step. If they require more information they can open the text log tab. This tab will inform the user what is happening at current and previous steps. This tab can be minimized again once no longer needed by the user.

2.3.7 Algorithm comparison

There is also another dropdown menu where users can select another algorithm to compare with the currently selected algorithm. This will split the canvas, displaying the algorithms side by side, letting the user progress through and compare the two algorithms at their own pace.

2.4 Constraints

Time constraints

We will follow our preliminary schedule as much as possible to ensure we develop our application to its full potential in the time we have. The final submission of our project is May 7 2021 and this places a constraint on the amount of work we will ultimately be able to put into the application. Ranking the functional requirements and using this information to plan our preliminary schedule helps to make sure we reach our deadline with the most important functions implemented.

User requirements

Users want our application to be simple and easy to understand when visualising algorithms. This will place a constraint on the complexity involved in the user interface of the application.

Hardware

The application will be limited to working on desktop only. We have decided that with the number of items to display on the screen, a mobile version will be too cramped, removing any simplicity.

Remote work

Due to the current circumstances, we are unable to meet each other and work together on one computer or side-by-side in person for protective reasons. This may create some complications in the development of our application but we will make use of GitLab and Zoom calls to work together, remotely.

3. Functional Requirements

3.1 Algorithm selection

Description
When a user first opens the web application, they are presented with a list of search algorithms to visualise. The list of algorithms to choose from is Depth-First Search, Breadth-First Search, Iterative Deepening Depth-First Search and A* Search. When appropriate, the user can also select the heuristic to be used in the search.

Criticality
Choosing which algorithm to visualise is the most important function of the web application. Without selecting the search algorithm, there is no way of the application knowing which one to use.

Technical issues
The search algorithms will be presented in a simple interface for users to choose from. It should only be possible to choose heuristics when the algorithm can use them.

Dependencies with other requirements
This function only depends on the User Interface of the web application being implemented.

3.2 Algorithm visualisation

Description
Once the algorithm and heuristics or other customisation have been selected, the web application is ready to visualise the search algorithm. The web application will use nodes to generate a search graph or tree. The user can then either step through the search at their own pace using buttons or press play and watch the visualisation play automatically. Nodes in the graph are colour coded to display which state (open/closed) they are in. The user should be able to see the process followed by the algorithm as it passes through the graph.

Criticality
This function provides the main feature of the entire web application so it is a crucial requirement. Without it, there is no need for the other functions.

Technical issues
The algorithm visualisation must be clear and easy for users to understand. The main goal of the visualisation is to help users learn more about the way search algorithms work. A complicated visualisation will not be helpful in this.

Dependencies with other requirements
This function depends on the algorithm selection function as it needs to know the type of search algorithm that will be visualised. The function also needs to know of any customisation so it depends on the custom nodes function.

3.3 Algorithm comparison

Description
Users can select another algorithm which will be compared to the one they had selected when they opened the web application. The AI Search Visualiser will use two of the same graph and visualise the two different algorithms as they search through it, side-by-side. This allows the user to see the different steps taken by each algorithm.

Criticality
This function is not necessarily essential to the overall system as no other functions depend on it, but it is a beneficial function we hope to implement.

Technical issues
It is essential that the comparison of algorithms is useful, so we must ensure that the two graphs are truly identical and that the visualisation is still simple and clear when two graphs are present on the screen.

Dependencies with other requirements
This function depends fully on the algorithm visualisation function as it re-uses this function to simultaneously visualise two algorithms.

3.4 Custom nodes

Description
Users can create their own search graphs or trees. They can create multiple nodes with custom values and choose which nodes they connect to. The custom graph made by the user can then be used to visualise a search algorithm.

Criticality
The web application can still be used without this function, but customisation is a handy feature for users. For example, a student could copy information from a search graph found in their lecture notes and run through it as a worked example.

Technical issues
Allowing users to customise nodes/graphs adds a lot of complexity to the web application. We will have to implement a method for drawing new nodes that is simple for any user to use. This function will have to correctly store all the node values and how they are connected to visualise the search properly.

Dependencies with other requirements
This function depends on the User Interface of the web application and any functions that may be used in initialising non-customised search graphs.

3.5 Algorithm steps text log

Description
When the web application is visualising a search algorithm, users have the option to turn on a steps text log. This text log will display a textual representation of the steps taken by the algorithm, to provide more context for moves made.

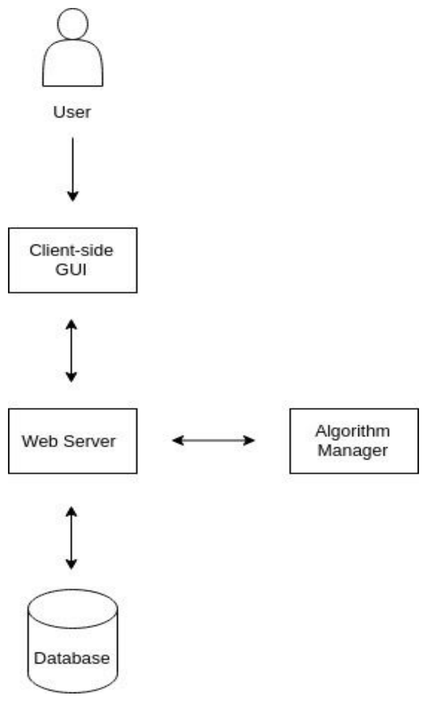
Criticality
This function is optional for users, so it is not as important as others. It is simply a bonus feature.

Technical issues
The text log needs to be concise, with enough information to help users understand what the algorithm is doing as it moves through the graph. The text log must be in sync with the visualisation of the algorithm, otherwise, it will give inaccurate information.

Dependencies with other requirements
This function depends on the visualisation function to receive the steps made by the algorithm.

4. System Architecture

4.1 Diagram



4.2 Description

User
The user is the actor in our system architecture diagram. They interact with the web application as stated in the operational scenarios.

Client-side GUI
The client-side GUI is the front end of the web application. It is what the user will see and interact with. It will be responsive and user friendly. It will be written using HTML, CSS and React.

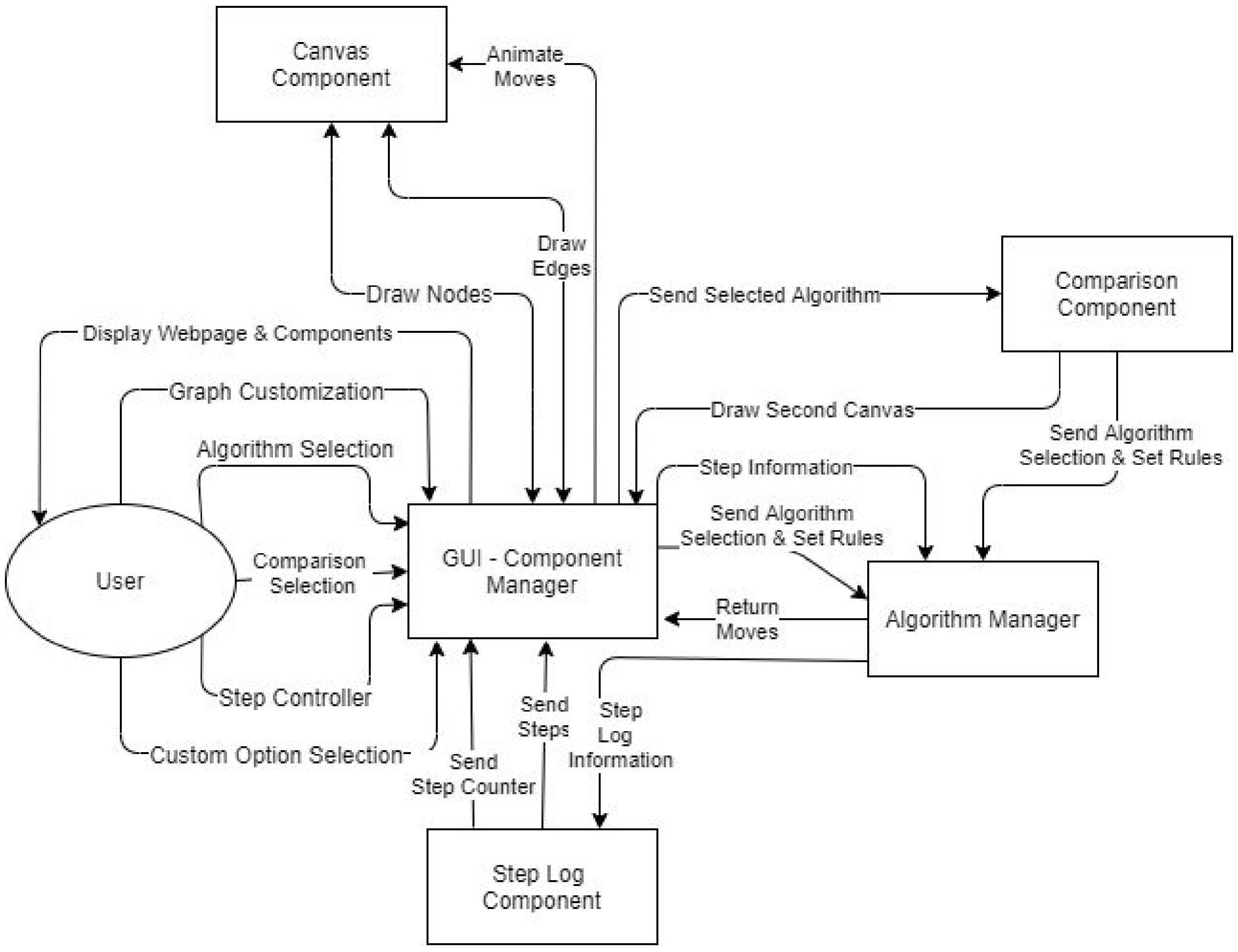
Web Server
The web server interacts with both the GUI and the algorithm manager, handling and applying the many requests made by the client. It will be written using javascript.

Algorithm Manager
This will manage the various algorithms offered by the system and deliver them to the web server once called.

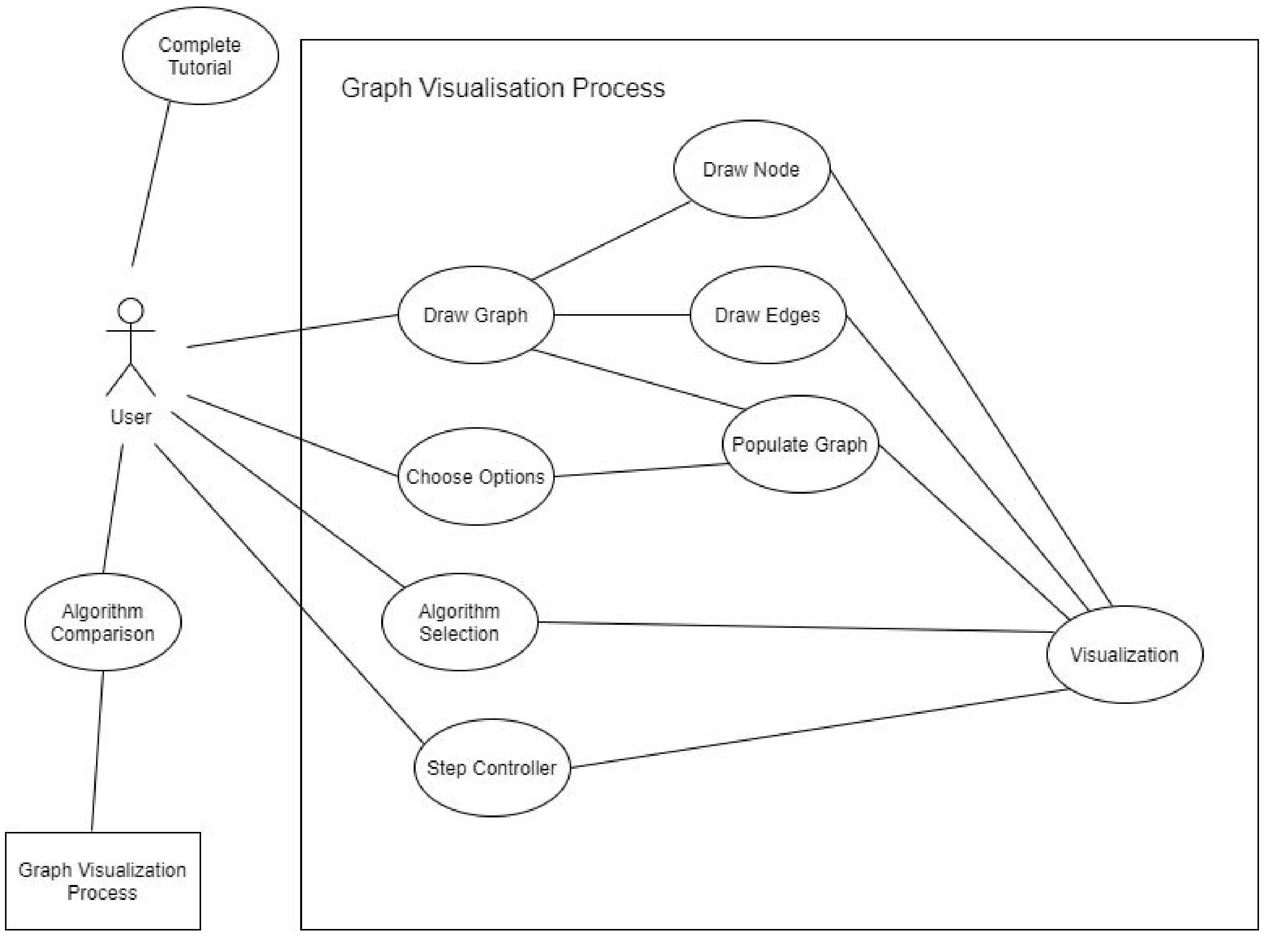
Database
This is where we will store node location and will interact with the webserver to sort and update the GUI.

5. High-Level Design

5.1 Context Diagram



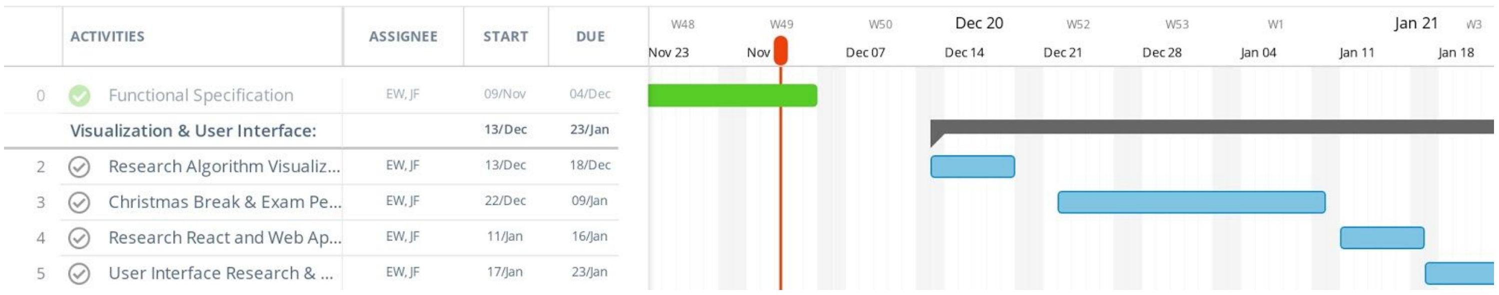
5.2 Use Case Diagram



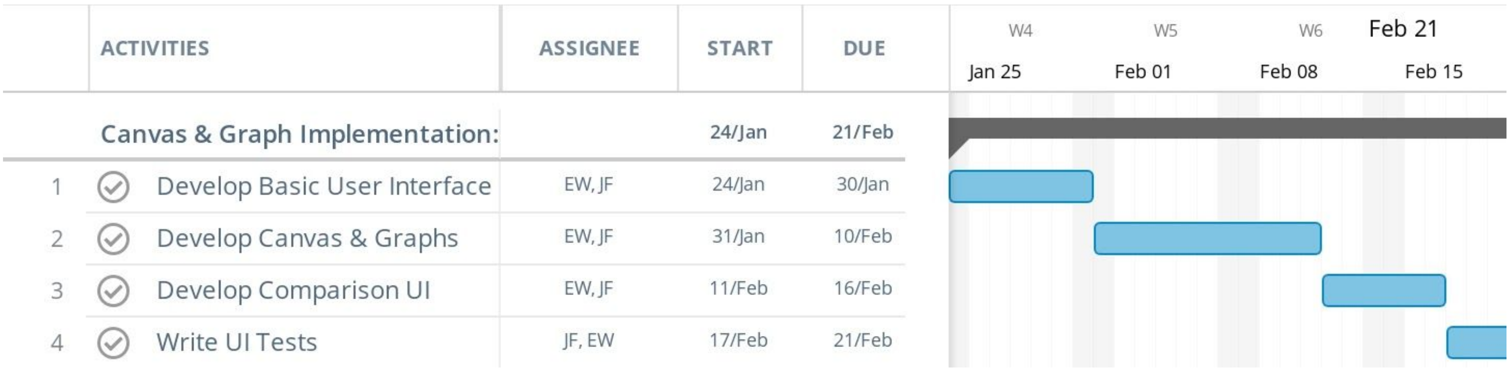
5.3 Description

Firstly, the user is given a quick tutorial on how to operate the web application. This tutorial can be passed through step by step or completely skipped but can be reopened at any point. The user can then draw a graph using nodes and then connecting the nodes with edges. After this, the user can choose from several options such as starting node and edge length. Once all of these variables have been set the graph will update with the correct values. An algorithm is then selected by the user which affects the number of steps in the step controller for the user. Finally, the user may wish to initiate the visualization using this or add another algorithm to compare to the current one, starting this process again.

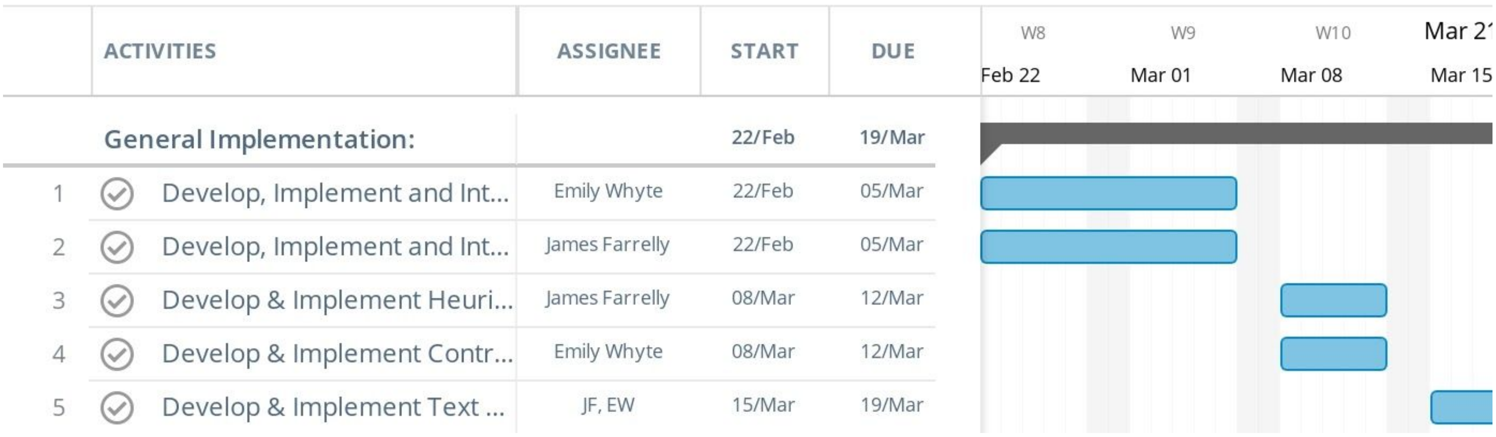
6. Preliminary Schedule



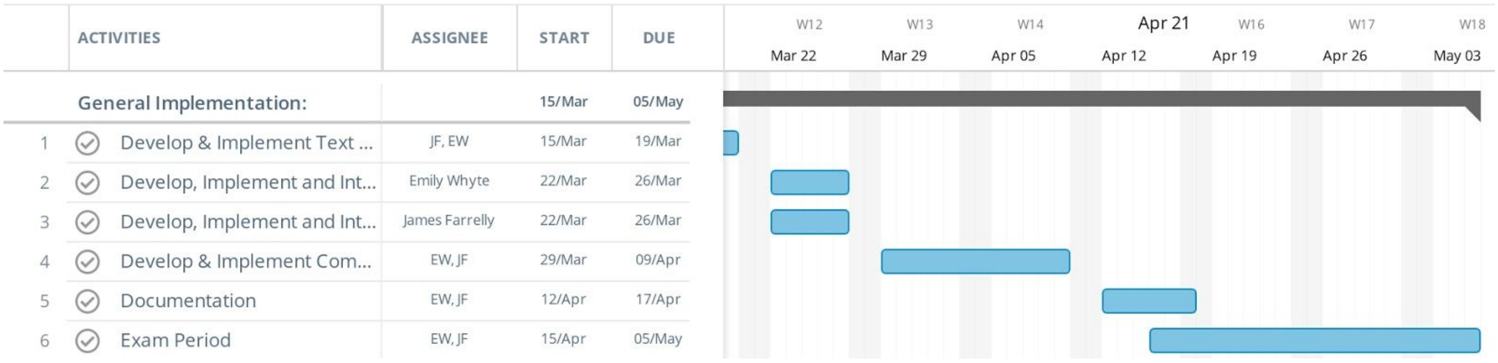
	ACTIVITIES	ASSIGNEE	START	DUE
0	✓ Functional Specification	EW, JF	09/Nov	04/Dec
	Visualization & User Interface:		13/Dec	23/Jan
2	✓ Research Algorithm Visualiz...	EW, JF	13/Dec	18/Dec
3	✓ Christmas Break & Exam Pe...	EW, JF	22/Dec	09/Jan
4	✓ Research React and Web Ap...	EW, JF	11/Jan	16/Jan
5	✓ User Interface Research & ...	EW, JF	17/Jan	23/Jan



		ACTIVITIES	ASSIGNEE	START	DUE
		Canvas & Graph Implementation:		**24/Jan**	**21/Feb**
1	✓	Develop Basic User Interface	EW, JF	24/Jan	30/Jan
2	✓	Develop Canvas & Graphs	EW, JF	31/Jan	10/Feb
3	✓	Develop Comparison UI	EW, JF	11/Feb	16/Feb
4	✓	Write UI Tests	JF, EW	17/Feb	21/Feb



		ACTIVITIES	ASSIGNEE	START	DUE
		General Implementation:		**22/Feb**	**19/Mar**
1	✓	Develop, Implement and Int...	Emily Whyte	22/Feb	05/Mar
2	✓	Develop, Implement and Int...	James Farrelly	22/Feb	05/Mar
3	✓	Develop & Implement Heuri...	James Farrelly	08/Mar	12/Mar
4	✓	Develop & Implement Contr...	Emily Whyte	08/Mar	12/Mar
5	✓	Develop & Implement Text ...	JF, EW	15/Mar	19/Mar



		ACTIVITIES	ASSIGNEE	START	DUE
		General Implementation:		**15/Mar**	**05/May**
1	✓	Develop & Implement Text ...	JF, EW	15/Mar	19/Mar
2	✓	Develop, Implement and Int...	Emily Whyte	22/Mar	26/Mar
3	✓	Develop, Implement and Int...	James Farrelly	22/Mar	26/Mar
4	✓	Develop & Implement Com...	EW, JF	29/Mar	09/Apr
5	✓	Documentation	EW, JF	12/Apr	17/Apr
6	✓	Exam Period	EW, JF	15/Apr	05/May