Facilitating Graph Exploration

Abstract:

Graphs have a long history dating back to the 18th century, and they've been fundamental in computer science. Graphs have been used in a wide range of applications from traffic routing, social media platforms, operating systems, computer networking, and much more.

The main motive of the project comes from the need to provide a solution for researchers, and students who require accessible tools for exploring graphs. The main goal of the project is to create a user-friendly website that combines two essential functionalities: graph visualisation and generation. During the development stage, the focus will primarily be on a subset of features known as the Minimum Viable Product (MVP) features. These features include inputting graphs in different formats, visualising them, and visualising various graph algorithms. The project will mainly use web languages such as HTML, CSS, and JavaScript for development.

1. Introduction:

The proposed project is mainly focused on 2 areas:

1.1 Graph & Graph Algorithm Visualisations

This part of the proposed project aims to create user-friendly visualisations for representing graphs and to make it easier to understand how graph algorithms work. One frequent use case of such a feature is that it is so often that programmers will go solve coding problems where a graph is given in the form of an Adjacency Matrix, Adjacency List, etc. and you want to quickly draw this graph and the classical approach is to manually use pen and paper, but with the help of the graph visualiser people can quickly create and view those graphs.

1.2 Graph Generations

This part of the proposed project aims to generate all sets of graphs with specific properties to be used for experimentation, benchmarking, and various other research applications.

2. Background & Motivation:

There have been previous projects that have similar ideas to this proposed project examples are:

1. CS Academy Graph Editor [3]: Used for graphically displaying Graphs
2. The Combinatorial Object Server [1]: Used to Generate unlabelled non-isomorphic graphs with a given number of nodes with various additional properties using the nauty library [2]
3. Graph Online [4]: Used for graphically displaying Graphs and running algorithm visualisations on Graphs
4. D3 Graph Theory [5]: Used to facilitate interactive learning of graph theory
5. CS USFCA [6]: Used for Algorithm and Data Structure Visualisations

All the above are examples of projects similar to the proposed project, but up to this day there has not been a service that supports both Graph Generation and Visualization at the same place and that is the main focus of the proposed project.

3. The Proposed Project:

3.1 Aims & Objectives:

As discussed in the introduction this project is concerned with facilitating the exploration of graphs. In this section, we will dive deeper into the exact requirements of the project. We will divide the requirements of the project into 3 categories:

A. **MVP Features:** Minimal Viable Product will contain all the set of requirements that is so critical to the project, and those requirements alone are enough for conveying the goals of the project
B. **Extra Features:** Features that would be nice if were included in the project, but if they were not included that would not be a big problem
C. **Miscellaneous Features:** Features that would be nice to have, but out of the scope of this project

A. MVP Features:

1. Allow the user to input a graph in any representation like Adjacency List, Adjacency Matrix, etc.., and display the drawing of the Graph on the website
    a. These features should be able to draw both undirected and directed graphs
2. Allow the user to visualise the following algorithms on the graph:
    a. Depth-First Search (The user has to specify the start and end nodes)
    b. Breadth-First Search (The user has to specify the start and end nodes)
    c. Dijkstra’s Shortest Path (The user should specify the start node)
    d. For all the above visualisation the user should have the ability to modify the speed of the visualisation process

B. Extra Features:

1. Allow the user to visualise the following algorithms on the Graph:
    a. Bellman-Ford Shortest Path (The user should specify the start node)
    b. Floyd Warshall Shortest Path (The user should specify the start node)
    c. Khan’s Algorithm
    d. Cycle Detection using DFS
2. Should create another webpage on the website that allows users to generate all combinations of undirected non-isomorphic graphs based on the following properties:
    a. Number of Nodes (required field)
    b. Minimum number of edges
    c. Maximum number of edges
    d. Minimum degree
    e. Maximum degree
3. The user should be able to specify the type/family of the graph he wants to generate including the following:
    a. Connected
    b. Biconnected
    c. Bipartite

4. The user should be able to choose the output format of the generated graphs including the following:
    a. Adjacency List
    b. Edge List
    c. Adjacency Matrix
5. All the output of the Generated Graphs should be displayed in the Browser
6. Allow the user to generate all Isomorphic undirected graphs, as the number of isomorphic undirected graphs with **n** nodes is exponential based on the following formula $2^{n(n-1)/2}$ there would be a limit to the number of nodes used
7. Allow the user to click on any of the generated graphs to display and visualise it

C. Misc Features:
1. Allow for additional graph types/families including the following:
    a. Triangle Free
    b. 4 Cycle Free
    c. Allow the users to download the generated graphs as a text file
2. Allow the user to download the graph visualisation as an image including the following image formats:
    a. jpg
    b. png
    c. SVG
3. The website should be able to support random generation and visualisation of mazes and should be able to run DFS & BFS on the maze by treating it as a graph

3.2 Methodology:
Mainly in the Methodology, we will discuss the following points:
a. Software Development Approach Used
b. Software Development Life Cycle (SDLC) of the Project
c. Main Languages and Technologies used in the Project

A. Software Development Approach Used:
The Software Development approach used for this project is the waterfall model, where all requirements are gathered before the development, and that is simply the case as the project has a well-defined set of requirements with a specific goal. By choosing this approach for development I will not be expecting a change in the requirements. One advantage of this approach compared with other approaches is that you have no ambiguity as all the requirements are defined from the start, and a design for the system as a whole can be created for best aiding those set of requirements without needing to think about designs that handle frequently changing requirements.

B. SDLC of the Project:
As we are using the waterfall model we are expected to pass by every stage of the SDLC only once, except for the design and the implementation phases as for each requirement category will have its own design and implementation. The Life Cycle of the project will be as follows:
1. **Requirements Analysis:** This part of the life cycle has already been done in the Aims & Objectives section

2. **Design:** The output of this stage should be a simple diagram representing the architecture of the project I might provide a draft Class Diagram at this stage, but there is a very high chance it will change in the future during the implementation phase
3. **Implementation:** This is the most fun phase at least for me, the expected outcome of this phase is a fully functional project with all MVP Features, and any additional features
4. **Testing:** This is the last phase in the Project Life Cycle, black-box testing will be used for the project and the expected outcome at this phase is a document containing different test cases and their results

C. Main Languages and Technologies used in the Project:
The Project will be implemented as a website, so the main Languages and Technologies used in the project are:

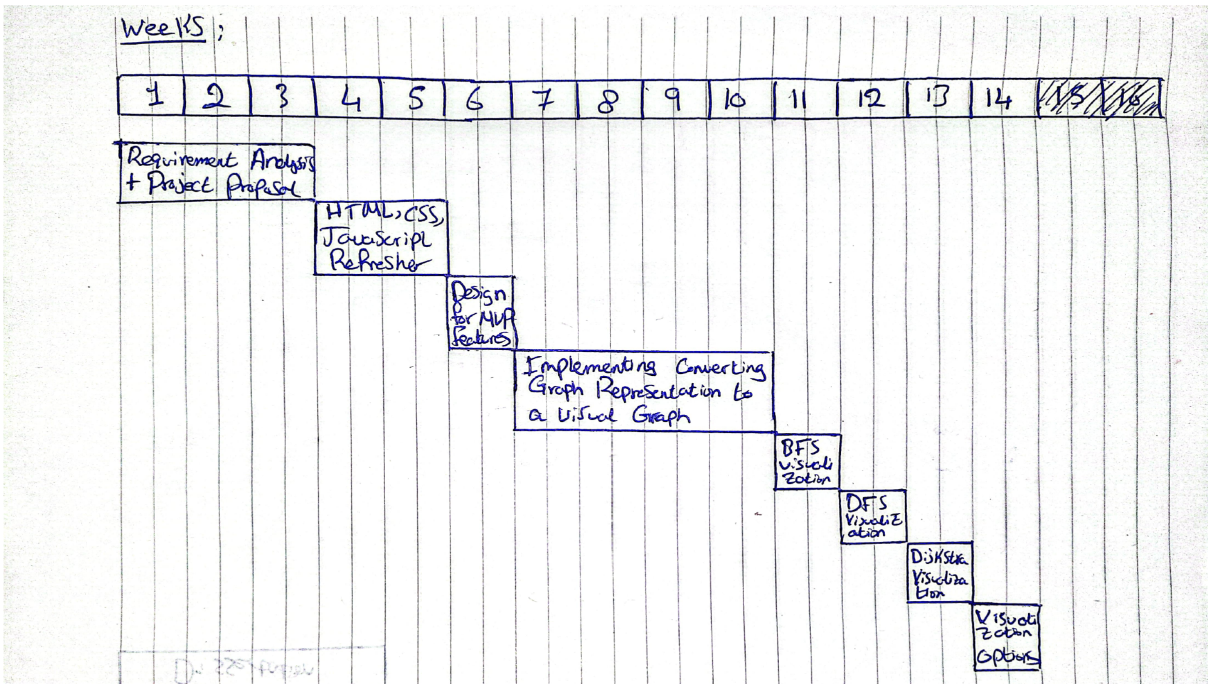
1. HTML
2. CSS
3. JavaScript/TypeScript

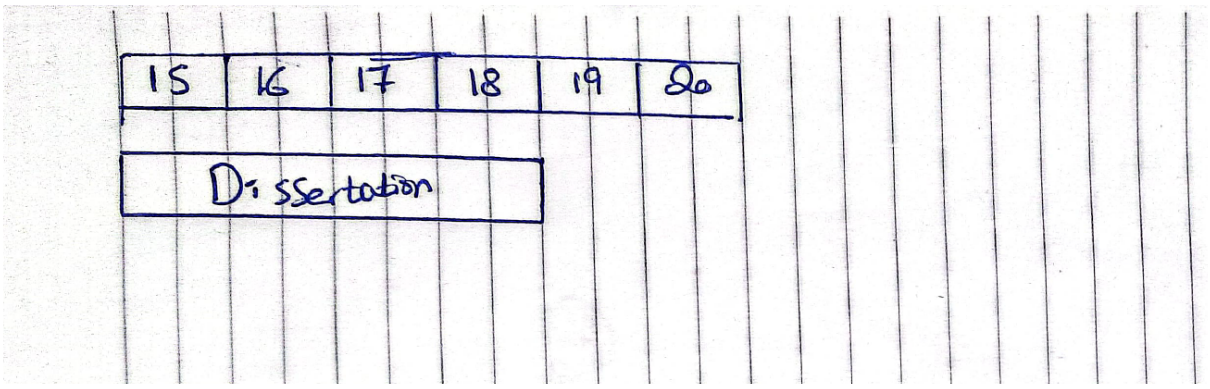
Also, additional frameworks and Libraries may be used, but at this point, it is not determined yet, as a point of reference the following might be used:

1. Frontend: React.js
2. Backend: Express.js
3. Drawing: d3.js or p5

4. Program of Work:

In this section we will discuss the timeline of the project, the timeline provided here is just a draft for reference, and it will only consider the MVP Features, and changes to the timeline during the duration of the project are expected and normal.





5. References:

1. Combinatorial Object Server
   http://combos.org/nauty
2. Nauty and Traces
   https://pallini.di.uniroma1.it/
3. CS Academy Graph Editor
   https://csacademy.com/app/graph_editor/
4. Graph Online
   https://graphonline.ru/en/
5. D3 Graph Theory
   https://d3gt.com/
6. Algorithm and Data Structure Visualisations, University of San Francisco
   https://www.cs.usfca.edu/~galles/visualization/Algorithms.html