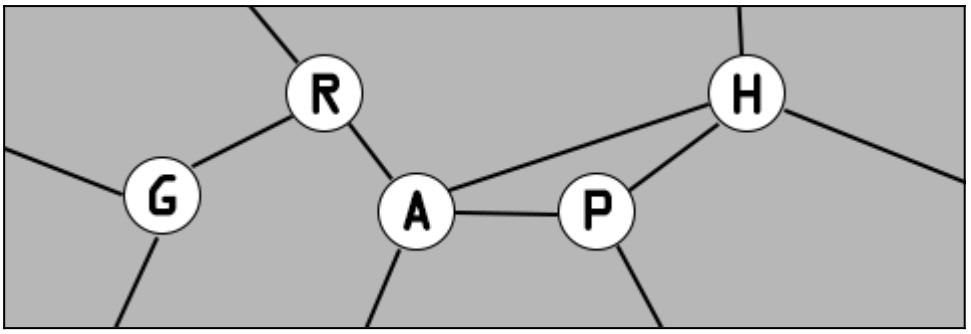
Candidate Name: Khalyl Boukhanouf Centre Name: George Spencer Academy
Candidate Number: 7030 Centre Number: 28278



Graph Visualiser
By Khalyl Boukhanouf
7517 A-Level Computer Science

Candidate Name: Khalyl Boukhanouf **Centre Name: George Spencer Academy**
Candidate Number: 7030 **Centre Number: 28278**

Contents

Analysis

Introduction

Graphs are an abstract data type which can be hard to visualise, especially for students who are new to the concept. Furthermore, as students learn about the algorithms they might need to know how they work so that they can understand more advanced concepts since they are usually the basis of other complex maths further on in their course. My aim is to make teaching graphs more interactive.

My program is aimed to allow students to design their own graphs and trees and use algorithms. It will allow them to understand the topics more which will help them in their studies to achieve better grades. The algorithms incorporated in my projects will consist of Prim's, Kruskal's and Dijkstra's.

End User

As described above this program will mainly be catered to teachers and students studying computer science and further maths. Teachers could use the program in adjacency to their lesson to educate their students on how the algorithms work. The students could also use it for their revision outside of lessons as it will help them achieve better grades. The program has to be user friendly so that someone can use it despite their IT skills being subpar. The number of students that will be using the program will be around 30 people which incorporates my further maths class as well as my computer science class.

The specific user is Mr Simpson. He teaches Further Maths and specialises in Discrete maths. This includes graphs which is why he is the targeted user. His diagrams are very detailed however the more edges there are in a graph the more messy it can become on a whiteboard or paper so the construction of this algorithm will allow the students to experience a less convoluted method of explanation. After asking him what he would like to see included in my project, he responded saying that it would be nice to see the working out for the algorithms. The method to do this is to project a list of all the nodes which are being connected as well as rejected edges which create a cycle. This makes it clearer to the user as to what the algorithm is doing.

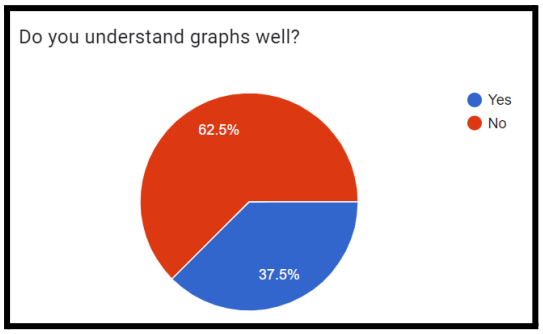
In addition I decided to ask my computer science teacher Miss Fell how I should extend myself in my project. She responded by recommending me to be able to create an import and export system so that graphs can be shared by different users. I decided to store the data in a json file and then export it so that it can be shared externally.

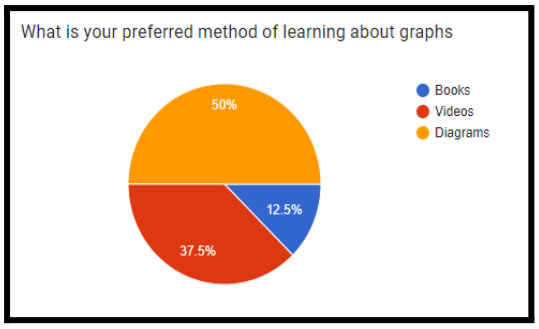
Candidate Name: Khalyl Boukhanouf Centre Name: George Spencer Academy
Candidate Number: 7030 Centre Number: 28278

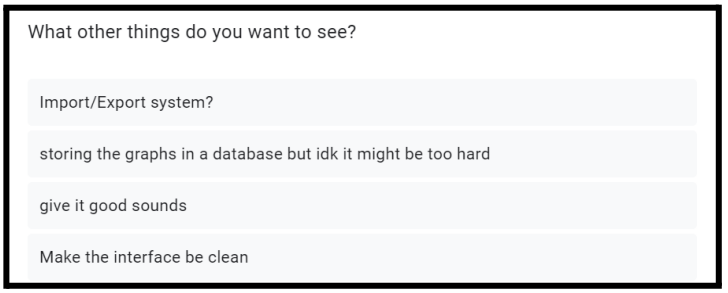
Investigation

The current system involves using a whiteboard and it includes a lot of crossing out which can appear messy and unclear as to what is actually happening. This can cause confusion for the students.

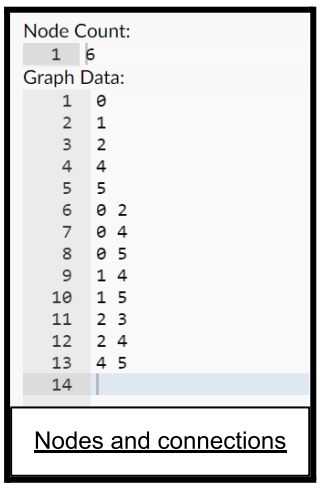
Students responses:





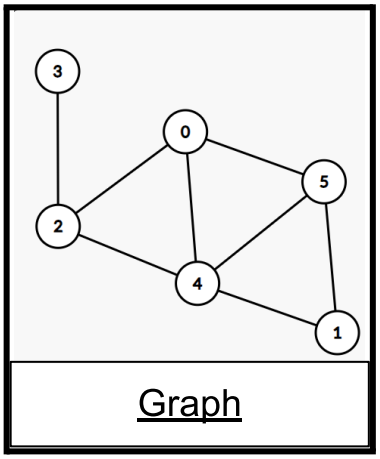


The majority of students do not understand graphs that well, mainly because they are an abstract data structure which can be very convoluted. THis means that I have to ensure that the program will be as accessible and easy to understand as possible. Also based on feedback from a questionnaire, the graph visualiser must be able to compute the basic algorithms from a graph that the user makes. In addition, it was recommended that I implement a system for sharing graphs such as an import/export system so that other users can acquire other users' graphs.



Nodes and connections

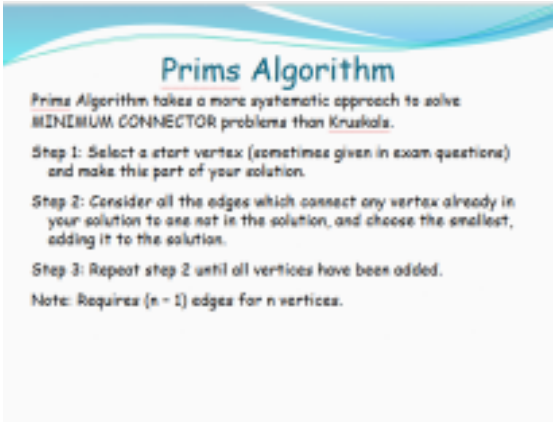
This website https://csacademy.com/app/graph_editor/ helps to visualise graphs by allowing the user to create their own nodes and arcs in a system and also allowing them to be named however, the GUI for this website is not very user friendly as it involves the user to write the actual nodes and the user cannot edit the location of each node by dragging it which could cause some edges to overlap with each other which would only lead to more unclear and messy explanations. In addition, the website doesn't allow for algorithms to



Graph

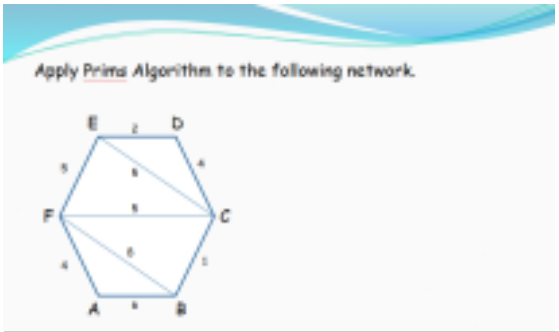
be carried out which is the main purpose of my program. However, the website provides a very simple method to switch between a directed and undirected graph which I would like to include in my own project.

Another system used is when Mr Simpson uses slides to teach students Prim's algorithm. This allows him to display examples on a screen at the front of the classroom while also running through the example with the students. The drawback of using this method is that creating the graph with text boxes and lines is a very arduous process, and on top of that, Mr Simpson usually likes to add animations which is also time consuming.



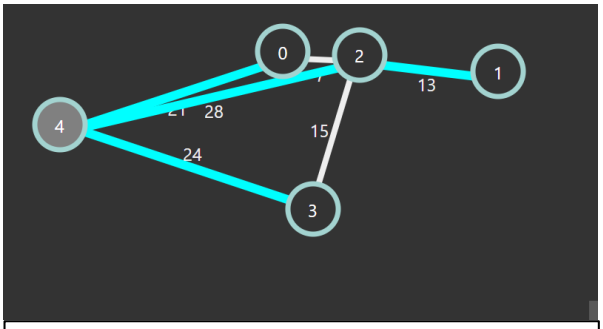
Explanation as to how it is taught

My plan is to make a solution which provides a method for graphs and matrices to be created and then have algorithms applied to them. This will allow them to be more interactive and for the class to understand them more. A step by step system needs to be shown so that the users can understand the algorithm. This will also reduce the need to use separate textbooks for the students.

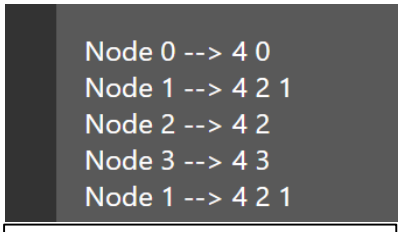


How the algorithm is shown

The website https://algo-dijkstra.vercel.app/ has a way to visualise dijkstra's algorithm and it is very similar to what i want to accomplish in terms of design where the user is able to create nodes and make a graph but on top of that the user is also able to select a specific node and choose another node to traverse to and then the algorithm will apply to the graph and records what the algorithm has done at every point as well.



Graph with Dijkstra's Algorithm applied to it



Explanation of what's happening

However, some disadvantages to this website is that once the user has put down a node they are unable to remove it or move it around which is

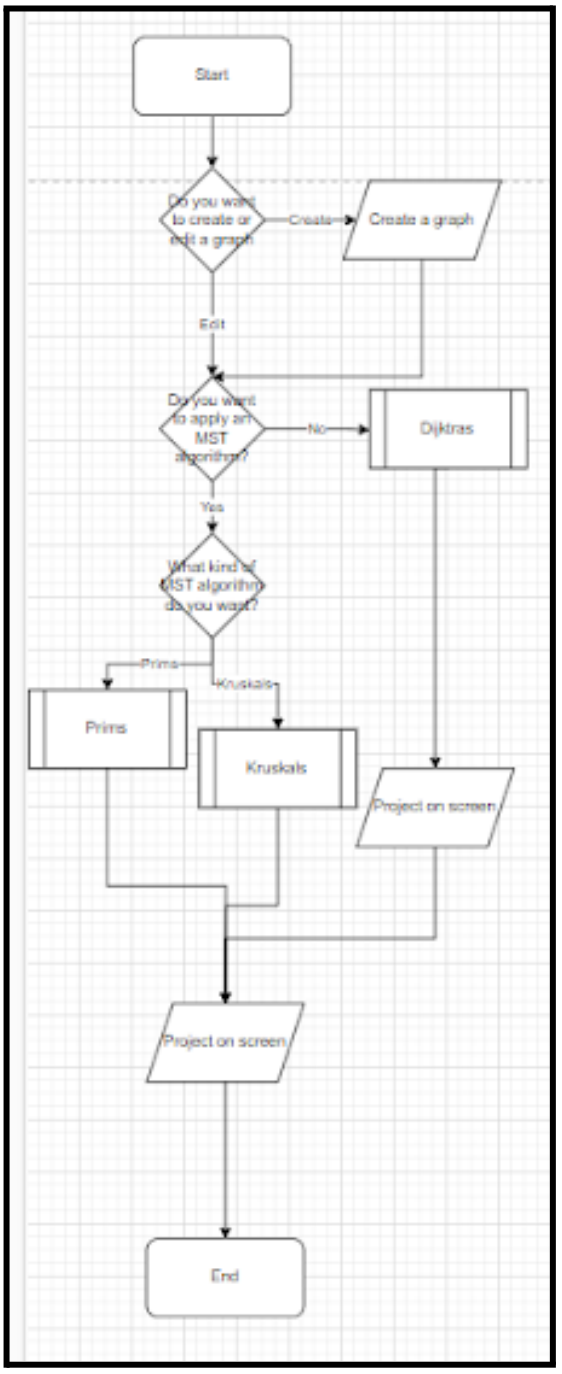
something that I would like to change about my project. In addition to this the project also does not have multiple algorithms being able to be applied to the graph created as it is specifically designed for dijkstra's. In my project I will allow multiple algorithms to be run on the graph. In addition, the website doesn't allow the user to edit the weight of the edges which means that they are set already which decreases the versatility of the program.

<table>
<tr><td></td><td>Graph editor</td><td>School system</td><td>Dijkstra visualiser</td></tr>
<tr><td rowspan="3">Advantages</td><td>Very easy to create nodes/edges</td><td>Demonstrates algorithms very well</td><td>Very easy to use, straightforward to understand</td></tr>
<tr><td rowspan="2">Easy to switch between directed/undirected graphs</td><td rowspan="2">Very easy to use</td><td>Shows each step of how it works</td></tr>
<tr><td>Very similar design to mine</td></tr>
<tr><td rowspan="4">Disadvantages</td><td>Not very user friendly</td><td rowspan="2">Can be difficult for some students to understand</td><td rowspan="2">Cannot delete or move nodes around the screen</td></tr>
<tr><td rowspan="2">No ability to apply algorithms</td></tr>
<tr><td rowspan="2">Cannot drag nodes around easily</td><td rowspan="2">Impractical for large graphs</td><td rowspan="2">Only uses a single algorithm</td></tr>
<tr><td>Can get messy</td><td>Cannot edit the weights of the edges</td></tr>
</table>

Based on this research I think that I should add a system where it is simple to create edges and nodes but also demonstrates the algorithms well. I want the user to also know the step by step process of joining the edges. However i also want the design to be user friendly so that it is as easy as possible for the user to use the program since some of the websites that i have researched aren't user friendly which makes it difficult for users to understand which contradicts the goal of the project which is to simplify graph theory for students. These will be some of my objectives.

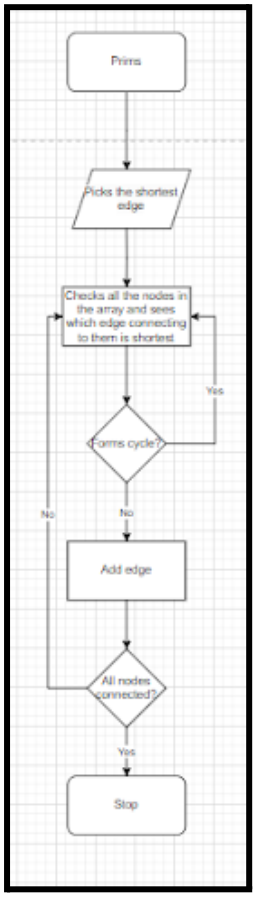
Flowchart system

The main program which will encompass the majority of the backbone of the system is to the right. It will be the main file. Inside this will be the main display and GUI and in it will allow the user to decide which kind of algorithm to apply to their graph that they have created. This involves splitting the different types of algorithms into MST algorithms and pathfinding. This allows the user to then differentiate between what kind of algorithm they want to apply to the graph. Then the algorithm will enter a subroutine and then apply the algorithm onto the graph and then after it finishes it will project it onto the screen.

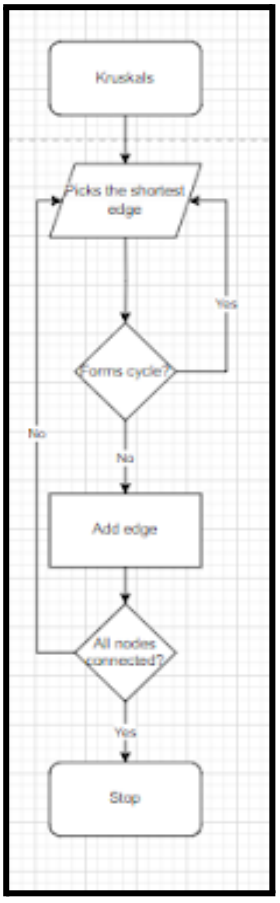


The other flowcharts below are the designs of the other algorithms. They describe how each algorithm works.

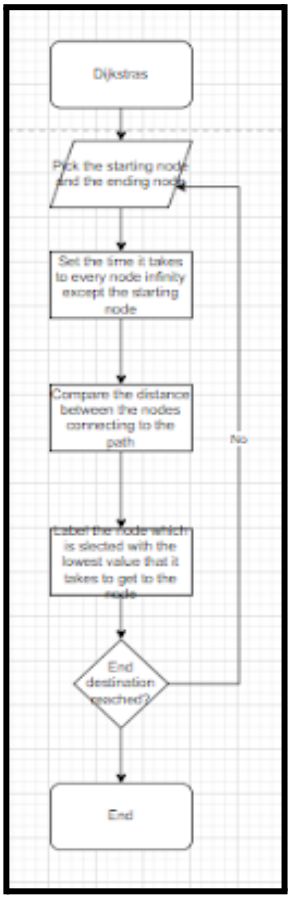
Prim's



Kruskal's



Dijkstra's



Complexities

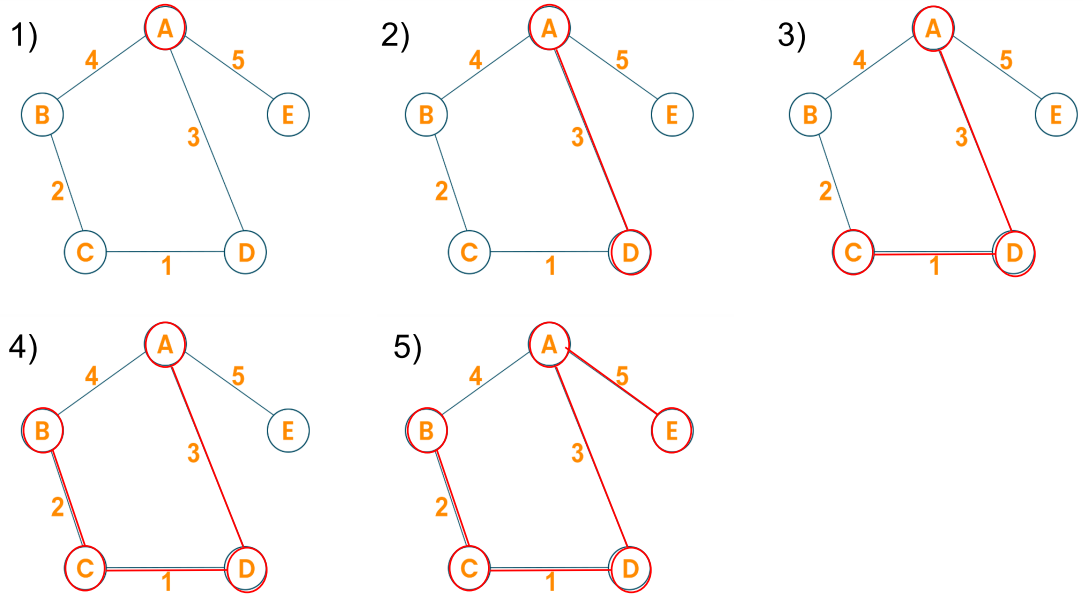
The main difficulty in this program will be the programmation of the graph algorithms.

Prim's algorithm:
The purpose of this algorithm is to create an MST within a graph, the way it achieves this is by selecting a starting node and it then sees the shortest edge connecting to that node and will select the edge for the MST and it will add the new node. It will then search through all the nodes which have been selected to find the shortest edge. If the edge creates a cycle then it will be ignored since it will not be a tree.

Originally I thought about implementing it from a matrix by removing the for all the selected nodes from the matrix so that no cycles occur, however in python, problems arise when trying to delete rows in the matrix since the order of the nodes will change meaning that it will become difficult to assign positions to each node if they're constantly changing.Instead, I decided to do it in a more efficient way by creating an array for nodes to be searched and then checking each node in it to find the shortest edge. Then to check for cycles, when adding the node to the array it will check whether the node is already in the array and if it is, then it means if it is added again it will form a cycle and therefore it can be prevented.

Example:

1) The node A is selected in the graph to start from it
2) The shortest node D is then connected to out current tree
3) Out of all the nodes in the MST, the shortest connecting edge is D-C
4) The next node with the shortest distance from the MST is B
5) The next shortest node is A but this creates a cycle so we disregard it
6) The next shortest is E which connects all the nodes which completes it



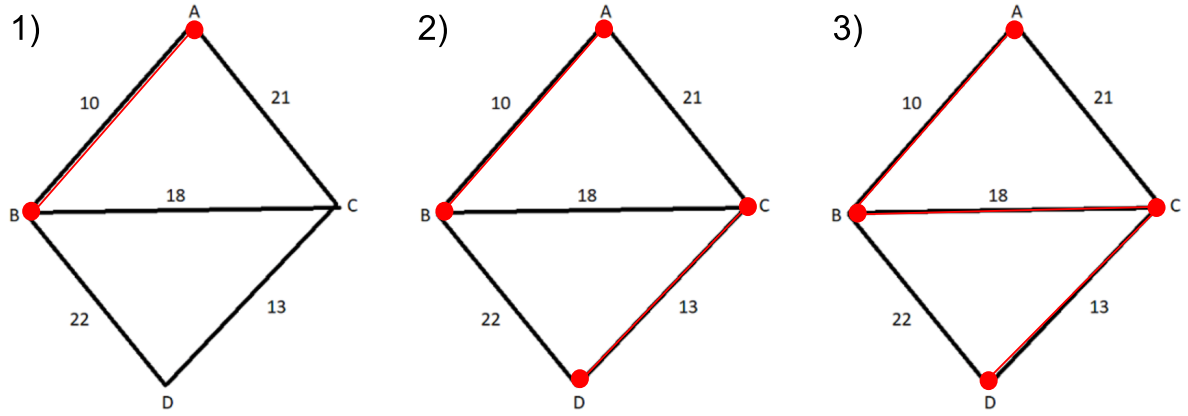
Kruskal's algorithm:

The purpose of this algorithm is the same as Prim's algorithm. It is just a different approach to finding the MST of a graph. This algorithm works by finding the shortest edge in the entire graph and then adding it to the MST. It will then find the next shortest and so on and it will repeat until it has all the nodes connected. However, once again it will not add an edge if it will create a cycle in the MST since MSts don't have cycles and are trees.

To program this I have decided to approach this similarly to my approach to Prim's algorithm and I will be adding the nodes to an array to identify whether a cycle has been created. The difference is that there will also need to be a counter to see whether 2 different sub graphs become connected to make sure that there will not be false positives when identifying cycles.

Example:

1) Select the shortest edge which is A-B, this connects both these nodes
2) Select the next shortest edge which is D-C and this connects nodes C,D
3) Repeating this gives us the edge B-C and because now all the nodes are connected together this completes the MST algorithm
4) If the shortest edge would have produced a cycle then it would be discarded and we would move on to the next edge



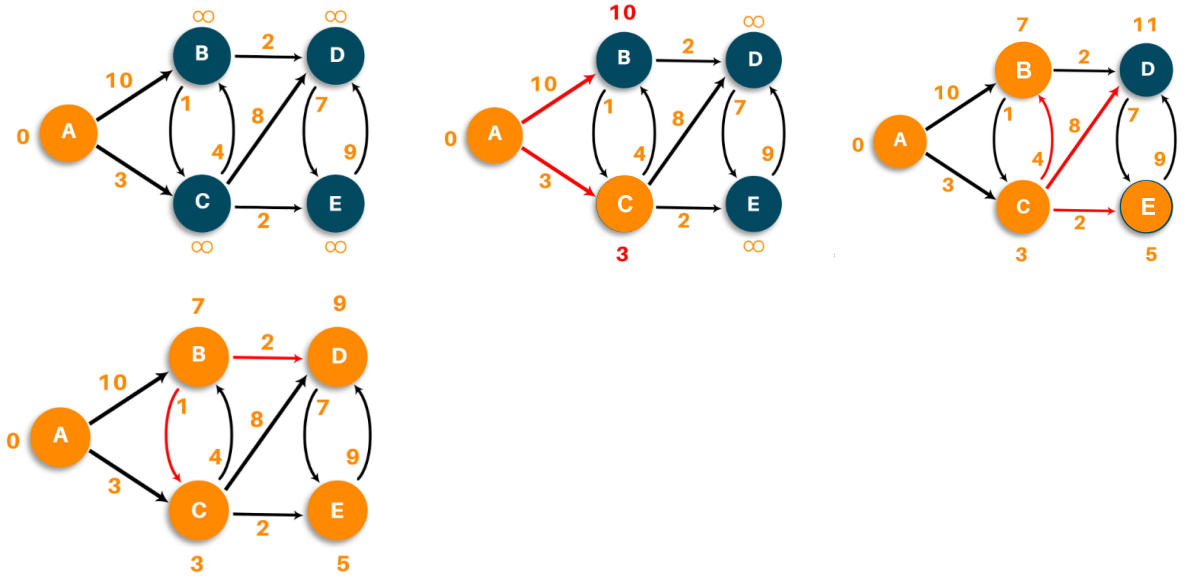
Candidate Name: Khalyl Boukhanouf Centre Name: George Spencer Academy
Candidate Number: 7030 Centre Number: 28278

Dijkstra's algorithm:
The purpose of this algorithm is to find the shortest path between 2 nodes in a graph. This works by starting at the starting node and labelling every other node as infinity. This means that it currently takes infinite time to reach that node. Then the neighbouring nodes are checked to see the quickest route. Then the node will be relabeled to its actual time to get to it. This will be the shortest time to get to that node. This process will then be repeated to all the nodes in the path until the ending node is achieved. This will then be the quickest time it will take for someone to travel the distance between the 2 nodes.

I plan on programming this by creating a separate array for all the times it takes to get to each node. They will initially be labelled with INF and then as they start to get labelled correctly they will then be relabeled as the algorithm proceeds. This will work from an adjacency matrix and similarly to prims where it will look through the matrix and check all the nodes connected to the current node and see what the minimum times will be and then it will add it to the algorithm.

Example:

1) Currently every node is set to have infinite time to reach.
2) Then the algorithm checks the shortest path it takes to get to each node and associates that node with that number B-10, C-3
3) The algorithm then repeats by picking the shortest path from the nodes. If any overlap occurs then the lower number is assigned to the node B-7, C-3,E-5,D-11
4) The process then repeats while checking each connection and updates the shortest path, D-9. When every node is checked the final distance is 9 because it's the last number to be associated with the finishing node



Pseudocode

The pseudocode below is for the different algorithms that I will be creating within my project. I have split them into 3 parts so that they are more legible and I have also commented them so that anyone can understand what is happening. I will incorporate them in the final project later so that the program runs smoothly.

Prims

```
MST-PRIM(G,w,r):
      #Searches each node and sets the value of it to infinity and
its parent to nothing
      For each u in G.V
            u.key = INF
            u.parent = NIL
      #Sets the root node to have a value to be incorporated into
the MST
      r.key = 0
      #The INF graph will be stored in Q so that it can be tampered
with without affecting the original graph
      Q = G.V
      #The program will loop until Q is empty
      While Q
            #The program will get the shortest edge which is
connected to the root node from Q and then check whether it will
form a cycle if added to G.V
            u = EXTRACT-MIN(Q)
            For each v in G.Adj[u]
                  #Checks each vertex in the graph and compares it to
see whether it will form a cycle
                  If v in Q and w(u,v) < v.key
                        #If it doesn't then it will add the edge to
the graph of v so that it becomes part of the tree
                        v.parent = u
                        v.key = w(u,v)
```

Kruskals

```
MST-Kruskal(G,w,r)
      #Searches each node and sets the value of it to infinity and
its parent to nothing
      For each u in V[G]
            Key[u] = INF
            u = 0
      #Sets the root node to have a value to be incorporated into
the MST
      Key[r] = 0
      #The INF graph will be stored in Q so that it can be tampered
with without affecting the original graph
      Q = V[G]
      #The program will loop until Q is empty
      While Q
            #The program will get the shortest edge in the whole
 graph and then check whether it creates a cycle or not
            U = EXTRACT-MIN(Q)
            For each v in Adj[u]
                  If v in Q and w(u,v) < key[v]
                  #If it doesn't then it will add the edge to the
      graph of v so that it becomes part of the tree
                        v = u
                        Key[v] = w(u,v)
```

Dijkstras

```
DIJKSTRA(G,w,s)
      #The program finds the starting node from the graph
      INITIALIZE-SINGLE-SOURCE(G,s)
      #Creates the array S for finding the path and once again
stores the old graph into Q so that it can be tampered with without
affecting the original graph
      S = []
      Q = G.V
      While Q
            #Finds the smallest edge and adds it to the array S so
that it can check whether it is the shortest way to get to that
vertex or not
            u = EXTRACT-MIN(Q)
            S = S and {u}
            For each vertex v in G.Adj[u]
                  #If it finds that the node has a quicker path then
it will shorten the time it takes by however much it should be
                  RELAX(u,v,w)
```

Data Dictionary

I will be using various different types of data within my program especially since it is mainly centred around 1 data type. However, I will not just be using graphs but also trees, arrays and of course variables.

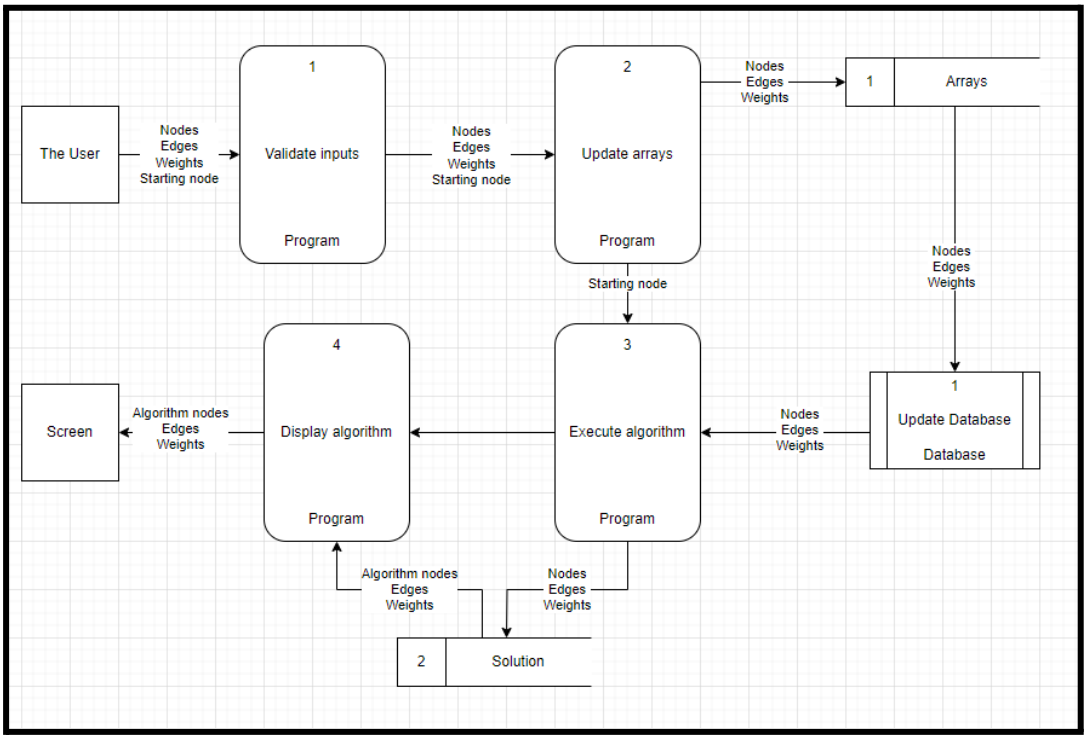
Name	Data type	Validation	Comments
Starting Node	String	Should be a named node in the graph	This will be selected when the user clicks on a specific node on the graph.
Neighbours	Array	Needs to consist of sub arrays which have 2 values: The name of the connected node and its weight	This will be inside the node array so that it can act as an adjacency list.
Nodes	Array	Should not contain any numbers	It is basically an adjacency list where it will consist of objects from a class. These objects will have attributes names, neighbours and position
Matrix	Array	Has to be either an integer or a "-"	It will be the adjacency matrix for the graph since it is easier to apply the algorithms on an adjacency matrix than an adjacency list
Algorithm	String	Has to be either Prim's, Kruskals or Dijkstras	This will determine what kind of algorithm will be applied to the graph and whether it is a pathfinding algorithm or an MST algorithm

Candidate Name: Khalyl Boukhanouf Centre Name: George Spencer Academy
Candidate Number: 7030 Centre Number: 28278

ISPO

Inputs	Storage	Processes	Outputs
The user places nodes on the screen and links them together using edges. This may also be done by creating a set of cells to make up a matrix Each edge/cell is given a weight by the user. A starting node/column is selected by the user. The user selects an algorithm to apply onto the graph.	The graph will be stored in 2 different ways, they will be stored as an adjacency list and also as a matrix. This is because it is easier to display the graph on the screen using the adjacency list but it is easier to apply the algorithms to the matrix. In addition the data for users will also be stored in a database so that the user can login to it multiple times	It is checked whether each node/column is connected to the. It is checked whether each weight is valid Each edge/cell and its weight is added to an array. The algorithm chosen is then applied to the graph which the user has created	The minimum spanning tree is displayed onto the screen if prims or kruskal's algorithm is picked The shortest path between 2 nodes is displayed onto the screen if dijkstra's algorithm is picked.

DFD



Data Volumes

The majority of the data used by my program will be entered by the user or generated by the program itself. For example, the nodes will be created by selecting a new node and the weights will be entered by clicking on an edge and changing its value. In addition they will also choose their starting node. There will be a numerous amount of variables in my program such as Boolean values solved whilst it runs. The weights of the edges and the connections formed can be changed from an adjacency matrix to an adjacency list with relative ease. My program will also sometimes be reading data from files such as a json file or a database file for user customisation

There will be no need for my program to use data from any other sources such as from an API since it has no requirement.

Limits and constraints

My project will have to have a large amount of storage since it will have to include multiple 2D arrays and the more nodes that the user incorporates, the more memory my program will use. This will make the program run slower as it will mean that every time the user opens the program, all the graphs will open as well. This means that every time an algorithm is run on a program there will have to be multiple selections to ensure that the right graph has the algorithm applied on.

To fix this problem, I intend to create external storage outside the program for the graphs which will mean every time a graph is opened it will load it from secondary storage. This means that it will only apply the algorithms on the only graph available. The way to store it will be in databases but to keep the database normalised, I will have to pickle the graph so that i can use it as a string and unpickle it when receiving it from the database. The user will be able to create as many graphs as they want and store them in the database just as long as they have enough storage on their local device. However, I will not need to check that the user has enough storage because the graphs will be stored as strings which means that they will only be taking up bytes of data and since most devices have around 200GB of storage, I think it's an overcomplication to add a feature that monitors the users storage.

Objectives

1) My project must be able to have a GUI system for moving nodes around the screen with the mouse. It needs to only select 1 node when they overlap and not multiple so that they don't group together. The method to incorporate this is to have each node on a separate layer so that if 2 nodes are on the same layer it will pick the node which is on the upper layer. I want to be able to achieve this system by October 2023

2) My project must also be able to create edges from nodes when holding down the click while the mouse is on the node and then they can drag it to another node to connect it. It will be a straight line from node to node and if the user does not connect to another node then as they let go it will just delete the edge. I want to be able to achieve this by November 2023

3) My project will also have a right click system on the graph editor page where the user can right click to have options such as create a new node or delete a node or copy and paste. This will be similar to microsoft word or powerpoint and how when they right click there is a menu on there. I want it to also deselect if they click off it. In addition if they don't select anything then the right click menu will have certain buttons greyed out as it will not be able to execute those functions. It will be achieved I want to achieve this by December 2023

4) My project must also have a menu selection screen so that users will be able to select whether to log in or to sign up to the system. It will work using databases and it will be the loading screen for the project. The menu will be able to switch from login to sign up using a button similar to a google login system. This can be achieved by having a variable set to either login or signup so that it determines whether to create a new record in the database or whether to just read from a record in the database. I want to achieve this by September 2023

5) My project will also have a login system using databases. It will consist of 3 tables in which it will store account information, graph information and folder information. The graph and folder information will consist of lists so they will have to be pickled to store them as a string in a database. The graphs will be stored as an adjacency matrix and the folders will be stored as a tree since it will be a hierarchy. I want to also achieve this by September 2023

6) My project should also be able to execute Prims and Kruskal's algorithms on a matrix as they are MST type algorithms so I will create a separate file to compute these algorithms and then add it onto my main file. This is so that the code will not be cluttered and I will be able to run the files on their own to test them out to see whether they work before I add them to my main project. I want this done by August 2023

7) Another type of algorithm I will be using are pathfinding algorithms and the one I will use is Dijkstra's which will also be created in an external file that will be imported into the project. I want to complete this one again in August 2023

8) I also want my algorithm to have a menu theme so that if a user is not happy with the colour scheme then they can change the accent colours or the background colours. I want to achieve this by February 2023

9) I also need to add selections of different graphs from the database which will include information about them and have them on display. It will be useful to see what kind of graphs they are. It will be completed by January 2024

10) I want to add an export and import system where the user can import a graph or export a graph. It would be preferable to open it as a xml file although a .dat file may be used instead for simplicity. This shall be completed by March 2023

11) I want to make the GUI user friendly so that my program is as easy to use as possible. The way I will achieve this is by explicitly stating how the user should be able to create a graph. This is to be completed by February 2024

Possible solutions

Javascript, HTML, CSS and PHP
Initially I was thinking about making my project a web app since this means that anyone will be able to access it from any computer provided that they have internet access. This also means that when the user logs in they can log in from any device not just the device it's downloaded on. Furthermore, a web app means that the user does not need to download anything and everything is online.

However, the major flaw with this method of programming my project is that there is not a lot of processing power with web apps. To create one which has short loading times it would need to be a thin client however it means that a lot of data will have to be sent to and from the server meaning that it could cause the networks to be very slow which would be a problem in itself. However the main disadvantage this method has is that a server will need to be running 24/7 for the website to be active full time.

Python

Programming my solution means that the user has to download a file to run the software but it means that it is easier to run complex algorithms since the computer does all the processing power. This means that the speed of the program will depend on the processing power that the user's computer has which makes it much more reliable. Furthermore, an exporting and importing system is used to compensate for the lack of portability in the network.

Despite its advantages, python needs to be compiled individually and isn't used straight up. It also means that mobile users cannot use the program contrary to the previous method where people can access the project on any device provided it has the ability to access the internet.

GUI system

In HTML, the GUI is very basic to use but difficult to design since it is easy to project items onto the screen but it is more arduous to design the items on the screen since that requires knowledge of css. In python, the GUI is much simpler where I can use an inbuilt module named pygame in which I can directly project shapes and text onto the user interface. Pygame is also specifically made for programming games which means it will be easier to detect inputs from the user (i.e. mouse or keyboard). However, it does not incorporate a built-in typing function which means that I will have to implement it myself if I am to use this module. This could be difficult and tedious compared to other methods.

Another possible way to display a GUI in python is using the tkinter module. However, this module has less versatility since it is organised in a grid which means that the nodes on the graph cannot be placed anywhere and have to be placed on a grid. This means that in some cases it could be harder to visualise the nodes. However, it also has better graphics than pygame which means that the display is more visually appealing. Its syntax is also much simpler and it is inbuilt in python. This means that an external download is not required to run the program.

Chosen solution

Overall I have decided to choose to program my project in python so that it is more reliable and requires less components (only a client side not a server side). I also have more experience programming in python since I have been doing it longer than javascript which means if i encounter any problems while coding it means that i will be able to debug it much easier. In addition, I will program the project using pygame to provide a user interface to project the graphs onto the screen instead of tkinter since it is more versatile and it allows me to include a main part of the project.

Pygame can also group objects together which becomes very useful when I want to combine multiple nodes and edges into a single sprite which is the graph which can be manipulated. It is also very simple to make different kinds of objects since it is a procedural language.

Tkinter also has a very outdated look where it seems like it is from the 20th century which differs from what the students said which is a minimalist modern design.

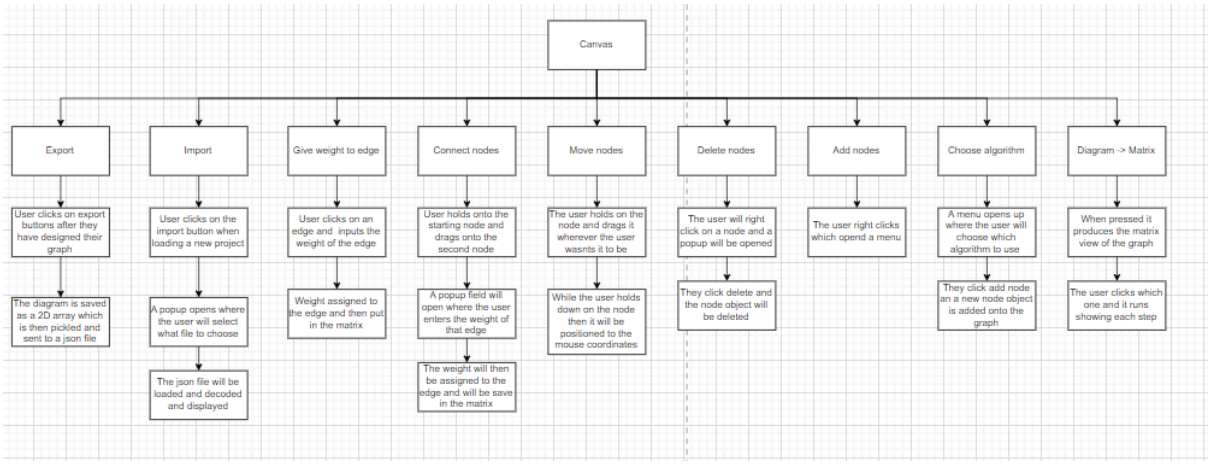
Design

I will be designing my program such that it will both function correctly as well as having a sleek design. It will consist of multiple pages which I will need to understand how they will be implemented as well as different files that need to be managed.

ISPO

Inputs	Storage	Processes	Outputs
The user logs into their account	Stored in the database and a variable so	Retrieves the graphs and folders associated with that user	The folders and graphs are displayed on the selection menu
The user signs up for an account	Stores it in the database as a new	Creates new presets for the user so that they start with a few folders	The folders then allow the user to create graphs and store them there
The user clicks to add a node	Node object recreated and also the matrix is updated	A new node is added onto the graph	The node will be displayed on the screen
The user clicks to delete a node	The node object is impeached and the matrix removes that node	The node is removed and so are all the edges connected to it	The node and edges connected to it will no longer be displayed
The users holds the node to move it around	The coordinates of the node are updated	The coordinates of the node become the same as the mouse	The node should move to wherever the mouse is when its held down
The user enters the weight of an edge	Weight of the edge is updates in memory	The edge will be incorporated into the matrix	The weight of the edge will be displayed onto the screen
The user selects an algorithm to apply to the nodes	The variable will store the name of the algorithm	The algorithm will be applied to the matrix	The Algorithms output will displayed on the screen

Structure



Data Dictionary

Name	Data type	Example	Use	Validation
Node	String	"A"	Used to identify the different nodes in the graph	Should be a string
Weight	Float	7	Stores the weight of an edge in entered by the user	Should only be rational numbers > 0
Coord	Array	[640,360]	Stores the location of the nodes on the canvas	Should be a an array with only 2 items which would both be integers
Edge	Array	["A","B",6]	An array which stores the array which are connected by that edge and its weight in the entire graph	Either an array with 3 items or 2 items depending if the graph is weighted or not and the first 2 items should be names of previous nodes and the 3rd should be a float
Algorithm	String	"Kruskal"	Used so that the program knows what algorithm to undergo when the RUN button is pressed and execute	It should be either one of 3 values: Kruskal, Prim or Dljkstra

MST Edges	Array	[[“A”,“B“,6], [“B“,“C“3]]	2D array which stores all the edges which are incorporated in the MST	Should be a 2D array in which all the subarrays have 3 different items in them. Where they also pass the validation check of Edge
Position	Integer	4	It determines what node is selected when nodes are stacked on top of each other and the user clicks	It should be an integer and should increment every time a new node is added similar to the program counter
Selected Node	String	“D“	Used so that when the user drags a node the program knows which to move	Has to just be a node which is in the graph with the lowest position
Right click	Boolean	True	So that the program knows whether the right click button has been pressed	It should be active when the user has right clicked but should be false when the user left clicks somewhere else
Add node	Boolean	False	So that the program knows that when the user left clicks it will create a new node and not move around the canvas	It should be true when the right click menu is open and the user clicks it
Move node	Boolean	True	So that the program can move the nodes around when the user clicks on them	Should be True when it is the lowest position and it has been held down and dragged across
Mouse coords	Array	[0,0]	So that the program knows where to move the node when the user holds on it	It should always change to whatever the current mouse position is.
AverageX coords	Integers	[100,100]	In the menu system it will be able to show a preview of the densest part of the graph	It must be changed whenever the user saves and exits the canvas
AverageY coords				

Matrix	2D Array	[[0,1],[1,0]]	Will be used to display the matrix form of the graph being produced as well as it being useful for the algorithms	It must be changed whenever a user adds a node/edge it must be updated.

Validation

These will be the techniques that I will use to validate the program. I have listed the different methods I will use.

Format check

This will check that the correct data type has been added so that errors don't arise. For example, the weight must be a float and the node name must be a string.

```
ValidInput = False
      WHILE ValidInput = False:
            INPUT String
            TRY:
                  CONVERT String to Integer
                  ValidInput = True
            EXCEPT:
                  OUTPUT "Enter a valid input"
      ENDWHILE
END
```

Presence check

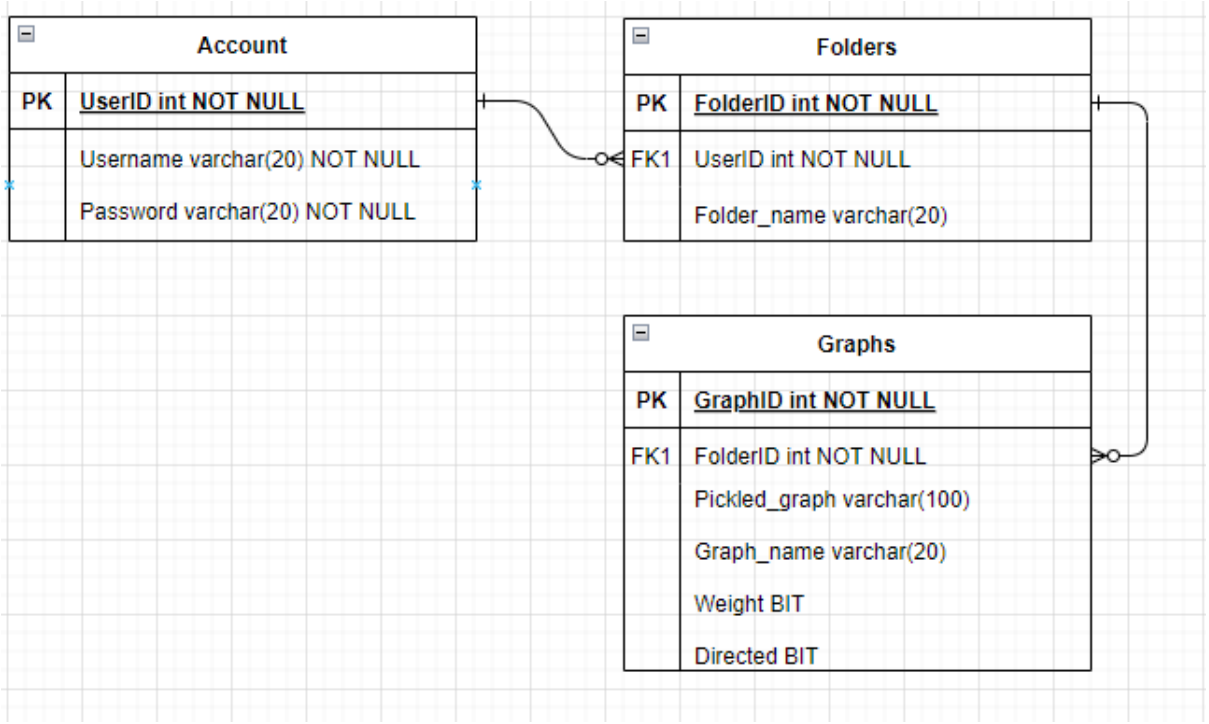
This will check that the user has actually entered something in the input fields because if they haven't then they won't be able to be displayed and it could potentially be a problem. The way that I will combat this is that for every input I will have the variable predefined which means when the user inputs something they are rewriting the variable and not assigning it to a new variable. This means that if the user does not enter anything it will use the pre-defined number.

User of GUI

I plan to use buttons and switches to make it more interactive for the user instead of other input methods such as text boxes to reduce the chance that the user inputs valid data. This will mean the user will only be able to enter data that is available.

ERD Diagram

My project will require databases to store account information so that the user can save their graphs and come back to them after another user has logged in. The way it works is that when the user logs into the account they will be able to access certain folders which will contain certain graphs.



Sample Algorithms

These will be some of the algorithms that I will be incorporating in my program. They will be used in different areas of the program but will mainly be in the root file since that is where the majority of the backbone of the program will be.

Choose login or signup

This program works simply by just having 2 variables, signupbutton and loginbutton which will be set to true if their respective buttons are pressed. This will be able to alter the page, making it either a signup page or a login page.

```
IF SignupButton is CLICKED
      THEN RUN SignupPage
ELSE IF LoginButton is CLICKED
      THEN RUN LoginPage
ENDIF
```

Choose Algorithm

This works in a similar way to the login/signup one where the user will click on the algorithm that they want to run and it will run the algorithm that they want.

```
IF PrimsButton is CLICKED
      THEN RUN PrimsAlgorithm
ELSE IF KruskalsButton is CLICKED
      THEN RUN KruskalsAlgorithm
ELSE IF DijkstrasButton is CLICKED
      THEN RUN DijkstrasAlgorithm
ENDIF
```

Connecting nodes

This program first has a list of all the nodes and sets the closest coords to be that of the mouse. This is the base case. Then while the node is right clicked. The program will check if the mouse has moved and carry out the calculations. This is because if the calculations were run every game loop then it would decrease performance. When it does move then the closest node will be found by iterating through all the nodes and seeing which has the smallest distance to the original node and selecting it.

```
Nodes = [List of all nodes]
Closest = COORDINATES OF MOUSE
Closest_node = Node
WHILE Node is RIGHT_CLICKED
      IF MOUSE HAS MOVED
            THEN Closest = COORDINATES OF MOUSE
            #This for loop is used to find the closest node to the
            mouse
            FOR i in Nodes
                  IF |i.pos| < |Closest|
                        THEN Closest = i.pos
                        Closest_node = i
                  ENDIF
            ENDFOR
      ENDIF
      HIGHLIGHT Closest_node
ENDWHILE
#This makes sure that no edge is draw to itself by accident
IF Closest != Node.pos
      THEN DRAW EDGE FROM Node to Closest_node
ELSE
      THEN CONFIRM A LOOP WILL BE CREATED
      IF CONFORMATION = True
            THEN DRAW EDGE FROM Node to Closest_node
      ENDIF
ENDIF
```

Create node

This algorithm works by first making sure that the add node button is clicked. This will initiate a variable called AddNode to be True. This allows a while loop to be initiated where it will check whether the screen has been clicked. If it has then a new node will be created at the coords of the mouse and it will also create a node object which references that point. Then the AddNode variable will be set to False since the user has already added a node and they will no longer be able to add more until they click the add button again.

```
IF AddButton is CLICKED
      #This boolean shows if a user is trying to place a node
      AddNode = True
ENDIF
WHILE AddNode = True
      IF Screen is CLICKED
            #The node is created at the coordinates of the mouse
            THEN CentreCoordinates = COORDINATES of MOUSE
            CREATE Node OBJECT
            COORDINATES of Node = CentreCoordinates
            AddNode = False
      ENDIF
ENDWHILE
```

Delete node

This function works similar to the add button where it checks whether delete node is on and it will then check every node and see its coords. If the user clicks and the coords match up then the node will be deleted and the DeleteNode variable will be turned off.

```
IF DeleteButton is CLICKED
      DeleteNode = True
ENDIF
WHILE DeleteNode = True
      #Used for disabling DeleteMode
      IF DeleteButton is CLICKED
            DeleteNode = False
      ENDIF
      IF Node is CLICKED
            DELETE Node
      ENDIF
ENDWHILE
```

Moving nodes

This very simple program will just relocate a node once it's clicked. When it is clicked the the does coords will just be changed to the mouse coords which allows it to be able to be dragged around

```
WHILE Node is LEFT_CLICKED
      #The location of the node is updated to match the location of
      the mouse while the user holds down on the node
      COORDINATES OF NODE = COORDINATES OF MOUSE
ENDWHILE
```

Converting a list into a matrix

This works by iterating through all the nodes and seeing which nodes are adjacent to each other, when it does, then it will input into the grid the weight of the connection, it knows which nodes are which because of the amount of times the program has looped which corresponds to the node number

```
def list_to_graph(nodes):

    graph = []
    #Cycles through all the nodes
    for i in nodes:
        temp_list = []
        for j in nodes:
            #checks to see if the node is connected to one selected
            length = "-"
            for k in i.neighbours:
                #Checks to see if it is a neighbour
                if j.name == k[0]:
                    length = k[1]

            #adds it to the row of the node currently being examined
            temp_list.append(length)
        #Adds the row to the final matrix
        graph.append(temp_list)

    return graph
```

Converting a matrix into a list

This one works by creating a list and looping through the matrix and seeing each node's neighbours, it then finds the weight and appends it to a list which is then integrated into the other list. Then this will repeat until all the neighbours are integrated into the list

```
def graph_to_list(matrix, old_nodes):
    new_nodes = []
    Iterates through all the rows in the matrix
    for a, i in enumerate(matrix):
        neighbours = []
        #Iterates through all the connections that the node has
        for b, j in enumerate(i):
            #Checks to see if it has a neighbour
            if j != "-":
                #Adds it to the neighbours array
                neighbours.append([old_nodes[b].name, j])
        #Adds the list of neighbours to the object being created
        With also the name and its other attributes
        new_nodes.append(node(old_nodes[a].name, old_nodes[a].pos,
old_nodes[a].x,old_nodes[a].y, neighbours))

    return new_node
```

Finding the highest

This just loops through all the edges and finds the highest edge by comparing the current highest with the current one being inspected. This then is returned by the function

```
def highest_algorithm(graph):
    highest = 0
    for i in graph:
        #Checks to see every nodes connection
        for j in i:
            #Makes sure there is a connection
            try:
                #Makes the comparison to see if its the highest
                if j > highest:
                    highest = j
            except:
                pass

    return highest
```

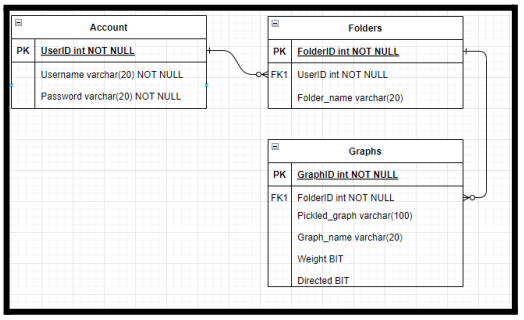
Creating the databases

These are the databases that will be created in the program and this the SQL that will be used to create them. There will also be ways to query the database so that the user is able to get the previous graphs that they have saved.

This will create the graph database which will store the graphs. They can be joined to the folders through the foreign key folder_num which will allow the 2 tables to be joined

```
CREATE TABLE graphs (
      folder_num int,
      graph str,
      name str,
      weighted bool,
      directed bool
);
```

(ERD)



This will create the folder database which will store the folder . They can be joined to the Users through the foreign key user_id which will allow the 2 tables to be joined

```
CREATE TABLE folders(
      user_id int,
      name str
);
```

This will create the folder database which will store the folder . They can be joined to the Users through the foreign key user_id which will allow the 2 tables to be joined

```
CREATE TABLE users(
      username str,
      password str
);
```

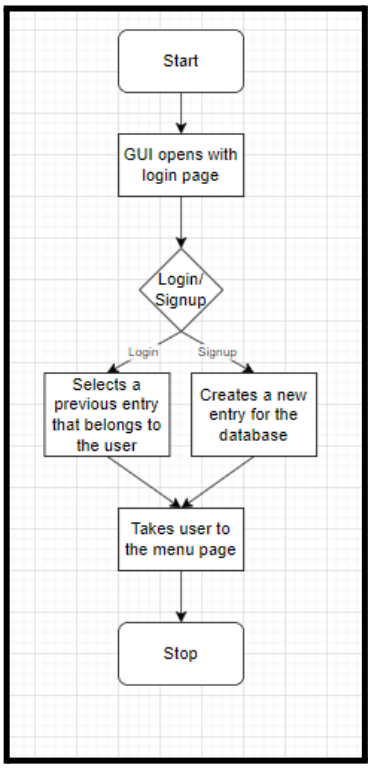
Querying the databases

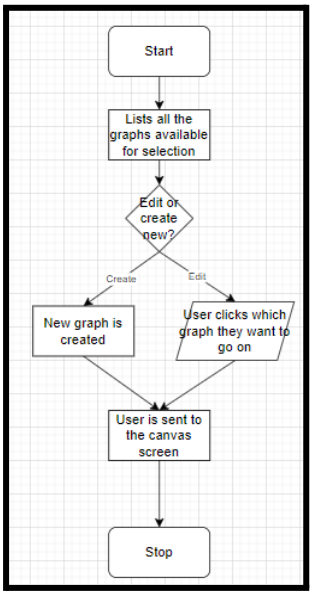
This will be the way to query the tables where the user will be able to select the graph and the name from the graphs/users database but the condition means that they will only be able to select it which match the username currently logged in

```
SELECT graph,name FROM graphs,users WHERE graphs.folder_num =
selected_folder AND users.username = username;
```

Flowchart

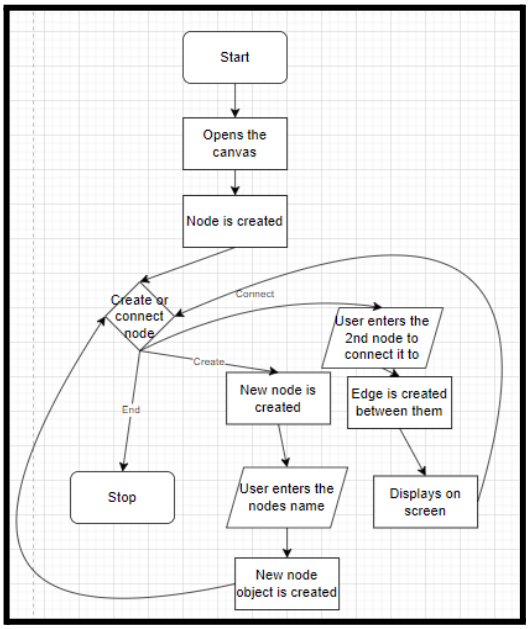
This flowchart will be what the user will first come across when opening the program. This will first open the GUI and then display the login system. This will then be free for the user to log in or sing up depending on what they choose



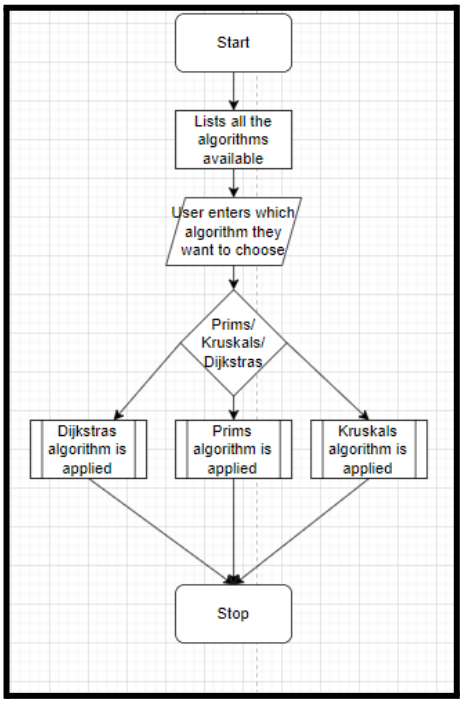


This flowchart will be after the user has already logged into the system and their graphs will now be displayed. This will show them their graphs from their database and then they can either create a new graph or edit a previously made graph.

This flowchart shows what happens when the user opens the canvas to either edit or create a graph. They will be greeted with the option to either create a node or make an edge and then they can carry on the process to complete their graph.

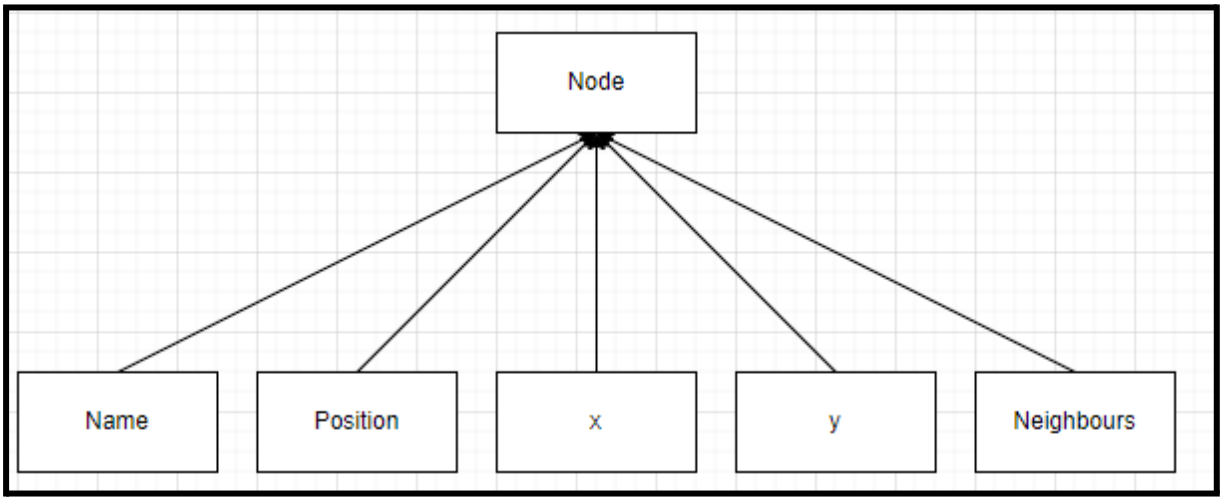


This last flowchart shows how the user can select which algorithm to pick on the screen. The 3 choices will be displayed for the user to click on and choose

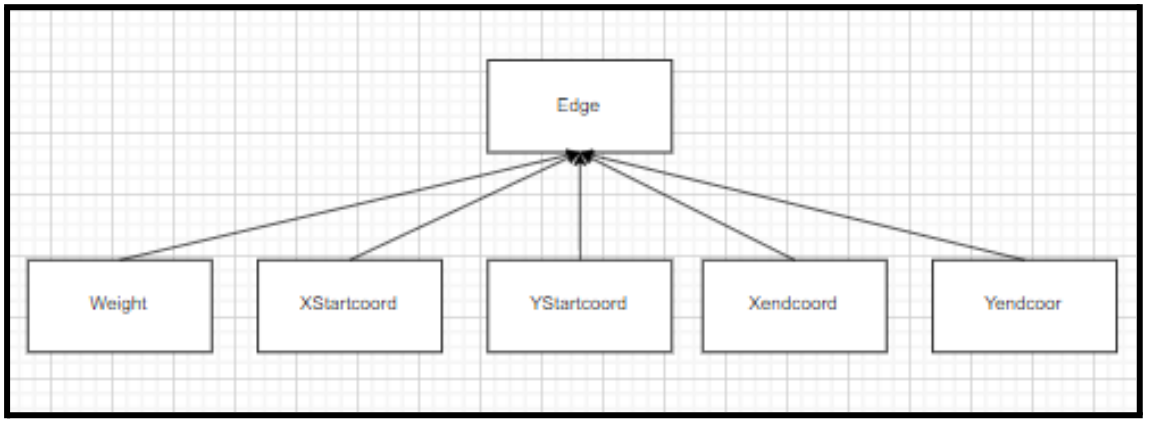


OOP

There will be 2 main classes in my program. The first being the class for the nodes so that all the nodes will be able to be stored efficiently. It will cause them to have attributes of the name of the node as well as its neighbours, coords and its order of the nodes. The last attribute will be used when a user clicks on a node. If the 2 or more nodes are on top of each other then the order will decide which node to move. Some of the methods I will incorporate will be connecting the nodes and also checking whether the mouse is hovering over the node which would signify that the user wants to click on it.



The second major class will be the edges in the graph which will have the attributes of the weight and the start and end coords. Some of the methods will be whether the edge has been clicked on, and updating the weight when it's changed. It will also calculate the position whenever the node is moved around.




```
Class Node
      Private Name
      Private Pos
      Private x
      Private y
      Private Neighbours

      Public
            Function Connect_node
      Public
            Function Is_hovering
Endclass

Edge = Class(Sprite)
      Private Weight Float
      Private XStartCoord Integer
      Private YStartCoord Integer
      Private XEndCoord Integer
      Private YEndCoord Integer

      Public
            Procedure CalculatePosition
      Public
            Procedure ClickedOn
      Public
            Procedure UpdateWeight

End
```

Data Structures

My program will need to use many different kinds of data structures for it to function correctly.

A lot of the data will be stored in arrays since they are the most versatile kind and they group multiple different variables which will mainly be used to store data such as using a 2D list to make a matrix to store the graph in. It will also be used to list the neighbours for all the nodes. In terms of the algorithms it will be used to store the does which will be included in the final result or the edges depending on the algorithm.

The majority of data will be held in variables. This is because they store a single part of the program. This can be used to distinguish what kind of algorithm is being used at the moment. It can also be used to store the username and password of the user when entered.

Evidently, I will be using graphs to store the actual graphs being produced within my program which will be stored as a 2D array as this will be the adjacency matrix. Furthermore, I will be storing the relationship between the folders as a tree so that it can be compactly stored efficiently.

File Structure

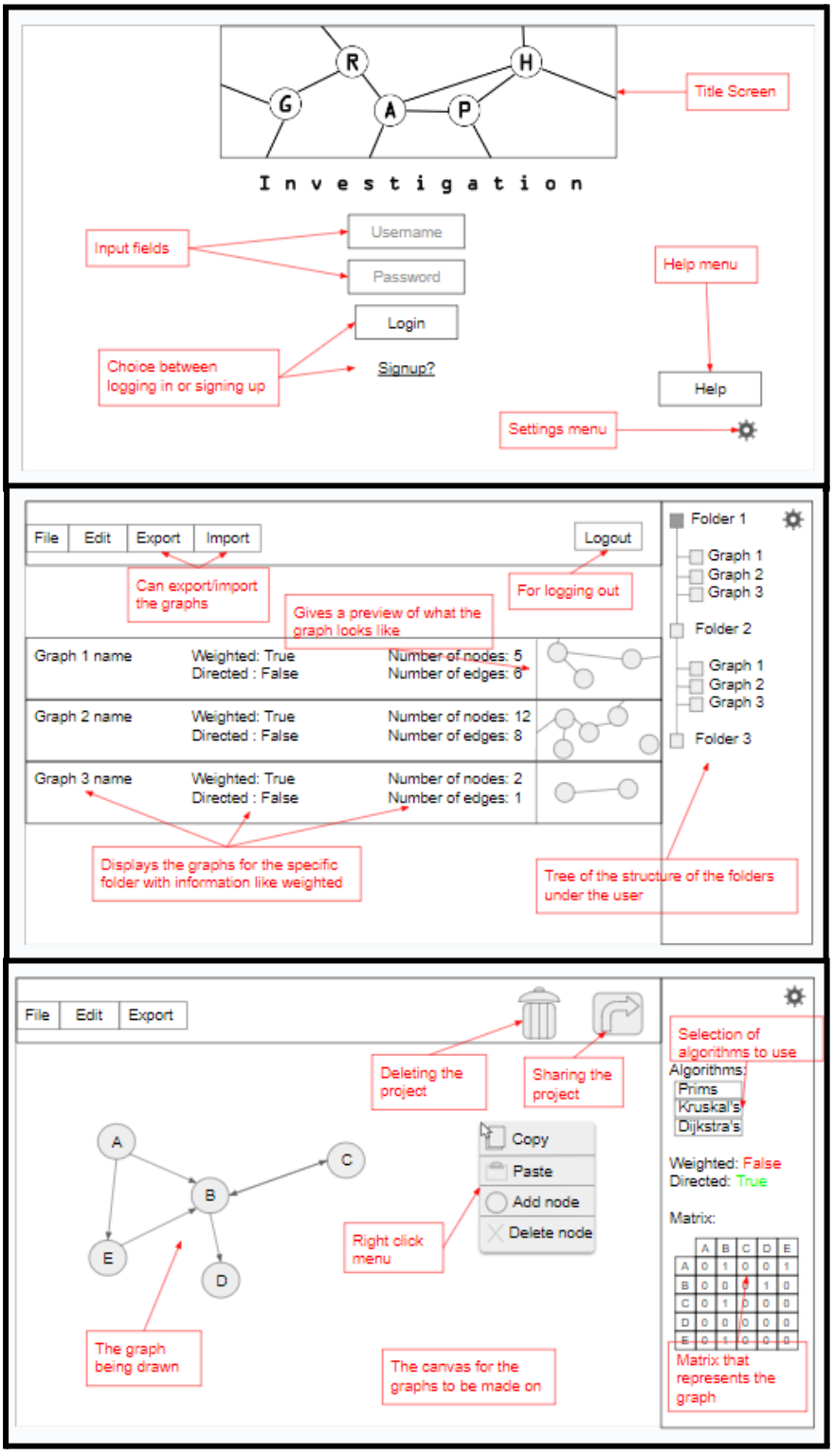
My aim is to make it possible for the user to save their projects in a database where they log in and log out. This will mean that users can save their progress if they ever need to create large graphs. This means tha tif a large project is needed then it can be done in separate times and not just at once.

A graph needs information about nodes and edges which are typically stored in a matrix. However, the database needs to be kept atomic which means that I will have to first pickle the matrix so that it can be stored as a string and then store the string in the database so that it can remain at 3rd normal form. When it is retrieved it will then be unpickled back into the original matrix

Additionally, I will also be creating an import/export system for easy file sharing. This will mean that other users will be able to send their graphs to each other. This will be done by first pickling the graph and then storing it in a json file and sending it off to another user to then import and then unpickle.

HCI

The pictures below show how the user will interact with my program. It will be the GUI of the program which shows each page of the program and how they work. This will be controlled by a variable and will also be showing how each part connected with each other.



Libraries

The main library I will be using is Pygame when creating my program. This is because it makes it easy to create a user interface and also has its own sprite system. This means that the dimensions of a sprite can be automatically calculated. You can also make groups of sprites which will be useful when you have multiple nodes and edges and you group them to make the graph. It also supports object oriented and event driven programming paradigms. This makes it more efficient.

The 2nd library I will be using is the pickle module. This module allows you to turn any type of a data type into an encoded string. This is usually used for sockets and wireless transmission, however, it is useful in this case to keep my database normalised so that I can store the entire matrix in a database. The string can then be unpickled to retrieve the coded information.

Security and integrity of data

I will have to validate all the inputs so that no data will be able to crash the system. This will not be as much since I will be making the program as interactive as possible. I will also need to sanitise the data entering the database since i will be using SQL, I will need to prevent anyone from using SQL injections to tamper with other users data in the system.

Furthermore, I will be using hashing algorithms to store passwords in the database to make it more secure so that even if the database is compromised, the passwords won't be able to be retrieved.

Last of all, when my program is finished I will compile the .py file into a .exe file so that the source code is not tampered with. This also means that python will not be required for a user to be able to access the program and it will be standalone.

Test strategy

I aim to test out my program by myself while also getting feedback from other students in further maths or computer science. They will be able to help me find any flaws in my program as well as tell me how they feel about the UI.
I will do formative testing during the development of the program. I will perform unit testing every time a significant feature is added. They will be recorded with every version in the revisions section. When completed I will apply summative testing against the objectives. I will also use black box testing for some users so that there are no explicit bugs that can be stumbled upon by the average user and I will also do white box testing to make sure that there are no exploits that can be found in my program if the source code gets leaked. The results of my testing are to be managed in the following table.

No.	Purpose of test	Data type	Expected outcome	Actual outcome	Explanation	Image

Revisions

I will initially develop programs which each carry out the 3 main algorithms that I will be constructing. They will just be displayed. Then I will create the UI and menu system and a database so that the user can log in and create their own account. Then I will make a menu system where the user will be able to add folders and graphs to their user. The next version will be implementing the creation of graphs (i.e. adding nodes and edges to the graphs). Then after that I will implement the algorithms to the program.

Version 1

This version consists of 3 independent files, each which carry out the algorithms that I want to implement in my program. In this version, the user will have predetermined nodes and predetermined weights and I want it so that the algorithm will be able to function and for it to be able to be displayed.

Program Code: Display and General Algorithms

These lines are just setting up the program. The first line just imports the modules I will be using.

The class below creates node objects with the following attributes.

Then I have predefined 8 nodes for the user to use.

The nodes list incorporates all the 8 nodes into 1 list. This means when I iterate through all the neighbours attributes for the nodes I can create an adjacency list.

```
#Imports
import pygame,time

#Class
class node:
    #Constructor
    def __init__(self,name,pos,x,y,neighbours):
        self.name = name
        self.pos = pos
        self.x = x
        self.y = y
        self.neighbours = neighbours


#Predefining the nodes
A = node("A",1,50,200,[["B",1],["C",2]])
B = node("B",2,150,100,[["A",1],["E",3],["H",8]])
C = node("C",3,150,300,[["A",2],["F",7],["D",6]])
D = node("D",4,250,300,[["C",6],["F",4]])
E = node("E",5,250,100,[["B",3],["F",4],["G",6]])
F = node("F",6,350,100,[["E",4],["C",7],["D",4],["G",5]])
G = node("G",7,350,300,[["E",6],["F",5]])
H = node("H",8,450,200,[["B",8]])

#Make them into a list
nodes = [A,B,C,D,E,F,G,H]
```

The first variable is important as it is the matrix form of the nodes. The rest of the variables are specific to pygame as they are what initiates the window/program

```
#Creates the matrix
matrix = list_to_graph(nodes)

#Initiates the pygame window
pygame.init()
#Designs the windows dimensions
win = pygame.display.set_mode((640,640))
#Sets the clock (for the fps)
clock = pygame.time.Clock()
#Sets the font for writing text on the screen
font = pygame.font.SysFont("Arial",32)
#This controls the gameloop
run = True
```

The algorithm below finds the highest weighted edge in the matrix and returns it. It does this using linear search by iterating through every edge and checking to see if it is less than or greater than the current highest found. When cycling through all the values, the highest weighted node is returned at the end. This will be used in MST and Pathfinding algorithms since usually, the algorithm has to compare the current weight with the highest one

```
#This function finds the highest value in the matrix
def highest_algorithm(matrix):
    #The default highest value will always be 0
    highest = 0
    #It will then cycle through each value in the matrix that was passed and check if the value of the edge is higher than the highest
    for i in matrix:
        for j in i:
            #The need for the try function is that a lack of an edge between 2 nodes is represented as a '-' which is a string
            #This means u cannot do mathematical operations to it and therefore if it is encountered it will just be ignored
            try:
                if j > highest:
                    #If a new highest is found then it will set that value as the new highest
                    highest = j
            except:pass

    #just returns the value
    return highest
```

This algorithm is provided with a matrix and a list of old nodes and returns a new list of nodes which corresponds to the edges given in the matrix. Since displaying the nodes using an adjacency list is much easier than an adjacency matrix, it means that even if the algorithms require the matrix to function, it is possible to just convert them into a list. The way this algorithm works is by cycling through all the nodes twice and checking to see if there is a connection between them. If there is then it will add it to a temporary neighbours list. When the algorithm has finished with the first node, it will add the details of the node (such as name and position) to a new list of new nodes. This cycle repeats for all nodes and then the list is then outputted.

```
#This function gets matrix and a list of nodes and sets the edges in the matrix to the edges in the list
def graph_to_list(matrix,old_nodes):
    #Sets a new empty list to blank
    new_nodes = []
    #Cycles through all the columns and indecies in the given matrix
    for a,i in enumerate(matrix):
        #Makes a new empty list of the neighbours, these are the neighbours of the nodes
        neighbours = []
        #Cycles through all the rows and indecies in the given matrix
        for b,j in enumerate(i):
            #Makes sure that there is actually a connection between the 2 nodes
            if j != "-":
                #Adds to the list of neighbours another list containing the name of the node and its weight
                neighbours.append([old_nodes[b].name,j])
        #Adds the nodes to the new nodes and give it the neighbours
        new_nodes.append(node(old_nodes[a].name,old_nodes[a].pos,old_nodes[a].x,old_nodes[a].y,neighbours))

    return new_nodes
```

This other algorithm is the opposite to the one above since it returns a matrix given a list of nodes. This works by finding the index of 2 nodes (e.g. a and b) and sets the value of their connection in the matrix to the weight of the edge connecting the. (e.g. matrix[a][b] = 5). When cycling through all the combinations of 2 nodes then it will complete the matrix and return it

```
#Does the opposite of the previous one where it outputs a matrix when given a list
def list_to_graph(nodes):
    #Initiates the matrix
    graph = []
    #Cycles through all the columns in the list
    for i in nodes:
        #Creates a temporary column
        temp_list = []
        #Cycles through all the rows in the list
        for j in nodes:
            #Initally sets no connected between the 2 nodes
            weight_of_graph = "-"
            #Cycles through the neighbours of that node
            for k in i.neighbours:
                #If it finds that if they are neighbours it changes the weight to its weight
                if j.name == k[0]:

                    weight_of_graph = k[1]

            #Appends the weight of the edge to the column whether its '-' or an actual connection
            temp_list.append(weight_of_graph)
        #Finally after the column has had all its values inputted it adds it to the matrix
        graph.append(temp_list)

    return graph
```

Program Code: Prims

This is one of the main algorithms of this version since it is the overall main focus of the program. This function computes the minimum spanning tree. It does this by looping through all the indices of the nodes until all the nodes have been connected. In each iteration It checks if each connection with the nodes currently connected to see which is the smallest. If a cycle is found then the edge should be discarded. The way I can do that in this case is for the node to only be appended if the node has not been selected before. Once all the nodes have been iterated over then it will add the indices to an adjacency matrix. Then using the function graph_to_list above, it turns the matrix into an adjacency list and returns it.

```
#Prims algorithm
def prims_algorithm(highest,matrix,starting_node,nodes):
    #sets the number of nodes to the length of the matrix
    length = len(matrix)
    #creates a 'fresh' matrix
    fresh = []
    #No current nodes are selected
    selected_nodes = []

    #This for loop just makes the fresh matrix have the value '-' for all the edges
    for i in range(0,length):
        temp_list = []
        for j in range(0,length):
            temp_list.append("-")
        fresh.append(temp_list)
        #This part means that all current nodes will be unselected at the start
        selected_nodes.append(False)

    #This variable is responsible for keeping track of which edge its currently on
    edge_count = 0
    #This adds the inital node to the selected nodes
    selected_nodes[starting_node.pos-1] = True
    #Makes sure every node is cycled through
    while (edge_count < length - 1):

        #Sets the inital minimum weigh of the edge to 1 more than the highest
        minimum = highest+1
        #Index 1
        a = 0
        #Index 2
        b = 0
        #Cycles through all indecies of the matrix
        for i in range(length):
            #checks to see if the current node being targetted is one of the selected nodes
            if selected_nodes[i]:
                #Iterates though all the nodes again
                for j in range(length):
                    #Makes sure that the node being examined is not the selected node or a node already examined
                    if ((not selected_nodes[j]) and matrix[i][j]):
                        #Uses try once again because you cant use < and > when dealing with strings
                        try:
                            #Checks to see if the edge weight is the lowest possible
                            if minimum > matrix[i][j]:
                                #Sets the new lowest
                                minimum = matrix[i][j]
                                a = i
                                b = j
                        except:pass
        #Adds the weight of that edge to the fresh matrix
        fresh[a][b] = matrix[a][b]
        #adds the new node to the seleted nodes
        selected_nodes[b] = True
        #Increases the edge count so it can iterate over the next node
        edge_count += 1
    #makes a list out of the matrix when finished
    MST_list = graph_to_list(fresh,nodes)
    return MST_list
```

Program Code: Kruskals

The first function is relatively small which given a list of parent nodes and a node i, it will backtrack until it finds the root node and at that point it will return it.

```
#This small functions finds parent of the node
def find(i,parent):
    while parent[i] != i:
        i = parent[i]
    return i
```

This algorithm below is Kruskal's algorithm. The way this works is by looping until all the nodes have been cycled over. While cycling through the nodes. It finds the shortest edge and connects those 2 nodes. However if there is a cycle then it doesn't connect the nodes. The way this algorithm detects cycles is by creating lists and appending new nodes to the lists. If a node is added more than once then it means a cycle has been created and that edge will be removed. When an edge is added then it adds it to a matrix it uses the index of each node to know where to place the edge in the matrix. When the whole program finishes then it outputs the list

```
def kruskals_algorithm(highest,matrix,nodes):
    #Creates an array with the numbers 1 to the number of nodes
    parent = []
    for i in range(0,len(matrix)):
        parent.append(i)

    #Once again creates a fresh matrix
    fresh = []
    for i in range(0,len(matrix)):
        temp = []
        for j in range(0,len(matrix)):
            temp.append("-")
        fresh.append(temp)

    #Sets each varible to what its parent is
    for i in range(len(matrix)):
        parent[i] = i

    #The edge count once again
    edge_count = 0
    #Loops untill all edges have been analysed
    while edge_count < len(matrix):
        #minimum value is set to 1 higher than the highest
        minimum = highest+1
        #Index 1
        a = -1
        #Index 2
        b = -1
        #Cycles through the indecies of the nodes
        for i in range(len(matrix)):
            for j in range(len(matrix)):
                #Makes sure that there is a connection
                if matrix[i][j] != '-':
                    #Finds the parents of both nodes and if they are not the same and the edge weight is smaller than the minimum
                    if find(i,parent) != find(j,parent) and matrix[i][j] < minimum:
                        #Sets the minimum value to the current matrix position being examiend
                        minimum = matrix[i][j]
                        a = i
                        b = j
        #Finds the parent of the 2nd node and sets the to that parent of the parent of the 1st
        parent[find(a,parent)] = find(b,parent)
        #Adds the edge to the new 'fresh' matrix
        fresh[a][b] = matrix[a][b]
        #increase count
        edge_count += 1
    #turns the matrix into a list
    for a,i in enumerate(fresh):
        for b,j in enumerate(fresh):
            if fresh[a][b] != '-':
                fresh[b][a] = fresh[a][b]
            if fresh[b][a] != '-':
                fresh[a][b] = fresh[b][a]
    kruskal_nodes = graph_to_list(fresh,nodes)
    kruskal_nodes = sorted(kruskal_nodes, key=lambda x: x.neighbours[0][1])
    return kruskal_nodes
```

Program Code: Dijkstra's

This algorithm works by setting the distance to each node to infinity. Then it goes to each node connected to the starting node and adds the minimum distance and the node index to a list. This process repeats and everytime a minimum distance is found for a node it updates the list. To calculate the path. It uses the list to find the shortest path between the start node and the end node which are supplied as parameters. Then it returns the list.

```
def dijkstras_algorithm(matrix, start, end):
    #Sets the distance to each node as infinity at the start
    dist = []
    for i in range(0,len(matrix)):
        dist.append(float('inf'))
    #sets the start node distance to 0
    dist[start] = 0
    #Sets the vitied nodes all to False
    visited = []
    for i in range(0,len(matrix)):
        visited.append(False)
    #Sets the parents of all the nodes to negative meaning they have none
    parents = []
    for i in range(0,len(matrix)):
        parents.append(-1)

    #Cycles through all the nodes
    for i in range(len(matrix)):
        #sets the minimum disance and the minimum index
        min_dist = float('inf')
        min_index = -1
        #Checks each node and finds the minimum distance and smallest index it can be
        for j in range(len(dist)):
            if dist[j] < min_dist and not visited[j]:
                min_dist = dist[i]
                min_index = j

        #Shows the node has been vitited
        visited[min_index] = True
        for j in range(len(matrix)):
            try:
                #Checks to see if the node is not visitived and that the length of the node works
                #And that that distance to the node is not infinity and that it is the shortest distance possible
                if (not visited[j] and matrix[min_index][j] != 0 and dist[min_index] != float('inf') and dist[min_index] + matrix[min_index][j] < dist[j]):
                    #Adds the node to the shortest distance
                    dist[j] = dist[min_index] + matrix[min_index][j]
                    parents[j] = min_index
            except:pass
    #Creates an empty array for the path
    path = []
    #sets the current node to the end and backtracks
    current = end
    while current != start:
        #Back trash
        path.append(current)
        current = parents[current]
    path.append(start)

    shortest_path = list(reversed(path))
    return shortest_path
```

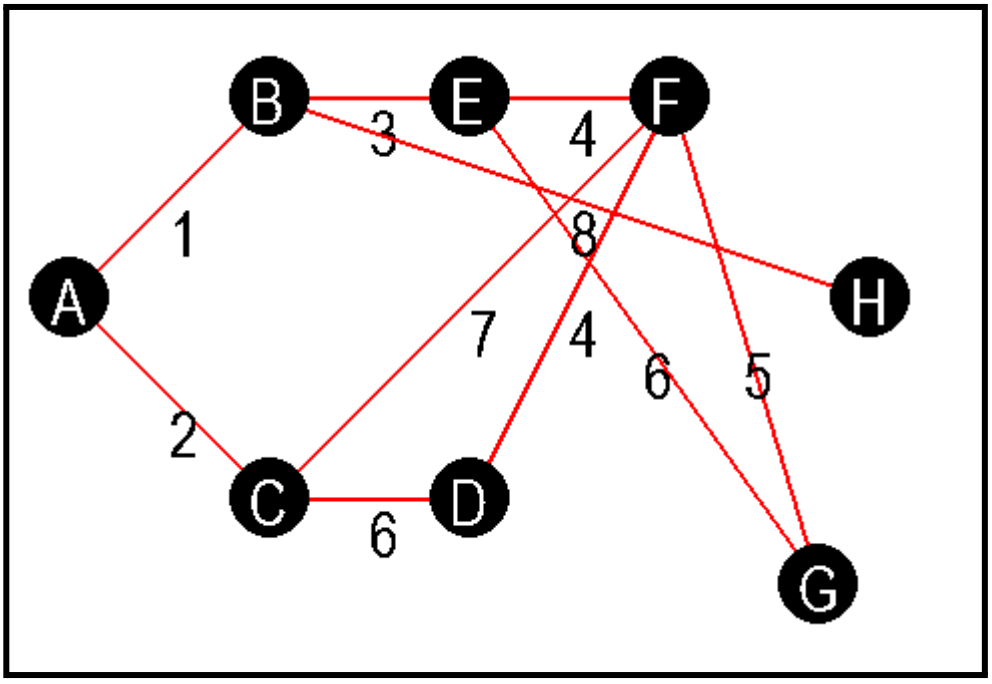
Program Code: Mainloop

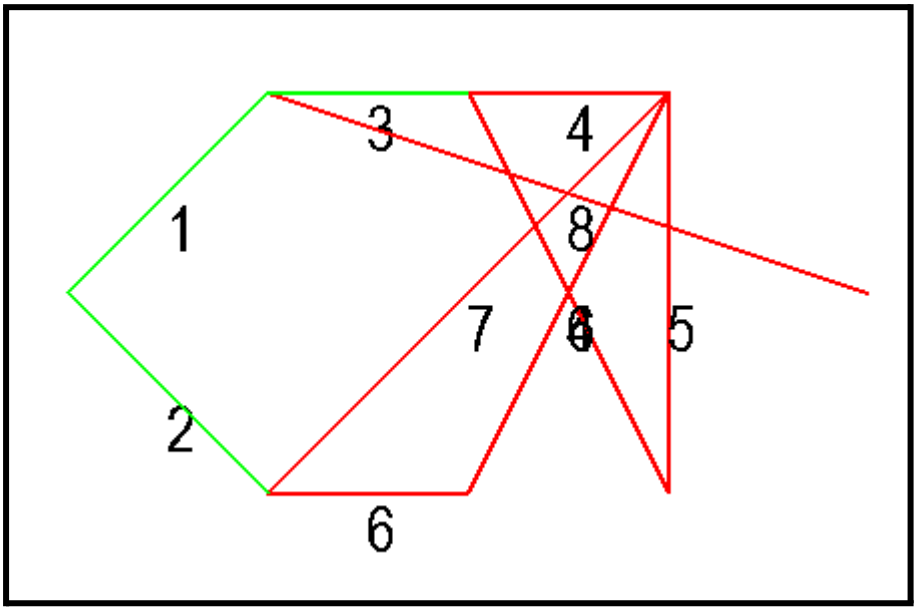
The algorithm below is the game loop of the program. It starts by drawing all the relevant things on the screen (i.e. the nodes and edges and weights). In the events loop, it also checks to see if the user is holding down on any of the buttons. If the user then drags their mouse then they move the position of the node and it changes in real time. This is detected in the events loop which cycles through all the events happening currently and detects any changes. To prevent picking up more than 1 node when the user holds down on an area it puts all the nodes selected into a list and checks which one of them has the lowest position. That will be the one that is actually moved around. In addition to moving around the nodes, this program also detects the press of the space bar which allows for the toggle the algorithm selected to be displayed or not.

```
display_MST = False
hold = False
#gameloop
while run:
    #resets the screen to white every time
    win.fill((255,255,255))
    #Sets the apps fps to 60 so it cycles 60 times every second at a maximum
    clock.tick(60)
    #retrieves the mouses coordinates every loop
    mousex,mousey = pygame.mouse.get_pos()

    #Cycles nodes to draw them on the screen
    for i in nodes:
        for j in i.neighbours:
            for k in nodes:
                if j[0] == k.name:
                    pygame.draw.line(win,(255,0,0),(i.x,i.y),(k.x,k.y),2)
                    win.blit(font.render(str(j[1]),0,(0,0,0)),((i.x+k.x)//2,(i.y+k.y)//2))
    #Checks every event happeneing on the screen
    for event in pygame.event.get():
        #If the user clicks the X then it closes the window
        if event.type == pygame.QUIT or (event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE):
            run = False
        #If the usr clicks the spacebar then it will run prims algorithm
        if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
            highest_number = highest_algorithm(matrix)
            MST_matrix = prims_algorithm(highest_number,matrix,A,nodes)
            MST_list = graph_to_list(MST_matrix,nodes)
            #This allows the user to toggle whether they want to see the MST or not
            if display_MST:
                display_MST = False
            else:
                display_MST = True
        #If the user holds down the moursebutton then they will be able to drag the nodes around
        if event.type == pygame.MOUSEBUTTONDOWN:
            hold = True
            selected_nodes = []
            #Checks each node that it is hovering over and adds them to a list
            for a,i in enumerate(nodes):
                if i.x-20<mousex<i.x+20 and i.y-20<mousey<i.y+20:
                    selected_nodes.append(a)

            # Finds the node with the lowest pos and sets that to the node that is being moved
            try:
                current_lowest = nodes[selected_nodes[0]].pos
                for i in selected_nodes:
                    if nodes[i].pos <= current_lowest:
                        moving_node = i

            except:
                hold = False


        #If the user lets go of their mouse then the program will acknowledge they are no longer holding
        if event.type == pygame.MOUSEBUTTONUP:
            hold = False
        #If the user is holding down the button then the node selected will have its x and y coordinates changed to that of the users mouse
        if hold:
            nodes[moving_node].x = mousex
            nodes[moving_node].y = mousey

    #Checks to see whether the user has toggled on or toggled off the MST and acts accordingly
    if display_MST:
        for i in MST_list:
            for j in i.neighbours:
                for k in MST_list:
                    if j[0] == k.name:
                        pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)

    #Draws the nodes onto the screen
    for i in nodes:
        pygame.draw.circle(win,(0,0,0),(i.x,i.y),20)
        win.blit(font.render(i.name,0,(255,255,255)),(i.x-10,i.y-16))

    #Updates any changes made to the screen
    pygame.display.update()
```

Screen Design: Prims

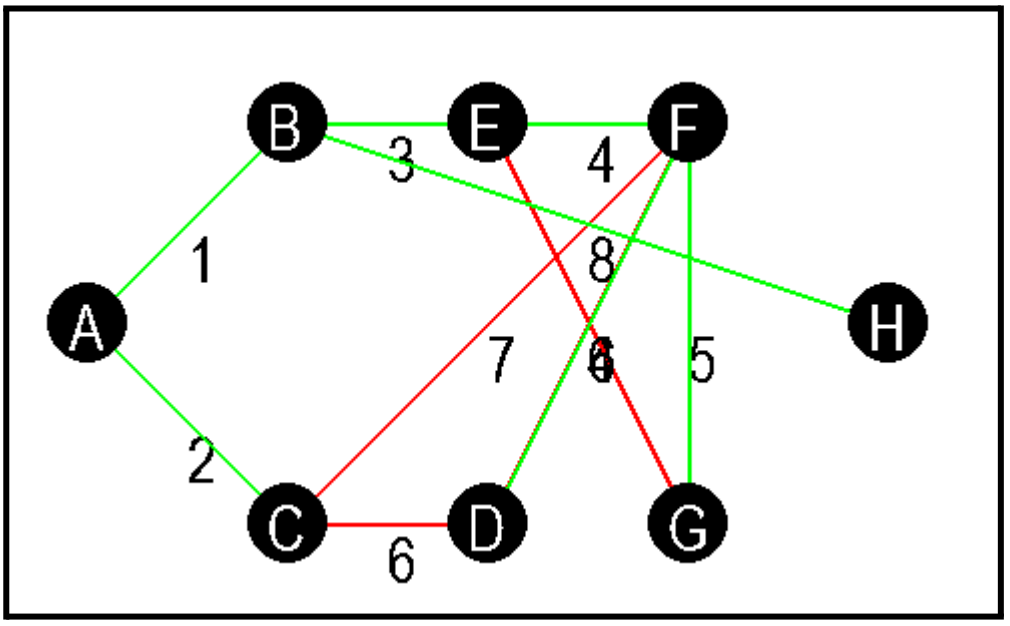
The graph being displayed onto the screen with numbers representing the weights of the edges. It is possible to move around the nodes so that it is easier to use the algorithms





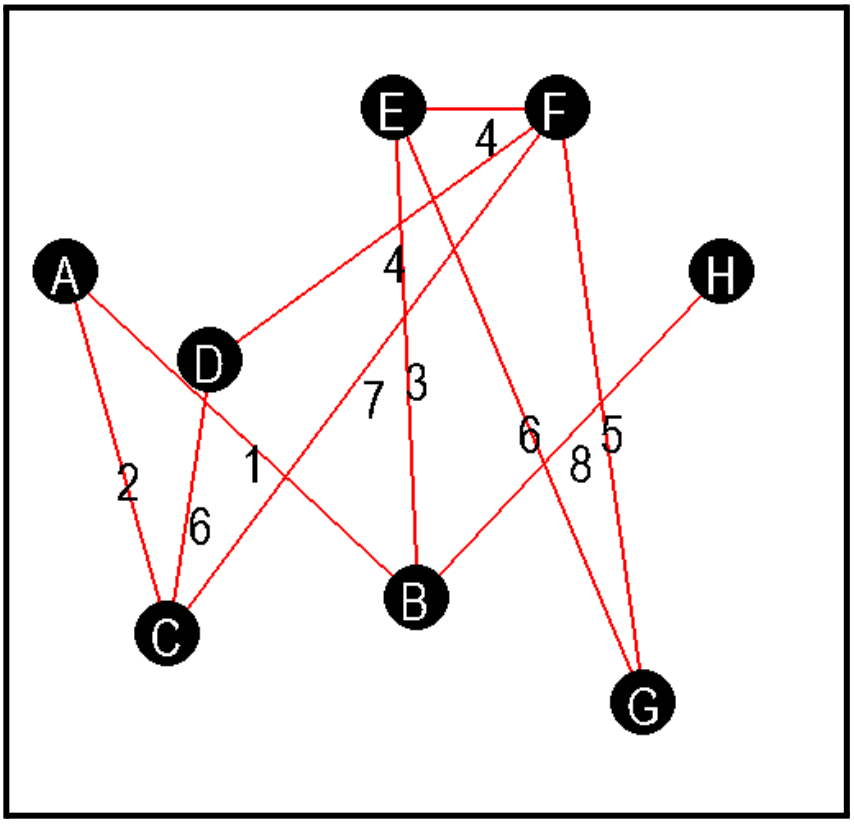
The algorithm going step by step to show the user how prim's algorithm actually works and how it connects the smallest weighted edge onto the current tree except that it removes the nodes when it is projecting the MST

The final completed MST for the graph using prim's algorithm is shown by having highlighted edges

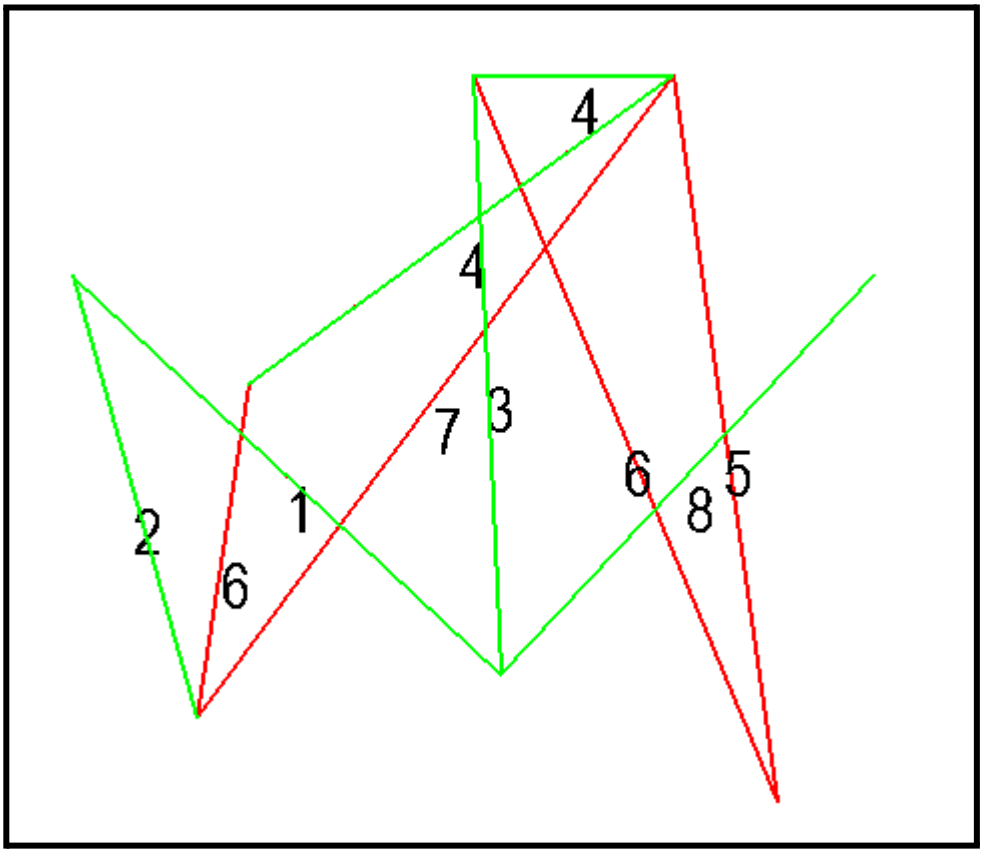


Screen Design: Kruskals

The graph displaying Kruskal's algorithm. This also shows how the nodes can be moved around since the graph is identical to the previous one except a different algorithm is used to get to the same result

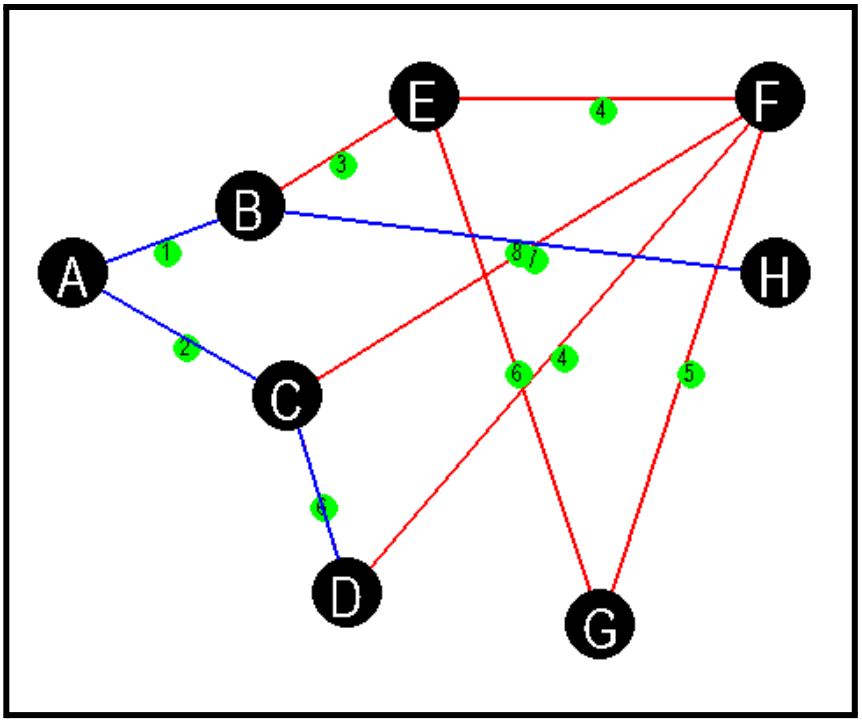


This shows how Kruskal's algorithm works by finding the shortest edges and connecting them together. Although the result of prims and kruskal's are the same, the methods are different so the user is able to see it properly and understand how they work



System Design: Dijkstras

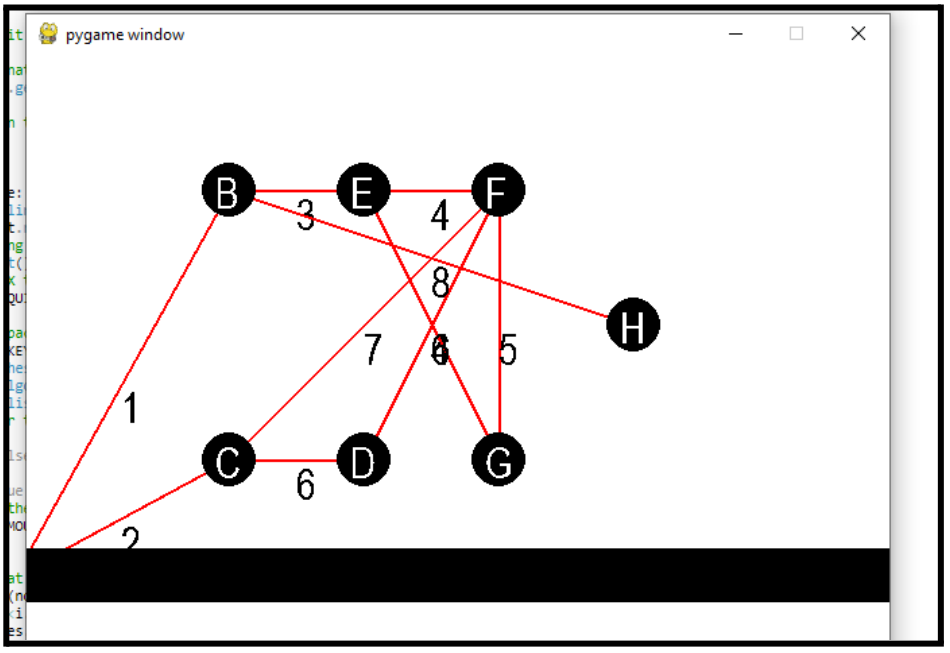
This is the 1 algorithm that does not currently have an animation to show the exact steps of how it came to dijkstra's because it is much more complex. It will be a problem I will tackle when I have completely finished the programming of the main file. However this is still the correct shortest path from node D to H so the actual algorithm still works as intended and displays the path correctly



Testing

Overall, the programs do the absolute basics which is to perform the algorithm on the graph. There is a bit of exception handling where the algorithms like to try to use mathematical operations on strings since a lack of a node is represented as a '-' in the matrix and it will raise an error is that happens so instead i added the try function so if it encounters that error it just skips over it since it doesn't matter about that edge because it doesn't exist.

Also, if a node is taken off the screen it creates a visual glitch. In addition, it becomes unrecoverable because the mouse cannot get it when it goes off the screen which means it will be permanent. To counter this I will have to create boundary conditions for when the user is dragging on a node.



Conclusion

All the algorithms work and function. There are a few bugs which need to be cleared up and a few errors that occur if the user does something wrong. Apart from that I need to combine it so that all the algorithms can be chosen at once.

Version 2

Program Code: Database

This sets up the database for it to be manipulated. It creates 3 tables, 1 for the account, 1 for folders and the last one for actually storing the graphs. It first tries to create the tables however if they already exist, for it not to overwrite the data it just ignores the request and carries on with the program

```
#Connection to the database
conn = sqlite3.connect('Resources/SaveData/SaveData.db')
#Creates a cursor for selecting items in a database
curs = conn.cursor()
#Will attempt to create a database if there isnt one already, otherwise nothing will happen
try:
    curs.execute("""CREATE TABLE Account_info (
            Username text,
            Password text
            )""")


    curs.execute("""CREATE TABLE Folder_info (
            Folder_id int,
            Name text,
            Account_id int
            )""")

    curs.execute("""CREATE TABLE Graph_info (
            Name text,
            Weighted bool,
            Directed bool,
            Graph text,
            List text,
            Account_id int,
            Folder_id int,
            Graph_id int
            )""")

except:pass

#Save the changes
conn.commit()
#Closes the connection
conn.close()
```

Program Code: Login Screen

Sets the fonts that will be required for the program, also adds the images that will be used for the login screen. Initialises the page to automatically ask the user to login.

```
#The standard font that will be used
font = pygame.font.SysFont("Cascadia Code",32)
#A smaller font
small_font = pygame.font.SysFont("Cascadia Code",12)
#Password font
pass_font = pygame.font.Font("Resources/Media/Pass_font.ttf",32)

#images
header = pygame.image.load("Resources/Media/header.png")
subheader = pygame.image.load("Resources/Media/subheader.png")
```

```
#This variable will control what the user is currently seeing
page = "login"
#Check key press variable
keys=pygame.key.get_pressed()
```

This procedure is what is used when the user tries to type something. THis will detect it and the variable current string to the current string + whatever new thing has been added. It detects key presses so it's easier for me to pull information from what the user types later on.

```
#Function for typing into the program
def type(key,page):
    global current_string,username,password
    #if the key is backspace then it removes the last character from the string
    if key.key == pygame.K_BACKSPACE:
        current_string = current_string[:-1]
    #If the key is tab it moves it from username -> password -> free in that order
    elif key.key == pygame.K_TAB:
        if page == "login" or page == "signup":
            if username:
                username = False
                password = True
                current_string = pass_string
            elif password:
                username = False
                password = False
            elif not username and not password:
                username = True
                password = False
                current_string = user_string
    #Otherwise it adds it to the curent string
    else:
        current_string += event.unicode
```

```
#Hashing algorithm for storing passwords
def hashs(plaintext):

    hashed = plaintext

    return hashed
```

This is meant to be the hashing algorithm however I have yet to implement it. I just need to find an algorithm to have it for it to be secure.

This is the login function. It checks the current string for the username with all the values in the database to make sure the username is not unique.. It then also checks to make sure that if u has the password that the user entered it is the same as the entry in the database. It then returns whether it's found a matching account with the credentials given or not.

```
#If the user chooses to log in
def login(user,passs):
    global current_account_id
    current_account_id = 0
    found = False
    #Connection to the database
    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
    #Creates a cursor for selecting items in a database
    curs = conn.cursor()
    #Fetches all the accounts
    curs.execute("SELECT * FROM Account_info")
    #Checks to see if the username and password are correct
    for a,i in enumerate(curs.fetchall()):
        if user == i[0] and hashs(passs) == i[1]:
            found = True
            current_account_id = a+1

    #Save the changes
    conn.commit()
    #Closes the connection
    conn.close()
    #reutnrs whehters its ture or not
    return found
```

The signup function is similar to the login function except it works in the opposite way. It checks to see if the user is already in the database. If it is then it returns that there is a duplicate username. However if it is unique then it will create a new entry in the database with the credentials given and return that there was no duplicate found.

```
#Signup function
def signup(user,passs):
    global current_account_id
    #No current dupes found
    dupe = False


    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
    curs = conn.cursor()

    curs.execute("SELECT * FROM Account_info")
    #Gets all the accounts from the database
    data = (curs.fetchall())

    #Checks to see if there is a dupe
    for i in data:
        if user == i[0]:
            dupe = True

    #If there is not a dupe username then it will input a new username and password into the table
    if not dupe:
        curs.execute("INSERT INTO Account_info VALUES (:Username,:Password)",
            {

            'Username': user,
            'Password': hashs(passs)

            })
        curs.execute("INSERT INTO Folder_info VALUES (:Folder_id,:Name,:Account_id)",
        {
        'Folder_id':0,
        'Name':"Root",
        'Account_id':len(data)+1

        })
        current_account_id = len(data)+1


    conn.commit()
    conn.close()
    #returns whether it found a dupe or not
    return dupe
```

This is the start of the game loop. It runs at 60fps and it colours the background in the theme currently chosen (there is only 1 at the moment but will be updated soon). It also gets the mouse coordinates. If the screen is login or signup then it projects all the images that are required for a correct display.

```
#Gameloop
while run:
    #The clock which will dictacte the ticks
    clock.tick(60)
    #This will fill the screen with this colour and since it is at the start of the loop this indicates that it is the background
    win.fill(colour1[theme])
    #mouse x and y coords
    mousex,mousey = pygame.mouse.get_pos()

    #This will assess which page the user is currently on
    if page == "login" or page =="signup":
        #Design
        pygame.draw.rect(win,(255,255,255),(400,40,480,160))
        #This inserts the image of the header
        win.blit(header,(400,40))
        pygame.draw.rect(win,(0,0,0),(400,40,480,160),2)
        #Subheader
        win.blit(subheader,(440,180))
        #This is the usrename/password box and text setup
        win.blit(font.render("Username:",0,(255,255,255)),(480,340))
        pygame.draw.rect(win,colour2[theme],(480,380,320,50))
        win.blit(font.render("Password:",0,(255,255,255)),(480,430))
        pygame.draw.rect(win,colour2[theme],(480,470,320,50))
        #If the user wants to login it will display the login button
        if page == "login":
            pygame.draw.rect(win,colour2[theme],(570,535,130,50))
            win.blit(font.render("Login",0,(0,0,0)),(590,540))


            win.blit(font.render("Signup?",0,(0,0,0)),(575,620))
            pygame.draw.line(win,(0,0,0),(575,660),(705,660),3)
        #If the user wants to signup it shows the signup button
        elif page == "signup":
            pygame.draw.rect(win,colour2[theme],(570,535,130,50))
            win.blit(font.render("Signup",0,(0,0,0)),(575,540))


            win.blit(font.render("Login?",0,(0,0,0)),(590,620))
            pygame.draw.line(win,(0,0,0),(575,660),(705,660),3)
```

This checks to see if the user has clicked. If they have clicked on the username box then they will be able to edit the username and if they have clicked on the password box then they will be able to edit the password. If they click somewhere else then it deselects. If they click signup? Then it switches to sign up otherwise if they click login? It switches to login. Instead if they click the login/signup button it does a validity check and then logins the user in. and switches the page to the menu

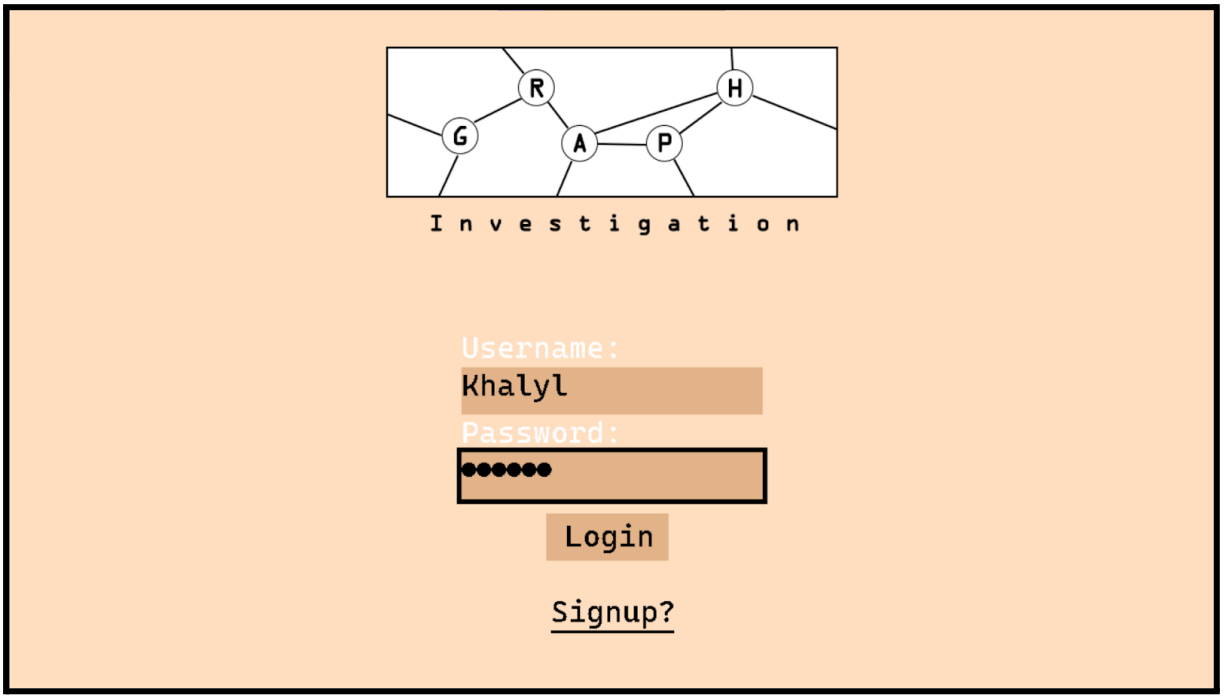
```
#If the user left clicks
if pygame.mouse.get_pressed()[0]:
    #If it is bounded withing the username box
    if mousex >= 480 and mousex <= 800 and mousey >= 380 and mousey <= 430:
        #Allows the user to enter the username
        username = True
        password = False
        current_string = user_string
    #If tit is mbonuded within the password box
    elif mousex >= 480 and mousex <= 800 and mousey >= 470 and mousey <= 520:
        #Allows the user to enter the password
        username = False
        password = True
        current_string = pass_string
    elif mousex >= 575 and mousex <= 705 and mousey >= 620 and mousey <= 660:
        #If the login/signup? button is clicked
        time.sleep(0.2)
        if page == "login":
            page = "signup"
        elif page == "signup":
            page = "login"

    elif mousex >= 570 and mousex <= 700 and mousey >= 535 and mousey <= 585:
        #If the login button is clicked
        if page == "login":
            #checks to see if the credentials are correct
            val = login(user_string,pass_string)
            #If not it will tell the user that the username or apss is wrong
            if not val:
                win.blit(font.render("User or pass incorrect",0,(255,0,0)),(480,300))
            #Otherwise it will send the user to the menu screen
            else:
                page = "menu"
                current_string = ""
        #If the signup button is clicked
        elif page == "signup":
            #It checks to see if a user with that username alr exists
            val = signup(user_string,pass_string)
            if val:
                win.blit(font.render("User alr exists",0,(255,0,0)),(480,300))
            else:
                page = "menu"
                current_string = ""


    else:
        #If the user clicks anywhere else then the username and password boxes are deseected
        username = False
        password = False
```

Checks to see if the user has clicked any buttons or keys on the keypad. It then updates any changes made to the screen

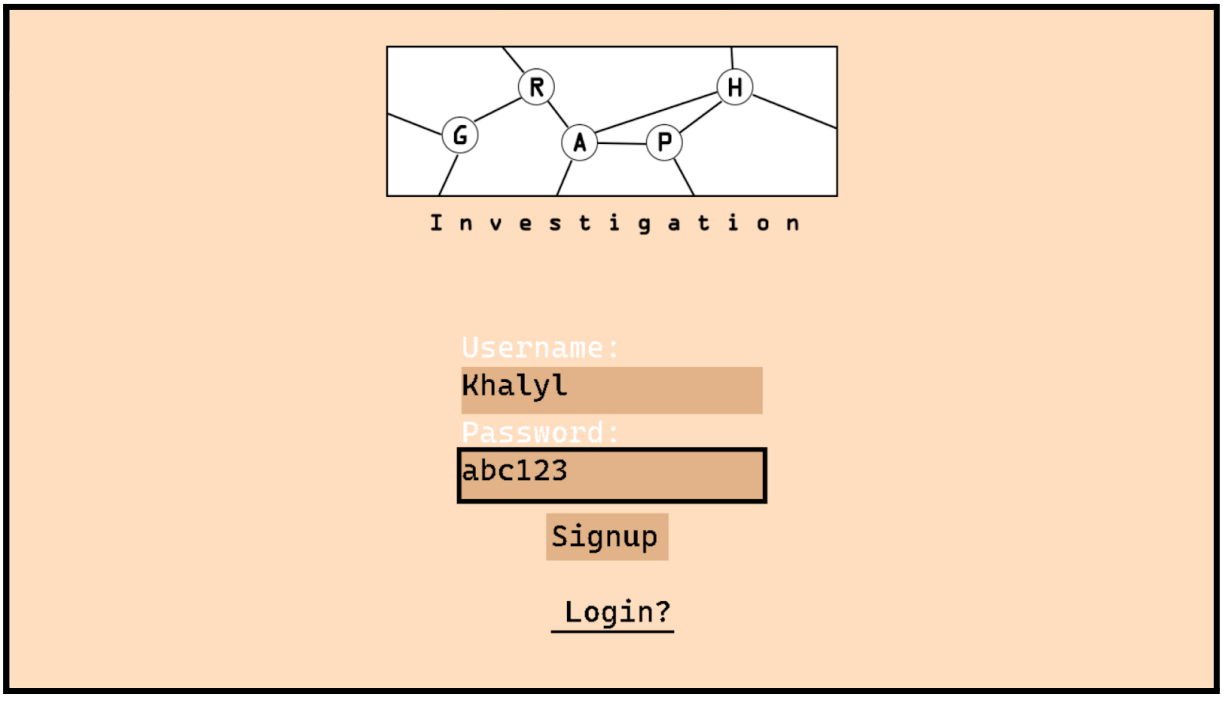
```
#This checks every event currently happening in the window
for event in pygame.event.get():
    #If the event is the big red X in the top corner being pressed or the ESCAPE key being pressed then it will close the program
    if event.type == pygame.QUIT or (event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE):
        #This just sets the variable which dictates the gameloop to false which means it will no longer run
        run = False
        #Checks to see if the user has typed anything
    if event.type == pygame.KEYDOWN:
        type(event,page)

#This updates any changes going on w the screen
pygame.display.update()
```

Screen Design

Login screen



Signup screen



Testing

Using this program, it displays everything in all 3 tables of the database. I can use this to see that everything is working correctly. And everything is working fine.

```
import sqlite3


conn = sqlite3.connect('Resources/SaveData/SaveData.db')
curs = conn.cursor()
curs.execute("SELECT * FROM Account_info")
a = curs.fetchall()
print(a)

curs.execute("SELECT * FROM Folder_info")
a = curs.fetchall()
print(a)

curs.execute("SELECT * FROM Graph_info")
a = curs.fetchall()
print(a)

conn.commit()
conn.close()
```

Conclusion

The login and signup system is working perfectly and there are no problems creating a new entry in the database although i do need to find a better hashing algorithm than just plaintext

Version 3

Program Code: Menu Screen

This checks to see that the page is the menu screen. It then will display all the buttons and the design of the program. In addition, all the graphs that are in the current folder selected will be displayed in the central panel. And all the folders in general wil be displayed on the right hand side panel. In addition, If the user clicks on a folder then it changes the current folder selected to that one.

```
elif page == "menu":
    #Shows the buttons on the menu
    pygame.draw.rect(win,colour2[theme],(0,0,950,100))
    pygame.draw.rect(win,(0,0,0),(0,0,950,100),2)

    pygame.draw.rect(win,colour4[theme],(950,0,330,720))
    pygame.draw.rect(win,(0,0,0),(950,0,330,720),2)

    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
    curs = conn.cursor()
    curs.execute(f"SELECT * FROM Folder_info WHERE Account_id={current_account_id}")
    #Selects all the folders associated with the acocunt fom the database
    folder_list = curs.fetchall()
    largest_folder_id = folder_list[len(folder_list)-1][0]
    #Displays all the folders associated witht the account of the panel on the right hand side
    for a,i in enumerate(folder_list):
        pygame.draw.rect(win,(0,0,0),(970,20+a*45,200,40),2)
        win.blit(font.render(f"{i[1]}",0,(255,0,0)),(975,22+a*45))

    conn.commit()
    conn.close()
    #Settings icon again
    win.blit(settings_icon,(870,20))
    #Sends the user to the settings menu
    if pygame.mouse.get_pressed()[0]:
        if mousex >= 870 and mousex <= 940 and mousey >= 20 and mousey <= 90:
            page = "settings_menu"

    #If the user clicks on a speciic folder then the current folder id wil be changed to that which they selected
    if pygame.mouse.get_pressed()[0]:
        for a,i in enumerate(folder_list):
            if mousex >= 970 and mousex <= 1170 and mousey >= 20+a*45 and mousey <= 60+a*45:
                current_folder_id = a


    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
    curs = conn.cursor()
    curs.execute(f"SELECT * FROM Graph_info WHERE Folder_id={current_folder_id} AND Account_id={current_account_id}")
    #retreve all the graphs in the database that are assocaited with the user
    graph_list = curs.fetchall()
    #will set the largest graph id to the length of the previous graph
    try:
        largest_graph_id = graph_list[len(graph_list)-1][len(graph_list[len(graph_list)-1])-1]
        #If it cant then it will set it to -1 (implying that there are no graphs currently made)
    except:
        largest_graph_id = -1
    #Draws all the graphs that are currently int the folder
    for a,i in enumerate(graph_list):
        pygame.draw.rect(win,(0,0,0),(40,120+a*45,200,40),2)
        win.blit(font.render(f"{i[0]}",0,(255,0,0)),(45,122+a*45))

    conn.commit()
    conn.close()
```

This section of code checks to see if any of the graphs are clicked on. If they are then the user will be directed to the canvas page (still not created). In addition a few more buttons are being displayed. These are the new project, new folder and import buttons.

```
#If the user clicks anyone of the graphs that are displayed then it will take them to the canvas where they can create graphs
if pygame.mouse.get_pressed()[0]:
    for a,i in enumerate(graph_list):
        if mousex >= 40 and mousex <= 240 and mousey >= 120+a*45 and mousey <= 160+a*45:
            #sets the current graph id to the one that has been selected
            current_graph_id = a
            #switches pages
            page = "canvas"
            #retreives the matrix and the nodes from the database of the graph that they selected and they are assigned to the matrix and nodes
            matrix = pickle.loads(graph_list[current_graph_id][3])
            nodes = pickle.loads(graph_list[current_graph_id][4])


#Displays the buttons for a new project, new folder and importing other graphs
pygame.draw.rect(win,(0,0,0),(20,20,230,60),2)
win.blit(font.render("New project",0,(0,0,0)),(30,30))

pygame.draw.rect(win,(0,0,0),(270,20,215,60),2)
win.blit(font.render("New folder",0,(0,0,0)),(280,30))

pygame.draw.rect(win,(0,0,0),(505,20,135,60),2)
win.blit(font.render("Import",0,(0,0,0)),(515,30))
```

If the user clicks on the new graph or new folder buttons then it will open a text box that will ask for the name and other options only if a text box is not already open

```
#If the user hasnt decided to create a graph or create a folder
if not text_box_folder and not text_box_graph:
    weighted = False
    current_string = ""
    if pygame.mouse.get_pressed()[0]:
        if mousex >= 20 and mousex <= 250 and mousey >= 20 and mousey <= 80:
            #onclick
            text_box_graph = True
            time.sleep(0.1)

        elif mousex >= 270 and mousex <= 485 and mousey >= 20 and mousey <= 80:
            #onclick
            text_box_folder = True
            time.sleep(0.1)
```

This checks to see if the text box for creating a new has been pressed and if it has then it will display a popup box asking the user to enter a name and to select a slider of whether the new graph is weighted or not. Then the user will be able to click enter. When they do a lot of validity checks will take place to make sure that the correct kind of data is entered and then it will add it to the table.

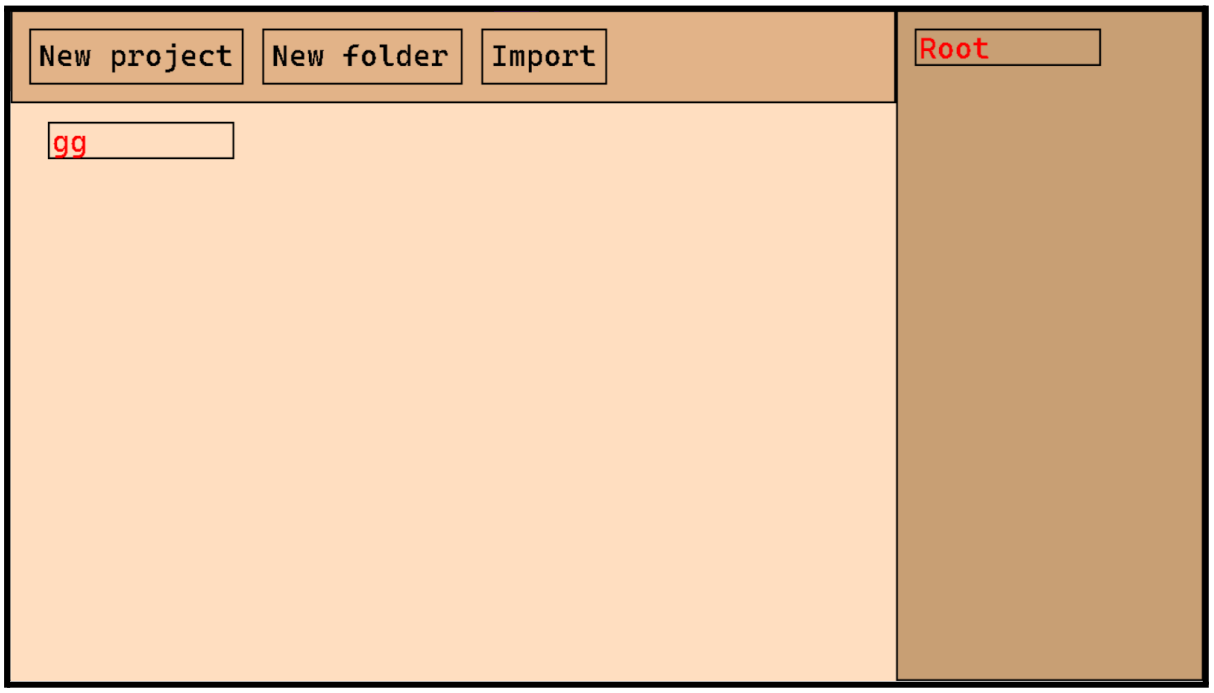
```
elif text_box_graph:
    #displays all the buttons for the text box for creating a graph
    pygame.draw.rect(win,colour4[theme],(320,200,640,320))
    pygame.draw.rect(win,(0,0,0),(320,200,640,320),2)

    #input field for the user to input the name
    win.blit(font.render("Enter graph name:",0,(255,255,255)),(400,220))
    pygame.draw.rect(win,colours[theme],(360,280,560,40))
    pygame.draw.rect(win,(0,0,0),(360,280,560,40),2)

    win.blit(font.render(current_string,0,(0,0,0)),(365,280))
    #displays whether the graph is weighted or not
    if weighted:
        win.blit(font.render("Weighted",0,(0,0,0)),(550,350))
    else:
        win.blit(font.render("Not weighted",0,(0,0,0)),(550,350))

    #design for the weighed? button
    pygame.draw.circle(win,(0,0,0),(620,420),18,2,draw_top_left=True,draw_bottom_left=True)
    pygame.draw.circle(win,(0,0,0),(660,420),18,2,draw_top_right=True,draw_bottom_right=True)
    pygame.draw.line(win,(0,0,0),(620,420-18),(660,420-18),2)
    pygame.draw.line(win,(0,0,0),(620,420+17),(660,420+17),2)

    pygame.draw.circle(win,colours[theme],(620,420),14,draw_top_left=True,draw_bottom_left=True)
    pygame.draw.circle(win,colour2[theme],(660,420),14,draw_top_right=True,draw_bottom_right=True)
    pygame.draw.rect(win,colours[theme],(620,420-14,40,28))

    #displays a green light for weighted and a red light for not weighted
    if not weighted:
        pygame.draw.circle(win,(255,0,0),(620,420),10)
    else:
        pygame.draw.circle(win,(0,255,0),(660,420),10)

    #enter button
    pygame.draw.rect(win,(0,0,0),(580,470,120,40),2)
    win.blit(font.render("Enter",0,(0,0,0)),(590,470))
    #checks for mouse button clicks
    if pygame.mouse.get_pressed()[0]:
        #if they click outside the box then it removes the text box
        if not(mousex >= 320 and mousex <= 960 and mousey >= 320 and mousey <= 600):
            text_box_graph = False
        #if the user clicks the weighte doutton then it changes it from weighted to not weighted and vice versa
        elif mousex >= 620-18 and mousex <= 660+18 and mousey >= 420-18 and mousey <= 420+18:
            #if its weighted then it will change it to not weighted
            if weighted:
                weighted = False
                win.blit(font.render("Not Weighted",0,(0,0,0)),(550,350))
                #otherwise it will be weighted
            else:
                weighted = True
                win.blit(font.render("Weighted",0,(0,0,0)),(550,350))
            #this waits 0.1 which acts like a buffer so that the person does not end up spamming
            time.sleep(0.1)
            #enter button
        elif mousex >= 580 and mousex <= 700 and mousey >= 470 and mousey <= 510:
            #makes sure to see theres a name inputted
            if current_string != "":
                conn = sqlite3.connect('resources/savedata/savedata.db')
                curs = conn.cursor()
                #inputs it into the database
                curs.execute(f"SELECT * FROM Graph_info WHERE Account_id={current_account_id} AND Folder_id={current_folder_id}")
                #checks for duplicates
                check = curs.fetchall()
                #checks to see if its already there or not
                alr = False
                for i in check:
                    if current_string == i[0]:
                        alr = True

                #if it is not already tehre then it will insert it into the table
                #It enters an empty from and an empty list because the user has just created the graph
                #The other stuff are variables which have already been defined
                if not alr:
                    curs.execute("INSERT INTO Graph_info VALUES (:Name,:Weighted,:Directed,:Graph,:List,:Account_id,:Folder_id,:Graph_id)",
                        {
                        'Name':current_string,
                        'Weighted':weighted,
                        'Directed':False,
                        'Graph':pickle.dumps([]),
                        'List':pickle.dumps([]),
                        'Account_id':current_account_id,
                        'Folder_id':current_folder_id,
                        'Graph_id':largest_graph_id+1

                        })
                    text_box_graph = False
                    #Increments the latest graph by 1
                    current_graph_id = largest_graph_id+1
                    #empties the current string
                    current_string = ""
                conn.commit()
                conn.close()
```

This creates a folder, it works in the same way as creating a graph in that it will set up a text box for the user to enter the name of the folder and when the user has entered it they can click enter and a lot of validity checks will take place to make sure that the data can be entered and it will then add it to the table of folders. This means that they will now be an option to display and the user can click on them.

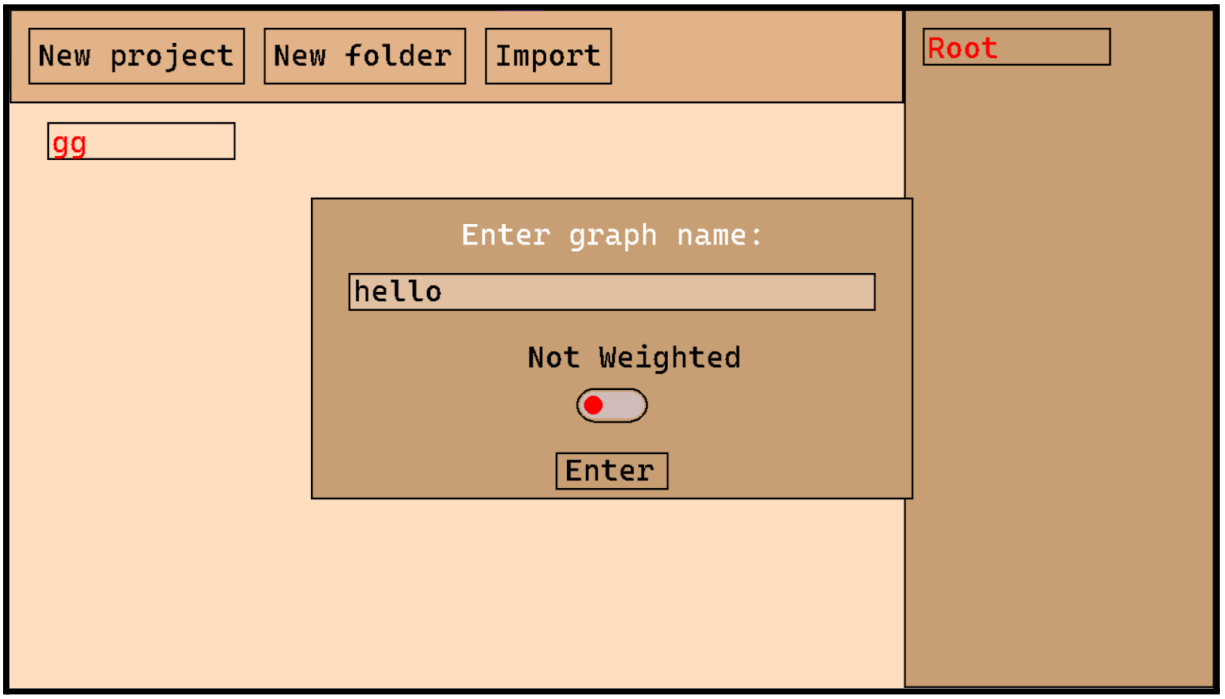
```
else:

    #Displays the folder popup text box and buttons
    pygame.draw.rect(win,colour4[theme],(320,240,640,240))
    pygame.draw.rect(win,(0,0,0),(320,240,640,240),2)
    #Entry field
    win.blit(font.render("Enter folder name:",0,(255,255,255)),(480,300))
    pygame.draw.rect(win,colour5[theme],(360,360,560,40))
    win.blit(font.render(current_string,0,(0,0,0)),(370,360))
    pygame.draw.rect(win,(0,0,0),(360,360,560,40),2)
    pygame.draw.rect(win,(0,0,0),(580,420,120,40),2)
    #Displays the enter button
    win.blit(font.render("Enter",0,(0,0,0)),(590,420))
    #Checks that the mouse has bene clicked
    if pygame.mouse.get_pressed()[0]:
        #Checks to see if the mouse was outside the box otherwise it deselects
        if not(mousex >= 320 and mousex <= 960 and mousey >= 240 and mousey <= 480):
            text_box_folder = False

        #If it clicks on the button
        elif mousex >= 580 and mousex <= 700 and mousey >= 420 and mousey <= 460:
            #makes sure that the box is not empty
            if current_string != "":
                #Connects to the database
                conn = sqlite3.connect('Resources/SaveData/SaveData.db')
                curs = conn.cursor()
                #retreives all the folders under the account
                curs.execute(f"SELECT * FROM Folder_info WHERE Account_id={current_account_id}")
                #sets them to a list
                check = curs.fetchall()
                #Check to see if there is a duplicate
                alr = False
                for i in check:
                    if current_string == i[0]:
                        alr = True
                #If there is not a duplicate it inputs it into the database under the user
                if not alr:
                    curs.execute("INSERT INTO Folder_info VALUES (:Folder_id,:Name,:Account_id)",
                        {
                        'Folder_id':largest_folder_id+1,
                        'Name':current_string,
                        'Account_id':current_account_id

                        })
                    text_box_folder = False
                    current_folder_id = largest_folder_id+1
                conn.commit()
                conn.close()
                current_string = ""
```

Screen Design

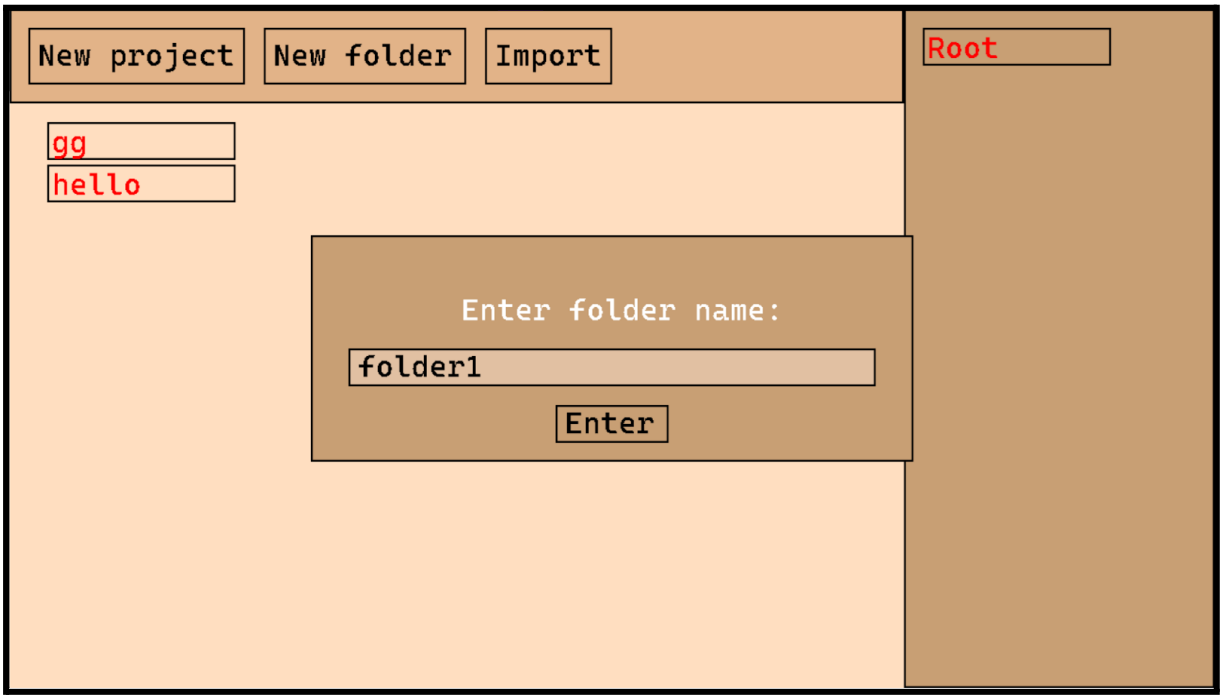
This is the menu screen where you can see the graphs are displayed in the centre and the folders are displayed on the right. The buttons are all displayed at the top



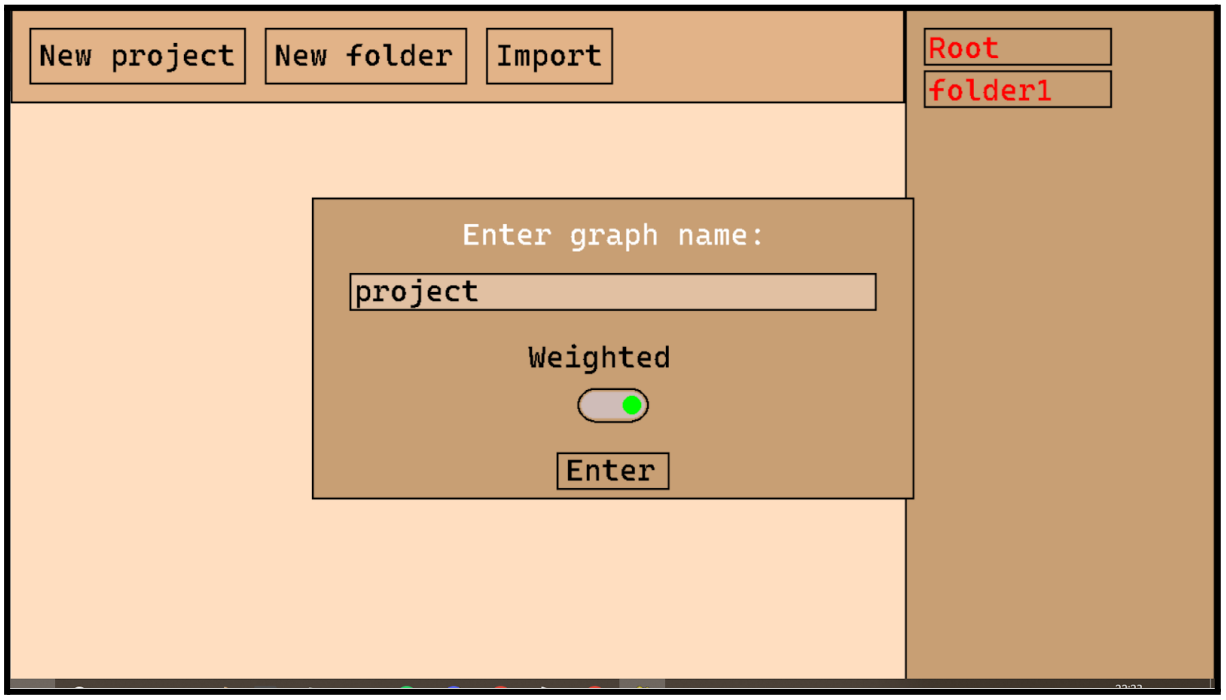
This is the popup for when the user wants to create a new project. It will ask them if they want it weighted or not and they can select it as well as have the input box for inputting the name of the graph



This is the folder input box. It is almost identical to the graph input box except for the lack of the weight since it does not make sense to have that



This is an example of adding a new graph to a folder. This time you can see what it looks like when you make the graph weighted.



Testing

The system works very well even though there was a lot of database manipulation and therefore there was a lot that could go wrong when doing this. However one problem is that you cannot know which folder you are currently on so I will need some way of indicating that. Either by writing its name at the top or perhaps by highlighting the box.

Conclusion

The menu system works completely fine and there aren't any errors.

Version 4

Program Code

This will be in the canvas page. It starts off by updating the matrix since it will cause the program to work smoothly without random interrupts which is the reason for it being at the start. In addition it synchronises the matrix along the leading diagonal making it symmetrical along it. In addition this section will draw the nodes and the edges of the graph. The reason the edges are drawn first is so that they are behind the nodes so it looks cleaner. In addition a list is created which includes all the graphs in the folder selected and under the account that the user is in.

```
elif page == "canvas":
    #Whenever the game loops it will always update the matrix by using the list_to_graph module.
    #This means whenever the matrix has an algorithm applied onto it, the matrix will be updated at the start of the loop
    matrix = list_to_graph(nodes)
    #This help with any errors that occur during the process of conversion. Sometimes the graph only fills half of the matrix.
    #This algorithm makes the matrix symmetrical along the leading diagonal
    #loops through all the columns
    for a,i in enumerate(matrix):
        #loops though all the rows
        for b,j in enumerate(matrix):
            #checks to see that there is an edge in the cell currently being examined
            if matrix[a][b] != '-':
                #will set the opposite value to it
                matrix[b][a] = matrix[a][b]
            #does the opposite for its mirror
            if matrix[b][a] != '-':
                matrix[a][b] = matrix[b][a]


    #loops through all the nodes
    for i in nodes:
        #loops through all the neighbours
        for j in i.neighbours:
            #loops through all the node again
            for k in nodes:
                #If the node is connected to the other node
                if j[0] == k.name:
                    #Draws the edge
                    pygame.draw.line(win,(255,0,0),(i.x,i.y),(k.x,k.y),2)
                    #If it is weighted then it also draws its weight
                    if weighted:
                        win.blit(font.render(str(j[1]),0,(255,255,255)),((i.x+k.x)//2,(i.y+k.y)//2))

    #loops through all the nodes
    for i in nodes:
        #draws the node and its name
        pygame.draw.circle(win,(255,255,255),(i.x,i.y),20)
        pygame.draw.circle(win,(0,0,0),(i.x,i.y),20,1)
        win.blit(small_font.render(i.name,0,(0,0,0)),(i.x-30,i.y-40))


    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
    curs = conn.cursor()
    #gets all the graphs from the current folder
    curs.execute(f"SELECT * FROM Graph_info WHERE Folder_id={current_folder_id} AND Account_id={current_account_id}")
    graph_list = curs.fetchall()
    conn.commit()
    conn.close()

    #checks to see if this graph is weighted
    try:
        weighted = graph_list[current_graph_id][1]
    except:
        weighted = graph_list[current_graph_id-1][1]

    #draws the pannels
```

Candidate Name: Khalyl Boukhanouf Centre Name: George Spencer Academy
Candidate Number: 7030 Centre Number: 28278

This part of the program checks to see whether the graph is weighted or not and it will display it in the sidebar. It also loads the buttons for the algorithms (prims, kurskals and dijkstra's). It also checks if they are selected and if they are it will highlight them so that the user can know that they've been selected. Finally the right click menu has been implemented where if the user right clicks then it will send them to the right click menu but if the user then clicks off while on the right click menu then it won't be displayed. It also checks whether the user has right clicked on a node or just in general.

```
if weighted:
    weight = "True"
else:
    weight = "False"

#displays whether it is weighed or not
win.blit(font.render(f"Weighted:{weight}",0,(0,0,0)),(955,10))
#Not yet made it able to be directed so for now all the graphs will be undirected
win.blit(font.render(f"Directed:False",0,(0,0,0)),(955,50))

#Displays the buttons for prims kruskals and dijkstras
pygame.draw.rect(win,(0,0,0),(955,90,200,40),1)
win.blit(font.render(f"Prims",0,(0,0,0)),(960,90))
#If the algorithm has been selected then it will highlight it red
if prims:
    pygame.draw.rect(win,(255,0,0),(955,90,200,40),1)
    win.blit(font.render(f"Prims",0,(255,0,0)),(960,90))
#same for kruskals
pygame.draw.rect(win,(0,0,0),(955,140,200,40),1)
win.blit(font.render(f"Kruskals",0,(0,0,0)),(960,140))
if kruskals:
    pygame.draw.rect(win,(255,0,0),(955,140,200,40),1)
    win.blit(font.render(f"Kruskals",0,(255,0,0)),(960,140))
#same for dijkstras
pygame.draw.rect(win,(0,0,0),(955,190,200,40),1)
win.blit(font.render(f"Dijkstras",0,(0,0,0)),(960,190))
if dijkstras:
    pygame.draw.rect(win,(255,0,0),(955,190,200,40),1)
    win.blit(font.render(f"Dijkstras",0,(255,0,0)),(960,190))
#reset button
pygame.draw.rect(win,(0,0,0),(955,240,200,40),1)
win.blit(font.render(f"Reset",0,(0,0,0)),(960,240))


#checks to see if the right click menu is active
if right_click_menu:
    #if they left click or middle click
    if pygame.mouse.get_pressed()[0] or pygame.mouse.get_pressed()[1]:
        #if they click outside the menu area
        if not(mousex > tempx and mousey > tempy and mousex < tempx+200 and mousey < tempy+340):
            #deselects the right click menu
            right_click_menu = False
    #if they right click tho then it just changes the corrdinates of the right click menu
    elif pygame.mouse.get_pressed()[2]:
        tempx,tempy=mousex,mousey
#if the menu isnt active
else:
    #if the user clicks in the region of the canvas
    if mousex >= 0 and mousex <= 950 and mousey >= 100 and mousey <= 720:
        #if they right click
        if pygame.mouse.get_pressed()[2]:
            #it will load the menu
            right_click_menu = True
            #check to see if any node has been clicked on at all
            current_node_right_clicked_on = None
            #cycles through all the node
```

```
        for i in nodes:
            #checks each of the nodes corrdinated and a 40x40 region around its center to check if the user has right clicked on that node
            if mousex >= i.x-20 and mousex <= i.x+20 and mousey >= i.y-20 and mousey <= i.y+20:
                #if so then it will set it to the current node right clicked on
                current_node_right_clicked_on = i
        #in addition when the user right clicks it sets tempx,temy to the coordatinated which were right clicked on. This precents it from following the mouse cursor
        tempx,tempy=mousex,mousey
```

This is the function of the right click menu. This first displays all the buttons. One feature I added is that if the user cannot do a certain action (for example paste when nothing is in the clipboard) then it will grey out that button on the right click menu and the user will not be able to do that. The right click menu functions by right clicking and clicking on a button. When the user clicks on add it will open the add_node popup so the user can add a node. When the user clicks on delete it checks to see if there is a node currently selected and it will remove it from the list and from the neighbours in other nodes. THe copy button copies the selected node and all its neighbours and the paste pastes that node along with all its neighbours into a new node. This means that this will have to be inserted into the database again

```
#if the right click menu is active
if right_click_menu:
    #displays the copy button
    pygame.draw.rect(win,(255,255,255),(tempx,tempy,200,340))
    pygame.draw.rect(win,(180,180,180),(tempx,tempy,200,340),1)
    pygame.draw.rect(win,(0,0,0),(tempx+20,tempy+20,40,45),1)
    pygame.draw.rect(win,(255,255,255),(tempx+24,tempy+16,40,45))
    pygame.draw.rect(win,(0,0,0),(tempx+24,tempy+16,40,45),1)
    win.blit(font.render("Copy",0,(0,0,0)),(tempx+80,tempy+20))
    #displays the paste button
    pygame.draw.line(win,(180,180,180),(tempx,tempy+80),(tempx+200,tempy+80),1)
    #if something exists in the clipboard then it will display black, otherwisde it will display grey
    if clipboard != None:
        win.blit(paste_icon[1],(tempx+10,tempy+90))
        win.blit(font.render("Paste",0,(0,0,0)),(tempx+80,tempy+105))
    else:
        win.blit(paste_icon[0],(tempx+10,tempy+90))
        win.blit(font.render("Paste",0,(150,150,150)),(tempx+80,tempy+105))

    pygame.draw.line(win,(180,180,180),(tempx,tempy+170),(tempx+200,tempy+170),1)
    #Displays the add button
    pygame.draw.circle(win,(0,0,0),(tempx+42,tempy+210),25,1)
    win.blit(font.render("Add",0,(0,0,0)),(tempx+90,tempy+190))

    pygame.draw.line(win,(180,180,180),(tempx,tempy+260),(tempx+200,tempy+260),1)
    #displays the delete button but displays it grey if o node is selected but displays black if there is
    if current_node_right_clicked_on != None:
        pygame.draw.line(win,(0,0,0),(tempx+20,tempy+270),(tempx+50,tempy+320))
        pygame.draw.line(win,(0,0,0),(tempx+20,tempy+320),(tempx+50,tempy+270))
        win.blit(font.render("Delete",0,(0,0,0)),(tempx+70,tempy+280))
    else:
        pygame.draw.line(win,(150,150,150),(tempx+20,tempy+270),(tempx+50,tempy+320))
        pygame.draw.line(win,(150,150,150),(tempx+20,tempy+320),(tempx+50,tempy+270))
        win.blit(font.render("Delete",0,(150,150,150)),(tempx+70,tempy+280))
    #if the user clicks
    if pygame.mouse.get_pressed()[0]:
        #checks to see if its in the range of the right click menu
        if mousex > tempx and mousex < tempx+200:
            #checks to see if its in the range of the copy button
            if mousey > tempy and mousey < tempy+80:
                #tries to add a node to the cliboard
                try:
                    clipboard = current_node_right_clicked_on
                except:pass
            #checks to see if its in the range of the paste button
            elif mousey > tempy+80 and mousey < tempy+170:
                #checks to see if there issomething in the clipboard
                if clipboard:
                    #checks to see if there is already a copy of that node
                    alr = False
                    temp_neighbours = []
                    #finds all the neighbours of the node that has been copied
                    for i in clipboard.neighbours:
                        temp_neighbours.append(i)
```

However, the entry in the database needs to be the correct graph that the user has been selecting so it will then have to be found using WHERE clauses. In addition, i f the user clicks the add menu then it will close the right click menu.

```
                #cycles through al lthe nodes and their neighbours to see if they share a neighbour with the node in the clipboard
                for i in nodes:
                    for j in i.neighbours:
                        if j[0] == clipboard.name:
                            #if they do then it wil again append it to the neighbours of the copying node
                            temp_neighbours.append([i.name,j[1]])

                #checks too see for  adulicate name
                for i in nodes:
                    if i.name == clipboard.name + " - Copy":
                        alr = True

                #adds the new copied node to the nodes list
                if not alr:
                    nodes.append(node(clipboard.name + " - Copy",len(nodes)+1,clipboard.x,clipboard.y,temp_neighbours))

                #remvoves the old node from clipboard
                clipboard = None

        #checks to see if it is in the range if the add button
        elif mousey > tempy+170 and mousey < tempy+260:
            #opens the add menu but closes the right click menu
            add_menu = True
            right_click_menu = False
        #checks to see if it is in the range of the delete button
        elif mousey > tempy+260 and mousey < tempy+340:
            try:
                #deletes the node from the nodes list
                for i in nodes:
                    if i.name == current_node_right_clicked_on.name:
                        nodes.remove(i)
                #goes through every other nodes neighbours and deletes the connection between theme
                for i in nodes:
                    for j in i.neighbours:
                        if j[0] == current_node_right_clicked_on.name:
                            i.neighbours.remove(j)
            except:pass
    #saves any changes made to the database that has been made
    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
    curs = conn.cursor()
    curs.execute(f"SELECT * FROM Graph_info WHERE Account_id={current_account_id} AND Folder_id={current_folder_id} AND Graph_id={current_graph_id}")
    check = curs.fetchall()

    check = check[0]
    curs.execute(f"""UPDATE Graph_info SET
        Name = :Name,
        Weighted = :Weighted,
        Directed = :Directed,
        Graph = :Graph,
        List = :List,
        Account_id = :Account_id,
        Folder_id = :Folder_id,
        Graph_id = :Graph_id
```

```
                WHERE Account_id = {current_account_id} AND Folder_id = {current_folder_id} AND Graph_id = {current_graph_id}
                """,
                {

                'Name':check[0],
                'Weighted':check[1],
                'Directed':check[2],
                'Graph':pickle.dumps(matrix),
                'List':pickle.dumps(nodes),
                'Account_id':check[5],
                'Folder_id':check[6],
                'Graph_id':check[7],


                })


            conn.commit()
            conn.close()
            current_string = ""
```

If the user has not right clicked then the program will check to see if they have selected to weight an edge. If they have then a popup box will open asking then what the weight of their edge is to be. The user will then input a number. If this is not a number then the program will refuse it however if they do enter a number then it will be added to the neighbours list in all the affected nodes and then it will be turned into a matrix. At this point, the entry in the table will be updated to account for this new edge that has been added to the graph.

```
elif node_menu:
    #if tje user is on the add node menu
    #displays the popup box
    pygame.draw.rect(win,colour4[theme],(320,240,640,240))
    pygame.draw.rect(win,(0,0,0),(320,240,640,240),2)
    #displays entry field
    win.blit(font.render("Enter edge weight:",0,(255,255,255)),(480,300))
    pygame.draw.rect(win,colour5[theme],(360,360,560,40))
    #tests to see if the string is an integer. If it isnt then it will remove the last digit
    try:
        int(current_string)
    except:
        current_string = current_string[:-1]
    #displats enter button
    win.blit(font.render(current_string,0,(0,0,0)),(370,360))
    pygame.draw.rect(win,(0,0,0),(360,360,560,40),2)
    pygame.draw.rect(win,(0,0,0),(580,420,120,40),2)
    win.blit(font.render("Enter",0,(0,0,0)),(590,420))
    #if the mouse is pressed
    if pygame.mouse.get_pressed()[0]:
        #if the user has clicked enter
        if mousex >= 580 and mousex <= 700 and mousey >= 420 and mousey <= 460:
            #if the string isnt empty
            if current_string != "":
                #closes the popup
                node_menu = False
                #it adds to the neighours list the node edge
                nodes[closest[2]].neighbours.append([nodes[second_closest[2]].name,int(current_string)])
                matrix = list_to_graph(nodes)
                #the matrix gets updated
                conn = sqlite3.connect('Resources/SaveData/SaveData.db')
                curs = conn.cursor()
                curs.execute(f"SELECT * FROM Graph_info WHERE Account_id={current_account_id} AND Folder_id={current_folder_id} AND Graph_id={current_graph_id}")
                check = curs.fetchall()
                check = check[0]
                #updates the database
                curs.execute(f"""UPDATE Graph_info SET
                    Name = :Name,
                    Weighted = :Weighted,
                    Directed = :Directed,
                    Graph = :Graph,
                    List = :List,
                    Account_id = :Account_id,
                    Folder_id = :Folder_id,
                    Graph_id = :Graph_id
                    WHERE Account_id = {current_account_id} AND Folder_id = {current_folder_id} AND Graph_id = {current_graph_id}
                    """,
                    {
                    'Name':check[0],
                    'Weighted':check[1],
                    'Directed':check[2],
                    'Graph':pickle.dumps(matrix),
                    'List':pickle.dumps(nodes),
                    'Account_id':check[5],
                    'Folder_id':check[6],
                    'Graph_id':check[7],})
```

This is the popup for adding a node to the graph. It works by asking the user to input a string for the name of the node. It then does a few validity checks and then enters into the database when it has seen that it is not a duplicate.

```
#if the add menu is selected
elif add_menu:
    #displays the buttons
    pygame.draw.rect(win,colour4[theme],(320,240,640,240))
    pygame.draw.rect(win,(0,0,0),(320,240,640,240),2)
    #displays the entry field
    win.blit(font.render("Enter node name:",0,(255,255,255)),(480,300))
    pygame.draw.rect(win,colour5[theme],(360,360,560,40))
    #limits the name of the node to be 10 characters long
    current_string = current_string[:10]
    #displays the name in the entry field
    win.blit(font.render(current_string,0,(0,0,0)),(370,360))
    pygame.draw.rect(win,(0,0,0),(360,360,560,40),2)
    pygame.draw.rect(win,(0,0,0),(580,420,120,40),2)
    #enter button
    win.blit(font.render("Enter",0,(0,0,0)),(590,420))
    #if the user clicks enter
    if pygame.mouse.get_pressed()[0]:
        #checks to see if the user has clicked on the right area
        if mousex >= 580 and mousex <= 700 and mousey >= 420 and mousey <= 460:
            #makes sure the node has a name
            if current_string != "":
                #check to see if a node already has this name
                alr = False
                for i in nodes:
                    if current_string == i.name:
                        alr = True
                #if it doesnt
                if not alr:
                    #finds the latest position of the nodes
                    temp_len = len(nodes)+1
                    #adds it to the nodes list
                    nodes.append(node(current_string,temp_len,tempx,tempy,[]))
                    #closes the menu
                    add_menu = False

                    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
                    curs = conn.cursor()
                    curs.execute(f"SELECT * FROM Graph_info WHERE Account_id={current_account_id} AND Folder_id={current_folder_id} AND Graph_id={current_graph_id}")
                    check = curs.fetchall()

                    check = check[0]
                    #adds it to the database
                    curs.execute(f"""UPDATE Graph_info SET
                        Name = :Name,
                        Weighted = :Weighted,
                        Directed = :Directed,
                        Graph = :Graph,
                        List = :List,
                        Account_id = :Account_id,
                        Folder_id = :Folder_id,
                        Graph_id = :Graph_id
```

```
            WHERE Account_id = {current_account_id} AND Folder_id = {current_folder_id} AND Graph_id = {current_graph_id}
            """,
            {
            'Name':check[0],
            'Weighted':check[1],
            'Directed':check[2],
            'Graph':pickle.dumps(matrix),
            'List':pickle.dumps(nodes),
            'Account_id':check[5],
            'Folder_id':check[6],
            'Graph_id':check[7],


            })


        conn.commit()
        conn.close()
        current_string = ""
```

This section of the program runs when there is no menu/popup being selected. This part just checks to detect the press of any of the buttons on the right hand side. It is a toggle mechanism where only one of the 3 algorithms can be run at any given time. There is a also a reset button to remove all of the algorithms traces

```
else:

    #checks to see if the user has clicked on any of the algorthm buttons
    if pygame.mouse.get_pressed()[0]:
        #if they click the back button then the user goes back to the menu
        if mousex >= 20 and mousex <= 90 and mousey >= 20 and mousey <= 90:
            page = "menu"
            time.sleep(0.2)
        #if the user clicks on the right sidebar
        if mousex >= 955 and mousex <= 1155:
            #checks to see if prims has been clicked
            if mousey >= 90 and mousey <= 130:
                #only enables prims
                prims = True
                kruskals = False
                dijkstras = False
            #checks to see if kruskals has been clicked
            elif mousey >= 140 and mousey <= 180:
                #only enables kruskals
                prims = False
                kruskals = True
                dijkstras = False
            #checks to see if dijkstras has been clicked
            elif mousey >= 190 and mousey <= 230:
                #only enables djikstras
                prims = False
                kruskals = False
                dijkstras = True
            #checks to see if the reset button has been clicked
            elif mousey >= 240 and mousey <= 280:
                #disables all of them and resets the nodes in the arays
                prims = False
                kruskals = False
                dijkstras = False
                prims_nodes = []
                kruskal_nodes = []
                dijkstras_nodes = []
```

Candidate Name: Khalyl Boukhanouf Centre Name: George Spencer Academy
Candidate Number: 7030 Centre Number: 28278

This section is for displaying the result of prims, kruskal's and dijkstra's algorithms. The list of nodes that are needed for these algorithms are inside each of these if statements. They will cycle through each node and draw the edge that connects the 2.

```
#if prims is active then it will display the prims edges
if prims:
    #cycles through all the prims nodes
    for i in prims_nodes:
        #updates the coordinates so that if the user moves around the nodes these edges will also move around
        i.update_coords(nodes)
        #cycles through neighbours
        for j in i.neighbours:
            #cycles through the nodes
            for k in prims_nodes:
                #checks to see if they are neighbours
                if j[0] == k.name:
                    #draws the edge
                    pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)


#if kruskal is active then it will display the kruskal edges
if kruskals:
    #cycles through all the kruskal nodes
    for i in kruskals_nodes:
        #updates the coordinates so that if the user moves around the nodes these edges will also move around
        i.update_coords(nodes)
        #cycles through neighbours
        for j in i.neighbours:
            #cycles through the nodes
            for k in kruskals_nodes:
                #checks to see if they are neighbours
                if j[0] == k.name:
                    #draws the edge
                    pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)

#if djikstras is active then it will display the dijkstras edges
if dijkstras:
    #cycles through all the nodes
    for i in range(0,len(dijkstras_nodes)):
        #displays the edges from the start to the end of each of the nodes
        try:
            pygame.draw.line(win, (0, 0, 255), (nodes[dijkstras_nodes[i]].x, nodes[dijkstras_nodes[i]].y), (nodes[dijkstras_nodes[i+1]].x, nodes[dijkstras_nodes[i+1]

        except:pass
```

This part checks that the shift key has been pressed. If prims dijkstra's or kruskal's are not selected then the user will be able to click on 2 separate nodes and be able to create an edge between them. It will then update the table so that it will be saved in it for future usage.

```
#checks to see if shift is pressed and that there are movre than 2 nodes
keys = pygame.key.get_pressed()
if (keys[pygame.K_LSHIFT] or keys[pygame.K_RSHIFT]) and len(nodes) >=2:
    #if prims and kruskals are not selected
    if not prims and not kruskals:
        #the user can select the first node
        if first_node:
            #sets the first node to the closest as a base case
            closest = [nodes[0].x,nodes[0].y,0]
            #goes through all the node
            for a,i in enumerate(nodes):
                #checks to see if the magnitude of the distance between the nodes is smaller and if it is then it will set that as the new closest
                if ((i.x-mousex)**2+(i.y-mousey)**2)**0.5 <= ((mousex-closest[0])**2+(mousey-closest[1])**2)**0.5:
                    closest = [i.x,i.y,a]

            #highlights the selected node
            pygame.draw.circle(win,(0,255,0),(nodes[closest[2]].x,nodes[closest[2]].y),20)
            pygame.draw.circle(win,(0,0,0),(nodes[closest[2]].x,nodes[closest[2]].y),20,1)
            win.blit(small_font.render(nodes[closest[2]].name,0,(0,0,0)),(nodes[closest[2]].x-30,nodes[closest[2]].y-40))
            #if the user clicks then the second node will have a chance to get selected
            if pygame.mouse.get_pressed()[0]:
                first_node = False
                second_node = True
                time.sleep(0.2)

        #if the second node neds to be chosen
        elif second_node:
            #highlights the original selected node
            pygame.draw.circle(win,(0,0,255),(nodes[closest[2]].x,nodes[closest[2]].y),20)
            pygame.draw.circle(win,(0,0,0),(nodes[closest[2]].x,nodes[closest[2]].y),20,1)
            #finds a random other node to make the second closest as a base case
            try:
                second_closest = [nodes[closest[2]+1].x,nodes[closest[2]+1].y,closest[2]+1]
            except:
                second_closest = [nodes[closest[2]-1].x,nodes[closest[2]-1].y,closest[2]-1]

            #once again cycles though all the values to find the closest value
            for a,i in enumerate(nodes):
                #checks for the magnitude to see if it is smaller
                if ((i.x-mousex)**2+(i.y-mousey)**2)**0.5 <= ((mousex-second_closest[0])**2+(mousey-second_closest[1])**2)**0.5:
                    #checks to see that the node is not the same as the starting node
                    if second_closest[2] != closest[2]:
                        #sets it as the 2nd node
                        second_closest = [i.x,i.y,a]

            #if the user clicks the mouse now
            if pygame.mouse.get_pressed()[0]:
                #resets the variables
                first_node = True
                second_node = False
                time.sleep(0.2)
                #checks to see for duplicate nodes
                alr = False
                for i in nodes[closest[2]].neighbours:
                    if i[0] ==nodes[second_closest[2]].neighbours:
                        alr = True
```

```
#if there are no dupes
if not alr:
    #if dijkstras is not selected
    if not dijkstras:
        #if it isnt weighted (it isnt atm but for future development)
        if not weighted:
            nodes[closest[2]].neighbours.append([nodes[second_closest[2]].name,1])
            alr = False
            for i in nodes:
                if current_string == i.name:
                    alr = True

            if not alr:

                conn = sqlite3.connect('Resources/SaveData/SaveData.db')
                curs = conn.cursor()
                #get the informaion from the graph urrently working on
                curs.execute(f"SELECT * FROM Graph_info WHERE Account_id={current_account_id} AND Folder_id={current_folder_id} AND Graph_id={current_graph_id}")
                check = curs.fetchall()

                check = check[0]
                #updates the database
                curs.execute(f"""UPDATE Graph_info SET
                    Name = :Name,
                    Weighted = :Weighted,
                    Directed = :Directed,
                    Graph = :Graph,
                    List = :List,
                    Account_id = :Account_id,
                    Folder_id = :Folder_id,
                    Graph_id = :Graph_id

                    WHERE Account_id = {current_account_id} AND Folder_id = {current_folder_id} AND Graph_id = {current_graph_id}
                    """,
                    {

                    'Name':check[0],
                    'Weighted':check[1],
                    'Directed':check[2],
                    'Graph':pickle.dumps(matrix),
                    'List':pickle.dumps(nodes),
                    'Account_id':check[5],
                    'Folder_id':check[6],
                    'Graph_id':check[7],


                    })



                conn.commit()
                conn.close()
                current_string = ""

            else:
                #opens node menu
```

```
                            conn.commit()
                            conn.close()
                            current_string = ""

                        else:
                            #opens node menu
                            node_menu = True
                            current_string = ""

                    else:
                        #if dijkstras is active then it will run dijkstras algorithm
                        dijkstras_nodes = dijkstras_algorithm(matrix,closest[2],second_closest[2])
            #highlights the 2nd closest node
            pygame.draw.circle(win,(0,255,0),(nodes[second_closest[2]].x,nodes[second_closest[2]].y),20)
            pygame.draw.circle(win,(0,0,0),(nodes[second_closest[2]].x,nodes[second_closest[2]].y),20,1)
            win.blit(small_font.render(nodes[second_closest[2]].name,0,(0,0,0)),(nodes[second_closest[2]].x-30,nodes[second_closest[2]].y-40))
```

This section of the code is what happens when the user clicks one of the algorithms. If the user clicks dijkstra's then it will be the same as if they tried to connect 2 nodes together except that a new edge won't be created but a path from the 2 nodes selected. It will highlight the 2 nodes selected as well. If prims is chosen then the program will ask the user to select a node. At which it will highlight it as the starting node. Then the MST will be displayed on the screen in the order that it is created. This means that it will teach the user how prim's algorithm works. Kruskal's works in a very similar way to prims except that there is not a starting node which is required for it to run which means that just pressing shift is sufficient for the algorithm to run.

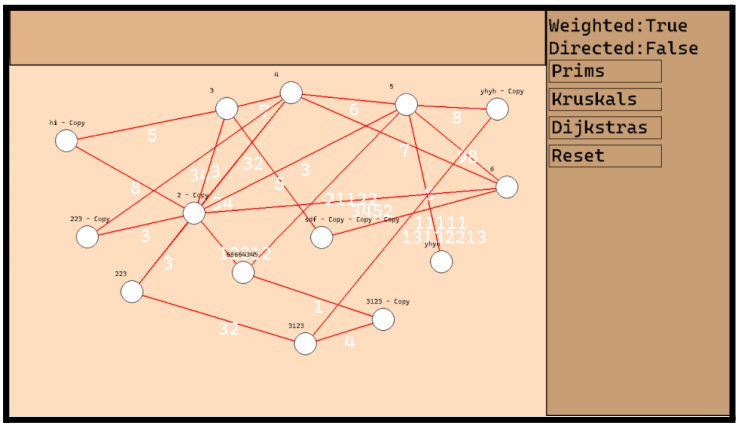
```
#if prims is active
elif prims:
    #the first node is the closes node (base case)
    closest = [nodes[0].x,nodes[0].y,0]
    #itearets through all the nodes to see if any of them are closr
    for a,i in enumerate(nodes):
        if ((i.x-mousex)**2+(i.y-mousey)**2)**0.5 <= ((mousex-closest[0])**2+(mousey-closest[1])**2)**0.5:
            closest = [i.x,i.y,a]
    #highlights selected node
    pygame.draw.circle(win,(0,255,0),(nodes[closest[2]].x,nodes[closest[2]].y),20)
    pygame.draw.circle(win,(0,0,0),(nodes[closest[2]].x,nodes[closest[2]].y),20,1)
    win.blit(small_font.render(nodes[closest[2]].name,0,(0,0,0)),(nodes[closest[2]].x-30,nodes[closest[2]].y-40))
    #if the user clicks
    if pygame.mouse.get_pressed()[0]:
    #prims algorithm runs
        prims_nodes = prims_algorithm(highest_algorithm(matrix),matrix,nodes[closest[2]],nodes)
        #initially shows the user how prims algorithm actually work
        #it cycles though the node and 1 by one displys them in the order that they were added in
        for i in prims_nodes:
            for j in i.neighbours:
                for k in prims_nodes:
                    if j[0] == k.name:
                        pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)
                        #the reason for this update is so that the user can actually see it in real time
                        pygame.display.update()
                        #that is also the same reason for this sleep which just pauses for 0.1 seconds
                        time.sleep(0.1)
#if kruskals is active
elif kruskals:
    #kruskals algorithm runs
    kruskals_nodes = kruskals_algorithm(highest_algorithm(matrix),matrix,nodes)
    #cycles though all of the node
    for i in kruskals_nodes:
        for j in i.neighbours:
            for k in kruskals_nodes:
                if j[0] == k.name:
                    #once again displays it in real time
                    pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)
                    pygame.display.update()
                    time.sleep(0.1)
```

This is the updated game loop. It incorporates the moving of the nodes which has been previously explained. Apart from this the only refinement is that it only works when the user is on the canvas and the user is not holding shift. There are also bounds now so now the user cannot drag the nodes outside the canvas

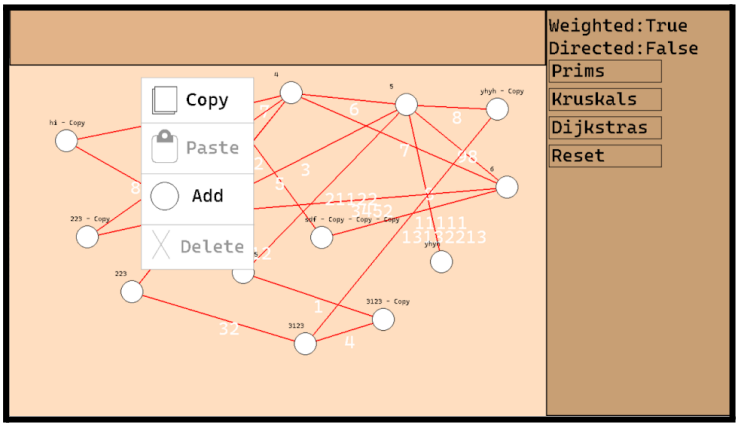
```
#This checks every event currently happening in the window
for event in pygame.event.get():
    #If the event is the big red X in the top corner being pressed or the ESCAPE key being pressed then it will close the program
    if event.type == pygame.QUIT or (event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE):
        #This just sets the variable which dictates the gameloop to false which means it will no longer run
        run = False
        #Checks to see if the user has typed anything
    if event.type == pygame.KEYDOWN:
        type(event,page)
    #checks to see if the page is canvas and the user has held down the button and the shift buttons havent been clicked
    if event.type == pygame.MOUSEBUTTONDOWN and page == "canvas" and not (keys[pygame.K_LSHIFT] or keys[pygame.K_RSHIFT]):
        #acknowledges the user is holding
        hold = True
        #no nodes currently selected
        selected_nodes = []
        #all the nodes that are in the area of mouse click are put into the selected nodes array
        for a,i in enumerate(nodes):
            if i.x-20<mousex<i.x+20 and i.y-20<mousey<i.y+20:
                selected_nodes.append(a)

        try:
            #this cycles through all the values in the selected array and checks to see if there is anything lower.
            current_lowest = nodes[selected_nodes[0]].pos
            for i in selected_nodes:
                if nodes[i].pos <= current_lowest:
                    #if it finds any lower then it will set that to the moving node
                    moving_node = i

        except:
            #otherwise it will deselect
            hold = False


    #if the user doesnt drag then they are not holding
    if event.type == pygame.MOUSEBUTTONUP:
        hold = False
#if they are holding then it changes the x and y corrds of thhe moving node to whatever the mouse x and y is
if hold:
    if mousex <= 950 and mousex >= 0:
        nodes[moving_node].x = mousex
    if mousey >=100 and mousey <=720:
        nodes[moving_node].y = mousey


#This updates any changes going on w the screen
pygame.display.update()
```

Screen Design

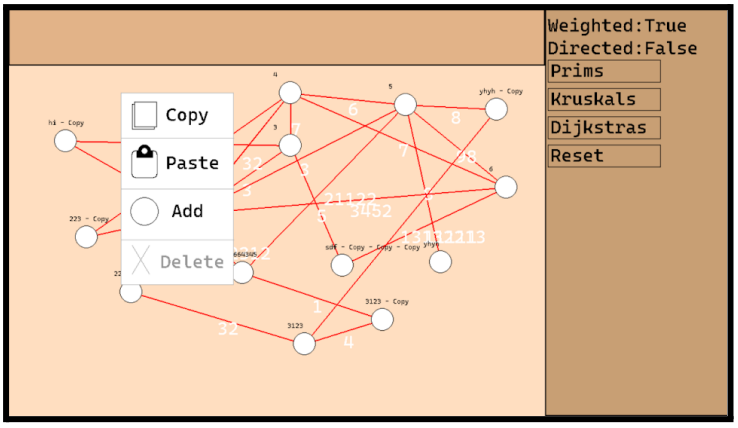
Canvas as normal



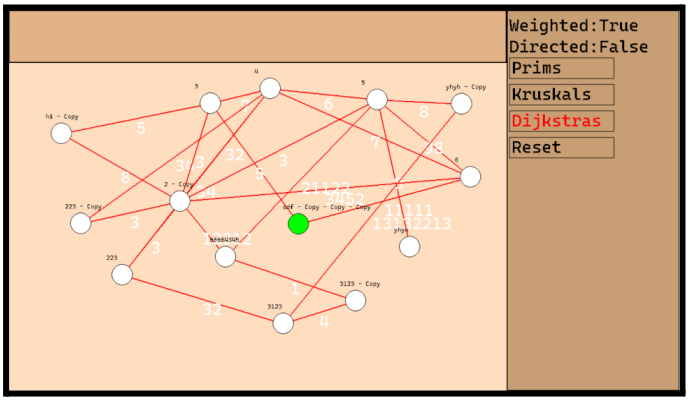
Right click menu (nothing in clipboard)



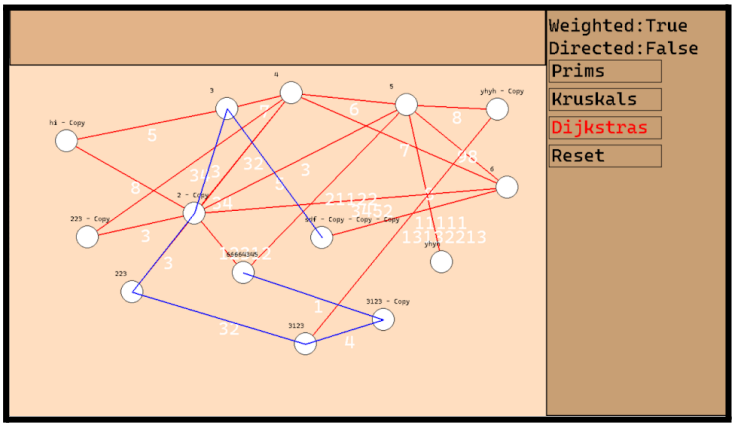
Right click menu (something in clipboard)



Selecting node



Dijkstra's algorithm



Testing

Some problem that I encountered is that the program crashes whenever the user tries to run dijkstra's on a non-connected graph. This is because if there is no path between the starting and ending node then it will just keep looking until the end of time at which it will just crash due to a stack overflow error.

Another small visual glitch happens when I load the right click menu since if you keep holding the right click you can drag it around. But that is not a big problem.

Apart from that the algorithms all work under normal circumstances and the right click menu functions fine. However I do want the user to be able to go back to other projects and edit them instead of needing to close the program entirely and re-open it.

Conclusion

The canvas is working harmoniously with the menu system and there aren't any database errors which is something I was concerned about.

Version 5

Program Code

This part occurs in the page if statement and it just detects if the user wants to go to the settings. If they do then it will check whether they came from the menu or the login page so when they go back it returns them to the place they were at. The only thing in the settings page so far is being able to change the colour scheme of the program which just works by clicking on which theme you want.

```
elif page == "settings_login" or page == "settings_menu":
    #Draws the settings art
    win.blit(back_icon,(20,20))
    win.blit(font.render("Theme 1",0,(0,0,0)),(600,70))
    win.blit(font.render("Theme 2",0,(0,0,0)),(600,110))
    #If user clicks
    if pygame.mouse.get_pressed()[0]:
        #If they select the back button then it returns them to the place they were at
        if mousex >= 20 and mousex <= 90 and mousey >= 20 and mousey <= 90:
            if page == "settings_login":
                page = "login"
            else:
                page = "menu"
                time.sleep(0.2)
        #If they click on theme1/theme 2 it changes the colours
        if mousex >= 600 and mousex <= 720:
            if mousey >= 70 and mousey <= 100:
                theme = 0
            elif mousey >= 110 and mousey <= 150:
                theme = 1
```

This section of code adds a back button to the canvas so that the user can go back to edit other projects instead of closing and reopening the program.

```
if pygame.mouse.get_pressed()[0]:
    #if they click the back button then the user goes back to the menu
    if mousex >= 20 and mousex <= 90 and mousey >= 20 and mousey <= 90:
        page = "menu"
        time.sleep(0.2)
```

This is the code for exporting a graph. When the user clicks the export button the current graph that is being edited will have all its information extracted from the table and put into a list. That list is then pickled (turned into bytes) and a .dat file is opened and it is written to the .dat file and put into the exports folder.

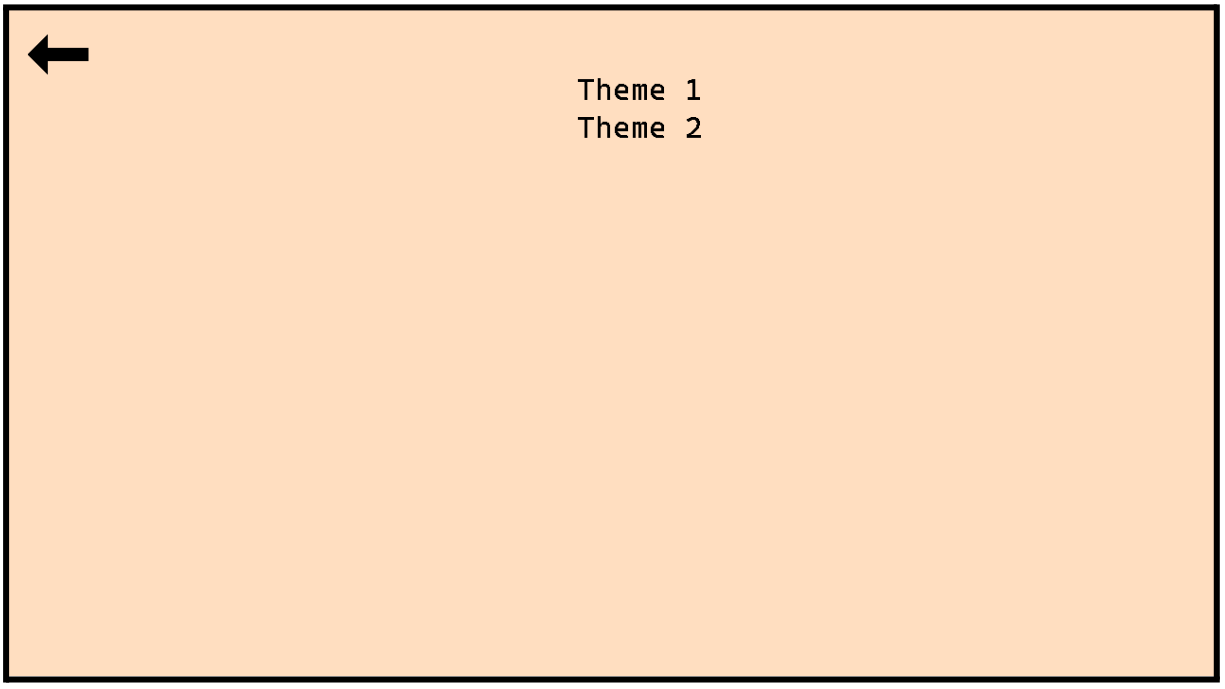
```
#is the user clicks on the export buttons
elif mousex >=870 and mousex <= 940 and mousey >=20 and mousey <=90:
    if pygame.mouse.get_pressed()[0]:
        #it collects all the information of the graph from graph list
        temp_import_graph = graph_list[current_graph_id]
        #it opens a new file with the name of th graph. the file is a dat file
        export_file = open("Resources/Exports/"+graph_list[current_graph_id][0]+".dat","wb")
        #pickles the information and stores it in the bnary file
        export_file.write(pickle.dumps(temp_import_graph))
        #closes the file so its saved
        export_file.close()
```

This is in the menu screen and it is for importing a graph. If the user clicks import then a windows explorer screen will popup and the user will be able to select the .dat file. The only kinds of files that are present are .dat files and in addition, it will automatically put the user into the imports folder. Then, when the .dat file is retrieved it will make sure that they name is different to every other graph in that folder and once the validity checks are done then it will input it into the table as a new graph and then the user will be able to see it on the menu screen

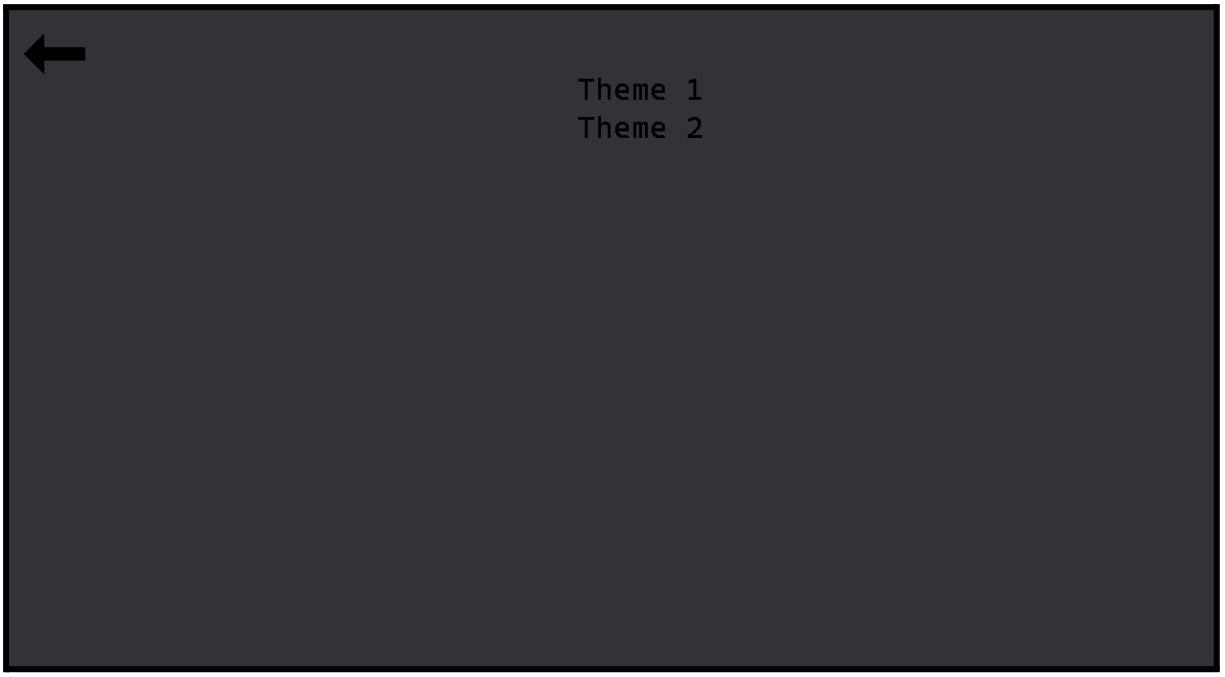
```
elif mousex >= 505 and mousex <= 640 and mousey >= 20 and mousey <= 80:
    #opens windows explorere on the imports folder ands gets the user to select a file
    import_graph = filedialog.askopenfilename(initialdir="Resources/Imports",title="Select A File",filetypes=(("dat files", "*.dat"),))
    #sets to the value being imported
    opening_import = open(import_graph,"rb")
    #unloads the binary file
    graph_pieces = pickle.loads(opening_import.read())
    opening_import.close()

    conn = sqlite3.connect('Resources/SaveData/SaveData.db')
    curs = conn.cursor()
    #creates a list with all the graphs in the folder in the account
    curs.execute(f"SELECT * FROM Graph_info WHERE Account_id={current_account_id} AND Folder_id={current_folder_id}")
    check = curs.fetchall()
    alr = False
    #Checks to see if its already in the databse
    for i in check:
        #if the names are the same
        if graph_pieces[0] == i[0]:
            #knows its already there
            alr = True
    #if it is not already there then it wil add it to the database
    if not alr:
        curs.execute("INSERT INTO Graph_info VALUES (:Name,:Weighted,:Directed,:Graph,:List,:Account_id,:Folder_id,:Graph_id)",
            {
            'Name':graph_pieces[0],
            'Weighted':graph_pieces[1],
            'Directed':graph_pieces[2],
            'Graph':graph_pieces[3],
            'List':graph_pieces[4],
            'Account_id':current_account_id,
            'Folder_id':current_folder_id,
            'Graph_id':largest_graph_id+1

            })
        #turns off the text box
        text_box_graph = False
        #increases the length of the graphs list
        current_graph_id = largest_graph_id+1
        #sets the string to nothing
        current_string = ""
    #closes the database
    conn.commit()
    conn.close()
```

Screen Design

Setting screen (theme 1)

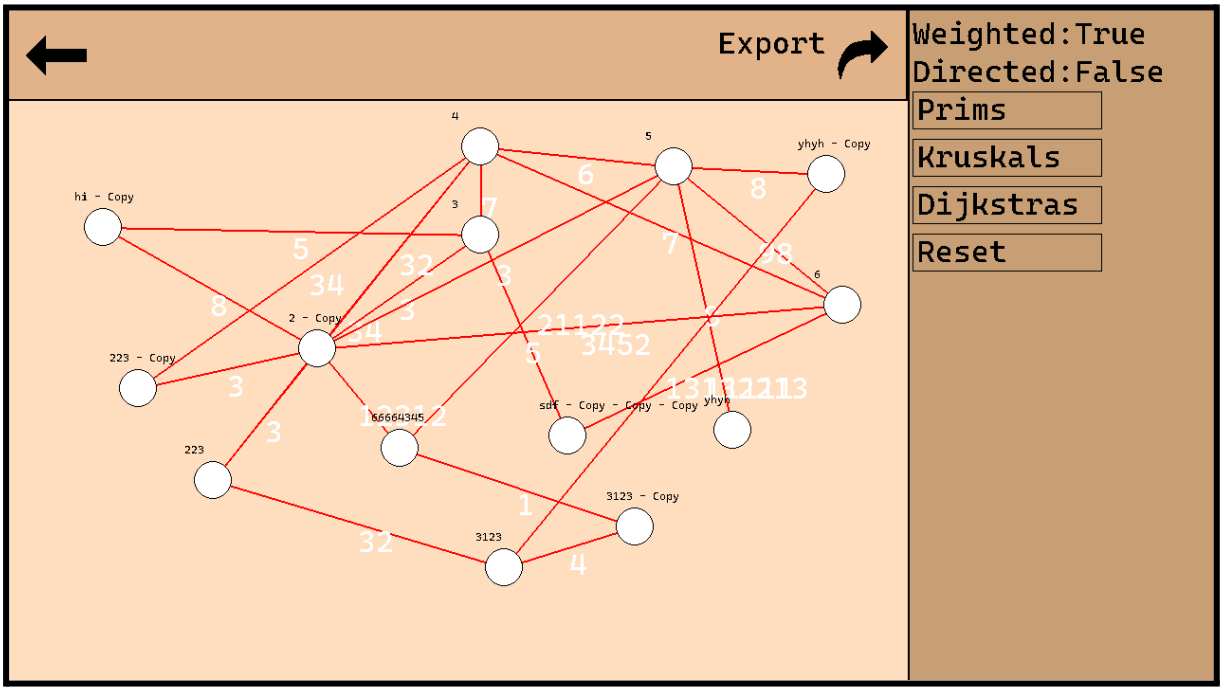


Settings screen (theme 2)

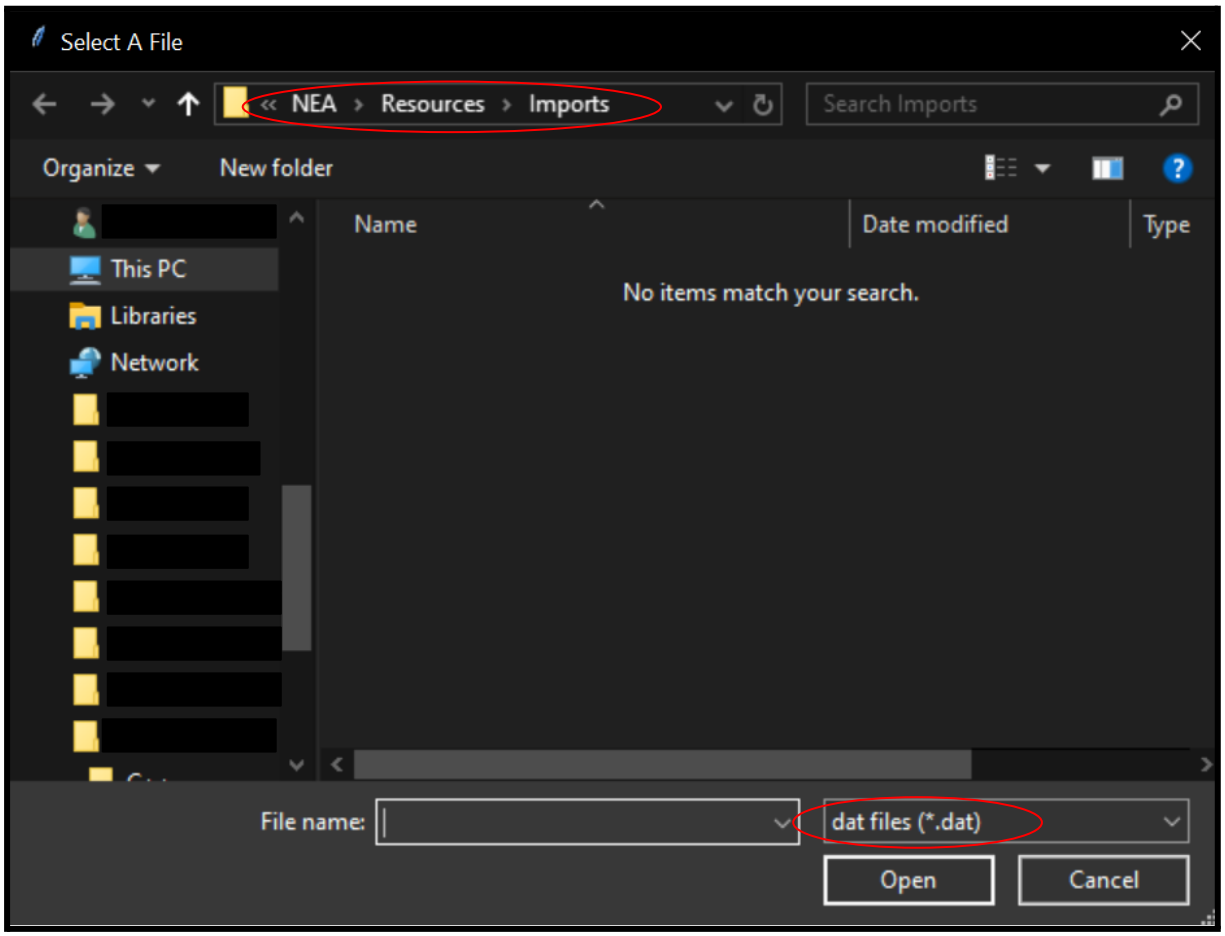


What the menu looks like in theme 2

What the exporting file image looks like



When importing a file, this is what the popup will look like



Testing

No errors have been encountered with the settings menu. However with import/exporting graphs, if the user does not select a file then the program will crash. Another error that has occurred is that if the user uploads a random .dat file that did not come from the program the program will also crash because the data would have not been organised in the same way.

These are the only errors I found though so overall there have not been many problems. In addition I added the back button that I was needing from the last version.

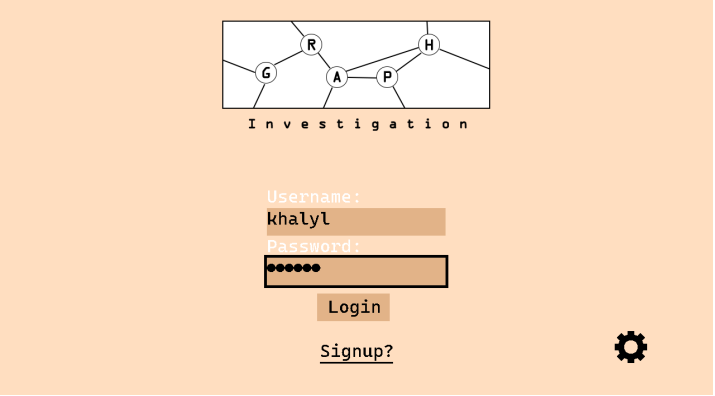
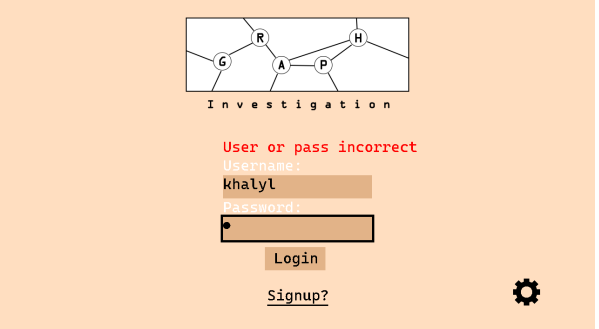
Conclusion

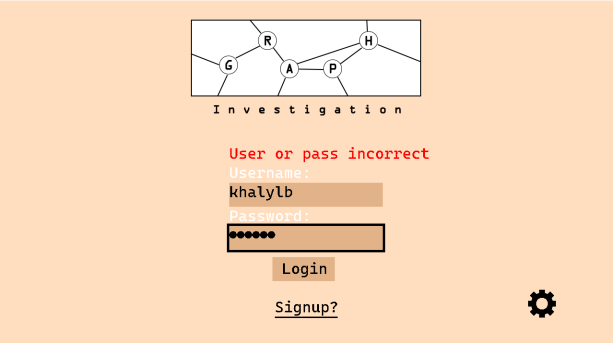
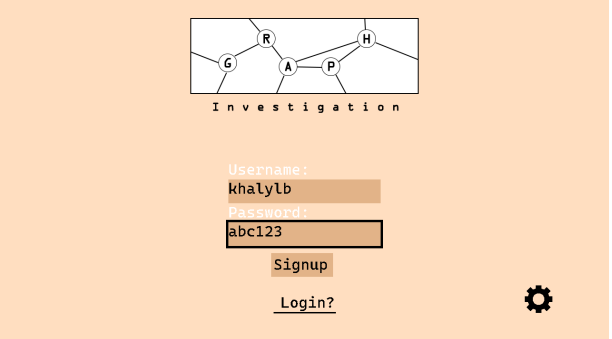
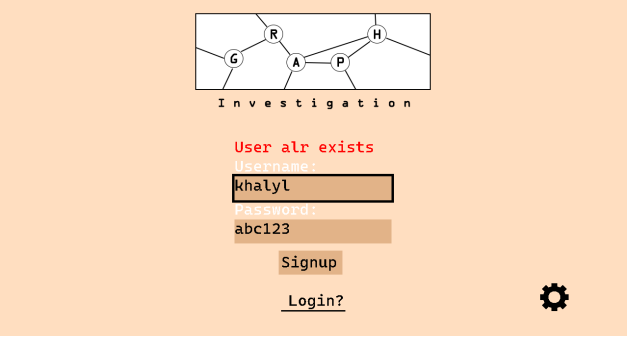
This is the final version of my project so this is my project in its completed form. There are still a few bugs that i will need to sort out to make this program run smoothly

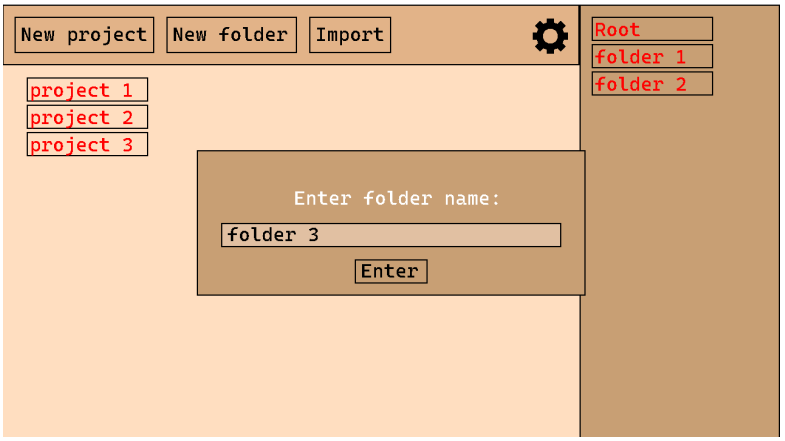
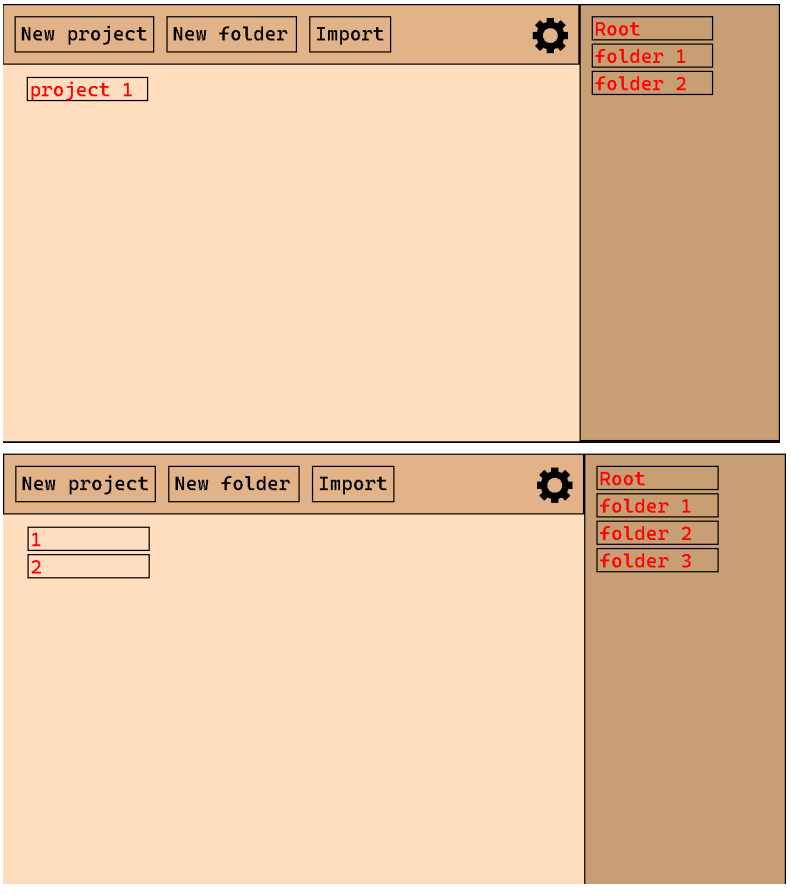
Testing

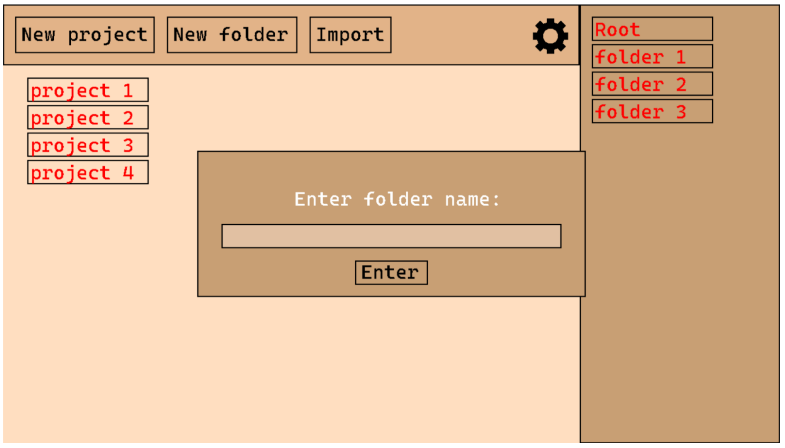
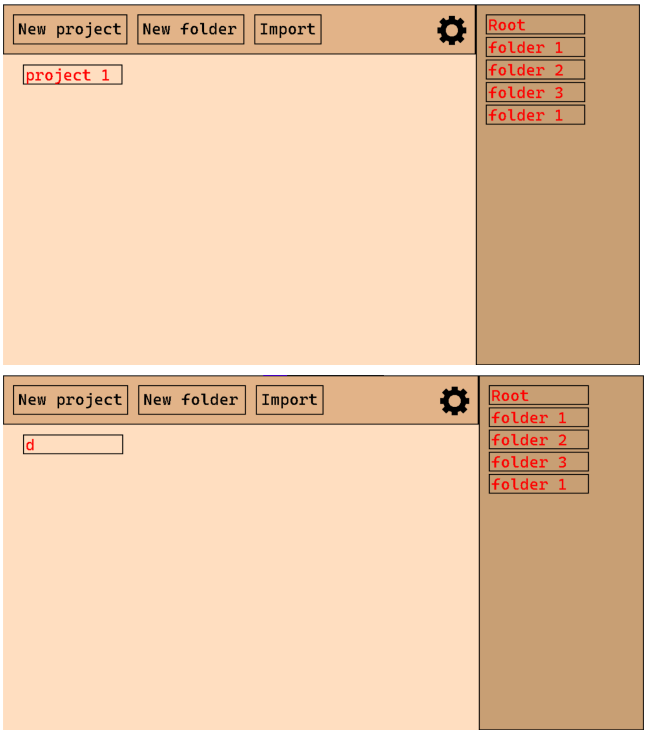
Test plan

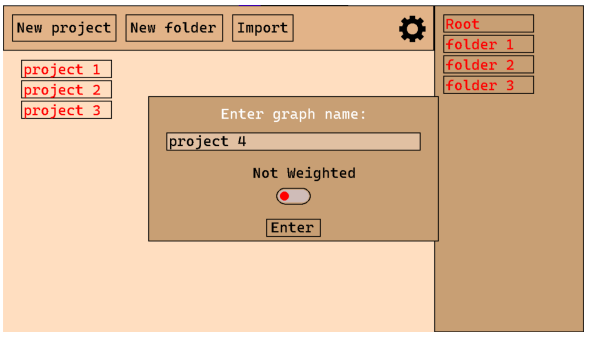
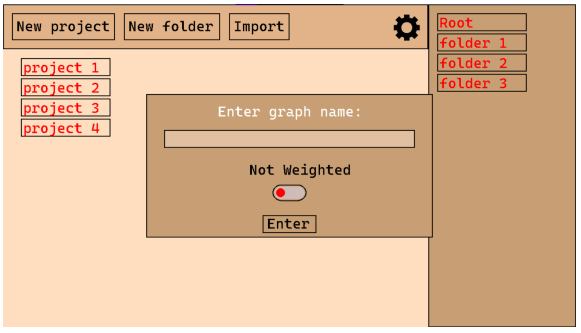
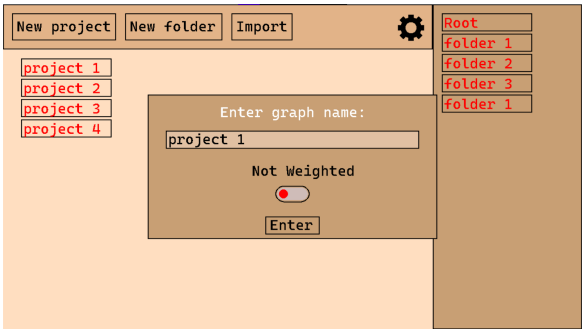
My plan for testing my algorithm to make sure that it works as intended and that there are no errors with the main program so that it runs smoothly. I have included a video on how the program functions: https://youtu.be/kLyesPmy3lA

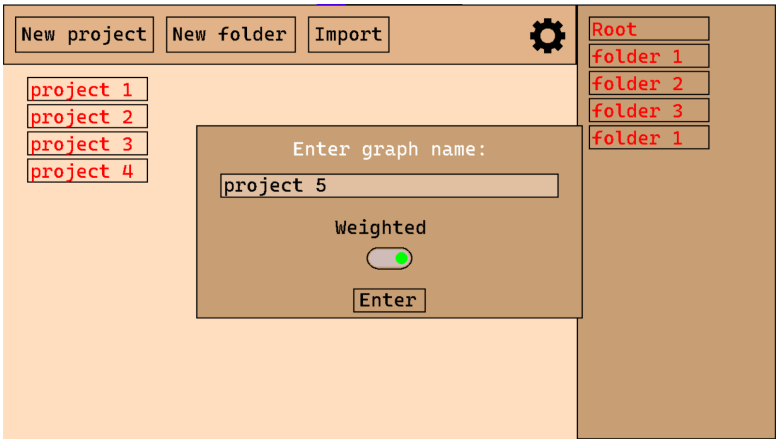
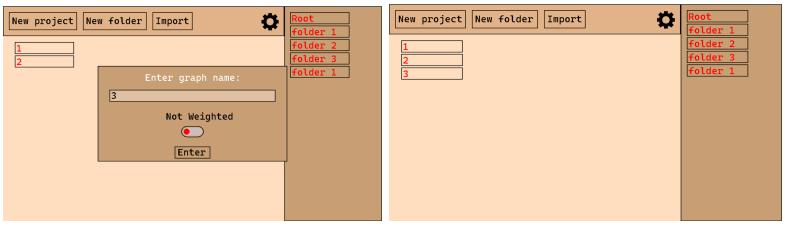
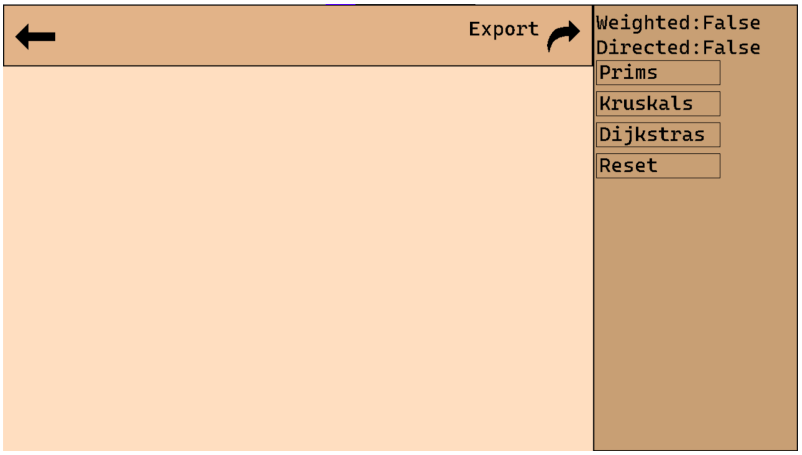
<table>
<tr><th colspan="2">Test No.</th><th>Purpose of Test</th><th>Data Type</th><th>Expected Outcome</th><th>Actual Outcome</th></tr>
<tr><td rowspan="4">1</td><td>1.1</td><td>Make sure that the user is able to log into an existing account if the credentials are correct</td><td>Normal</td><td>Allows the user to log in and directs them to the menu page where they can see their graphs</td><td>As expected</td></tr>
<tr><td colspan="2">The text box that the user is currently selecting is highlighted in black so the user knows if they're entering the username or password. It also blurs out the password for security reasons</td><td colspan="3"> 
</td></tr>
<tr><td>1.2</td><td>Make sure that the user is unable to log in if the password is wrong</td><td>Erroneous</td><td>Prevents the user from logging in and notifies them the user or pass is wrong</td><td>The program tells the user but the text quickly flashes so the user may not have time to see it</td></tr>
<tr><td colspan="2">Text flashes onto the screen notifying the user that they have entered the username or password incorrectly</td><td colspan="3"> 
</td></tr>
</table>

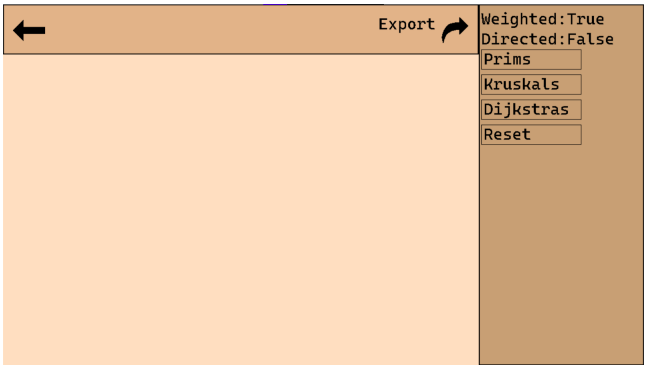
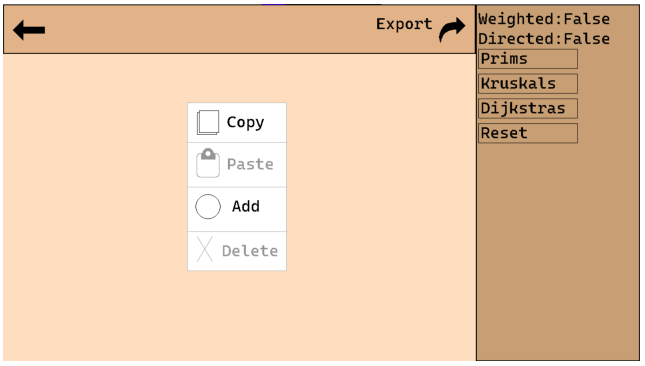
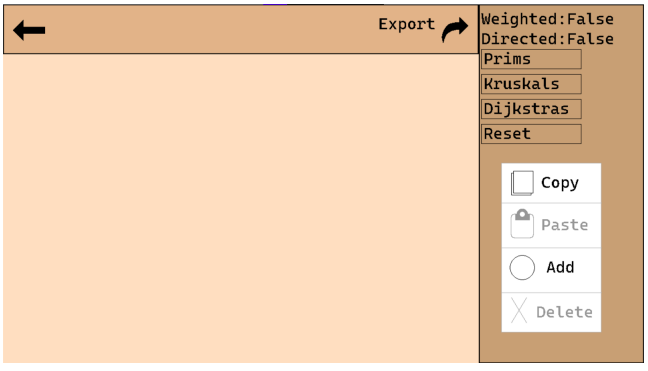
	1.3	Make sure that user is unable to log in if the user does not exist	Erroneous	Prevents the user from logging in and notifies them the user or pass is wrong	The program tells the user but the text quickly flashes so the user may not have time to see it
	The text flashes correctly like the one above				
2	2.1	Make sure that the user is able to create a new account which has a unique username	Normal	New account created in the database	As expected
	The password is shown this time instead of being blurred so that the user can make sure there's no typo in the password they have put in.				
	2.2	Make sure that the user is unable to create an account is the username already exists	Erroneous	It should flash a message telling the user that that username already exists	Once again the text flashes too quickly so the user might not know what it said
	Flashes on the screen that this user already exists and that they are unable to proceed to the menu page				

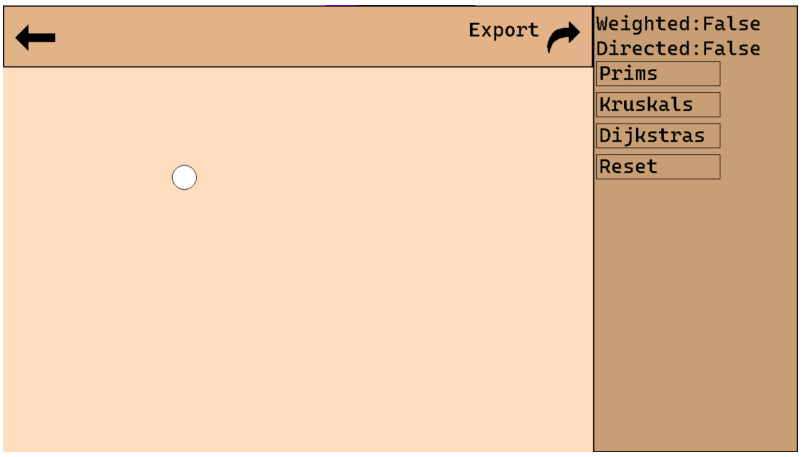
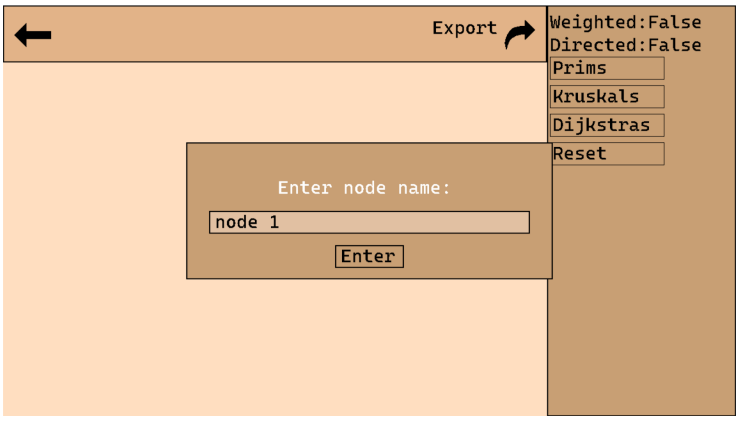
3	3.1	Allow the user to create a folder in their account name	Normal	A new folder should appear in the menu	As expected
	The popup for the user to create a new folder. Clicking outside the box disables this popup				
	3.2	Allow the user to navigate between the folders	Normal	The user should see the graphs in the new folder display on screen	As expect but it is sometimes unclear is the folders are empty so a heading is probably needed
	The graphs displayed are different because the user has switched from folder 1 to folder 3				

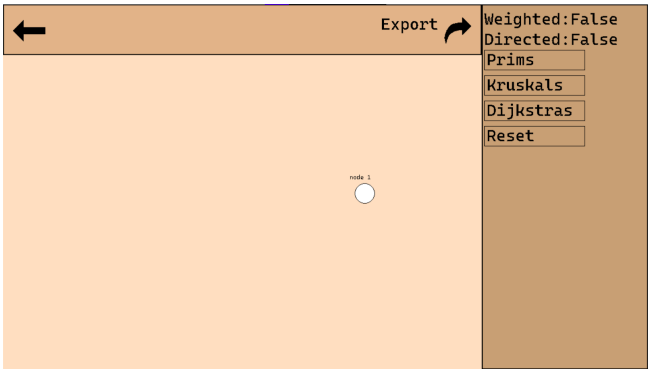
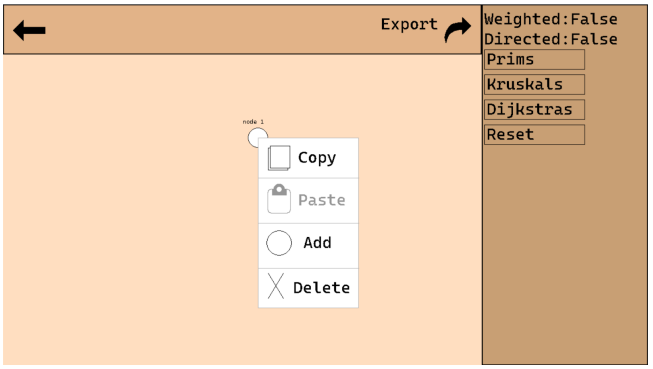
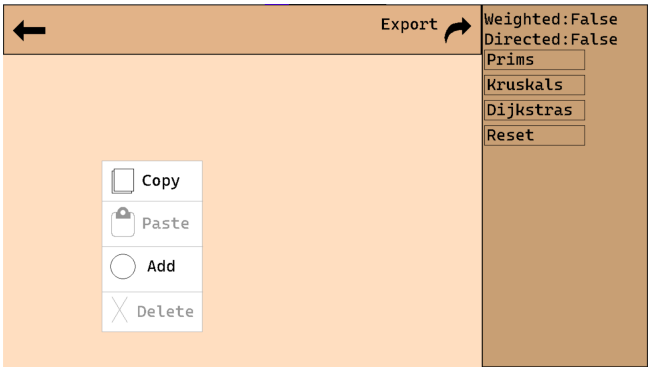
	3.3	Make sure the user enters a name for the folder	Erroneous	The user should not be able to click the enter button and a popup should tell the user to enter a name	The user cannot click the button but the popup doesn't work
	The user is clicking the enter button yet it is not working.				
	3.4	Make sure the user doesn't enter duplicate names for them	Erroneous	The user should not be able to click the enter button and a popup should tell the user to not enter a duplicate name	The user can create a new folder with the same name although the graphs do not get duplicated over.
	The folder "folder 1" already exists so the user should not be able to open a duplicate folder. In this case they can however the graphs inside these folders are different.				

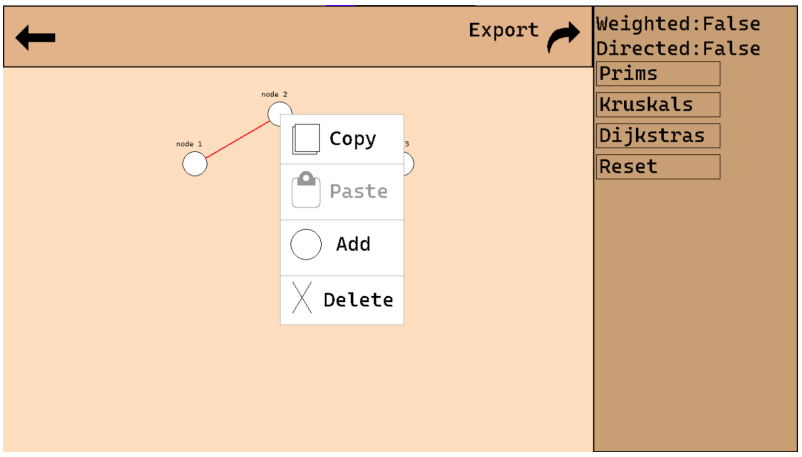
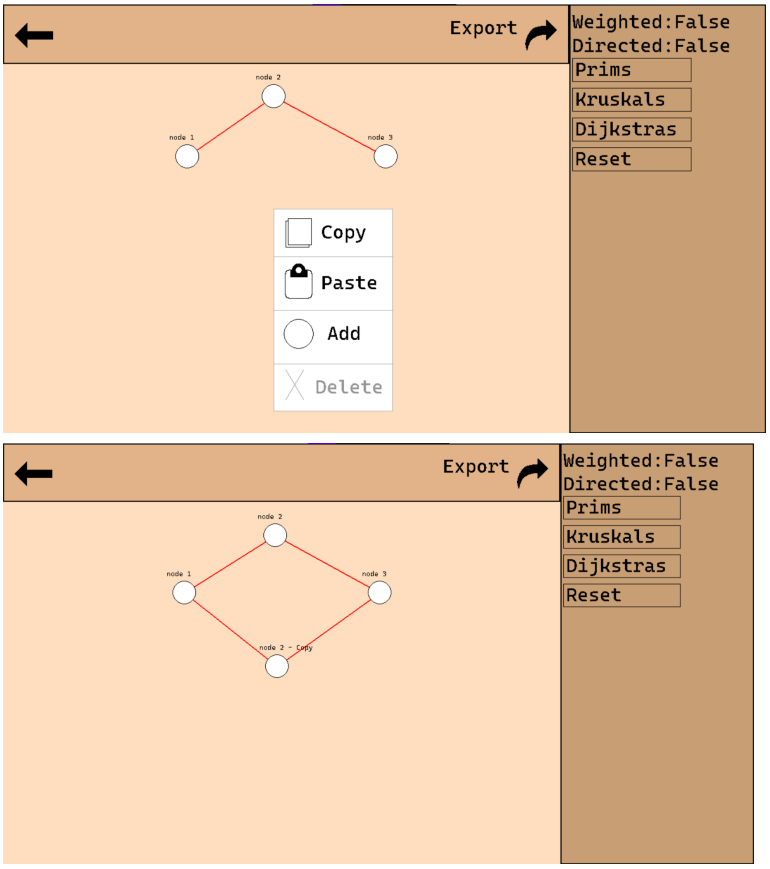
4	4.1	Allow the user to create a new graph and input its name.	Normal	A new graph should appear on screen as a selectable option	As expected except that the button is slightly small so the text spills over
	Popup appears where the user can enter the name. Clicking outside the popup disables the popup				
	4.2	Make sure that the user does enter no name	Erroneous	The enter button should not work and should notify the user that they need to enter a name	The user cannot click the button but the popup doesn't work
	The user is clicking to try and create a new graph but there is no string so the user is unable to proceed.				
	4.3	Make sure the user does not enter a duplicate name for the graph	Erroneous	The enter button should not work and the user should be notified that this name already exists in this folder	The user cannot click the button but the popup doesn't work
	The program is unwilling to accept a duplicate name in the same folder and will not allow the user to enter this graph name				

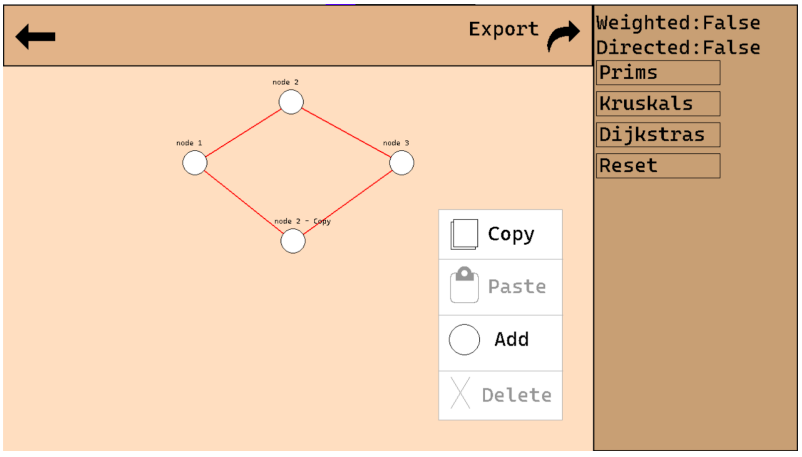
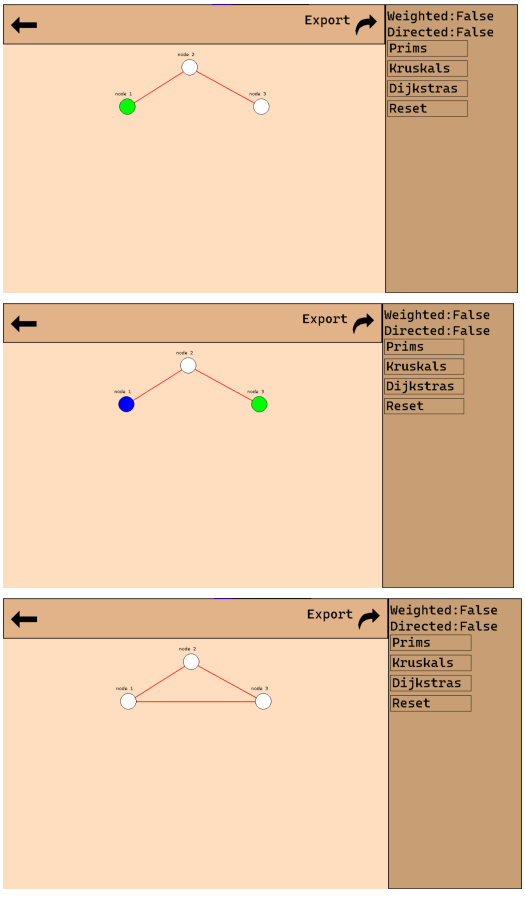
	4.4	Allow the user to select whether the graph it is weight or not	Normal	A toggleable button should appear on the popup	As expected
	The weighted toggle button is below the text box to allow the user to click it and toggle the weight on and off				
	4.5	Allow the user to specify which folder to store their graph in	Normal	The graph should appear in the desired folder	As expected
	The user is able to specify which folder to enter the new graph				
5	5.1	Create a GUI when the user selects a graph to edit	Normal	The GUI appears when the user clicks on a graph	As expected
	Includes buttons to send the user back to the menu page as well as a sidebar				

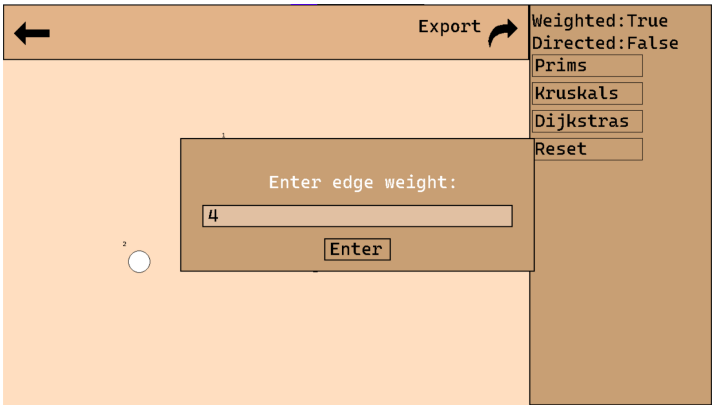
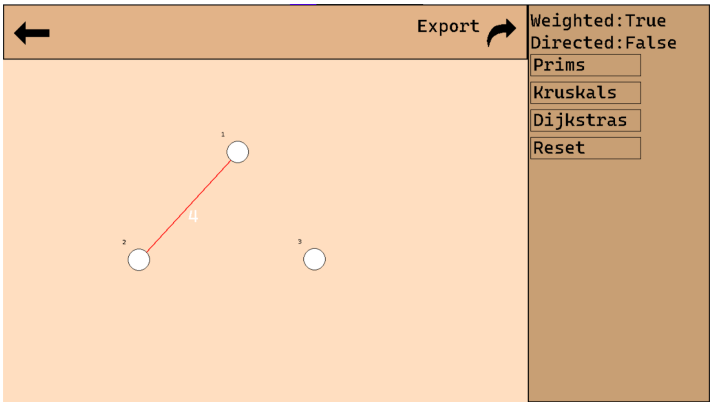
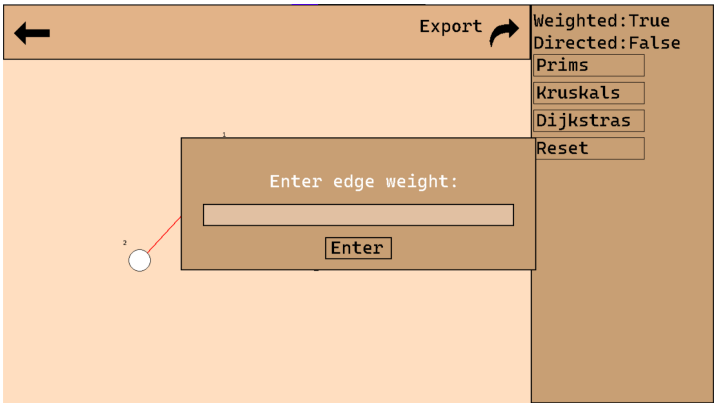
	5.2	GUI displays information on the graph such as if it is weight or not	Normal	Information is displayed on the sidebar	As expected
	Tells the user whether their graph is weight for directed				
6	6.1	Test that the right click menu appears when the mouse is on the canvas	Normal	The right click menu opens	As expected
	The user is right clicking on the canvas which opens up the menu. Some options are greyed out since they are unable to be carried out at the moment				
	6.2	Test that the right click menu does not appear when on the canvas	Erroneous	Nothing happens	Does not work at all. Requires boundary conditions for it to work
	The user is right clicking on the sidebar and the right click menu is not opening				

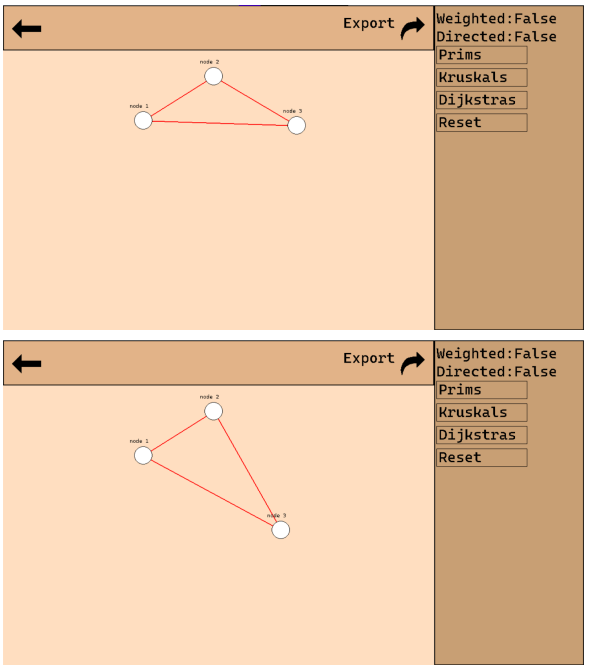
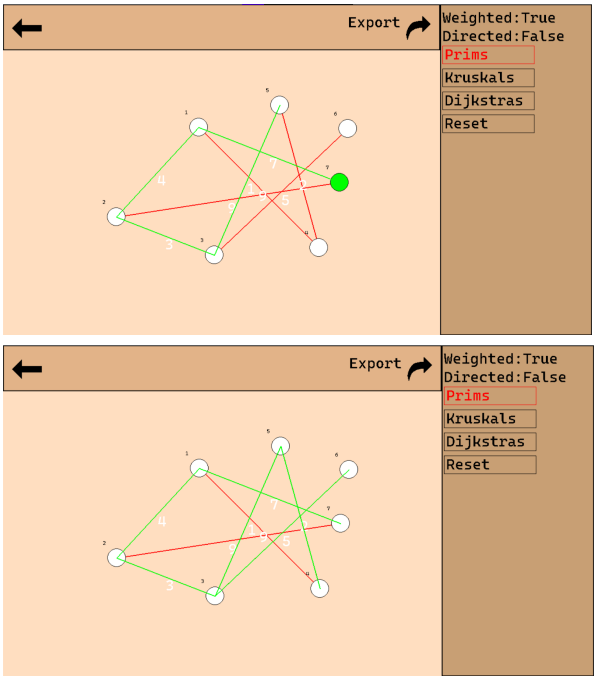
7	7.1	Make sure that the user is able to create a node and for it to display on the screen	Normal	A new node is displayed onto the screen	As expected
	The user is right clicking and creating a new node by clicking the "Add node" button				
	7.2	Allows the user to enter the name of the node in the popup	Normal	The popup appears asking for the users input	As expected
	The user can enter the name and click enter for the popup to disappear and for the node to be displayed onto the screen		 		

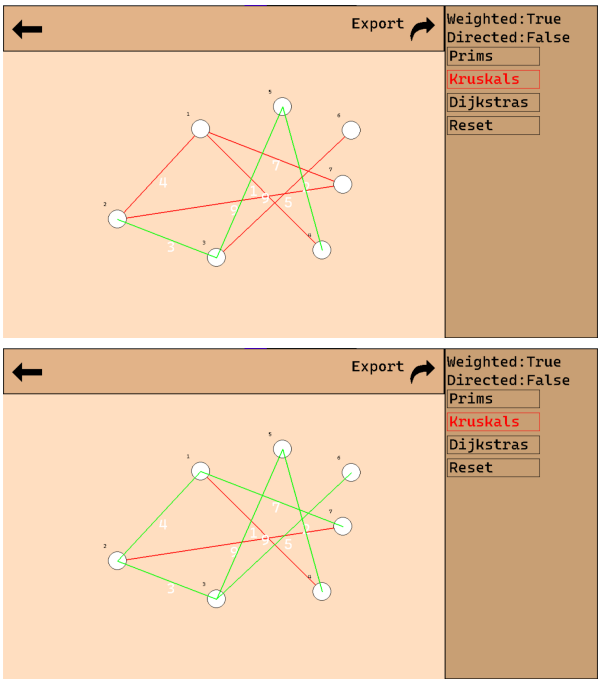
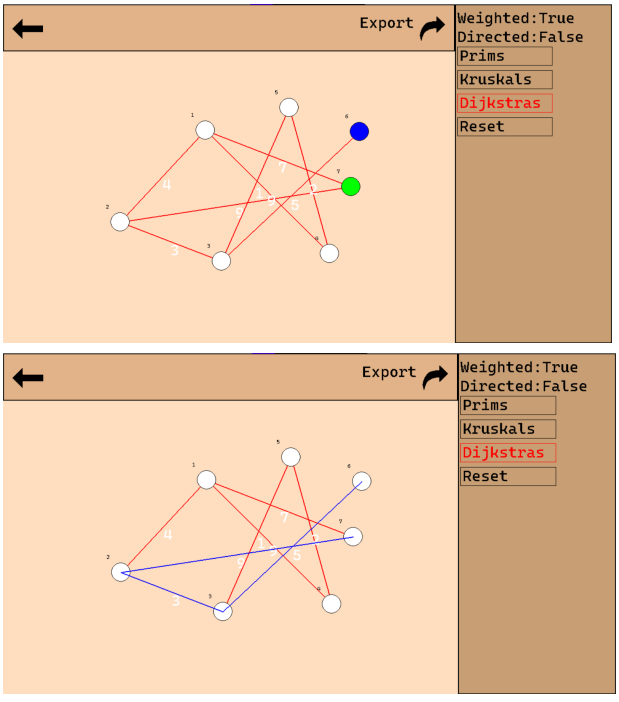
	7.3	Make sure the user is able to move around the nodes on the canvas	Normal	The user can hold down on a node and drag it across the screen	As expected
	The user is holding on the node and moving it around to wherever they want				
8	8.1	Makes sure that the user is able to delete nodes and that they are displayed	Normal	The node selected is removed from the screen	As expected
	The user hovers over a node and right clicks on it and deletes it. The button is now highlighted black since it is possible to perform that action				
	8.2	The delete button is greyed out if the user is not hovering over a node	Erroneous	Nothing happens	As expected
	The user is hovering over an empty place in the canvas and right clicking shows that the delete button is greyed out so the action cannot be performed				

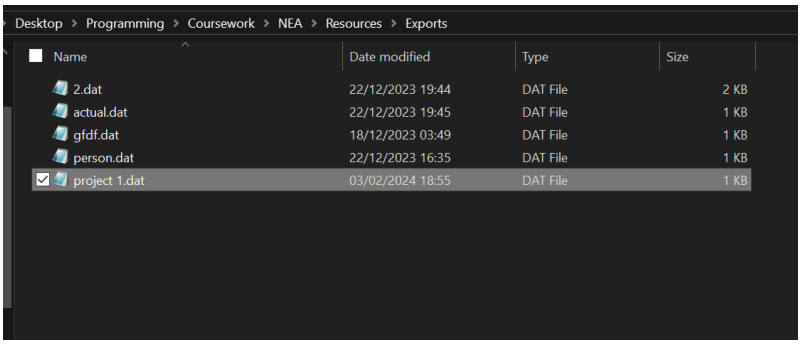
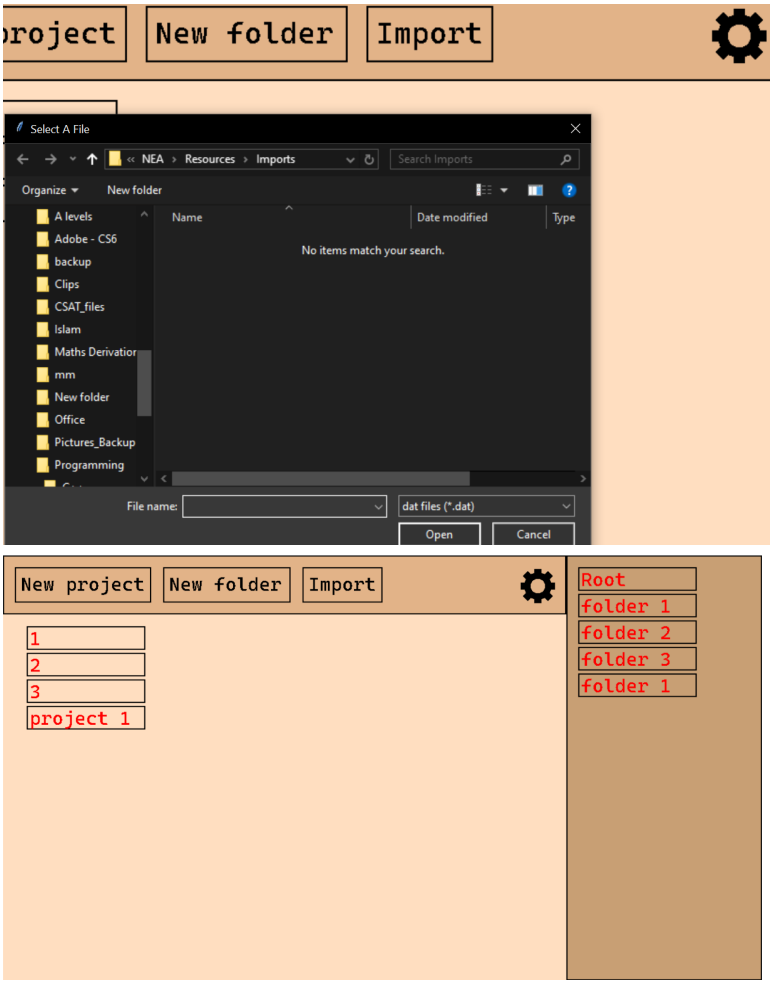
9	9.1	Makes sure that the user is able to copy a node	Normal	The node is copied and added to the clipboard	As expected
	The user right clicks on a node which enables them to click copy to copy a node				
	9.2	Makes sure that the user is able to paste a node if there is one stored in the clipboard	Normal	The node in the clipboard is pasted onto the screen along with the edges connected to it	As expected
	The user pastes the node along with all its connections by clicking on the paste button in the right click menu. It is highlighted since there is a node in the clipboard				

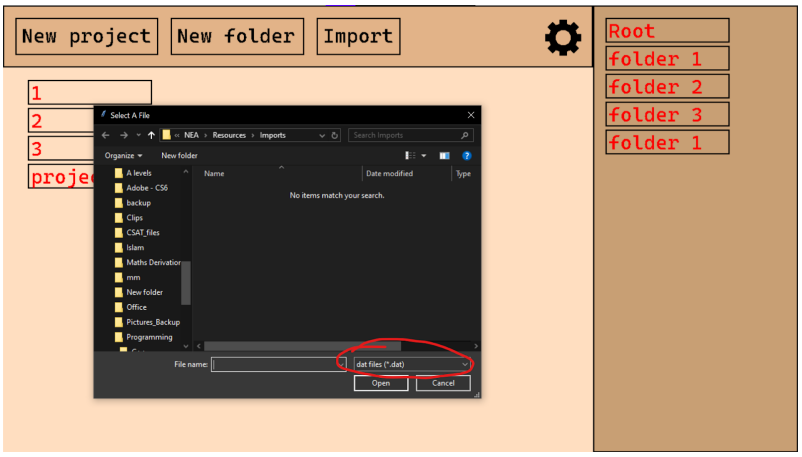
	9.3	Makes sure that if there is nothing in the clipboard the user is unable	Erroneous	Nothing is pasted because there is nothing in the clipboard	As expected
	The user can see the paste button in the right click menu is greyed out because there is no node in the clipboard for them to paste				
10	10.1	Allows the user to create edges and join 2 nodes together	Normal	A new edge is added onto the graph connecting the 2 nodes	As expected
	The user shift clicks to select the first node and while holding shift they click on the 2nd which creates an edge				

	10.2	If the graph is weighted the user is able to enter the edge weight	Normal	A popup appears asking the user to enter the weight of the node	As expected
	A popup opens when the user has created an edge between the 2 nodes asking for a number to associate with the edge		 		
	10.3	Makes sure that the user cannot enter a string for the weight	Erroneous	The program will not show a string if they have entered a character	As expected
	The user is trying to enter a string into the textbox but the program is rejecting this				

	10.4	If the user moves the nodes around the edges move as well	Normal	As the user moves around the nodes the edges move around with them	As expected
	The user is moving around the nodes as mentioned in 7.3 and the edges are moving around with them				
11	11.1	Test Prim's algorithm works on the created graph	Normal	Prim's algorithm displays onto the graph	As expected
	The user holds shift and clicks on a node to initiate Prim's algorithm on the starting node				

	11.2	Test Kruskal's algorithm works on the created graph	Normal	Kruskal's algorithm displays onto the graph	As expected
	The user presses shift to initiate Kruskal's algorithm				
	11.3	Test Dijkstra's algorithm only works if the graph is weighted	Normal	Dijkstra's Algorithm displays onto the graph	As expected
	The user holds shift and clicks on the 2 nodes that they want to select to find the path between them				

<table>
<tr><td rowspan="4">12</td><td>12.1</td><td>Check to ensure that the user is able to export graphs</td><td>Normal</td><td>Clicking the export button sends it to the exports folder</td><td>As expected</td></tr>
<tr><td colspan="3">The user clicks export so the graph they're on is sent to their hard drives folder called exports</td><td colspan="2"> 
</td></tr>
<tr><td>12.2</td><td>Check to ensure that the user is able to import graphs from the menu page</td><td>Normal</td><td>Clicking the import button opens up the choice to select which file to get</td><td>As expected</td></tr>
<tr><td colspan="3">Windows explorer opens up so that the user can look for the file they want</td><td colspan="2"> 
</td></tr>
</table>

	12.3	Check to make sure that the user is only able to import .dat files	Normal	Only displays .dat files so that they are the only ones that are able to be imported	As expected
	Only selected flies are to be shown so that crashes do not happen. It is shown in the bottom right where only .dat files are shown				

Corrections

1.2 and 1.3

By adding a time.sleep(0.3) I am able to delay the progression of the program and therefore the text will display for longer meaning that this time it won't just flash but instead will remain on the screen for a period of time and then go off. The use of pygame.display.update() is so that the screen is updated before it waits otherwise it will only display after the wait and therefore nothing will have happened.

```
627                 #checks to see if the credentials are correct
628                 val = login(user_string,pass_string)
629                 #If not it will tell the user that the username or apss is wrong
630                 if not val:
631                     win.blit(font.render("User or pass incorrect",0,(255,0,0)),(480,300))
632                     pygame.display.update()
633                     time.sleep(0.3)
634
```

2.2

This problem was very similar to the issue in 1.2 and 1.3 so i have fixed it in the same way so that the user is able to read the text before it disappears

```
640             elif page == "signup":
641                 #It checks to see if a user with that username alr exists
642                 val = signup(user_string,pass_string)
643                 if val:
644                     win.blit(font.render("User alr exists",0,(255,0,0)),(480,300))
645                     pygame.display.update()
646                     time.sleep(0.3)
647
```

3.2

This just draws a red rectangle over the box of the current folder that is being edited. This means that the user will be able to know which folder they are in.

```
        pygame.draw.rect(win,(0,0,0),(970,20+a*45,200,40),2)
        win.blit(font.render(f"{i[1]}",0,(255,0,0)),(975,22+a*45))

    pygame.draw.rect(win,(255,0,0),(970,20+current_folder_id*45,200,40),2)
```

3.3

This is very similar to 1.2 and 1.3 where it will make sure the text won't flash when it is displayed and will remain long enough for the user to be able to see it

```
    else:
        win.blit(font.render("Please enter a name",0,(255,0,0)),(480,150))
        pygame.display.update()
        time.sleep(0.3)
```

3.4

This part was a logic error where previously I was comparing current_string with i[0] which is the id of the folder instead of the name of the folder.

```
    #Check to see if there is a duplicate
    alr = False
    for i in check:
        if current_string == i[1]:
            alr = True
```

At this part i just added the same fix to it similar to 1.2 and 1.3

```
    else:
        win.blit(font.render("Folder already exists",0,(255,0,0)),(480,150))
        pygame.display.update()
        time.sleep(0.3)
    conn.commit()
```

4.1

This was a simple fix where I needed to increase the length of the box so I changed the length from 200 to 600.

```
    #Draws all the graphs that are currently int the folder
    for a,i in enumerate(graph_list):
        pygame.draw.rect(win,(0,0,0),(40,120+a*45,600,40),2)
        win.blit(font.render(f"{i[0]}",0,(255,0,0)),(45,122+a*45))
```

4.2 and 4.3

This is the same fix in 1.2 and 1.3

```
    else:
        win.blit(font.render("Please enter a name",0,(255,0,0)),(480,150))
        pygame.display.update()
        time.sleep(0.3)
```

```
    else:
        win.blit(font.render("Name already exists",0,(255,0,0)),(480,150))
        pygame.display.update()
        time.sleep(0.3)
```

6.2
I needed to add the boundary conditions for the right click menu to activate. Simple fix required. Just making sure that the coordinates of mouse must be encapsulated in the region designated fo the menu to popup

```
    #if they right click tho then it just changes the corrdinates of the right click menu
    elif pygame.mouse.get_pressed()[2] and mousex >= 0 and mousex <= 950 and mousey >= 100 and mousey <= 720:
        tempx,tempy=mousex,mousey
#if the menu isnt active
```

Evaluation

End User Feedback

I interviewed several further maths students about my program and their opinions on it. This is so that I can gain an understanding of what my program accomplishes and whether I have met my end goal.

What do you like about my program?	
"I like the customizability of the program and how it gives a lot of freedom to the user" "It is a very straightforward program to use" "Helpful to understand how the algorithms described work as it show a visual representation of what each algorithm does" "GUI is very clean" "I like the export and import system which is something very unique about your program as it allows me to share my graphs with other people which can be useful if my friends need help with a problem." "Very fluid movement when moving around the nodes as they move seamlessly and there are no bugs with the edges when you move the nodes around" "I really like the right click menu since it makes me feel like i'm running an advanced software"	The majority of test users like the GUI and how easy the program was to use. This shows that I have been able to complete objective 12. In addition the import export system was a component i added to differentiate myself from other similar program which i believe i have successfully accomplished

What would you change about my program?	
"Creating edges is a little bit buggy since sometimes it doesn't let me click on the node that i want to and i end up having to deselect and reselect the node that i want" "Some parts look unrefined and could do with a polishing such as the settings menu with the themes as there are not boxes around the buttons" "I would like there to be an option to make your graphs directed since it tells the user that their graph is not directed but there isn't an option to make it directed" "Make a display for the adjacency matrix to that the user can see that as well so that the user can check to see if the graph they have created is the one they want" "Add more themes" "Make it so multiple people can work on a single project at the same time" "You can't log out"	The majority of the criticisms of the program is that it is unrefined and could do with polishing such as adding a logout button or making sure connecting nodes is cleaner. This could be simply rectified by combing through the program and making small changes. A major change people would make about my program is the addition of directed graphs which is a feature that was slightly too complicated for me to implement in the time that was allotted. However with some fine tuning it seems possible to amend these changes.

Comparison to initial objectives

No.	Objective	Commentary
1	My project must be able to have a GUI system for moving nodes around the screen with the mouse. It needs to only select 1 node when they overlap and not multiple so that they don't group together.	This objective has been completed since i have been able to create a GUI which allows the user to move around nodes on the screen without having any bugs or glitches
2	My project must also be able to create edges from nodes when holding down the click while the mouse is on the node and then they can drag it to another node to connect it. It will be a straight line from node to node and if the user does not connect to another node then as they let go it will just delete the edge.	This objective has been met since the user is able to join 2 nodes together by creating an edge between them. However the execution of this feature is different to the plan since now they shift click 2 nodes instead of holding to join them together
3	My project will also have a right click system on the graph editor page where the user can right click to have options such as create a new node or delete a node or copy and paste. This will be similar to microsoft word or powerpoint and how when they right click there is a menu on there. I want it to also deselect if they click off it. In addition if they don't select anything then the right click menu will have certain buttons greyed out as it will not be able to execute those functions.	This objective has been completely met where the user will be able to right click on the canvas to open up the right click menu. All of the buttons on the right click menu work as they have been tested. In addition certain buttons which should not work under specific conditions are greyed out which means the latter part of this objective has been met
4	My project must also have a menu selection screen so that users will be able to select whether to log in or to sign up to the system. The menu will be able to switch from login to sign up using a button similar to a google login system.	This objective has also been met and tested rigorously to make sure no errors occur in the database although some parts of this system were not user friendly so they had to be rectified in the final version
5	My project will also have a login system using databases. It will consist of 3 tables in which it will store account information, graph information and folder information. The graph and folder information will consist of lists so they will have to be pickled to store them as a string in a database. The graphs will be stored as an adjacency matrix and the folders will be stored as a tree since it will be a hierarchy.	This objective has also been met as I have had to create 3 tables in the database so that it can store the accounts, folders and graphs. The login system works flawlessly and the pickled matrices work well without any problems

6	My project should also be able to execute Prims and Kruskal's algorithms on a matrix as they are MST type algorithms.	The MST algorithms are highlighted in green so that the user knows and these work very well and instruct the user well on how each of the algorithms work.
7	Another type of algorithm I will be using are pathfinding algorithms and the one I will use is Dijkstra's which will find the shortest between 2 selected nodes on the graph	Dijkstra's algorithm is highlighted in blue instead since it is a different type of algorithm so the user can see the distinction between them
8	I also want my algorithm to have a menu theme so that if a user is not happy with the colour scheme then they can change the accent colours or the background colours.	The menu has been successfully implemented and the user is able to choose between dark and light theme so this objective has been completed
9	I also need to add selections of different graphs from the database which will include information about them and have them on display. It will be useful to see what kind of graphs they are.	The user is able to set the graph to be weighted or not and on the sidebar in the canvas it tells the user whether they a weighted/directed graph
10	I want to add an export and import system where the user can import a graph or export a graph. It would be preferable to open it as a xml file although a .dat file may be used instead for simplicity.	This objective has been fulfilled since the user is able to click export to export any of the graphs to the exports folder so they can externally share them
11	I want to make the GUI user friendly so that my program is as easy to use as possible. The way I will achieve this is by explicitly stating how the user should be able to create a graph.	This has been achieved based on the feedback that i have received since the majority of users said it is a clean GUI

Potential improvements

My program is a very well put together piece of software and it has received praise from the test users however it is not perfect. Some of the users would have preferred different aspects of the program to be changed so this is a compilation of potential improvements:

- Improve the method of joining 2 nodes together (creating an edge) by making it more consistent.
- Add a zoom function since some graphs may be too big so view on a set screen size so i could add a method of zooming in and out of the canvas
- I could add other algorithms such as Kosaraju's algorithm to further develop the program
- I could add a matrix view so the user can see the matrix associated with their graph they've created
- I could also add a mode in which the program asks the user exam questions so that they can solve them using the program and understand how they work
- Make the interface look more complete by polishing rough edges such as adding boxes over some text to make it stand out more as a button
- Add directed graphs so that the user can understand unidirectional graphs. This would come in useful with Kosaraju's algorithm.
- Add a logout button so that the users are able to switch between each others account on the same machine without having to close the program and reopen it

None of these improvements affect a major aspect of my program and do not prevent the program from meeting its objectives however these are some ideas that i would consider adding if i am to further this project or rework the project

Conclusion

Overall my project has satisfied the objectives i had set for it which means i have accomplished my goal. I believe this project could be a great aid to students and teachers in order to help teach or learn about graph theory. I believe it will help understand an abstract topic such as this. In addition I believe the time it takes to create a graph on my program is much less than drawing it out and performing the algorithms on a sheet of paper.

A major change that I would make if I were to program this again is to make the program able to operate over the internet so that multiple users will be able to modify the same graphs at the same time given that they have been shared with the host user. This would be beneficial for remote learning as the teacher will be able to demonstrate the algorithms visually coherently across different places which could help to improve online learning for students.

There are also some minor changes I would like to make in order to improve the cohesion of the program. These improvements would not affect the program in a big way.

Ultimately my program achieves my goal of aiding students and teachers with understanding the topic of graph theory.

References

https://favtutor.com/blogs/prims-algorithm-python
https://favtutor.com/blogs/dijkstras-algorithm-cpp
https://csacademy.com/app/graph_editor/
https://algo-dijkstra.vercel.app/

Appendix

Main.py

```
#Python library imports
import pygame,math,sqlite3,time,pickle
from tkinter import filedialog


#Connection to the database
conn = sqlite3.connect('Resources/SaveData/SaveData.db')
#Creates a cursor for selecting items in a database
curs = conn.cursor()
#Will attempt to create a database if there isnt one already,
otherwise nothing will happen
try:
      curs.execute("""CREATE TABLE Account_info (
                  Username text,
                  Password text
                  )""")


      curs.execute("""CREATE TABLE Folder_info (
                  Folder_id int,
                  Name text,
                  Account_id int
                  )""")
```

```
      curs.execute("""CREATE TABLE Graph_info (
                  Name text,
                  Weighted bool,
                  Directed bool,
                  Graph text,
                  List text,
                  Account_id int,
                  Folder_id int,
                  Graph_id int
                  )""")

except:pass

#Save the changes
conn.commit()
#Closes the connection
conn.close()

#This variable decides the theme of the app
theme = 0

#These are the possible themes that the user can select when on
the app
colour1 = [(255,222,192),(49,51,56)]
colour2 = [(226,179,136),(32,38,43)]
colour3 = [(207,188,184),(85,98,102)]
colour4 = [(200,159,116),(43,45,49)]
colour5 = [(225,192,162),(35,36,40)]



#This class is for the nodes which allows them to be created
class node:
      #This is the constructor
      def __init__(self,name,pos,x,y,neighbours):
            #Name of the node
            self.name = name
            #Position of the node (row and column)
            self.pos = pos
            #x coord
            self.x = x
            #y coord
            self.y = y
            #Neighbours of the node
            self.neighbours = neighbours
      #These methods are used for encapsulation and they allow the
program to only be able to access the attributes through methods
      def get(self,var):
            if var == "name":
                  return self.name
            if var == "pos":
                  return self.pos
```

```
            if var == "x":
                  return self.x
            if var == "y":
                  return self.y
            if var == "neighbours":
                  return self.neighbours

      def set(self,var,val):
            if var == "name":
                  self.name = val
            if var == "pos":
                  self.pos = val
            if var == "x":
                  self.x = val
            if var == "y":
                  self.y = val
            if var == "neighbours":
                  self.neighbours = val
      #This updates the coordinates of any nodes
      def update_coords(self,nodes):
            for i in nodes:
                  if i.name == self.name:
                        self.x = i.x
                        self.y = i.y

#This function finds the highest value in the matrix
def highest_algorithm(matrix):
      #The default highest value will always be 0
      highest = 0
      #It will then cycle through each value in the matrix that was
passed and check if the value of the edge is higher than the
highest
      for i in matrix:
            for j in i:
                  #The need for the try function is that a lack of
an edge between 2 nodes is represented as a '-' which is a string
                  #This means u cannot do mathematical operations to
it and therefore if it is encountered it will just be ignored
                  try:
                        if j > highest:
                              #If a new highest is found then it will
set that value as the new highest
                              highest = j
                  except:pass

      #just returns the value
      return highest

#This function gets matrix and a list of nodes and sets the edges
in the matrix to the edges in the list
def graph_to_list(matrix,old_nodes):
      #Sets a new empty list to blank
      new_nodes = []
```

```
      #Cycles through all the columns and indecies in the given
matrix
      for a,i in enumerate(matrix):
            #Makes a new empty list of the neighbours, these are
the neighbours of the nodes
            neighbours = []
            #Cycles through all the rows and indecies in the given
matrix
            for b,j in enumerate(i):
                  #Makes sure that there is actually a connection
between the 2 nodes
                  if j != "-":
                        #Adds to the list of neighbours another list
containing the name of the node and its weight
                        neighbours.append([old_nodes[b].name,j])
            #Adds the nodes to the new nodes and give it the
neighbours

new_nodes.append(node(old_nodes[a].name,old_nodes[a].pos,old_nodes
[a].x,old_nodes[a].y,neighbours))

      return new_nodes

#Does the opposite of the previous one where it outputs a matrix
when given a list
def list_to_graph(nodes):
      #Initiates the matrix
      graph = []
      #Cycles through all the columns in the list
      for i in nodes:
            #Creates a temporary column
            temp_list = []
            #Cycles through all the rows in the list
            for j in nodes:
                  #Initally sets no connected between the 2 nodes
                  weight_of_graph = "-"
                  #Cycles through the neighbours of that node
                  for k in i.neighbours:
                        #If it finds that if they are neighbours it
changes the weight to its weight
                        if j.name == k[0]:

                              weight_of_graph = k[1]

                  #Appends the weight of the edge to the column
whether its '-' or an actual connection
                  temp_list.append(weight_of_graph)
            #Finally after the column has had all its values
inputted it adds it to the matrix
            graph.append(temp_list)

      return graph
```

```
#Prims algorithm
def prims_algorithm(highest,matrix,starting_node,nodes):
      #sets the number of nodes to the length of the matrix
      length = len(matrix)
      #creates a 'fresh' matrix
      fresh = []
      #No current nodes are selected
      selected_nodes = []

      starter = 0
      for a,i in enumerate(nodes):
            if i.name == starting_node.name:
                  starter = a

      order = [starter]

      #This for loop just makes the fresh matrix have the value '-'
for all the edges
      for i in range(0,length):
            temp_list = []
            for j in range(0,length):
                  temp_list.append("-")
            fresh.append(temp_list)
            #This part means that all current nodes will be
unselected at the start
            selected_nodes.append(False)

      #This variable is responsible for keeping track of which edge
its currently on
      edge_count = 0
      #This adds the inital node to the selected nodes
      selected_nodes[starter] = True

      #Makes sure every node is cycled through
      while (edge_count < length - 1):

            #Sets the inital minimum weigh of the edge to 1 more
than the highest
            minimum = highest+1
            #Index 1
            a = 0
            #Index 2
            b = 0
            #Cycles through all indecies of the matrix
            for i in range(length):
                  #checks to see if the current node being targetted
is one of the selected nodes
                  if selected_nodes[i]:
                        #Iterates though all the nodes again
                        for j in range(length):
                              #Makes sure that the node being
examined is not the selected node or a node already examined
```

```
                              if ((not selected_nodes[j]) and
matrix[i][j]):
                                    #Uses try once again because you
cant use < and > when dealing with strings
                                    try:
                                          #Checks to see if the edge
weight is the lowest possible
                                          if minimum > matrix[i][j]:
                                                #Sets the new lowest
                                                minimum = matrix[i][j]
                                                a = i
                                                b = j
                                    except:pass
            #Adds the weight of that edge to the fresh matrix
            fresh[a][b] = matrix[a][b]
            order.append(b)
            #adds the new node to the seleted nodes
            selected_nodes[b] = True
            #Increases the edge count so it can iterate over the
next node
            edge_count += 1
      #makes a list out of the matrix when finished
      MST_list = graph_to_list(fresh,nodes)
      ordered_nodes = []
      for i in order:
            ordered_nodes.append(MST_list[i])
      return ordered_nodes


#This small functions finds parent of the node
def find(i,parent):
      while parent[i] != i:
            i = parent[i]
      return i


def kruskals_algorithm(highest,matrix,nodes):
      #Creates an array with the numbers 1 to the number of nodes
      parent = []
      for i in range(0,len(matrix)):
            parent.append(i)

      #Once again creates a fresh matrix
      fresh = []
      for i in range(0,len(matrix)):
            temp = []
            for j in range(0,len(matrix)):
                  temp.append("-")
            fresh.append(temp)

      #Sets each varible to what its parent is
      for i in range(len(matrix)):
            parent[i] = i
```

```
      #The edge count once again
      edge_count = 0
      #Loops untill all edges have been analysed
      while edge_count < len(matrix):
            #minimum value is set to 1 higher than the highest
            minimum = highest+1
            #Index 1
            a = -1
            #Index 2
            b = -1
            #Cycles through the indecies of the nodes
            for i in range(len(matrix)):
                  for j in range(len(matrix)):
                        #Makes sure that there is a connection
                        if matrix[i][j] != '-':
                              #Finds the parents of both nodes and if
they are not the same and the edge weight is smaller than the
minimum
                              if find(i,parent) != find(j,parent) and
matrix[i][j] < minimum:
                                    #Sets the minimum value to the
current matrix position being examiend
                                    minimum = matrix[i][j]
                                    a = i
                                    b = j
            #Finds the parent of the 2nd node and sets the to that
parent of the parent of the 1st
            parent[find(a,parent)] = find(b,parent)
            #Adds the edge to the new 'fresh' matrix
            fresh[a][b] = matrix[a][b]
            #increase count
            edge_count += 1
      #turns the matrix into a list
      for a,i in enumerate(fresh):
            for b,j in enumerate(fresh):
                  if fresh[a][b] != '-':
                        fresh[b][a] = fresh[a][b]
                  if fresh[b][a] != '-':
                        fresh[a][b] = fresh[b][a]
      kruskal_nodes = graph_to_list(fresh,nodes)
      kruskal_nodes = sorted(kruskal_nodes, key=lambda x:
x.neighbours[0][1])
      return kruskal_nodes

def dijkstras_algorithm(matrix, start, end):
      #Sets the distance to each node as infinity at the start
      dist = []
      for i in range(0,len(matrix)):
            dist.append(float('inf'))
      #sets the start node distance to 0
      dist[start] = 0
      #Sets the vitied nodes all to False
```

```
      visited = []
      for i in range(0,len(matrix)):
            visited.append(False)
      #Sets the parents of all the nodes to negative meaning they
have none
      parents = []
      for i in range(0,len(matrix)):
            parents.append(-1)

      #Cycles through all the nodes
      for i in range(len(matrix)):
            #sets the minimum disance and the minimum index
            min_dist = float('inf')
            min_index = -1
            #Checks each node and finds the minimum distance and
smallest index it can be
            for j in range(len(dist)):
                  if dist[j] < min_dist and not visited[j]:
                        min_dist = dist[i]
                        min_index = j

            #Shows the node has been vitited
            visited[min_index] = True
            for j in range(len(matrix)):
                  try:
                        #Checks to see if the node is not visitived
and that the length of the node works
                        #And that that distance to the node is not
infinity and that it is the shortest distance possible
                        if (not visited[j] and matrix[min_index][j]
!= 0 and dist[min_index] != float('inf') and dist[min_index] +
matrix[min_index][j] < dist[j]):
                              #Adds the node to the shortest distance
                              dist[j] = dist[min_index] +
matrix[min_index][j]
                              parents[j] = min_index
                  except:pass
      #Creates an empty array for the path
      path = []
      #sets the current node to the end and backtracks
      current = end
      while current != start:
            #Back trach
            path.append(current)
            current = parents[current]
      path.append(start)

      shortest_path = list(reversed(path))
      return shortest_path


#Initialisation of pygame
pygame.init()
```

```
#Sets the window display size
win = pygame.display.set_mode((1280,720))
#Sets the clock which is the amount of cycles the gameloop will
run per second
clock = pygame.time.Clock()
#The variable which controls the gameloop
run = True
#The standard font that will be used
font = pygame.font.SysFont("Cascadia Code",32)
#A smaller font
small_font = pygame.font.SysFont("Cascadia Code",12)
#Password font
pass_font = pygame.font.Font("Resources/Media/Pass_font.ttf",32)

#images
header = pygame.image.load("Resources/Media/header.png")
subheader = pygame.image.load("Resources/Media/subheader.png")

settings_icon = pygame.image.load("Resources/Media/settings.png")
back_icon = pygame.image.load("Resources/Media/Back.png")
export_icon = pygame.image.load("Resources/Media/share.png")
delete_icon = pygame.image.load("Resources/Media/bin.png")
paste_icon =
[pygame.image.load("Resources/Media/clipboard_off.png"),pygame.ima
ge.load("Resources/Media/clipboard_on.png")]


#This variable will control what the user is currently seeing
page = "login"
#Check key press variable
keys=pygame.key.get_pressed()

#Username of the user
global username
username = False
#Password of the user
global password
password = False
#The current string is whatever the user types into the keyboard
global current_string
current_string = ""
#string of the username
user_string = ""
#string of the password
pass_string = ""

#The current account that the user is on
global current_account_id
current_account_id = 0

#The current folder the user is in
current_folder = 0
```

```
#Checks to see if the text box for inputting a new folder is true
text_box_folder = False
#Checks to see if the text box for inputting a new graph is true
text_box_graph = False



#Function for typing into the program
def type(key,page):
      global current_string,username,password
      #if the key is backspace then it removes the last character
from the string
      if key.key == pygame.K_BACKSPACE:
            current_string = current_string[:-1]
      #If the key is tab it moves it from username -> password ->
free in that order
      elif key.key == pygame.K_TAB:
            if page == "login" or page == "signup":
                  if username:
                        username = False
                        password = True
                        current_string = pass_string
                  elif password:
                        username = False
                        password = False
                  elif not username and not password:
                        username = True
                        password = False
                        current_string = user_string
      #Otherwise it adds it to the curent string
      else:
            current_string += event.unicode

#Hashing algorithm for storing passwords
def hashs(plaintext):

      hashed = plaintext

      return hashed

#If the user chooses to log in
def login(user,passs):
      global current_account_id
      current_account_id = 0
      found = False
      #Connection to the database
      conn = sqlite3.connect('Resources/SaveData/SaveData.db')
      #Creates a cursor for selecting items in a database
      curs = conn.cursor()
      #Fetches all the accounts
      curs.execute("SELECT * FROM Account_info")
      #Checks to see if the username and password are correct
```

```
      for a,i in enumerate(curs.fetchall()):
            if user == i[0] and hashs(passs) == i[1]:
                  found = True
                  current_account_id = a+1

      #Save the changes
      conn.commit()
      #Closes the connection
      conn.close()
      #reutnrs whehters its ture or not
      return found

#Signup function
def signup(user,passs):
      global current_account_id
      #No current dupes found
      dupe = False


      conn = sqlite3.connect('Resources/SaveData/SaveData.db')
      curs = conn.cursor()

      curs.execute("SELECT * FROM Account_info")
      #Gets all the accounts from the database
      data = (curs.fetchall())

      #Checks to see if there is a dupe
      for i in data:
            if user == i[0]:
                  dupe = True

      #If there is not a dupe username then it will input a new
username and password into the table
      if not dupe:
            curs.execute("INSERT INTO Account_info VALUES
(:Username,:Password)",
                  {

                  'Username': user,
                  'Password': hashs(passs)

                  })
            curs.execute("INSERT INTO Folder_info VALUES
(:Folder_id,:Name,:Account_id)",
            {
            'Folder_id':0,
            'Name':"Root",
            'Account_id':len(data)+1

            })
            current_account_id = len(data)+1
```

```
      conn.commit()
      conn.close()
      #returns whether it found a dupe or not
      return dupe

#current folder and largest folder that the usre is on
current_folder_id = 0
largest_folder_id = 0
#current id of the graph
current_graph_id = 0
#Inittaly sets the graph to not weighted
weighted = False
##Sets the right click menu to false
right_click_menu = False
#Sets the adding new nodes enu to false
add_menu = False
#The user is not holding the mouse down atm
hold = False

#The first node is currently being selected and the second isnt
first_node = True
second_node = False
#The menu for adding the weight of the edge is False
node_menu = False
#Currently there are no nodes in the graph selected and there is
not matrix either
nodes = []
matrix = []
#All the algorithms are not activated
prims = False
kruskals = False
dijkstras = False
#The nodes for the algorithms are all empty
prims_nodes = []
kruskals_nodes = []
dijkstras_nodes = []
#There is nothing in the clipboard atm
clipboard = None
#No node has been right clicked on atm
current_node_right_clicked_on = None

#Gameloop
while run:
      #The clock which will dictacte the ticks
      clock.tick(60)
      #This will fill the screen with this colour and since it is
at the start of the loop this indicates that it is the background
      win.fill(colour1[theme])
      #mouse x and y coords
      mousex,mousey = pygame.mouse.get_pos()

      #This will assess which page the user is currently on
      if page == "login" or page =="signup":
```

```
            #Design
            pygame.draw.rect(win,(255,255,255),(400,40,480,160))
            #This inserts the image of the header
            win.blit(header,(400,40))
            pygame.draw.rect(win,(0,0,0),(400,40,480,160),2)
            #Subheader
            win.blit(subheader,(440,180))
            #This is the usrename/password box and text setup

win.blit(font.render("Username:",0,(255,255,255)),(480,340))
            pygame.draw.rect(win,colour2[theme],(480,380,320,50))

win.blit(font.render("Password:",0,(255,255,255)),(480,430))
            pygame.draw.rect(win,colour2[theme],(480,470,320,50))
            #If the user wants to login it will display the login
button
            if page == "login":

pygame.draw.rect(win,colour2[theme],(570,535,130,50))
                  win.blit(font.render("Login",0,(0,0,0)),(590,540))



win.blit(font.render("Signup?",0,(0,0,0)),(575,620))

pygame.draw.line(win,(0,0,0),(575,660),(705,660),3)
            #If the user wants to signup it shows the signup button
            elif page == "signup":

pygame.draw.rect(win,colour2[theme],(570,535,130,50))

win.blit(font.render("Signup",0,(0,0,0)),(575,540))



win.blit(font.render("Login?",0,(0,0,0)),(590,620))

pygame.draw.line(win,(0,0,0),(575,660),(705,660),3)

            #Renders the username
            win.blit(font.render(user_string,0,(0,0,0)),(480,380))
            #If the user is signing up then the password will
render shown, otherwise itll be hiddne
            if page == "login":

win.blit(pass_font.render(pass_string,0,(0,0,0)),(480,470))
            elif page == "signup":

win.blit(font.render(pass_string,0,(0,0,0)),(480,470))
            #If the username box is selected then the uesrname will
be editted
            if username:
                  user_string = current_string
```

```
                pygame.draw.rect(win,(0,0,0),(475,375,330,60),5)
            #Otherwise if the password box is selected then the
passowrd wil lbe editted
            elif password:
                pass_string = current_string
                pygame.draw.rect(win,(0,0,0),(475,465,330,60),5)

            #Displays the settings icon
            win.blit(settings_icon,(1100,600))

            #If the user left clicks
            if pygame.mouse.get_pressed()[0]:
                #If it is bounded withing the username box
                if mousex >= 480 and mousex <= 800 and mousey >=
380 and mousey <= 430:
                    #Allows the user to enter the username
                    username = True
                    password = False
                    current_string = user_string
                #If tit is mbonuded within the password box
                elif mousex >= 480 and mousex <= 800 and mousey >=
470 and mousey <= 520:
                    #Allows the user to enter the password
                    username = False
                    password = True
                    current_string = pass_string
                elif mousex >= 575 and mousex <= 705 and mousey >=
620 and mousey <= 660:
                    #If the login/signup? button is clicked
                    time.sleep(0.2)
                    if page == "login":
                        page = "signup"
                    elif page == "signup":
                        page = "login"

                elif mousex >= 570 and mousex <= 700 and mousey >=
535 and mousey <= 585:
                    #If the login button is clicked
                    if page == "login":
                        #checks to see if the credentials are
correct
                        val = login(user_string,pass_string)
                        #If not it will tell the user that the
username or apss is wrong
                        if not val:
                            win.blit(font.render("User or
pass incorrect",0,(255,0,0)),(480,300))
                            pygame.display.update()
                            time.sleep(0.3)

                        #Otherwise it will send the user to the
menu screen
                        else:
```

```
                                    page = "menu"
                                    current_string = ""
                        #If the signup button is clicked
                        elif page == "signup":
                              #It checks to see if a user with that
username alr exists
                              val = signup(user_string,pass_string)
                              if val:
                                    win.blit(font.render("User alr
exists",0,(255,0,0)),(480,300))
                                    pygame.display.update()
                                    time.sleep(0.3)

                              else:
                                    page = "menu"
                                    current_string = ""
                  elif mousex >= 1100 and mousex <= 1170 and mousey
>= 600 and mousey <= 670:
                        #if the settings button is cliecked it sends
the user to the settings button
                        page = "settings_login"

                  else:
                        #If the user clicks anywhere else then the
username and password boxes are deseected
                        username = False
                        password = False

      elif page == "settings_login" or page == "settings_menu":
            #Draws the settings art
            win.blit(back_icon,(20,20))
            win.blit(font.render("Theme 1",0,(0,0,0)),(600,70))
            win.blit(font.render("Theme 2",0,(0,0,0)),(600,110))
            #If user clicks
            if pygame.mouse.get_pressed()[0]:
                  #If they select the back button then it returns
them to the place they were at
                  if mousex >= 20 and mousex <= 90 and mousey >= 20
and mousey <= 90:
                        if page == "settings_login":
                              page = "login"
                        else:
                              page = "menu"
                              time.sleep(0.2)
                  #If they click on theme1/theme 2 it changes the
colours
                  if mousex >= 600 and mousex <= 720:
                        if mousey >= 70 and mousey <= 100:
                              theme = 0
                        elif mousey >= 110 and mousey <= 150:
                              theme = 1
```

```
      #
      elif page == "menu":
            #Shows the buttons on the menu
            pygame.draw.rect(win,colour2[theme],(0,0,950,100))
            pygame.draw.rect(win,(0,0,0),(0,0,950,100),2)

            pygame.draw.rect(win,colour4[theme],(950,0,330,720))
            pygame.draw.rect(win,(0,0,0),(950,0,330,720),2)

            conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
            curs = conn.cursor()
            curs.execute(f"SELECT * FROM Folder_info WHERE
Account_id={current_account_id}")
            #Selects all the folders associated with the acocunt
fom the database
            folder_list = curs.fetchall()
            largest_folder_id = folder_list[len(folder_list)-1][0]
            #Displays all the folders associated witht the account
of the panel on the right hand side
            for a,i in enumerate(folder_list):

pygame.draw.rect(win,(0,0,0),(970,20+a*45,200,40),2)

win.blit(font.render(f"{i[1]}",0,(255,0,0)),(975,22+a*45))


pygame.draw.rect(win,(255,0,0),(970,20+current_folder_id*45,200,40
),2)

            conn.commit()
            conn.close()
            #Settings icon again
            win.blit(settings_icon,(870,20))
            #Sends the user to the settings menu
            if pygame.mouse.get_pressed()[0]:
                  if mousex >= 870 and mousex <= 940 and mousey >=
20 and mousey <= 90:
                        page = "settings_menu"

            #If the user clicks on a speciic folder then the
current folder id wil be changed to that which they selected
            if pygame.mouse.get_pressed()[0]:
                  for a,i in enumerate(folder_list):
                        if mousex >= 970 and mousex <= 1170 and
mousey >= 20+a*45 and mousey <= 60+a*45:
                              current_folder_id = a


            conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
            curs = conn.cursor()
            curs.execute(f"SELECT * FROM Graph_info WHERE
```

```
Folder_id={current_folder_id} AND
Account_id={current_account_id}")
            #retreve all the graphs in the database that are
assocaited with the user
            graph_list = curs.fetchall()
            #will set the largest graph id to the length of the
previous graph
            try:
                  largest_graph_id =
graph_list[len(graph_list)-1][len(graph_list[len(graph_list)-1])-1
]
                  #If it cant then it will set it to -1 (implying
that there are no graphs currently made)
            except:
                  largest_graph_id = -1
            #Draws all the graphs that are currently int the folder
            for a,i in enumerate(graph_list):

pygame.draw.rect(win,(0,0,0),(40,120+a*45,600,40),2)

win.blit(font.render(f"{i[0]}",0,(255,0,0)),(45,122+a*45))

            conn.commit()
            conn.close()
            #If the user clicks anyone of the graphs that are
displayed then it will take them to the canvas where they can
create graphs
            if pygame.mouse.get_pressed()[0]:
                  for a,i in enumerate(graph_list):
                        if mousex >= 40 and mousex <= 640 and mousey
>= 120+a*45 and mousey <= 160+a*45:
                              #sets the current graph id to the one
that has been selected
                              current_graph_id = a
                              #switches pages
                              page = "canvas"
                              #retreives the matrix and the nodes
from the database of the graph that they selected and they are
assigned to the matrix and nodes
                              matrix =
pickle.loads(graph_list[current_graph_id][3])
                              nodes =
pickle.loads(graph_list[current_graph_id][4])


            #Displays the buttons for a new project, new folder and
importing other graphs
            pygame.draw.rect(win,(0,0,0),(20,20,230,60),2)
            win.blit(font.render("New project",0,(0,0,0)),(30,30))

            pygame.draw.rect(win,(0,0,0),(270,20,215,60),2)
            win.blit(font.render("New folder",0,(0,0,0)),(280,30))
```

```
            pygame.draw.rect(win,(0,0,0),(505,20,135,60),2)
            win.blit(font.render("Import",0,(0,0,0)),(515,30))

            #If the user hasnt decided to create a graph or create
a folder
            if not text_box_folder and not text_box_graph:
                  weighted = False
                  current_string = ""
                  if pygame.mouse.get_pressed()[0]:
                        if mousex >= 20 and mousex <= 250 and mousey
>= 20 and mousey <= 80:
                              #onclick
                              text_box_graph = True
                              time.sleep(0.1)

                        elif mousex >= 270 and mousex <= 485 and
mousey >= 20 and mousey <= 80:
                              #onclick
                              text_box_folder = True
                              time.sleep(0.1)

                        elif mousex >= 505 and mousex <= 640 and
mousey >= 20 and mousey <= 80:
                              #opens windows explorere on the imports
folder ands gets the user to select a file
                              import_graph =
filedialog.askopenfilename(initialdir="Resources/Imports",title="S
elect A File",filetypes=(("dat files", "*.dat"),))
                              #sets to the value being imported
                              opening_import =
open(import_graph,"rb")
                              #unloads the binary file
                              graph_pieces =
pickle.loads(opening_import.read())
                              opening_import.close()

                              conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
                              curs = conn.cursor()
                              #creates a list with all the graphs in
the folder in the account
                              curs.execute(f"SELECT * FROM Graph_info
WHERE Account_id={current_account_id} AND
Folder_id={current_folder_id}")
                              check = curs.fetchall()
                              alr = False
                              #Checks to see if its already in the
databse
                              for i in check:
                                    #if the names are the same
                                    if graph_pieces[0] == i[0]:
                                          #knows its already there
                                          alr = True
```

```
                              #if it is not already there then it wil
add it to the database
                              if not alr:
                                    curs.execute("INSERT INTO
Graph_info VALUES
(:Name,:Weighted,:Directed,:Graph,:List,:Account_id,:Folder_id,:Gr
aph_id)",
                                                 {
                                                 'Name':graph_pieces[0],
                                                 'Weighted':graph_pieces[1],
                                                 'Directed':graph_pieces[2],
                                                 'Graph':graph_pieces[3],
                                                 'List':graph_pieces[4],

'Account_id':current_account_id,

'Folder_id':current_folder_id,

'Graph_id':largest_graph_id+1

                                                 })
                                    #turns off the text box
                                    text_box_graph = False
                                    #increases the length of the
graphs list

                                    current_graph_id =
largest_graph_id+1
                                    #sets the string to nothing
                                    current_string = ""
                              #closes the database
                              conn.commit()
                              conn.close()



          elif text_box_graph:
                    #displays all the buttons for the text box for
creating a graph

pygame.draw.rect(win,colour4[theme],(320,200,640,320))
                    pygame.draw.rect(win,(0,0,0),(320,200,640,320),2)

                    #Input field for the user to input the name
                    win.blit(font.render("Enter graph
name:",0,(255,255,255)),(480,220))

pygame.draw.rect(win,colour5[theme],(360,280,560,40))
                    pygame.draw.rect(win,(0,0,0),(360,280,560,40),2)


win.blit(font.render(current_string,0,(0,0,0)),(365,280))
                    #Displays whether the graph is weighted or not
                    if weighted:
```

```
win.blit(font.render("Weighted",0,(0,0,0)),(550,350))
                else:
                        win.blit(font.render("Not
Weighted",0,(0,0,0)),(550,350))

                #Design for the weighed? button

pygame.draw.circle(win,(0,0,0),(620,420),18,2,draw_top_left=True,d
raw_bottom_left=True)

pygame.draw.circle(win,(0,0,0),(660,420),18,2,draw_top_right=True,
draw_bottom_right=True)

pygame.draw.line(win,(0,0,0),(620,420-18),(660,420-18),2)

pygame.draw.line(win,(0,0,0),(620,420+17),(660,420+17),2)


pygame.draw.circle(win,colour3[theme],(620,420),14,draw_top_left=T
rue,draw_bottom_left=True)

pygame.draw.circle(win,colour3[theme],(660,420),14,draw_top_right=
True,draw_bottom_right=True)

pygame.draw.rect(win,colour3[theme],(620,420-14,40,28))

                #Displays a green light for weighted and a red
light for not weighted
                if not weighted:

pygame.draw.circle(win,(255,0,0),(620,420),10)
                else:

pygame.draw.circle(win,(0,255,0),(660,420),10)

                #Enter button
                pygame.draw.rect(win,(0,0,0),(580,470,120,40),2)
                win.blit(font.render("Enter",0,(0,0,0)),(590,470))

                #Checks for mouse button clicks
                if pygame.mouse.get_pressed()[0]:
                        #If they click outside the box then it
removes the text box
                        if not(mousex >= 320 and mousex <= 960 and
mousey >= 120 and mousey <= 600):
                                text_box_graph = False
                        #If the user clicks the weighte dbutton then
it changes it from weighted to not weighted and vice versa
                        elif mousex >= 620-18 and mousex <= 660+18
and mousey >= 420-18 and mousey <= 420+18:
                                #if its weighted then it will change it
to not weighted
```

```
                              if weighted:
                                    weighted = False
                                    win.blit(font.render("Not
Weighted",0,(0,0,0)),(550,350))
                                    #otherwise it will be weighted
                              else:
                                    weighted = True

win.blit(font.render("Weighted",0,(0,0,0)),(550,350))
                              #this waits 0.1 which acts like a
buffer so that the person does not end up spamming
                              time.sleep(0.1)
                              #Enter button
                        elif mousex >= 580 and mousex <= 700 and
mousey >= 470 and mousey <= 510:
                              #Makes sure to see theres a name
inputted
                              if current_string != "":
                                    conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
                                    curs = conn.cursor()
                                    #Inputs it into the database
                                    curs.execute(f"SELECT * FROM
Graph_info WHERE Account_id={current_account_id} AND
Folder_id={current_folder_id}")
                                    #Checks for duplicates
                                    check = curs.fetchall()
                                    #Checks to see if its already
there or not
                                    alr = False
                                    for i in check:
                                          if current_string == i[0]:
                                                alr = True

                                    #If it is not already tehre then
it will insert it into the table
                                    #It enters an empty from and an
empty list becauase the user has just created the graph
                                    #The other stuff are variables
which have already been defined
                                    if not alr:
                                          curs.execute("INSERT INTO
Graph_info VALUES
(:Name,:Weighted,:Directed,:Graph,:List,:Account_id,:Folder_id,:Gr
aph_id)",
                                                      {
                                                      'Name':current_string,
                                                      'Weighted':weighted,
                                                      'Directed':False,

'Graph':pickle.dumps([]),

'List':pickle.dumps([]),
```

```
'Account_id':current_account_id,

'Folder_id':current_folder_id,

'Graph_id':largest_graph_id+1

                                                  })
                                            text_box_graph = False
                                            #Increments the latest graph
by 1
                                            current_graph_id =
largest_graph_id+1
                                            #empties the current string
                                            current_string = ""
                                      else:
                                            win.blit(font.render("Name
already exists",0,(255,0,0)),(480,150))
                                            pygame.display.update()
                                            time.sleep(0.3)
                                      conn.commit()
                                      conn.close()
                                else:
                                      win.blit(font.render("Please
enter a name",0,(255,0,0)),(480,150))
                                      pygame.display.update()
                                      time.sleep(0.3)



                else:

                      #Displays the folder popup text box and buttons

pygame.draw.rect(win,colour4[theme],(320,240,640,240))
                      pygame.draw.rect(win,(0,0,0),(320,240,640,240),2)
                      #Entry field
                      win.blit(font.render("Enter folder
name:",0,(255,255,255)),(480,300))

pygame.draw.rect(win,colour5[theme],(360,360,560,40))

win.blit(font.render(current_string,0,(0,0,0)),(370,360))
                      pygame.draw.rect(win,(0,0,0),(360,360,560,40),2)
                      pygame.draw.rect(win,(0,0,0),(580,420,120,40),2)
                      #Displays the enter button
                      win.blit(font.render("Enter",0,(0,0,0)),(590,420))
                      #Checks that the mouse has bene clicked
                      if pygame.mouse.get_pressed()[0]:
                            #Checks to see if the mouse was outside the
box otherwise it deselects
                            if not(mousex >= 320 and mousex <= 960 and
mousey >= 240 and mousey <= 480):
```

```
                              text_box_folder = False

                        #If it clicks on the button
                        elif mousex >= 580 and mousex <= 700 and
mousey >= 420 and mousey <= 460:
                              #makes sure that the box is not empty
                              if current_string != "":
                                    #Connects to the database
                                    conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
                                    curs = conn.cursor()
                                    #retreives all the folders under
the account
                                    curs.execute(f"SELECT * FROM
Folder_info WHERE Account_id={current_account_id}")
                                    #sets them to a list
                                    check = curs.fetchall()

                                    #Check to see if there is a
duplicate
                                    alr = False
                                    for i in check:
                                          if current_string == i[1]:
                                                alr = True
                                    #If there is not a duplicate it
inputs it into the database under the user
                                    if not alr:
                                          curs.execute("INSERT INTO
Folder_info VALUES (:Folder_id,:Name,:Account_id)",
                                                       {

'Folder_id':largest_folder_id+1,
                                                       'Name':current_string,

'Account_id':current_account_id


                                                       })
                                          text_box_folder = False
                                          current_folder_id =
largest_folder_id+1

                                    else:
                                          win.blit(font.render("Folder
already exists",0,(255,0,0)),(480,150))
                                          pygame.display.update()
                                          time.sleep(0.3)
                                    conn.commit()
                                    conn.close()

                              else:
                                    win.blit(font.render("Please
enter a name",0,(255,0,0)),(480,150))
                                    pygame.display.update()
```

```
                                    time.sleep(0.3)

      #This will be all the code for the kanvas
      elif page == "canvas":
            #Whenever the game loops it will always update the
matrix by using the list_to_graph module.
            #This means whenever the matrix has an algorithm
applied onto it, the matrix will be updated at the start of the
loop
            matrix = list_to_graph(nodes)
            #This help with any errors that occur during the
process of conversion. Sometimes the graph only fills half of the
matrix.
            #This algorithm makes the matrix symmetrical along the
leading diagonal
            #loops through all the columns
            for a,i in enumerate(matrix):
                  #loops though all the rows
                  for b,j in enumerate(matrix):
                        #checks to see that there is an edge in the
cell currently being examined
                        if matrix[a][b] != '-':
                              #will set the opposite value to it
                              matrix[b][a] = matrix[a][b]
                        #does the opposite for its mirror
                        if matrix[b][a] != '-':
                              matrix[a][b] = matrix[b][a]


            #loops through all the nodes
            for i in nodes:
                  #loops through all the neighbours
                  for j in i.neighbours:
                        #loops through all the node again
                        for k in nodes:
                              #If the node is connected to the other
node
                              if j[0] == k.name:
                                    #Draws the edge

pygame.draw.line(win,(255,0,0),(i.x,i.y),(k.x,k.y),2)
                                    #If it is weighted then it also
draws its weight
                                    if weighted:

win.blit(font.render(str(j[1]),0,(255,255,255)),((i.x+k.x)//2,(i.y
+k.y)//2))

            #loops through all the nodes
            for i in nodes:
                  #draws the node and its name
                  pygame.draw.circle(win,(255,255,255),(i.x,i.y),20)
                  pygame.draw.circle(win,(0,0,0),(i.x,i.y),20,1)
```

```
win.blit(small_font.render(i.name,0,(0,0,0)),(i.x-30,i.y-40))


            conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
            curs = conn.cursor()
            #gets all the graphs from the current folder
            curs.execute(f"SELECT * FROM Graph_info WHERE
Folder_id={current_folder_id} AND
Account_id={current_account_id}")
            graph_list = curs.fetchall()
            conn.commit()
            conn.close()

            #checks to see if this graph is weighted
            try:
                  weighted = graph_list[current_graph_id][1]
            except:
                  weighted = graph_list[current_graph_id-1][1]

            #draws the pannels
            pygame.draw.rect(win,colour2[theme],(0,0,950,100))
            pygame.draw.rect(win,(0,0,0),(0,0,950,100),2)
            #draws the export button
            pygame.draw.rect(win,colour4[theme],(950,0,330,720))
            pygame.draw.rect(win,(0,0,0),(950,0,330,720),2)
            win.blit(font.render(f"Export",0,(0,0,0)),(750,20))
            win.blit(export_icon,(870,20))
            #allows the user to go back
            win.blit(back_icon,(20,20))
            #sets the string version of the weighted variable to
whatever it. It needs to be a tring so that it can be displayed
onto the screen
            if weighted:
                  weight = "True"
            else:
                  weight = "False"

            #displays whether it is weighed or not

win.blit(font.render(f"Weighted:{weight}",0,(0,0,0)),(955,10))
            #Not yet made it able to be directed so for now all the
graphs will be undirected

win.blit(font.render(f"Directed:False",0,(0,0,0)),(955,50))

            #Displays the buttons for prims kruskals and dijkstras
            pygame.draw.rect(win,(0,0,0),(955,90,200,40),1)
            win.blit(font.render(f"Prims",0,(0,0,0)),(960,90))
            #If the algorithm has been selected then it will
highlight it red
            if prims:
```

```
                pygame.draw.rect(win,(255,0,0),(955,90,200,40),1)

win.blit(font.render(f"Prims",0,(255,0,0)),(960,90))
            #same for kruskals
            pygame.draw.rect(win,(0,0,0),(955,140,200,40),1)
            win.blit(font.render(f"Kruskals",0,(0,0,0)),(960,140))
            if kruskals:
                  pygame.draw.rect(win,(255,0,0),(955,140,200,40),1)

win.blit(font.render(f"Kruskals",0,(255,0,0)),(960,140))
            #same for dijkstras
            pygame.draw.rect(win,(0,0,0),(955,190,200,40),1)
            win.blit(font.render(f"Dijkstras",0,(0,0,0)),(960,190))
            if dijkstras:
                  pygame.draw.rect(win,(255,0,0),(955,190,200,40),1)

win.blit(font.render(f"Dijkstras",0,(255,0,0)),(960,190))
            #reset button
            pygame.draw.rect(win,(0,0,0),(955,240,200,40),1)
            win.blit(font.render(f"Reset",0,(0,0,0)),(960,240))


            #checks to see if the right click menu is active
            if right_click_menu:
                  #if they left click or middle click
                  if pygame.mouse.get_pressed()[0] or
pygame.mouse.get_pressed()[1]:
                        #if they click outside the menu area
                        if not(mousex > tempx and mousey > tempy and
mousex < tempx+200 and mousey < tempy+340):
                              #deselects the right click menu
                              right_click_menu = False
                  #if they right click tho then it just changes the
corrdinates of the right click menu
                  elif pygame.mouse.get_pressed()[2] and mousex >= 0
and mousex <= 950 and mousey >= 100 and mousey <= 720:
                        tempx,tempy=mousex,mousey
            #if the menu isnt active
            else:
                  #if the user clicks in the region of the canvas
                  if mousex >= 0 and mousex <= 950 and mousey >= 100
and mousey <= 720:
                        #if they right click
                        if pygame.mouse.get_pressed()[2]:
                              #it will load the menu
                              right_click_menu = True
                              #check to see if any node has been
clicked on at all
                              current_node_right_clicked_on = None
                              #cycles through all the node
                              for i in nodes:
                                    #checks each of the nodes
corrdinated and a 40x40 region around its center to check if the
```

```
user has right clicked on that node
                                    if mousex >= i.x-20 and mousex <=
i.x+20 and mousey >= i.y-20 and mousey <= i.y+20:
                                        #if so then it will set it
to the current node right clicked on

current_node_right_clicked_on = i
                                #in addition when the user right clicks
it sets tempx,temy to the coordatinated which were right clicked
on. This precents it from following the mouse cursor
                                tempx,tempy=mousex,mousey
                    #is the user clicks on the export buttons
                    elif mousex >=870 and mousex <= 940 and mousey
>=20 and mousey <=90:
                          if pygame.mouse.get_pressed()[0]:
                                #it collects all the information of the
graph from graph list
                                temp_import_graph =
graph_list[current_graph_id]
                                #it opens a new file with the name of
th graph. the file is a dat file
                                export_file =
open("Resources/Exports/"+graph_list[current_graph_id][0]+".dat","
wb")
                                #pickles the information and stores it
in the bnary file

export_file.write(pickle.dumps(temp_import_graph))
                                #closes the file so its saved
                                export_file.close()



            #if the right click menu is active
            if right_click_menu:
                  #displays the copy button

pygame.draw.rect(win,(255,255,255),(tempx,tempy,200,340))

pygame.draw.rect(win,(180,180,180),(tempx,tempy,200,340),1)

pygame.draw.rect(win,(0,0,0),(tempx+20,tempy+20,40,45),1)

pygame.draw.rect(win,(255,255,255),(tempx+24,tempy+16,40,45))

pygame.draw.rect(win,(0,0,0),(tempx+24,tempy+16,40,45),1)

win.blit(font.render("Copy",0,(0,0,0)),(tempx+80,tempy+20))
                    #displays the paste button

pygame.draw.line(win,(180,180,180),(tempx,tempy+80),(tempx+200,tem
py+80),1)
                    #if something exists in the clipboard then it will
```

```
display black, otherwisde it will display grey
                if clipboard != None:
                      win.blit(paste_icon[1],(tempx+10,tempy+90))

win.blit(font.render("Paste",0,(0,0,0)),(tempx+80,tempy+105))
                else:
                      win.blit(paste_icon[0],(tempx+10,tempy+90))

win.blit(font.render("Paste",0,(150,150,150)),(tempx+80,tempy+105)
)


pygame.draw.line(win,(180,180,180),(tempx,tempy+170),(tempx+200,te
mpy+170),1)
                #Displays the add button

pygame.draw.circle(win,(0,0,0),(tempx+42,tempy+210),25,1)

win.blit(font.render("Add",0,(0,0,0)),(tempx+90,tempy+190))


pygame.draw.line(win,(180,180,180),(tempx,tempy+260),(tempx+200,te
mpy+260),1)
                #displays the delete button but displays it grey
if o node is selected but displays black if there is
                if current_node_right_clicked_on != None:

pygame.draw.line(win,(0,0,0),(tempx+20,tempy+270),(tempx+50,tempy+
320))

pygame.draw.line(win,(0,0,0),(tempx+20,tempy+320),(tempx+50,tempy+
270))

win.blit(font.render("Delete",0,(0,0,0)),(tempx+70,tempy+280))
                else:

pygame.draw.line(win,(150,150,150),(tempx+20,tempy+270),(tempx+50,
tempy+320))

pygame.draw.line(win,(150,150,150),(tempx+20,tempy+320),(tempx+50,
tempy+270))

win.blit(font.render("Delete",0,(150,150,150)),(tempx+70,tempy+280
))
                #if the user clicks
                if pygame.mouse.get_pressed()[0]:
                      #checks to see if its in the range of the
right click menu
                      if mousex > tempx and mousex < tempx+200:
                            #checks to see if its in the range of
the copy button
                            if mousey > tempy and mousey <
tempy+80:
```

```
                                    #tries to add a node to the
cliboard
                                    try:
                                          clipboard =
current_node_right_clicked_on
                                    except:pass
                               #checks to see if its in the range of
the paste button
                               elif mousey > tempy+80 and mousey <
tempy+170:
                                    #checks to see if there
issomething in the clipboard
                                    if clipboard:
                                          #checks to see if there is
already a copy of that node
                                          alr = False
                                          temp_neighbours = []
                                          #finds all the neighbours of
the node that has been copied
                                          for i in
clipboard.neighbours:

temp_neighbours.append(i)
                                          #cycles through al lthe
nodes and their neighbours to see if they share a neighbour with
the node in the clipboard
                                          for i in nodes:
                                                for j in i.neighbours:
                                                      if j[0] ==
clipboard.name:
                                                            #if they do
then it wil again append it to the neighbours of the copying node

temp_neighbours.append([i.name,j[1]])

                                          #checks too see for
adulicate name
                                          for i in nodes:
                                                if i.name ==
clipboard.name + " - Copy":
                                                      alr = True


                                          #adds the new copied node to
the nodes list
                                          if not alr:

nodes.append(node(clipboard.name + " -
Copy",len(nodes)+1,clipboard.x,clipboard.y,temp_neighbours))

                                          #remvoves the old node from
clipboard
                                          clipboard = None
```

```
                              #checks to see if it is in the range if
the add button
                              elif mousey > tempy+170 and mousey <
tempy+260:
                                    #opens the add menu but closes
the right click menu
                                    add_menu = True
                                    right_click_menu = False
                              #checks to see if it is in the range of
the delete button
                              elif mousey > tempy+260 and mousey <
tempy+340:
                                    try:
                                          #deletes the node from the
nodes list
                                          for i in nodes:
                                                if i.name ==
current_node_right_clicked_on.name:
                                                      nodes.remove(i)
                                          #goes through every other
nodes neighbours and deletes the connection between theme
                                          for i in nodes:
                                                for j in i.neighbours:
                                                      if j[0] ==
current_node_right_clicked_on.name:

i.neighbours.remove(j)
                                    except:pass
                        #saves any changes made to the database that
has been made
                        conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
                        curs = conn.cursor()
                        curs.execute(f"SELECT * FROM Graph_info
WHERE Account_id={current_account_id} AND
Folder_id={current_folder_id} AND Graph_id={current_graph_id}")
                        check = curs.fetchall()

                        check = check[0]
                        curs.execute(f"""UPDATE Graph_info SET
                              Name = :Name,
                              Weighted = :Weighted,
                              Directed = :Directed,
                              Graph = :Graph,
                              List = :List,
                              Account_id = :Account_id,
                              Folder_id = :Folder_id,
                              Graph_id = :Graph_id

                              WHERE Account_id = {current_account_id}
AND Folder_id = {current_folder_id} AND Graph_id =
{current_graph_id}
                              """,
```

```
                              {
                              'Name':check[0],
                              'Weighted':check[1],
                              'Directed':check[2],
                              'Graph':pickle.dumps(matrix),
                              'List':pickle.dumps(nodes),
                              'Account_id':check[5],
                              'Folder_id':check[6],
                              'Graph_id':check[7],


                              })



                        conn.commit()
                        conn.close()
                        current_string = ""
                  #if the add menu is selected
                  elif add_menu:
                        #displays the buttons

pygame.draw.rect(win,colour4[theme],(320,240,640,240))
                  pygame.draw.rect(win,(0,0,0),(320,240,640,240),2)
                  #displays the entry field
                  win.blit(font.render("Enter node
name:",0,(255,255,255)),(480,300))

pygame.draw.rect(win,colour5[theme],(360,360,560,40))
                  #limits the name of the node to be 10 characters
long
                  current_string = current_string[:10]
                  #displays the name in the entry field

win.blit(font.render(current_string,0,(0,0,0)),(370,360))
                  pygame.draw.rect(win,(0,0,0),(360,360,560,40),2)
                  pygame.draw.rect(win,(0,0,0),(580,420,120,40),2)
                  #enter button
                  win.blit(font.render("Enter",0,(0,0,0)),(590,420))
                  #if the user clicks enter
                  if pygame.mouse.get_pressed()[0]:
                        #checks to see if the user has clicked on
the right area
                        if mousex >= 580 and mousex <= 700 and
mousey >= 420 and mousey <= 460:
                              #makes sure the node has a name
                              if current_string != "":
                                    #check to see if a node already
has this name
                                    alr = False
                                    for i in nodes:
                                          if current_string == i.name:
                                                alr = True
```

```
                                    #if it doesnt
                                    if not alr:
                                          #finds the latest position
of the nodes
                                          temp_len = len(nodes)+1
                                          #adds it to the nodes list

nodes.append(node(current_string,temp_len,tempx,tempy,[]))
                                          #closes the menu
                                          add_menu = False

                                          conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
                                          curs = conn.cursor()
                                          curs.execute(f"SELECT * FROM
Graph_info WHERE Account_id={current_account_id} AND
Folder_id={current_folder_id} AND Graph_id={current_graph_id}")
                                          check = curs.fetchall()

                                          check = check[0]
                                          #adds it to the database
                                          curs.execute(f"""UPDATE
Graph_info SET
                                                Name = :Name,
                                                Weighted = :Weighted,
                                                Directed = :Directed,
                                                Graph = :Graph,
                                                List = :List,
                                                Account_id =
:Account_id,
                                                Folder_id = :Folder_id,
                                                Graph_id = :Graph_id

                                                WHERE Account_id =
{current_account_id} AND Folder_id = {current_folder_id} AND
Graph_id = {current_graph_id}
                                                """,
                                                {

                                                'Name':check[0],
                                                'Weighted':check[1],
                                                'Directed':check[2],

'Graph':pickle.dumps(matrix),

'List':pickle.dumps(nodes),
                                                'Account_id':check[5],
                                                'Folder_id':check[6],
                                                'Graph_id':check[7],


                                                })
```

```
                                        conn.commit()
                                        conn.close()
                                        current_string = ""


            elif node_menu:
                  #if tje user is on the add node menu
                  #displays the popup box

pygame.draw.rect(win,colour4[theme],(320,240,640,240))
                  pygame.draw.rect(win,(0,0,0),(320,240,640,240),2)
                  #displays entry field
                  win.blit(font.render("Enter edge
weight:",0,(255,255,255)),(480,300))

pygame.draw.rect(win,colour5[theme],(360,360,560,40))
                  #tests to see if the string is an integer. If it
isnt then it will remove the last digit
                  try:
                        int(current_string)
                  except:
                        current_string = current_string[:-1]
                  #displats enter button

win.blit(font.render(current_string,0,(0,0,0)),(370,360))
                  pygame.draw.rect(win,(0,0,0),(360,360,560,40),2)
                  pygame.draw.rect(win,(0,0,0),(580,420,120,40),2)
                  win.blit(font.render("Enter",0,(0,0,0)),(590,420))
                  #if the mouse is pressed
                  if pygame.mouse.get_pressed()[0]:
                        #if the user has clicked enter
                        if mousex >= 580 and mousex <= 700 and
mousey >= 420 and mousey <= 460:
                              #if the string isnt empty
                              if current_string != "":
                                    #closes the popup
                                    node_menu = False
                                    #it adds to the neighours list
the node edge

nodes[closest[2]].neighbours.append([nodes[second_closest[2]].name
,int(current_string)])
                                    matrix = list_to_graph(nodes)
                                    #the matrix gets updated
                                    conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
                                    curs = conn.cursor()
                                    curs.execute(f"SELECT * FROM
Graph_info WHERE Account_id={current_account_id} AND
Folder_id={current_folder_id} AND Graph_id={current_graph_id}")
                                    check = curs.fetchall()
```

```
                                    check = check[0]
                                    #updates the database
                                    curs.execute(f"""UPDATE
Graph_info SET
                                          Name = :Name,
                                          Weighted = :Weighted,
                                          Directed = :Directed,
                                          Graph = :Graph,
                                          List = :List,
                                          Account_id = :Account_id,
                                          Folder_id = :Folder_id,
                                          Graph_id = :Graph_id
                                          WHERE Account_id =
{current_account_id} AND Folder_id = {current_folder_id} AND
Graph_id = {current_graph_id}
                                          """,
                                          {
                                          'Name':check[0],
                                          'Weighted':check[1],
                                          'Directed':check[2],

'Graph':pickle.dumps(matrix),
                                          'List':pickle.dumps(nodes),
                                          'Account_id':check[5],
                                          'Folder_id':check[6],
                                          'Graph_id':check[7],})
                                    conn.commit()
                                    conn.close()
                                    current_string = ""

          else:

                #checks to see if the user has clicked on any of
the algorthm buttons
                if pygame.mouse.get_pressed()[0]:
                      #if they click the back button then the user
goes back to the menu
                      if mousex >= 20 and mousex <= 90 and mousey
>= 20 and mousey <= 90:
                            page = "menu"
                            time.sleep(0.2)
                      #if the user clicks on the right sidebar
                      if mousex >= 955 and mousex <= 1155:
                            #checks to see if prims has been
clicked
                            if mousey >= 90 and mousey <= 130:
                                  #only enables prims
                                  prims = True
                                  kruskals = False
                                  dijkstras = False
                            #checks to see if kruskals has been
clicked
                            elif mousey >= 140 and mousey <= 180:
```

```
                                    #only enables kruskals
                                    prims = False
                                    kruskals = True
                                    dijkstras = False
                              #checks to see if dijkstras has been
clicked
                              elif mousey >= 190 and mousey <= 230:
                                    #only enables djikstras
                                    prims = False
                                    kruskals = False
                                    dijkstras = True
                              #checks to see if the reset button has
been clicked
                              elif mousey >= 240 and mousey <= 280:
                                    #disables all of them and resets
the nodes in the arays
                                    prims = False
                                    kruskals = False
                                    dijkstras = False
                                    prims_nodes = []
                                    kruskal_nodes = []
                                    dijkstras_nodes = []


                  #if prims is active then it will display the prims
edges
                  if prims:
                        #cycles through all the prims nodes
                        for i in prims_nodes:
                              #updates the coordinates so that if the
user moves around the nodes these edges will also move around
                              i.update_coords(nodes)
                              #cycles through neighbours
                              for j in i.neighbours:
                                    #cycles through the nodes
                                    for k in prims_nodes:
                                          #checks to see if they are
neighbours
                                          if j[0] == k.name:
                                                #draws the edge

pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)


                  #if kruskal is active then it will display the
kruskal edges
                  if kruskals:
                        #cycles through all the kruskal nodes
                        for i in kruskals_nodes:
                              #updates the coordinates so that if the
user moves around the nodes these edges will also move around
                              i.update_coords(nodes)
                              #cycles through neighbours
```

```
                              for j in i.neighbours:
                                    #cycles through the nodes
                                    for k in kruskals_nodes:
                                          #checks to see if they are
neighbours
                                          if j[0] == k.name:
                                                #draws the edge

pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)

                  #if djikstras is active then izt will display the
dijkstras edges
                  if dijkstras:
                        ##cycles through all the nodes
                        for i in range(0,len(dijkstras_nodes)):
                              #displays the edges from the start to
the end of each of the nodes
                              try:
                                    pygame.draw.line(win, (0, 0,
255), (nodes[dijkstras_nodes[i]].x, nodes[dijkstras_nodes[i]].y),
(nodes[dijkstras_nodes[i+1]].x, nodes[dijkstras_nodes[i+1]].y),2)

                              except:pass

                  #checks to see if shift is pressed and that there
are movre than 2 nodes
                  keys = pygame.key.get_pressed()
                  if (keys[pygame.K_LSHIFT] or
keys[pygame.K_RSHIFT]) and len(nodes) >=2:
                        #if prims and kruskals are not selected
                        if not prims and not kruskals:
                              #the user can select the first node
                              if first_node:
                                    #sets the first node to the
closest as a base case
                                    closest =
[nodes[0].x,nodes[0].y,0]
                                    #goes through all the node
                                    for a,i in enumerate(nodes):
                                          #checks to see if the
magnitude of the distance between the nodes is smaller and if it
is then it will set that as the new closest
                                          if
((i.x-mousex)**2+(i.y-mousey)**2)**0.5 <=
((mousex-closest[0])**2+(mousey-closest[1])**2)**0.5:
                                                closest = [i.x,i.y,a]

                                    #highlights the selected node

pygame.draw.circle(win,(0,255,0),(nodes[closest[2]].x,nodes[closes
t[2]].y),20)

pygame.draw.circle(win,(0,0,0),(nodes[closest[2]].x,nodes[closest[
```

```
2]].y),20,1)

win.blit(small_font.render(nodes[closest[2]].name,0,(0,0,0)),(node
s[closest[2]].x-30,nodes[closest[2]].y-40))
                                    #if the user clicks then the
second node will have a chance to get selected
                                    if pygame.mouse.get_pressed()[0]:
                                          first_node = False
                                          second_node = True
                                          time.sleep(0.2)

                                #if the second node neds to be chosen
                                elif second_node:
                                    #highlights the original selected
node

pygame.draw.circle(win,(0,0,255),(nodes[closest[2]].x,nodes[closes
t[2]].y),20)

pygame.draw.circle(win,(0,0,0),(nodes[closest[2]].x,nodes[closest[
2]].y),20,1)
                                    #finds a random other node to
make the second closest as a base case
                                    try:
                                          second_closest =
[nodes[closest[2]+1].x,nodes[closest[2]+1].y,closest[2]+1]
                                    except:
                                          second_closest =
[nodes[closest[2]-1].x,nodes[closest[2]-1].y,closest[2]-1]

                                    #once again cycles though all the
values to find the closest value
                                    for a,i in enumerate(nodes):
                                          ##checks for the magnitude
to see if it is smaller
                                          if
((i.x-mousex)**2+(i.y-mousey)**2)**0.5 <=
((mousex-second_closest[0])**2+(mousey-second_closest[1])**2)**0.5
:
                                                #checks to see that the
node is not the same as the starting node
                                                if second_closest[2] !=
closest[2]:
                                                      #sets it as the
2nd node
                                                      second_closest =
[i.x,i.y,a]

                                    #if the user clicks the mouse now
                                    if pygame.mouse.get_pressed()[0]:
                                          #resets the variables
                                          first_node = True
                                          second_node = False
```

```
                                        time.sleep(0.2)
                                        #checks to see for duplicate
nodes
                                        alr = False
                                        for i in
nodes[closest[2]].neighbours:
                                            if i[0]
==nodes[second_closest[2]].neighbours:
                                                alr = True
                                        if nodes[closest[2]].name ==
nodes[second_closest[2]].name:
                                            alr = True
                                        #if there are no dupes
                                        if not alr:
                                            #if dijkstras is not
selected
                                            if not dijkstras:
                                                #if it isnt
weighted (it isnt atm but for future development)
                                                if not weighted:

nodes[closest[2]].neighbours.append([nodes[second_closest[2]].name
,1])
                                                        alr = False
                                                        for i in
nodes:
                                                            if
current_string == i.name:

alr = True

                                                        if not alr:

                                                            conn =
sqlite3.connect('Resources/SaveData/SaveData.db')
                                                            curs =
conn.cursor()
                                                            #get
the informaion from the graph urrently working on

curs.execute(f"SELECT * FROM Graph_info WHERE
Account_id={current_account_id} AND Folder_id={current_folder_id}
AND Graph_id={current_graph_id}")
                                                            check
= curs.fetchall()

                                                            check
= check[0]

#updates the database

curs.execute(f"""UPDATE Graph_info SET
```

```
Name = :Name,

Weighted = :Weighted,

Directed = :Directed,

Graph = :Graph,

List = :List,

Account_id = :Account_id,

Folder_id = :Folder_id,

Graph_id = :Graph_id


WHERE Account_id = {current_account_id} AND Folder_id =
{current_folder_id} AND Graph_id = {current_graph_id}

""",
                                                                    {


'Name':check[0],

'Weighted':check[1],

'Directed':check[2],

'Graph':pickle.dumps(matrix),

'List':pickle.dumps(nodes),

'Account_id':check[5],

'Folder_id':check[6],

'Graph_id':check[7],


})



conn.commit()

conn.close()

current_string = ""
```

```
                                                      else:
                                                            #opens node
menu
                                                            node_menu =
True

current_string = ""

                                                else:
                                                      #if dijkstras is
active then it will run dijkstras algorithm
                                                      dijkstras_nodes =
dijkstras_algorithm(matrix,closest[2],second_closest[2])
                                          #highlights the 2nd closest node

pygame.draw.circle(win,(0,255,0),(nodes[second_closest[2]].x,nodes
[second_closest[2]].y),20)

pygame.draw.circle(win,(0,0,0),(nodes[second_closest[2]].x,nodes[s
econd_closest[2]].y),20,1)

win.blit(small_font.render(nodes[second_closest[2]].name,0,(0,0,0)
),(nodes[second_closest[2]].x-30,nodes[second_closest[2]].y-40))
                              #if prims is active
                              elif prims:
                                    #the first node is the closes node
(base case)
                                    closest = [nodes[0].x,nodes[0].y,0]
                                    #itearets through all the nodes to see
if any of them are closr
                                    for a,i in enumerate(nodes):
                                          if
((i.x-mousex)**2+(i.y-mousey)**2)**0.5 <=
((mousex-closest[0])**2+(mousey-closest[1])**2)**0.5:
                                                closest = [i.x,i.y,a]
                                    #highlights selected node

pygame.draw.circle(win,(0,255,0),(nodes[closest[2]].x,nodes[closes
t[2]].y),20)

pygame.draw.circle(win,(0,0,0),(nodes[closest[2]].x,nodes[closest[
2]].y),20,1)

win.blit(small_font.render(nodes[closest[2]].name,0,(0,0,0)),(node
s[closest[2]].x-30,nodes[closest[2]].y-40))
                                    #if the user clicks
                                    if pygame.mouse.get_pressed()[0]:
                                    #prims algorithm runs
                                          prims_nodes =
prims_algorithm(highest_algorithm(matrix),matrix,nodes[closest[2]]
,nodes)
                                          #initially shows the user how
prims algorithm actually work
```

```
                                    #it cycles though the node and 1
by one displys them in the order that they were added in
                                    for i in prims_nodes:
                                          for j in i.neighbours:
                                                for k in prims_nodes:
                                                      if j[0] ==
k.name:

pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)
                                                                  #the reason
for this update is so that the user can actually see it in real
time

pygame.display.update()
                                                                  #that is
also the same reason for this sleep which just pauses for 0.1
seconds

time.sleep(0.1)
                        #if kruskals is active
                        elif kruskals:
                              #kruskals algorithm runs
                              kruskals_nodes =
kruskals_algorithm(highest_algorithm(matrix),matrix,nodes)
                              #cycles though all of the node
                              for i in kruskals_nodes:
                                    for j in i.neighbours:
                                          for k in kruskals_nodes:
                                                if j[0] == k.name:
                                                      #once again
displays it in real time

pygame.draw.line(win,(0,255,0),(i.x,i.y),(k.x,k.y),2)

pygame.display.update()
                                                      time.sleep(0.1)




                  #otherwise the system is just rest
                  else:
                        first_node = True
                        second_node = False



      #This checks every event currently happening in the window
      for event in pygame.event.get():
            #If the event is the big red X in the top corner being
pressed or the ESCAPE key being pressed then it will close the
program
            if event.type == pygame.QUIT or (event.type ==
```

```
pygame.KEYDOWN and event.key == pygame.K_ESCAPE):
                #This just sets the variable which dictates the
gameloop to false which means it will no longer run
                run = False
                #Checks to see if the user has typed anything
           if event.type == pygame.KEYDOWN:
                type(event,page)
           #checks to see if the page is canvas and the user has
held down the button and the shift buttons havent been clicked
           if event.type == pygame.MOUSEBUTTONDOWN and page ==
"canvas" and not (keys[pygame.K_LSHIFT] or keys[pygame.K_RSHIFT]):
                #acknowledges the user is holding
                hold = True
                #no nodes currently selected
                selected_nodes = []
                #all the nodes that are in the area of mouse click
are put into the selected nodes array
                for a,i in enumerate(nodes):
                     if i.x-20<mousex<i.x+20 and
i.y-20<mousey<i.y+20:
                          selected_nodes.append(a)

                try:
                     #this cycles through all the values in the
selected array and checks to see if there is anything lower.
                     current_lowest =
nodes[selected_nodes[0]].pos
                     for i in selected_nodes:
                          if nodes[i].pos <= current_lowest:
                               #if it finds any lower then it
will set that to the moving node
                               moving_node = i

                except:
                     #otherwise it will deselect
                     hold = False


           #if the user doesnt drag then they are not holding
           if event.type == pygame.MOUSEBUTTONUP:
                hold = False
     #if they are holding then it changes the x and y corrds of
the moving node to whatever the mouse x and y is
     if hold:
           if mousex <= 950 and mousex >= 0:
                nodes[moving_node].x = mousex
           if mousey >=100 and mousey <=720:
                nodes[moving_node].y = mousey


     #This updates any changes going on w the screen
     pygame.display.update()
```