Contents

[Analysis 3](#_Toc478635763)

[Identification 3](#_Toc478635764)

[Aim 3](#_Toc478635765)

[Investigation Methods 3](#_Toc478635766)

[How Dijkstra’s Algorithm Works 3](#_Toc478635767)

[Observation 4](#_Toc478635768)

[Questionnaires 5](#_Toc478635769)

[Results 5](#_Toc478635770)

[Questionnaire Analysis 6](#_Toc478635771)

[Interview 6](#_Toc478635772)

[Questions 6](#_Toc478635773)

[Interview Analysis 7](#_Toc478635774)

[Existing System 8](#_Toc478635775)

[IPSO Chart for Existing System 8](#_Toc478635776)

[Data Flow Diagram 8](#_Toc478635777)

[Document Specification 9](#_Toc478635778)

[Example of Existing System 10](#_Toc478635779)

[Flowchart of Process 11](#_Toc478635780)

[Requirements 12](#_Toc478635781)

[Design 13](#_Toc478635782)

[Inputs and Outputs (Graphical Design) 13](#_Toc478635783)

[Main Menu / Home Screen 13](#_Toc478635784)

[User Login 13](#_Toc478635785)

[Help Screen 14](#_Toc478635786)

[Time Trial 14](#_Toc478635787)

[Process 15](#_Toc478635788)

[Class Plan 18](#_Toc478635789)

[Class Diagram – Info above 19](#_Toc478635790)

[Storage 20](#_Toc478635791)

[Database 20](#_Toc478635792)

[Solution Development 22](#_Toc478635793)

[Main Menu 22](#_Toc478635794)

[User Login 24](#_Toc478635795)

[Help 29](#_Toc478635796)

[User Display 31](#_Toc478635797)

[Leaderboards 43](#_Toc478635798)

[Techniques Used 44](#_Toc478635799)

[Testing 45](#_Toc478635800)

[Evaluation 48](#_Toc478635801)

[Requirements Met 48](#_Toc478635802)

[User feedback 50](#_Toc478635803)

[Analysis of end user feedback 50](#_Toc478635804)

[Future improvements 51](#_Toc478635805)

# Analysis

## Identification

Client  
Godalming College Maths Department:  
Tim Hills

Contact  
Tim Hills  
Tuesley Lane  
Godalming  
Surrey  
GU7 1RS   
(01483 423526)

## Aim

Godalming College is educational instate located in Godalming, Surrey. The maths department consists of 8 teachers, covering a variety of the 18 modules that make up A Level Maths, Further Maths, and Further Additional Maths. Throughout the Modules, many students consider D1 the easiest, however it is only easy if the effort is put in, since there are algorithms that need to be learnt and rules that need to be followed to get all of the working marks. In D1, A lot of paper is used and thrown away as the different topics are taught and there is an inconsistency of clear notes.

My goal is to redesign the system in order to give the students a quick and easy way to practice Dijkstra’s algorithm, one of the harder topics in the module. This will help familiarize the Students with Dijkstra’s algorithm before the exam. This new system will also allow the teacher to monitor who is putting in the extra effort, who is exceling at it, and who needs a little extra attention. By using a computer, less paper will be wasted, all the students’ progress can be monitored individually and the college’s average score in D1 should increase slightly, since anyone who is struggling gets the attention they need and there is a quick and easy way for everyone to practice Dijkstra’s algorithm.

## Investigation Methods

As this system will primarily be used by students, it is important that I determine how most D1 students feel about the current teaching methods throughout the module. The best way to do this is through a questionnaire.

I will also study the current teaching method and how the data is recorded by the students to get a deeper understanding of the course and what the system needs to teach. Being a maths student, I have access to all the documentation you are given when learning D1.

I will interview Tim Hills, my end client in order to find out what he needs form the system. I will use this interview to determine the objectives the new system should fulfil.

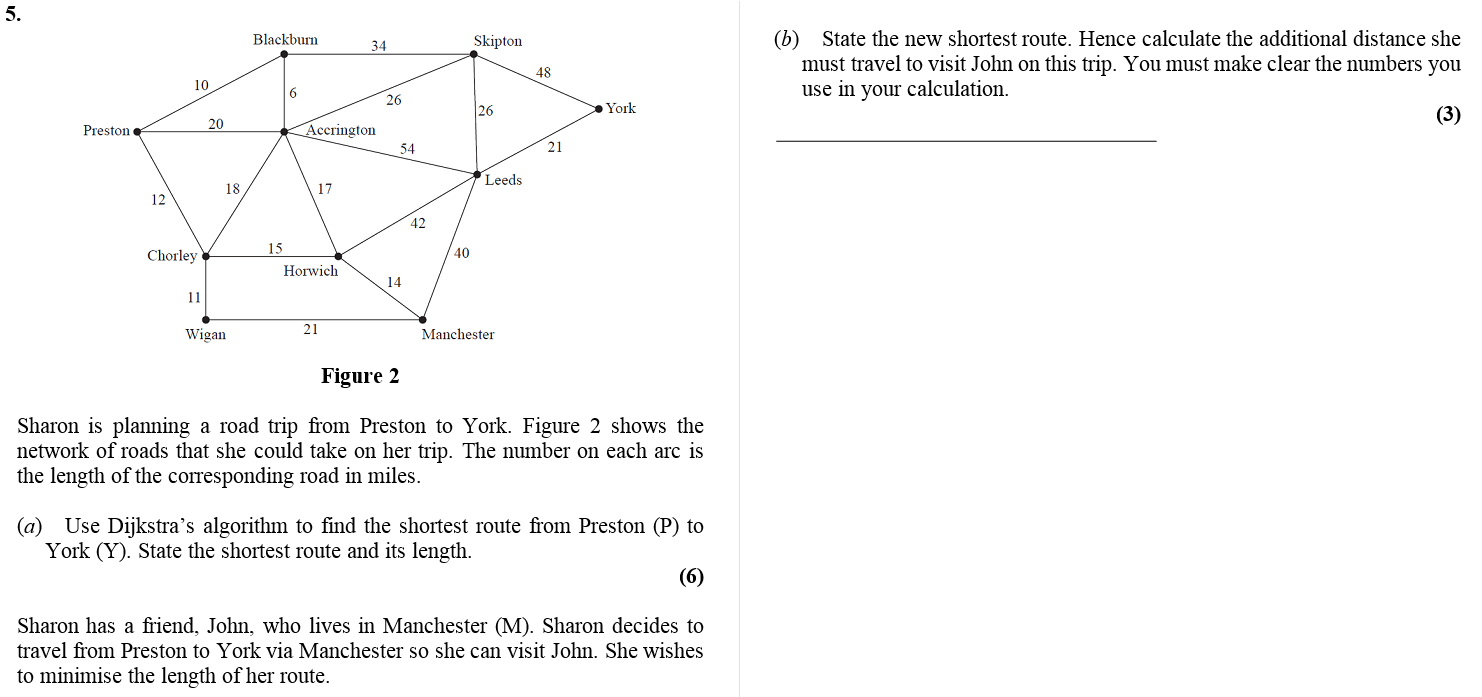
### How Dijkstra’s Algorithm Works

1. Label the start vertex with permanent label 0 and order label 1

* Assign temporary labels to all the vertices that can be reached directly from the start
* Select the vertex with the smallest temporary label and make its label permanent. Add the correct order label.
* Put temporary labels on each vertex that can be reached directly from the vertex you have just made permanent. The temporary label must be equal to the sum of the permanent label and the direct distance from it.  If there is an existing temporary label at a vertex, it should be replaced only if the new sum is smaller.
* Select the vertex with the smallest temporary label and make its label permanent by boxing it.
* Repeat until the finishing vertex has a permanent (boxed) label.
* To find the shortest paths(s), trace back from the end vertex to the start vertex. Write the route forwards and state the length.

## Observation

I have personally been taught D1 and therefore Dijkstra’s algorithm by Tim Hills and therefore have a first-hand experience on how the course is taught.

Below is an example of a typical D1 exam question. This shows me what areas will need to be assessed. The exam question below was from May 2016. This was the exam I sat:

For this question, Part A requires you to state the shortest path and its length. I do not want this to turn into a typing competition and therefore I plan to only ask for the shortest route. I will however expect each box to be filled in saying the shortest path to that vertex. This will reduce the risk of the problem being solved without following Dijkstra’s algorithm.

For part B, the user is required to rearrange the algorithm a little, in order to find the shortest route going through a vertex. This uses the same principle since it is just doing Dijkstra’s algorithm twice. It is because of this, I will not involve wordy questions like this in the new system. If students want to practice these types of questions, they can do past papers at home.

The positions of the vertices does not affect the problem since there is no scale to the drawing, allowing graph templates to be used. This would limit the shape, but still provide a large sample of different answers. If the number of vertices was going to change, an adjacency matrix would need to be used. The graph could then be drawn on a 2 – dimensional Cartesian coordinate system.

Practicing this cannot be assessed using method marks as easily however certain accuracy marks (A1) will be checked throughout the problem.

## Questionnaires

The questionnaire will be given to students to find out how they think they learn best and to get an anonymous, unbiased opinion on the proposal for the new system. Below are the questions I will ask and why I will ask them:

1. **In your personal opinion, how do you learn best?**

This question will be multiple choice however there will also be an ‘other’ box to allow an open answer. This is important to make sure modernising the current system is the best way to go to help teach Dijkstra’s algorithm.

1. **Would you be open to the idea of learning a topic of D1 on the computer?**

This will again make sure that my system will be an IMPROVEMENT on the current system.

1. **Do you think adding a level of competition will make D1 more enjoyable?**

This will allow the students to indirectly decide if they want the times visible. I will not ask the question directly since it will force hard work which some people are against.

1. **Would you like to be able to access the software at home?**

If the majority is yes, the system can not only run on the closed college network. It will therefore need to be accessed through the internet or transferrable via a hard disk.

1. **Is there anything specific you would like the system to do?**

This is the final question to allow students to have an input in the way the system will run. This question will be optional since some people do not like giving open answers.

The questionnaire will be anonymous so no names will be provided with the questionnaire.

## Results

The results have been collected and tabulated for ease of understanding and interpretation. This will also allow the results to be accessed quickly and easily.

|  |  |
| --- | --- |
| QUESTION 1 | |
| PowerPoints | 2 |
| Teacher Examples | 4 |
| Past Papers | 4 |
| Other | N/A |

|  |  |
| --- | --- |
| QUESTION 2 | |
| Yes | 7 |
| No | 3 |

|  |  |
| --- | --- |
| QUESTION 3 | |
| Yes | 6 |
| No | 4 |
| Comment | “Can’t be any less enjoyable” |

|  |  |
| --- | --- |
| QUESTION 4 | |
| Yes | 3 |
| No | 1 |
| Doesn’t affect me | 6 |

|  |
| --- |
| QUESTION 5 |
| N/A.              No one gave a (sensible) answer out of the 10 students interviewed |

Questionnaire Analysis

From completing this questionnaire I now know that the update in technology will mainly be taken well. This will allow me to move forward with my idea. I have also confirmed that competition will enhance the learning experience. With this information in mind, I will include leaderboards, ordered by time for the top 10 students.

## Interview

I carried out an interview with Tim Hills in order to find out some more information on how the current syllabus is taught and discover how it can be improved, allowing the students to be taught more efficiently and allow a greater number to feel comfortable with Dijkstra’s Shortest Path Algorithm.

In order to get a full understanding of the end user’s specifications, I will obtain information from a one-to-one interview. The interview will consist of a face-to-face conversation that will ensure reliability and an improved sense of clarity in the communication of what the end user wants from the system.

Tim Hills is the head of the Maths department at Godalming College, however he also does a lot of teaching, giving him a full insight into the current system used to teach Decision maths. He also has a prior knowledge on how children learn best and how much encouragement children need when they are practicing and toning their skills.

## Questions

I will be asking a series of questions to Tim Hills, the head of the maths department at Godalming College to get a clearer understanding of the current system and his general requirements for the replacement.

The questions I plan to ask are as follows:

1. **On average, how many students study D1 every year?**

This question will be asked in order to find out how many records the database will require. It will help when programming to know how much data will need to be stored in order to keep the program efficient.

1. **How do you want the student’s progress monitored?**

This question will be asked in order to find out what fields will be required in the database, and therefore how many instances the database will have, i.e. student, results.

1. **Will you be able to access the computer room for at least 1 lesson?**

This will be asked to make sure the concept will work in college and will actually be useful in the future.

1. **How would you like the students data ordered for ease of access?**

This will be asked to see how the database will be ordered, i.e. Student ID, Student Last Name etc.

1. **Would you like there to be a competitive aspect to the system?**

Informs me if I should put a timer for a set number of questions and let the students view the fastest times and their personal best times.

Below is the transcript of my conversation with Tim Hills:

Approximately, how many students study D1 every year at Godalming College?

It varies due to the number of student who choose to study further maths, since D1 is a set module taught here.

But if I pressed you for a number?

I would estimate it around 65 pupils

OK. Now, how do you consider the current system put in place by Godalming College?

I think it allows many of the children to pass their exams, as Godalming College achieves very good grades however, I do think that the students are limited by time constraints since the Graphs are needed in order to allow the students to practice for the exam.

And I assume that this wastes a lot of paper, limiting the schools resources?

For each student each lesson, 2 to 3 sheets of paper are required.

Would you be open to the students using a computer application instead to practice D1?

I can’t see why that would be a problem, as long as their work can be checked and the new system can make sure they are practicing.

No problem, how would you like the students results displayed.

I would like the student’s results to be easily accessible from my computer however I do not want everyone to be able to see the results.

So you would like the results saved to a database, ordered by College ID I presume?

That will work fine.

So, you want the results to be like a test. Would you there to be a set number of questions, with a time limit?

I will leave how the questions are set up to you, as long as the questions closely resemble exam style questions.

OK, I am running out of time, but quickly, pleas confirm you will be able to obtain a computer room during some D1 lessons.

I think that will be ok.

Thank you very much for your time, I’ll get it touch if any more queries come to mind.

## Interview Analysis

Firstly, during the interview, I discovered that the number of records in the database will have to be a variable integer, since people can change their mind throughout the year. I also found out that the system will have to access up to 65 records very quickly. The system will also have to run at the same time on multiple computers, since a class can have anywhere from 10 to 30 students. The times will also have to be updated constantly to provide the level of competition throughout the lesson.

I also found out that the current system completes the job at hand, however it does not do it efficiently, wasting a lot of paper in the process. It also does not let Tim check every student’s progress since in class, when everyone if working from the same set of questions, students can cheat and HW’s are nearly always plagiarised. This system will need to provide a way to assess the progress without allowing anyone to copy from the person sitting next to them. It is because of this; the questions will have to be randomly generated. An adjacency matrix can be used to achieve this.

When it comes to the computer side of the new system, the Maths Department is open to the idea of a more technical approach to the problem and believe a computer room will be accessible, allowing each student to have their own account of the system, making the database a much simpler concept. This does present the issue of needing different privileges on different accounts however this should be easy to overcome.

Since the test will need to closely resemble exam style questions, the student will be required to enter certain steps of Dijkstra’s algorithm in order to get the mark for the question. The question will be worded simply which is unlike a real exam style question, however the same process will need to be followed in the exam. The system will be required to check the user’s progress as the lengths of the arcs will be randomly generated. Will make sure they are solving the problem as intended; forcing them to follow the same steps they will in the exam.

Overall, this interview has taught me a lot about what the new system will need to include from its ease of access to the tabulation of results. There will need to be a login system so the students results can be displayed, and the teacher will need an account with different privileges. The graph will need to be displayed clearly and interactive in order to keep the questions closely resembling a real exam question.

## Existing System

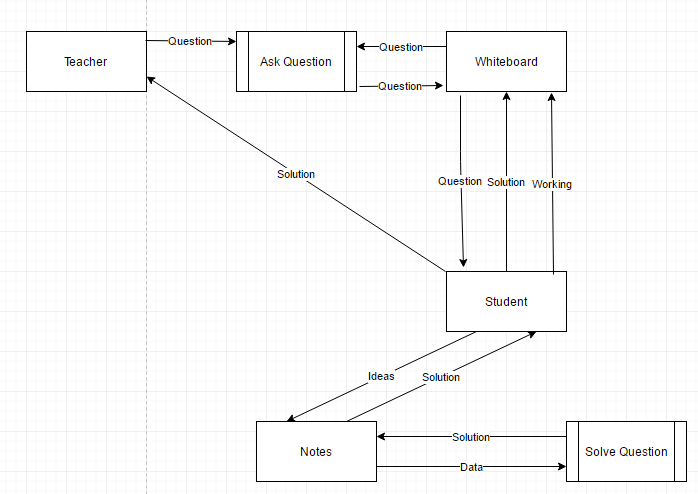
The existing system is currently one of the math’s teachers lecturing the class on Dijkstra’s algorithm, working through an example, and then getting the students to practice the theory behind it with pen and paper. The answer sheets need to be printed which is a waste of paper and the answers are then given, not allowing the teachers to see how the individual students are progressing.

## IPSO Chart for Existing System

|  |  |
| --- | --- |
| INPUT | PROCESS |
| Value for nodes  Value for edges  Student’s name  Student’s working  Start and end node  Names of the nodes | Obtain Question  Calculate Distances to all joining nodes  Find closest node  calculate distances to all adjacent  nodes  Find next Closest node  Check solution  Restart  New Question  Quit |
| STORAGE | OUTPUT |
| Students working  Students answer  Student's name  Correct answer  Students time taken to complete  Total attempts/ Percentage correct  Work completed (Boolean) | Graphic representation of graph  Shortest root  Solution  Marks available  Marks earned  Total time taken  Number of mistakes  Advice/ Feedback |

## Data Flow Diagram

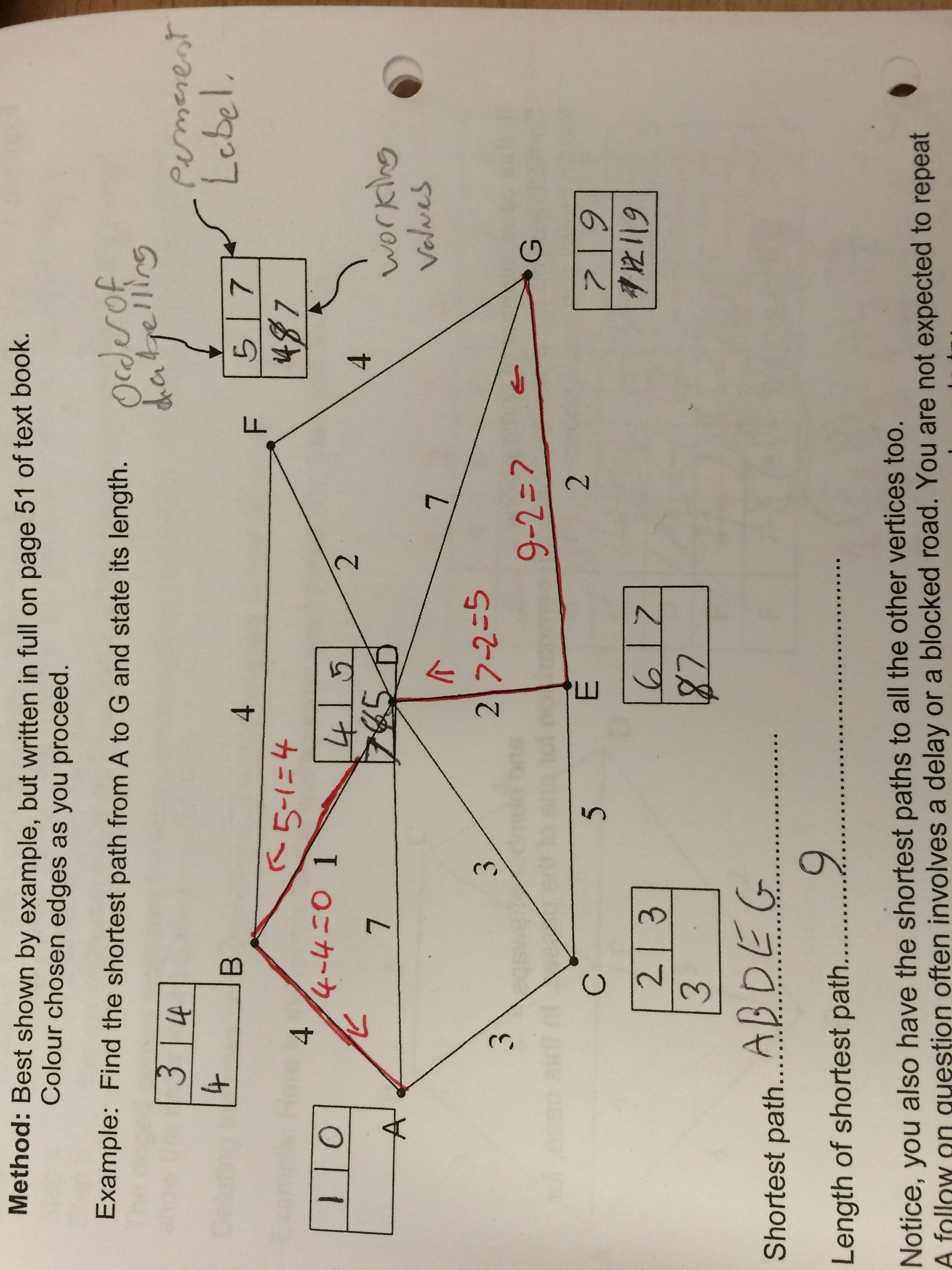
The data flow diagrams below show how data is processed by the system and where the data goes. It also shows the different processes that require data in the system.



## Document Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Volumetrics** | | | | | |
| **Data Dictionary** | | | | | |
| Ref | Name | Data Type | Regex | Occurrence | Source of data / description |
| 1 | Length of edge | Integer | /0[5-9]|1[0-9] | In-between each node in the graph | Weight of each edge in the graph |
| 2 | Name of Node | Character | /^[A-L]$ | Top left of each box next to each node | For Reference |
| 3 | Working Values | List of Integers | /0[1-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-5] | Bottom of box next to each node | User inputs the distance to each node. Shortest distance chosen |
| 4 | Order of Labelling | Integer | /0[1-8] | Top middle of each box next to node | User inputs the nodes in order |
| 5 | Permanent Label | Integer | /0[1-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-5] | Top right of each box next to node | User inputs the shortest distance to each node |
| 6 | Shortest Path | String | /^[A-L]+$ | Below the graph | You are required to write what nodes the shortest path goes through |
| 7 | Length of Shortest Path | Integer | /0[1-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-5] | Below the graph | The number in the permanent label box you want |
| 8. | Instructions | String | /^[a-zA-Z0-9.,:" ]+$ | At the top of the page | Text that explains how to complete the graph |

## Example of Existing System



1

21

31

41

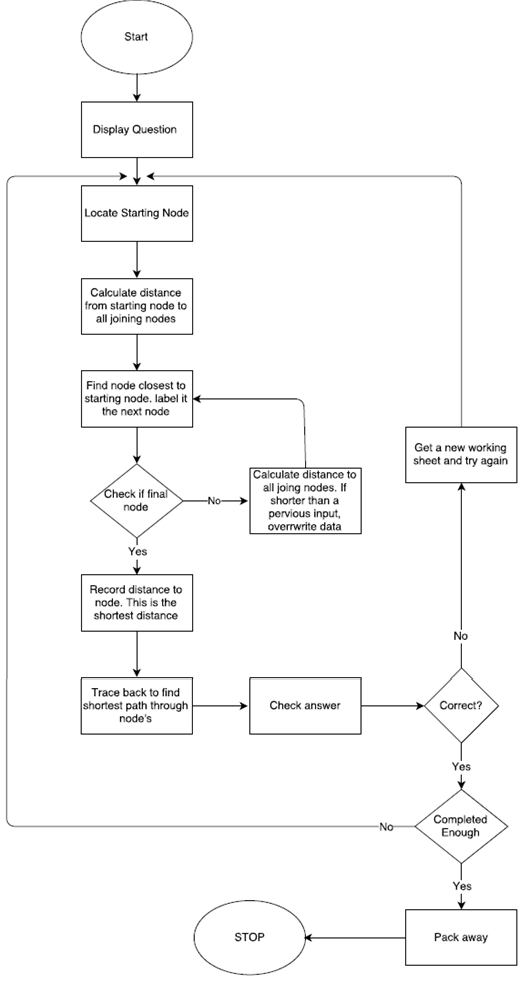
51

61

71

81

## Flowchart of Process



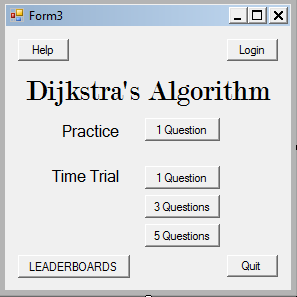
# Requirements

1. Visual display
   1. The graph’s edges should not cross
   2. The user should be able to switch between inputs quickly and easily due to time limit
   3. Inputs need to have validation so the system does not crash
   4. The text should fit in the box regardless of the number
   5. Your time should be visible at all times
   6. The core information in the problem should be quick and easy to spot
   7. There should not be a delay between boards unless incorrect answer
   8. The system will be easy to navigate
      1. There will be a menu when you log in showing the leaderboards
      2. There will be a help screen to explain how to use the new system
      3. You can Quit at any time
      4. It will be easy to choose how many questions you wish to complete
2. Timer
   1. All times should be input to the database
   2. The leaderboards should update real time
   3. There should be different leaderboards based on the number of shortest paths found
   4. You should be able to see how far your attempt was behind first place and top 10
3. Database
   1. The database should not be able to crash due to irregular or extreme inputs
   2. The student should not be allowed to enter invalid information
   3. The database should be in 3rd normal form
   4. The student should not be able to crash the database through irregular input
   5. The student should not be able to add themselves to the database
4. User Login
   1. The student's username should be displayed upon login
   2. It should be easy to login.
   3. You should be able to close the program at any time.
   4. The password should be encoded
5. Solving the problem to check answer
   1. The program should be able to solve the problem efficiently and check if the students working is correct
   2. There should be a very small chance of repeated graphs/answers.
   3. The program should be able to check if there are multiple shortest paths.
   4. The student should not be able to get the fastest time by guessing.

# Design

## Inputs and Outputs (Graphical Design)

### Main Menu / Home Screen



Let’s you sign in.

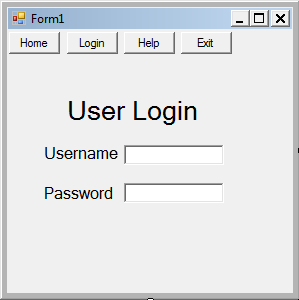
Let’s you choose length of challenge.

Allows practice without the teachers knowledge.

Help button.

This will display the leaderboards.

### User Login



e.g. 153712. Unique for every Student.

Each student can come up with their own password. This cannot be seen by the teacher.

Allows the student to get information on how to use the new system, and return to the home screen.

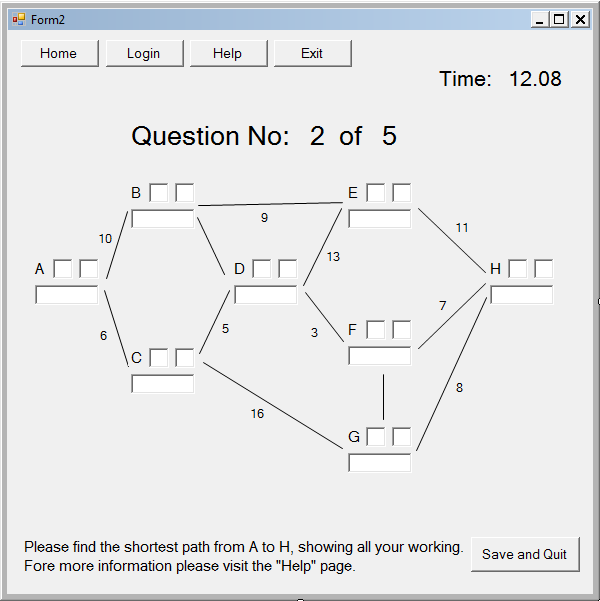
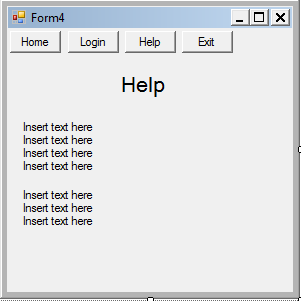
The ‘Login’ button will be a different colour to show it is selected.

Simple, clear title. The fonts will not look the same as in the image above. That will be decided later.

### Help Screen

Will explain how to use the new system so the students can get the largest benefit out of it.

Simple button layout for easy navigation.



Live timer.

This time trial consists of 5 questions.

Simple button layout for easy navigation.

Basic instructions on what to do.

A was to save current progress if limited time.

Simple input system to show and check working. This prevents lucky guesses.

Randomly generated edges and nodes to keep the solution different every time.

### Time Trial

## Process

In order to create this Project, I will start by creating a 12 by 12 matrix containing no edges. I will then decide how many nodes I will have in each column, before running through a variety of selection statements in order to place in the connections. A lot of this will be randomly generated. Since the matrix will be symmetrical along the leading diagonal, I will only have to worry about one half of the matrix. When I replace the unit vectors with the lengths, I can fill in the other half of the matrix.

Randomly generating how many of each node will be in each column:

NumberOfNodes1 <- 1

NumberOfNodes2 <- Math.Floor(Rnd() \* 3) + 3

NumberOfNodes3 <- Math.Floor(Rnd() \* 2) + 2

NumberOfNodes4 <- totalNumberOfNodes - (2 + NumberOfNodes(1) + NumberOfNodes(2))

NumberOfNodes5 <- 1

In order to define the lengths, I will iterate through the matrix. The lowest and highest values will be constants in order to allow future programmers to vary the numbers so they can vary the difficulty. Below is the pseudocode of how I plan to replace the 1’s.

For row <- 0 To 11 ' Iterating through matrix

For cell <- 0 To 11

If AdjacencyMatrix(row, column) = 1 Then

var = Math.Floor(Rnd() \* (largestLength - smallestLength + 1)) + smallestLength

AdjacencyMatrix(row, column) <- var

AdjacencyMatrix(column, row) <- var

End If

Next

Next

To display the nodes, I will have a second Class called Node. I will use composition to create a list of this class:

Dim nodes as list(Of Node)

Within this class I will have some getters in order to access the properties of the class. This class will need to contain the input box’s location, the node number/letter and an input box for the user to find the shortest path.

Next I will draw the lines in between the nodes. In order to see where the lines need to go, I will iterate through the matrix, with the rows representing the left node and the columns representing the right node.

In order to do I plan to import the refrence ‘powerpacks’. I experimented with handles mybase.paint in vb forms however it did not work as intended so I will use powerpacks instead.

At the moment, I do not think I have enough programming knowledge to design this at the moment however I plan to dynamically design this during the development of the solution. I will complete some research before I start this so I can complete this section of the code in an efficient way.

The program will need to calculate the shortest path to every node in the graph. I will design this in vb’s console to make sure it works before I implement it into my final project.

Module Module1

Sub main()

'adjacency matrix. Diagonal will always be 0

Dim adjacencyMatrix As Integer(,) = {

{0, 4, 0, 0, 0},

{0, 0, 8, 0, 0},

{0, 0, 0, 7, 0},

{0, 0, 0, 0, 5},

{0, 0, 0, 0, 0}

}

' graph, starting vertex, number of verticies

Dijkstra(adjacencyMatrix, 0, 5)

Console.ReadLine()

End Sub

Private Function MinimumDistance(ByVal distance As Integer(), ByVal shortestPathTreeSet As Boolean(), ByVal noOfVertices As Integer) As Integer

Dim min As Integer = Integer.MaxValue

Dim minIndex As Integer = 0

For v As Integer = 0 To noOfVertices - 1

If shortestPathTreeSet(v) = False And distance(v) <= min Then

min = distance(v)

minIndex = v

End If

Next

Return minIndex

End Function

Private Sub Print(ByVal distance As Integer(), ByVal noOfVertices As Integer)

Console.WriteLine("Vertex Distance from source")

For i As Integer = 0 To noOfVertices - 1

Console.WriteLine("{0}" & vbTab & " {1}", i, distance(i))

Next

End Sub

Public Sub Dijkstra(ByVal adjacencyMatrix As Integer(,), ByVal source As Integer, ByVal noOfVertices As Integer)

Dim distance As Integer() = New Integer(noOfVertices - 1) {}

Dim shortestPathTreeSet As Boolean() = New Boolean(noOfVertices - 1) {}

For i As Integer = 0 To noOfVertices - 1

distance(i) = Integer.MaxValue

shortestPathTreeSet(i) = False

Next

distance(source) = 0

For count As Integer = 0 To noOfVertices - 2

Dim u As Integer = MinimumDistance(distance, shortestPathTreeSet, noOfVertices)

shortestPathTreeSet(u) = True

For v As Integer = 0 To noOfVertices – 1

If Not shortestPathTreeSet(v) And Convert.ToBoolean(adjacencyMatrix(u, v)) And distance(u) <> Integer.MaxValue And distance(u) + adjacencyMatrix(u, v) < distance(v) Then

distance(v) = distance(u) + adjacencyMatrix(u, v)

End If

Next

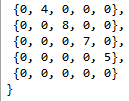
Next

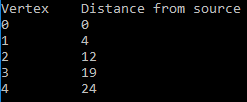
Print(distance, noOfVertices)

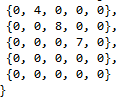
End Sub

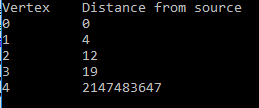
End Module

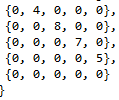
This is the fully working Dijkstra’s I designed in console. It takes the matrix as an input (designed as a 5 by 5 for ease of testing) and prints to the console the shortest distance to each node. Below are some screenshots of the algorithm working for a set matrix. If there is no path the program will say the shortest distance is the max integer count and if the size of the matrix doesn’t match the value input into the Dijkstra sub, the program will crash however invalid inputs will not be possible in the final project due to validation.



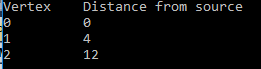








Screen Clipping



The timer will be created using forms inbuilt timer and the result will be displayed to a Label every second, unlike the nodes, this will not have to be dynamically generated so the label will exist on the form at the beginning.

The database will be created using MySQL and will be viewed through Xampp. I will need to reference SQL at the start of the form and import MySQL in order to use the queries and connect to the database.

MySQL string will be:

Dim string As New MySqlConnection("server=127.0.0.1;user=root;database=databasename;port=3306;password=;")

[CREATE](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/create-table.html) `leaderboard`.`time` ( `ID` [INT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/numeric-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL , `UserID` [VARCHAR](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/string-types.html)(6) [NOT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL , `Time` [INT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/numeric-types.html)(10) [NOT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL ) ENGINE = InnoDB;

Finally, I will need a dispose function to reset the form so the next graph can be generated. This will be vital for the completion of the project or you will not know which lines are relevant for the graph.

I will implement an interface called ‘IDisposable’ which contains the functions dispose and finalize. This interface has its functions called by the garbage collector on the disposal of an object. By overriding the interface functions, we can write our own custom code that will be executed when the object is destroyed.

### Class Plan

Black = List of Classes

Red = variable names

Green = subs/ Functions

1. Login
   1. Username, password
   2. User login – takes username and password
2. Main menu
   1. Username, 1/3/5 graphs
   2. Sub for each button on form
3. Graph
   1. Number of nodes, matrix (11 by 11), nodes (defined as a list of class node), smallest weight, largest weight, edges
   2. Graph Weight
   3. Graph Edges
   4. Display
   5. Create matrix
   6. Check valid matrix
4. User interface
   1. Time, 1/3/5 graphs, validation (true/false)
   2. Main
   3. Timer
5. Node
   1. Position (can be a point), label text
   2. Properties
   3. Getters/ Setters
6. Help
   1. Sub for each button on form
7. Database
   1. Connection (my SQL String), string builder, reader
   2. Check username and password
   3. Users Time
   4. Open database
   5. Close Database
   6. Add Time

### Class Diagram – Info above

Main Menu

Help

Login

Database

Node

User Interface

Graph

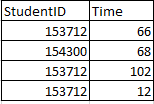
## Storage

### Database

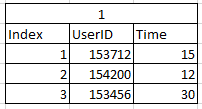
#### Un-normalised

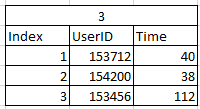
#### 

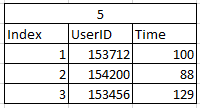
#### 1st normal form



#### 2nd Normal Form







# Screen ClippingSolution Development

## Main Menu

This form allows the user to choose how many boards they wish to complete, display the competition by navigating to the leaderboards, quit the program and view the help page, based on which button is clicked, the form will be hidden and a new from will open.

Public Class MainMenu

Private noOfGraphs As Integer = 0

Private practiceNoTimer As Boolean = False

Private UserID As String

Private Sub Help\_Home\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Help\_Home.Click ' open help and hide main menu

Dim openHelp As Help = Help

Me.Hide()

openHelp.Show()

openHelp = Nothing

End Sub

Private Sub Login\_Home\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) 'open login and hide main menu

Dim openUserLogin As UserLogin = UserLogin

Me.Hide()

openUserLogin.Show()

openUserLogin = Nothing

End Sub

Private Sub OneQP\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OneQP.Click ' opens visual display when button is clicked

noOfGraphs = 1

practiceNoTimer = True

OpenRandomGraph()

End Sub

Private Sub Quit\_Home\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Quit\_Home.Click

Application.Exit()

End Sub

Private Sub MainMenu\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Me.SetBounds(10, 10, 300, 300) ' defines size of form (top left coordinate and length, height)

UsernameLabel.Text = UserID

End Sub

Private Sub OneQTT\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OneQTT.Click ' opens visual display when button is clicked

noOfGraphs = 1

practiceNoTimer = False

OpenRandomGraph()

End Sub

Private Sub ThreeQTT\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ThreeQTT.Click ' opens visual display when button is clicked

noOfGraphs = 3

practiceNoTimer = False

OpenRandomGraph()

End Sub

Private Sub FiveQTT\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles FiveQTT.Click ' opens visual display when button is clicked

noOfGraphs = 5

practiceNoTimer = False

OpenRandomGraph()

End Sub

Public Sub setUserID(ByVal ID) ' gets UserID from a different form into this form

UserID = ID

End Sub

Private Sub OpenRandomGraph() ' opens the form with the randomly generated graph

Dim openUserDisplay As UserDisplay = UserDisplay

Me.Hide()

openUserDisplay.setNumberOfGraphs(noOfGraphs)

openUserDisplay.setPracticeNoTimer(practiceNoTimer)

openUserDisplay.setUserID(UserID)

openUserDisplay.Show()

openUserDisplay = Nothing

End Sub

Private Sub LeaderboardsButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LeaderboardsButton.Click ' opens the leaderboard form

Dim openLeaderboards As Leaderboards = Leaderboards

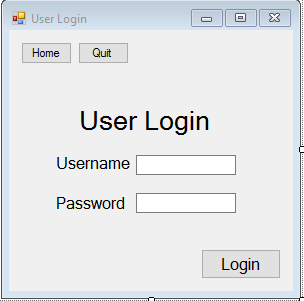
Me.Hide()

openLeaderboards.Show()

openLeaderboards = Nothing

End Sub

End Class



## User Login

This section of my code is responsible for allowing the user to login. All the database work is done within this form. There is validation to ensure that the user cannot crash the data or input false information. If the user does try, then they will be told that the username/password is incorrect. The user will not be able to access the graphs unless they sign in.

I link to the database through SQL commands so My sql has to be imported. I also encrypt the password using an inbuilt encryption within VB forms. His is the ‘system.security.cryptography’ I import at the start of this form. The data is read using a data reader. Within this section nothing is written to the database.

The class Database will initially connect to the database and check the users UserID and password to make sure they are a valid user. This will allow the main menu form to open if all inputs are valid.

The class leaderboard will access the times in the database and display the top 10 users fastest times in the leaderboards.

Usertime is a class designed to hold each users ID and time when it is collected from the database. A list of usertime is created and passed into a quicksort algorithm to order the data based on fastest time. Then the user ID duplications are removed and the first 10 times are taken from this new list, creating the leaderboards.

The quicksort algorithm takes in a list of the class Usertime; containing each of the times in the database, and orders them using a quick sort. This list could be very long, making the quicksort an efficient choice. The program will take a value from the list, then split the list into 2, one containing the numbers greater than the chosen number; and one containing the numbers lower than the chosen value.

Imports System.Security.Cryptography

Imports System.Text

Imports MySql.Data

Imports System.Data

Imports MySql.Data.MySqlClient

Public Class UserLogin

Private Sub HomeLogin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Me.Close()

MainMenu.Show()

End Sub

Private Sub Quit\_Login\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Quit\_Login.Click

Application.Exit()

End Sub

Private Sub Form2\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Me.SetBounds(10, 10, 300, 300)

End Sub

Private Sub EnterUsernamePassword\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles EnterUsernamePassword.Click

Dim username As String = UsernameText.Text

Dim password As String = PasswordText.Text

Dim sha256 As New SHA256CryptoServiceProvider()

Dim convertToByte As Byte() = Encoding.ASCII.GetBytes(password)

Dim hashed As String = Encoding.ASCII.GetString(sha256.ComputeHash(convertToByte))

Dim db As New database

db.CheckIDAndPassword(username, hashed)

db = Nothing

End Sub

End Class

Class database

Dim MySqlCon As New MySqlConnection("server=127.0.0.1;user=root;database=leaderboard;port=3306;password=;") ‘ Links to Database

Private cmd As MySqlCommand

Dim sbCmd As New System.Text.StringBuilder

Private Sub OpenDatabase(ByVal MySqlCon)

MySqlCon.Open()

End Sub

Private Sub CloseDatabase(ByVal MySqlCon)

MySqlCon.Close()

End Sub

Public Sub CheckIDAndPassword(ByVal username, ByVal password)

OpenDatabase(MySqlCon)

Dim sqlQuery As String

Dim dbPassword As String

Dim Reader As MySql.Data.MySqlClient.MySqlDataReader

Try

sqlQuery = "SELECT Password FROM leaderboard.Student WHERE Student.UserID = ('" & username & "')"

cmd = New MySqlCommand(sqlQuery, MySqlCon)

Catch ex As Exception

MsgBox("Invalid Username")

End Try

Reader = cmd.ExecuteReader

While Reader.Read

Dim temp = Reader.GetString("Password")

dbPassword = temp

End While

Reader.Close()

MySqlCon.Close()

If dbPassword = password Then

MsgBox("Logged in Successfully")

Dim passUserID As MainMenu = MainMenu

passUserID.setUserID(username)

MainMenu.Show()

UserLogin.Hide()

Else

MsgBox("Incorrect Username/ Password")

End If

End Sub

Public Sub insertTime(ByVal userName As String, ByVal time As Integer, ByVal numberOfGraphs As Integer)

OpenDatabase(MySqlCon)

Dim sqlQuery As String

Try

If numberOfGraphs = 1 Then

sqlQuery = " INSERT INTO `time1` (`UserID`, `time`) VALUES ('" & userName & "', '" & time & "');"

ElseIf numberOfGraphs = 3 Then

sqlQuery = " INSERT INTO `time3` (`UserID`, `time`) VALUES ('" & userName & "', '" & time & "');"

ElseIf numberOfGraphs = 5 Then

sqlQuery = " INSERT INTO `time5` (`UserID`, `time`) VALUES ('" & userName & "', '" & time & "');"

Else

MsgBox("Program Not Working!")

End If

cmd = New MySqlCommand(sqlQuery, MySqlCon)

cmd.ExecuteNonQuery()

Catch ex As Exception

MsgBox(ex.Message)

End Try

MySqlCon.Close()

End Sub

Public Sub displayLeaderboard(ByVal leaderboardNumber)

OpenDatabase(MySqlCon)

Dim sqlQuery As String

Dim Reader As MySqlDataReader

Dim dbPassword As String

Dim userTimes As New List(Of UserTime)

Dim tempTime As Integer

Dim tempID As String

Dim CurrentLeaderBoard As Leaderboard

Try

sqlQuery = "SELECT UserID, time FROM time" & leaderboardNumber

cmd = New MySqlCommand(sqlQuery, MySqlCon)

Reader = cmd.ExecuteReader

While Reader.Read()

tempID = Reader.GetString("UserID")

tempTime = Reader.GetString("time")

userTimes.Add(New UserTime(tempID, tempTime))

End While

Reader.Close()

Catch ex As Exception

MsgBox("DATABASE FAILURE!")

End Try

CurrentLeaderBoard = New Leaderboard(userTimes)

Try

CurrentLeaderBoard.sort()

Catch ex As Exception

MsgBox("No Times in Database")

End Try

userTimes = CurrentLeaderBoard.getLeaderboard

Dim transferTimes As Leaderboards = Leaderboards

transferTimes.LeaderboardDisplayLabel.Text = "Rank UserID Time" & Environment.NewLine

Dim counterSTR As String

Dim newUserList As New List(Of UserTime)

Dim noValue As Boolean = False

For Each name In userTimes

Dim alreadyExists As Boolean = False

For Each element In newUserList

If name.id = element.id Then

alreadyExists = True

End If

Next

If alreadyExists = False Then

newUserList.Add(name)

End If

Next

If newUserList.Count > 10 Then

For counter = 1 To 10

If counter < 10 Then

counterSTR = "0" & counter.ToString()

Else

counterSTR = counter.ToString()

End If

Dim minutes As Integer = Math.Floor(newUserList(counter - 1).time / 60)

Dim seconds As Integer = newUserList(counter - 1).time Mod 60

If seconds < 10 And minutes < 10 Then

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " 0" & minutes & ":0" & seconds & Environment.NewLine

ElseIf minutes < 10 Then

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " 0" & minutes & ":" & seconds & Environment.NewLine

ElseIf seconds < 10 Then

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " " & minutes & ":0" & seconds & Environment.NewLine

Else

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " " & minutes & ":" & seconds & Environment.NewLine

End If

Next

Else

For counter = 1 To newUserList.Count

If counter < 10 Then

counterSTR = "0" & counter.ToString()

Else

counterSTR = counter.ToString()

End If

Dim minutes As Integer = Math.Floor(newUserList(counter - 1).time / 60)

Dim seconds As Integer = newUserList(counter - 1).time Mod 60

If seconds < 10 And minutes < 10 Then

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " 0" & minutes & ":0" & seconds & Environment.NewLine

ElseIf minutes < 10 Then

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " 0" & minutes & ":" & seconds & Environment.NewLine

ElseIf seconds < 10 Then

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " " & minutes & ":0" & seconds & Environment.NewLine

Else

transferTimes.LeaderboardDisplayLabel.Text = transferTimes.LeaderboardDisplayLabel.Text & counterSTR & ") " & newUserList(counter - 1).id & " " & minutes & ":" & seconds & Environment.NewLine

End If

Next

End If

End Sub

End Class

Class UserTime

Property id As String

Property time As Integer

Sub New(ByVal userId As String, ByVal timetaken As Integer)

id = userId

time = timetaken

End Sub

End Class

Class Leaderboard

Dim users As List(Of UserTime)

Sub New(ByRef users As List(Of UserTime))

Me.users = users

End Sub

Sub sort()

quicksort(users, 0, users.Count - 1)

End Sub

Public Sub quicksort(ByRef users As List(Of UserTime), ByVal LO As Integer, ByVal HI As Integer) ' basic quicksort algorithm

Dim x As Integer

Dim y As Integer

Dim mid As UserTime

Dim temp As UserTime

x = LO

y = HI

mid = users(Int((x + y) / 2))

Do While x <= y

Do While users(x).time < mid.time

x = x + 1

Loop

Do While mid.time < users(y).time

y = y - 1

Loop

If x <= y Then

temp = users(x)

users(x) = users(y)

users(y) = temp

x = x + 1

y = y - 1

End If

Loop

If LO < y Then Call quicksort(users, LO, y)

If HI > x Then Call quicksort(users, x, HI)

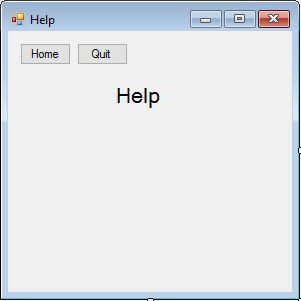
End Sub

Function getLeaderboard() As List(Of UserTime)

Return users

End Function

End Class



## Help

This class will open a text file and display it. This will explain to the user how to operate the program. A text file has been used so the instructions can be changed easily if it is ever required in the future. From this form, the user can return to the main menu, or quit the program.

Public Class Help

Private Sub Form4\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Me.SetBounds(10, 10, 300, 300)

OpenFile() ' will read a text file located in the bin folder created by vb. Allows the help page to be edited easily

End Sub

Private Sub Help\_Quit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Help\_Quit.Click

Application.Exit() ' closes the program

End Sub

Private Sub Login\_Help\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ' opens user login when button is clicked

Dim openLogin As UserLogin = UserLogin

Me.Close()

openLogin.Show()

openLogin = Nothing

End Sub

Private Sub Home\_Help\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Home\_Help.Click ' opens help when button is clicked

Me.Close()

MainMenu.Show()

End Sub

Private Sub OpenFile()

Try

' Create an instance of StreamReader to read from a file.

Using sr As System.IO.StreamReader = New System.IO.StreamReader("help.txt")

Dim line As String

' Read and display lines from the file until the end of

line = sr.ReadLine()

While (line <> Nothing)

Info.Text = Info.Text & Environment.NewLine & line ' Environment.Newline puts it on a new line

line = sr.ReadLine()

End While

End Using

Catch e As Exception

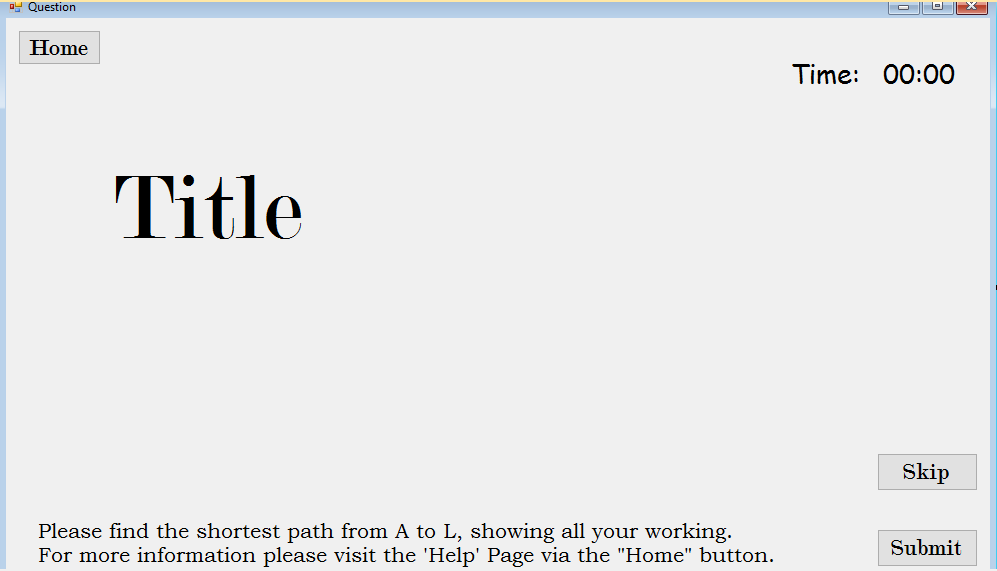
' Let the user know what went wrong.

MsgBox("The file could not be read:")

End Try

End Sub

End Class



## User Display

This class contains the majority of my code and dynamically generates the user interface for the graph. Powerpacks is imported at the start of the program to allow me to use the inbuilt functions shapecontainer(). This will allow me to draw lines on the form without too much effort.

The first class is called ‘UserDisplay’ and it is made up of many functions and subs.

Within ‘UserDisplay’ you get setNumberOfGraphs(). This allows me to access the variable number of graphs from within other forms even though the variable is private. It will take the private variable and assign it to a variable within the current form.

The next functions are solely for the purpose of accessing the variables around the different classes. They are simple to understand from the code so I will not go into detail.

The sub userDisplay\_Load() will run when the form loads. This only defines the dimensions of the display.

The sub clearGraph() will remove the graph from the program by calling a dispose function, with help from the garbage collector. This will effectively reset the form so another graph can be generated. It will not, however, stop or reset the timer.

Initialise graph is the main bode of the program. It calls all the functions relating to the class Graph so they run in the correct order. It also randomizes the inbuilt randomizer so the numbers generated are truly random. This class also starts the timer if the user selected time trial from main menu.

The sub title will create the text telling the user what graph they are on and how many graphs they have left to complete (i.e. Graph 2 out of 5).

The timer class increases the timer each second until it is told to stop. It then displays the time passed in an hour:minute format.

SendTimeToDatabsase will run the sub within Database, allowing the UserID and time to be passed in.

When the button, ‘submit’ is clicked. The program will calculate the shortest path using Dijkstra’s algorithm and check the users input based on the results of the algorithm. If the user is correct, the program will move on to the next graph and the current graph will be increased by one. If, however, they are incorrect, the can try to see where they went wrong or skip the graph. Once the user is correct or skips, initialize graph will be called again allowing the user to continue using the program.

The class ‘Graph’ is there to create the visual display of the graph. This includes the nodes edges and weights. The sub new() will run when the class is first defined. This will define how many nodes are in each column on the visual display.

The sub graph edges calculates the start and end coordinates of each line and then draw them using the inbuilt function, powerpacks. The program will iterate through the matrix and if there should be a line there, the program will draw it from the left centre of the left node to the right centre of the right node.

The sub graphweight adds the distance between each connected node to the centre of the edge connecting them. It defines the font and font size then adds the dynamically generated label to the form. If the value is less than 10, it is written with a 0 at the start. This keeps the labels all the same size.

The sub createDisplay works out the coordinates of each node based on the 1000 x 600-pixel display. The heights are calculated based on how many nodes there are in that column. It then uses form.controls.add() to add the node boxes to the visual display.

Adjacency matrix randomly comes up with a number between the highest and lowest constant. Then it replaces all 1’s in the matrix with a random number. These are the weights. It does it on both sides to make it symmetrical. The code would still work without this however it would not be a true matrix.

Check matrix checks every link for every possible node in every possible formation. i.e. if there are 3 nodes in the current column then 5 nodes in the next column. Certain connections are randomly generated if they do not need to be there. If there are 2 nodes there is a vertical line down. This is the only exception. If there is a connection then the 0 in the matrix is replaced with a 1. This 1 is updated later in the adjacency matrix class.

Within the class node the properties define the text boxes and labels that make up a node. There are some getters within this class that allow me to access the positions of each text box/label within the class. The sub display will set the font size and the position of each text box/label based on the adjacent text boxes/label.

The class Calculate Dijkstra is in charge of calculating the shortest path from the adjacency matrix and checking the users input when they hit submit to see if they are correct.

The sub Dijkstra sets the distance to max value (as a precaution in case there is no route to a node). The sub then calculates the shortest path to each node by looking at the current node and adding on the weight of the connection. This is obtained from the matrix. For more detail the code is designed in the design section of the documentation.

To check if the user is correct, you check the final value box to see if it is correct and check to see if the order box is an integer. If it is, the order box is calculated from the final value box with the aid of an ordered list of all the distances. This should be close to the correct order so a bubble sort is used since this should be efficient. From the start of the sub the user is correct. If anything is wrong the user is incorrect. Innocent until proven guilty!

Imports Microsoft.VisualBasic

Imports Microsoft.VisualBasic.PowerPacks

Public Class UserDisplay

Inherits System.Windows.Forms.Form

Private timer() As Integer = {0, 0}

Private currentGraph As Graph

Public Canvas As New ShapeContainer()

Private numberOfGraphs As Integer

Private numberOfGraphsRemaining As Integer

Private practiceNoTimer As Boolean

Private correctSoFar As Integer = True

Private firstTime As Boolean = True

Private userID As String

Public Sub setNumberOfGraphs(ByVal numgraphs As Int16)

numberOfGraphs = numgraphs

End Sub

Public Sub setPracticeNoTimer(ByVal practice As Boolean)

practiceNoTimer = practice

End Sub

Public Sub setUserID(ByVal ID As String)

userID = ID

End Sub

Public Function getNumberOfGraphsRemaining()

Return numberOfGraphsRemaining

End Function

Private Sub UserDisplay\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Me.SetBounds(10, 10, 1000, 600)

UsernameLabel2.Text = userID

InitialiseGraph()

End Sub

Public Sub ClearGraph()

If Not IsNothing(currentGraph) Then

currentGraph.Dispose()

End If

End Sub

Private Sub InitialiseGraph()

Randomize()

If firstTime = True Then

numberOfGraphsRemaining = numberOfGraphs

End If

firstTime = False

ClearGraph()

Title()

If practiceNoTimer = False Then

Timer1.Start() ' start the timer

Time.Show()

Time2.Show()

Else

Time.Hide()

Time2.Hide()

End If

currentGraph = New Graph()

currentGraph.AdjacencyMatrix()

currentGraph.CreateDisplay()

currentGraph.GraphEdges()

End Sub

Public Sub Title()

With (graphNo)

.Location = New Point(310, 20)

.Height = 65

.Width = 380

.Font = New System.Drawing.Font("Comic Sans MS", 36, FontStyle.Regular)

.Text = "Question " & (numberOfGraphs - numberOfGraphsRemaining + 1) & " of " & numberOfGraphs

End With

End Sub

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

If timer(1) >= 60 Then

timer(0) = timer(0) + Math.Floor(timer(1) / 60)

timer(1) = timer(1) Mod 60

End If

With (Time2)

If timer(0) < 10 And timer(1) < 10 Then

.Text = "0" & timer(0) & ":0" & timer(1)

ElseIf timer(1) < 10 Then

.Text = timer(0) & ":0" & timer(1)

ElseIf timer(0) < 10 Then

.Text = "0" & timer(0) & ":" & timer(1)

Else

.Text = timer(0) & ":" & timer(1)

End If

End With

timer(1) = timer(1) + 1

End Sub

Private Sub Home\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Home.Click

Me.Close()

Dim openMainMenu As MainMenu = MainMenu

openMainMenu.Show()

End Sub

Private Sub Submit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Submit.Click

correctSoFar = True

currentGraph.ShortestPath(correctSoFar)

If correctSoFar = True Then

numberOfGraphsRemaining = numberOfGraphsRemaining - 1

End If

If numberOfGraphsRemaining = 0 Then

Timer1.Stop()

sendTimetoDB()

Me.Close()

MainMenu.Show()

ElseIf correctSoFar = True Then

InitialiseGraph()

End If

End Sub

Private Sub sendTimetoDB()

Dim db As New database

db.insertTime(userID, timer(0) \* 60 + timer(1), numberOfGraphs)

End Sub

Private Sub Skip\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Skip.Click

InitialiseGraph()

End Sub

Private Sub graphNo\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles graphNo.Click

End Sub

End Class

Class Graph

Implements IDisposable

Protected Disposed As Boolean = False

Private Const totalNumberOfNodes As Integer = 12

Private NumberOfNodes As Integer() = {0, 0, 0, 0, 0}

Private Matrix(11, 11) As Integer

Private random As Integer

Private Nodes As New List(Of Node)

Private height() As Integer = {155, 225, 295, 365, 435}

Private currentColumn As Integer

Private variableHeight As Integer

Private counter As Integer

Private Lines As New List(Of LineShape)

Private WeightLabels As New List(Of Label)

Private Const smallestLength As Integer = 5 ' defines the distances the edges can have

Private Const largestLength As Integer = 20 ' defines the distances the edges can have

Sub New()

NumberOfNodes(0) = 1

NumberOfNodes(1) = Math.Floor(Rnd() \* 3) + 3

NumberOfNodes(2) = Math.Floor(Rnd() \* 2) + 2

NumberOfNodes(3) = totalNumberOfNodes - (2 + NumberOfNodes(1) + NumberOfNodes(2))

NumberOfNodes(4) = 1

End Sub

Protected Overridable Sub Dispose(ByVal Disposing As Boolean)

If Not Me.Disposed Then

If Disposing Then

For Each Line In Lines

UserDisplay.Canvas.Shapes.Remove(Line)

Next

For Each WeightLabel In WeightLabels

UserDisplay.Controls.Remove(WeightLabel)

Next

Lines.Clear()

WeightLabels.Clear()

For Each Node In Nodes

Node.Clear()

Next

End If

End If

End Sub

Public Sub Dispose() Implements IDisposable.Dispose ' clear the graph when skip is hit

Dispose(True)

GC.SuppressFinalize(Me)

End Sub

Protected Overrides Sub Finalize()

Dispose(False)

MyBase.Finalize()

End Sub

Public Sub GraphEdges() ' checks where to put graph edges and inserts them into the form

Dim x1, y1, x2, y2 As Integer

Dim Line As LineShape

For row As Integer = 0 To 11

For column As Integer = 0 To 11

If row < column Then

If Matrix(row, column) <> 0 Then

If Nodes(row).GetPosition.x = Nodes(column).GetPosition.x Then

x1 = Nodes(row).GetPosition.x + 45

y1 = Nodes(row).GetPosition.y + 40

x2 = Nodes(column).GetPosition.x + 45

y2 = Nodes(column).GetPosition.y

Else

x1 = Nodes(row).GetPosition.x + 90

y1 = Nodes(row).GetPosition.y + 20

x2 = Nodes(column).GetPosition.x

y2 = Nodes(column).GetPosition.y + 20

End If

Line = New LineShape(x1, y1, x2, y2) ' draws the line in the form

Line.BorderColor = Color.FromArgb(255, 100, 100, 100)

Line.BorderWidth = 2

UserDisplay.Canvas.Shapes.Add(Line)

Lines.Add(Line)

graphWeight(Matrix(row, column), x1, y1, x2, y2)

End If

End If

Next

Next

End Sub

Sub graphWeight(ByVal weight, ByVal x1, ByVal y1, ByVal x2, ByVal y2) ' defines the weight of each vertex and the position of the number on screen.

Dim label As New Label

With (label) ' label attributes. defines what it looks like

.Location = New Point((x1 + x2) / 2 - 10, (y1 + y2) / 2 - 10)

.Width = 21.5

.Height = 15

If weight < 10 Then ' alows numbers below 10 tobe displades as 2 characters

.Text = "0" & weight

Else

.Text = weight

End If

.Font = New System.Drawing.Font("Bookman Old Style", 8, FontStyle.Italic)

End With

UserDisplay.Controls.Add(label)

WeightLabels.Add(label)

End Sub

Sub ShortestPath(ByRef correctSoFar) ' starts solving dijkstra

Dim distanceToNode As New CalculateDijkstra

distanceToNode.StartDijkstra(Matrix, Nodes, correctSoFar)

End Sub

Public Sub CreateDisplay()

UserDisplay.Controls.Add(UserDisplay.Canvas)

For currentNode As Integer = 0 To totalNumberOfNodes - 1 ' iterate through list

If currentNode = 0 Then

currentColumn = 0

ElseIf currentNode < NumberOfNodes(1) + 1 Then

currentColumn = 1

ElseIf currentNode < NumberOfNodes(2) + NumberOfNodes(1) + 1 Then

currentColumn = 2

ElseIf currentNode < 11 Then

currentColumn = 3

Else

currentColumn = 4

End If

If currentNode = 0 Or currentNode = 1 Or currentNode = NumberOfNodes(0) + NumberOfNodes(1) Or currentNode = NumberOfNodes(0) + NumberOfNodes(1) + NumberOfNodes(2) Or currentNode = 11 Then

counter = 0

End If

If NumberOfNodes(currentColumn) = 1 Then

variableHeight = 2

ElseIf NumberOfNodes(currentColumn) = 2 Then

If counter = 0 Then

variableHeight = 1

ElseIf counter = 1 Then

variableHeight = 3

End If

ElseIf NumberOfNodes(currentColumn) = 3 Then

If counter = 0 Then

variableHeight = 0

ElseIf counter = 1 Then

variableHeight = 2

ElseIf counter = 2 Then

variableHeight = 4

End If

ElseIf NumberOfNodes(currentColumn) = 4 Then

If counter = 0 Then

variableHeight = 0

ElseIf counter = 1 Then

variableHeight = 1

ElseIf counter = 2 Then

variableHeight = 3

ElseIf counter = 3 Then

variableHeight = 4

End If

Else

variableHeight = counter

End If

counter += 1

Nodes.Add(New Node) ' adds the nodes to the form

Nodes(currentNode).Display(height, variableHeight, currentColumn, currentNode)

UserDisplay.Controls.Add(Nodes(currentNode).getLabel())

UserDisplay.Controls.Add(Nodes(currentNode).getWorkingBox())

UserDisplay.Controls.Add(Nodes(currentNode).getOrder())

UserDisplay.Controls.Add(Nodes(currentNode).getFinalValue())

Next

End Sub

Public Sub AdjacencyMatrix()

Dim randomLength As Integer

For column As Integer = 0 To 3

CheckMatrix(column)

Next

For row = 0 To totalNumberOfNodes - 1 ' Iterrating through matrix

For column = 0 To totalNumberOfNodes - 1

If Matrix(row, column) = 1 Then

randomLength = Math.Floor(Rnd() \* (largestLength - smallestLength + 1)) + smallestLength ' defines the variation in the edge weights

Matrix(row, column) = randomLength

Matrix(column, row) = randomLength

End If

Next

Next

End Sub

Sub CheckMatrix(ByVal column)

Dim column2 As Integer

random = RandomConnections()

If column = 0 Then

For cell = 1 To NumberOfNodes(column + 1)

Matrix(0, cell) = 1

Next

ElseIf column = 3 Then

For cell = (totalNumberOfNodes - (1 + NumberOfNodes(column))) To 10

Matrix(cell, 11) = 1

Next

End If

If column = 2 Then ' applies to both columns

column2 = NumberOfNodes(1)

Else

column2 = 0

End If

If NumberOfNodes(column) = 3 And NumberOfNodes(column + 1) = 3 Then

Matrix(1 + column2, 4 + column2) = 1 ' node 2 (6) goes to node 5 (9)

Matrix(2 + column2, 5 + column2) = 1 ' node 3 (7) goes to node 6 (10)

If random = 0 Or random = 2 Then ' node 2 (6) can go to node 6 (10) or node 4 (8) can go to node 6 (10) or both

Matrix(1 + column2, 5 + column2) = 1

End If

If random = 1 Or random = 2 Then

Matrix(3 + column2, 5 + column2) = 1

End If

Matrix(3 + column2, 6 + column2) = 1 ' node 4 (8) goes to node 7 (11)

End If

If NumberOfNodes(column) = 3 And NumberOfNodes(column + 1) = 2 Then

Matrix(1 + column2, 4 + column2) = 1 ' node 2 (7) goes to node 5 (10)

If random = 0 Or random = 2 Then ' node 3 (8) can go to node 5 (10) or node 6 (11) or both

Matrix(2 + column2, 4 + column2) = 1

End If

If random = 1 Or random = 2 Then

Matrix(2 + column2, 5 + column2) = 1

End If

Matrix(3 + column2, 5 + column2) = 1 ' node 4 (9) goes to node 6 (11)

Matrix(4 + column2, 5 + column2) = 1 ' node 5 (10) goes to node 6 (11)

End If

If column = 1 Then

If NumberOfNodes(column) = 4 And NumberOfNodes(column + 1) = 2 Then

Matrix(1, 5) = 1 ' node 2 goes to node 6

Matrix(2, 5) = 1 ' node 3 goes to node 6

Matrix(3, 6) = 1 ' node 4 goes to node 7

Matrix(4, 6) = 1 ' node 5 goes to node 7

Matrix(5, 6) = 1 ' node 6 goes to node 7

End If

If NumberOfNodes(column) = 4 And NumberOfNodes(column + 1) = 3 Then

Matrix(1, 5) = 1 ' node 2 goes to node 6

Matrix(2, 5) = 1 ' node 3 goes to node 6

If random = 0 Or random = 2 Then ' node 3 can go to node 7 or node 4 can go to node 7 or both can go to node 7

Matrix(2, 6) = 1

End If

If random = 1 Or random = 2 Then

Matrix(3, 6) = 1

End If

Matrix(3, 7) = 1 ' node 4 goes to node 8

Matrix(4, 7) = 1 ' node 5 goes to node 8

End If

If NumberOfNodes(column) = 5 And NumberOfNodes(column + 1) = 2 Then

Matrix(1, 6) = 1 ' node 2 goes to node 7

Matrix(2, 6) = 1 ' node 3 goes to node 7

If random = 0 Or random = 2 Then ' node 4 can go to node 7 or node 8 or both

Matrix(3, 6) = 1

End If

If random = 1 Or random = 2 Then

Matrix(3, 7) = 1

End If

Matrix(4, 7) = 1 ' node 5 goes to node 8

Matrix(5, 7) = 1 ' node 6 goes to node 8

Matrix(6, 7) = 1 ' node 7 goes to node 8

End If

If NumberOfNodes(column) = 5 And NumberOfNodes(column + 1) = 3 Then

Matrix(1, 6) = 1 ' node 2 goes to node 7

Matrix(2, 6) = 1 ' node 3 goes to node 7

Matrix(3, 7) = 1 ' node 4 goes to node 8

If random = 0 Or random = 2 Then ' node 4 goes to node 7 or node 9 or both

Matrix(3, 6) = 1

End If

If random = 0 Or random = 2 Then

Matrix(3, 8) = 1

End If

Matrix(4, 8) = 1 ' node 5 goes to node 9

Matrix(5, 8) = 1 ' node 6 goes to node 9

End If

End If

If column = 2 Then

If NumberOfNodes(column) = 2 And NumberOfNodes(column + 1) = 3 Then

Matrix(6, 8) = 1 ' node 7 goes to node 9

Matrix(6, 9) = 1 ' node 7 goes to node 10

Matrix(7, 9) = 1 ' node 8 goes to node 10

Matrix(7, 10) = 1 ' node 8 goes to node 11

End If

If NumberOfNodes(column) = 2 And NumberOfNodes(column + 1) = 4 Then

Matrix(5, 7) = 1 ' node 6 goes to node 8

Matrix(5, 8) = 1 ' node 6 goes to node 9

Matrix(6, 9) = 1 ' node 7 goes to node 10

Matrix(6, 10) = 1 ' node 7 goes to node 11

End If

If NumberOfNodes(column) = 2 And NumberOfNodes(column + 1) = 5 Then

Matrix(4, 6) = 1 ' node 5 goes to node 7

Matrix(4, 7) = 1 ' node 5 goes to node 8

If random = 0 Or random = 1 Then ' node 4 and node 5 can go to node 9

Matrix(4, 8) = 1

End If

If random = 1 Or random = 2 Then

Matrix(5, 8) = 1

End If

Matrix(5, 9) = 1 ' node 6 goes to node 10

Matrix(5, 10) = 1 ' node 6 goes to node 11

End If

If NumberOfNodes(column) = 3 And NumberOfNodes(column + 1) = 4 Then

Matrix(4, 7) = 1 ' node 5 goes to node 8

Matrix(4, 8) = 1 ' node 5 goes to node 9

Matrix(5, 8) = 1 ' node 6 goes to node 9

Matrix(5, 9) = 1 ' node 6 goes to node 10

Matrix(6, 9) = 1 ' node 7 goes to node 10

Matrix(6, 10) = 1 ' node 7 goesto node 11

End If

End If

End Sub

Function RandomConnections()

Dim randomEdges As Integer = Math.Floor(Rnd() \* 3)

Return randomEdges

End Function

Function getMatrix()

Return Matrix

End Function

End Class

Class Node

Private label As New Label

Private workingBox As New TextBox

Private order As New TextBox

Private finalValue As New TextBox

Private position As New Point

Dim variableHeight As Integer

Dim counter As Integer

Public Function GetPosition()

Return position

End Function

Public Sub Display(ByVal height, ByVal variableHeight, ByVal currentColumn, ByVal currentNode) ' need to look at sub new in other form

position.X = 50 + (200 \* currentColumn)

position.Y = height(variableHeight)

With (label)

.Location = position

.Width = 30

.Height = 20

.Text = Chr((Asc("A") + (currentNode)))

.Font = New System.Drawing.Font("Bookman Old Style", 12, FontStyle.Regular)

'.TextAlign = ContentAlignment.MiddleCenter

End With

With (workingBox)

.Location = New Point(position.X, position.Y + 20)

.Width = 90

End With

With (order)

.Location = New Point(position.X + 30, position.Y)

.Width = 30

End With

With (finalValue)

.Location = New Point(position.X + 60, position.Y)

.Width = 30

End With

End Sub

Function getLabel() As Label

Return label

End Function

Function getWorkingBox() As TextBox

Return workingBox

End Function

Public Function getOrder() As TextBox

Return order

End Function

Public Function getFinalValue() As TextBox

Return finalValue

End Function

Public Sub Clear()

UserDisplay.Controls.Remove(label)

UserDisplay.Controls.Remove(workingBox)

UserDisplay.Controls.Remove(order)

UserDisplay.Controls.Remove(finalValue)

End Sub

End Class

Class CalculateDijkstra

Private distance As Integer()

Sub StartDijkstra(ByVal Matrix, ByVal nodes, ByRef correctSoFar)

' graph, starting vertex, number of verticies

Dijkstra(Matrix, 0, 12)

CheckIfUserCorrect(distance, nodes, correctSoFar)

End Sub

Private Function MinimumDistance(ByVal distance As Integer(), ByVal shortestPathTreeSet As Boolean(), ByVal noOfVertices As Integer) As Integer

Dim min As Integer = Integer.MaxValue

Dim minIndex As Integer = 0

For v As Integer = 0 To noOfVertices - 1

If shortestPathTreeSet(v) = False And distance(v) <= min Then

min = distance(v)

minIndex = v

End If

Next

Return minIndex

End Function

Public Sub Dijkstra(ByVal Matrix As Integer(,), ByVal source As Integer, ByVal noOfVertices As Integer)

distance = New Integer(noOfVertices - 1) {}

Dim shortestPathTreeSet As Boolean() = New Boolean(noOfVertices - 1) {}

For i As Integer = 0 To noOfVertices - 1

distance(i) = Integer.MaxValue

shortestPathTreeSet(i) = False

Next

distance(source) = 0

For count As Integer = 0 To noOfVertices - 2

Dim u As Integer = MinimumDistance(distance, shortestPathTreeSet, noOfVertices)

shortestPathTreeSet(u) = True

For v As Integer = 0 To noOfVertices - 1

If Not shortestPathTreeSet(v) And Convert.ToBoolean(Matrix(u, v)) And distance(u) <> Integer.MaxValue And distance(u) + Matrix(u, v) < distance(v) Then

distance(v) = distance(u) + Matrix(u, v)

End If

Next

Next

End Sub

Sub CheckIfUserCorrect(ByVal distance As Integer(), ByVal nodes As List(Of Node), ByRef correctSoFar As Boolean)

Dim orderedDistance() As Integer = distance.Clone()

BubbleSort(orderedDistance, orderedDistance.Length)

For Node = 0 To 11

Try

If nodes(Node).getFinalValue().Text <> distance(Node).ToString() Then

correctSoFar = False

ElseIf orderedDistance(CInt(nodes(Node).getOrder.Text) - 1) <> CInt(nodes(Node).getFinalValue.Text) Then

correctSoFar = False

End If

Catch ex As Exception

correctSoFar = False

End Try

If correctSoFar = True Then

orderedDistance(nodes(Node).getOrder.Text - 1) = orderedDistance(nodes(Node).getOrder.Text - 1) + 100 ' stops the user from inputting the same order number twice if there are 2 or more occurances of that number

'MsgBox("node " & Node + 1 & " is correct") ' I am bad at dijkstra. wont be in final program

End If

Next

correctSoFar = True ' FOR TESTING

If correctSoFar = True Then

MsgBox("CORRECT", MsgBoxStyle.Information)

Else

MsgBox("INCORRECT", MsgBoxStyle.Exclamation)

End If

End Sub

Sub BubbleSort(ByRef distanceFromSource() As Integer, ByVal numberOfNodes As Integer) ' complete a bubblesort on the distances to each node from A

Dim i, j As Integer

For i = 0 To numberOfNodes

Dim sorted As Boolean = True

For j = numberOfNodes - 1 To i + 1 Step -1

If (distanceFromSource(j) < distanceFromSource(j - 1)) Then

Dim temp As Integer = distanceFromSource(j)

distanceFromSource(j) = distanceFromSource(j - 1)

distanceFromSource(j - 1) = temp

sorted = False

End If

Next

If sorted = True Then

Exit Sub

End If

Next

End Sub

End Class



## Leaderboards

The leaderboard form will open the database and display the leaderboards based on which leaderboard the user wishes to view. The user can also quit the program and return to the main menu from this screen.

Public Class Leaderboards

Private LeaderboardNumber As Integer

Public Function getLeaderboardNumber()

Return LeaderboardNumber

End Function

Private Sub Form5\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Me.SetBounds(10, 10, 300, 380)

End Sub

Private Sub Quit\_Login\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Quit\_Login.Click

Application.Exit()

End Sub

Private Sub HomeLeaderboards\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles HomeLeaderboards.Click

Me.Close()

MainMenu.Show()

End Sub

Private Sub leaderboard1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles leaderboard1.Click

LeaderboardNumber = 1

openDatabase()

End Sub

Private Sub leaderboard3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles leaderboard3.Click

LeaderboardNumber = 3

openDatabase()

End Sub

Private Sub leaderboard5\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles leaderboard5.Click

LeaderboardNumber = 5

openDatabase()

End Sub

Private Sub openDatabase() ' runs the class database in form 2

Dim db As New database

db.displayLeaderboard(getLeaderboardNumber())

End Sub

End Class

## Techniques Used

Below is a list of techniques I have used to allow a different programmer to understand the code. In some instances, it also makes the program more efficient.

This program contains reading from and writing to a database via MySQL that contains multiple tables, reading from a text file, dynamically generating forms based on an OOP model, OOP with arrays of objects, defined instances of a class, passing a list of objects into a quick sort and ordering them, a recursive quick sort, a bubble sort, drawing to a form, using a garbage collector to clear the form, and advanced matrix calculations to find shortest path.

# Testing

Link to video: (It is unlisted on YouTube)  
“<https://youtu.be/dBzsE6lsniI>”

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Outcome** | **Actual Outcome (Time on video)** |
| *User Login* | | |
| Invalid Username | The user will be told the username/ password was incorrect | 00:38 |
| Invalid Password | The user will be told the username/ password was incorrect | 01:00 |
| No Username | The user will be told the username/ password was incorrect | 00:54 |
| Random Username | The user will be told the username/ password was incorrect | 00:45 |
| No Password | The user will be told the username/ password was incorrect | 01:00 |
| Quit Button | The program will close | 01:16 |
| Correct Username and password | The program will display the main menu | 01:05 |
| *Main Menu* | | |
| Help Button | The program will load the help screen from a file and display it to the user | 01:40 |
| Home Button | The program will display the main menu | 01:44 |
| Quit Button | The program will close | 01:52 |
| *Leaderboards* | | |
| Leaderboards 1 | The current fastest times will be displayed for 1 graph | 02:15 |
| Leaderboards 3 | The current fastest times will be displayed for 3 graph | 02:20 |
| Leaderboards 5 | The current fastest times will be displayed for 5 graph | 02:22 |
| Leaderboard 5 if empty | The program will tell the user there are no current times in the database | 03:12 |
| Home Button | The main menu will be displayed to the user | 03:12 |
| Quit Button | The program will close | 02:31 |
| Each users’ fastest time is displayed. | The program will only display the users fastest time | 09:40 |
| Each user is only shown once | The leaderboard cannot be filled up by one person | 08:55 |
| Only top 10 are displayed | The leaderboard is limited to 10 different users | See picture below |
| *Practice* | | |
| Practice Question Clicked | A graph will be displayed. There will be no timer displayed. | 03:52 |
| Number of Questions displayed | The title will be changed to the correct number of graphs and the correct current graph | 03:32 |
| All lengths displayed | All the weights for the edges will be displayed | 03:32 |
| No crossing edges | None of the lines will cross | 03:32 |
| Tab between boxes | You can quickly move between boxes with the use of tab |  |
| Correct username displayed | The current users’ ID will be displayed in the top right of the screen | 02:06 |
| Randomly generated. Skip Randomly generates another | If the skip button is clicked, a new graph will be randomly generated | 03:35 |
| Home | The current graph will be destroyed and the user will be returned to the main menu. The attempt will not be saved to the leaderboards if on time trial. | 03:47 |
| No inputs submit clicked | The user will be told they are incorrect. The program will not crash | 03:41 |
| Non-integer inputs select clicked | The user will be told they are incorrect. The program will not crash | 11:43 |
| Correct working box but incorrect node | The user will be told they are incorrect. The program will not crash | 11:35 |
| 2 nodes the same (correct) | The user will be told they are incorrect. The program will not crash | 05:50 |
| Correct node number but incorrect working box | The user will be told they are incorrect. The program will not crash | 11:27 |
| Correct Answer | The user will be told they are correct. Once they click OK the next graph will appear | 06:55 |
| After correct answer is input and all graphs solved | The user will be returned to the main menu | 06:55 |
| Times not added to leaderboards/database | Unless time trial, the times will not be added to the database |  |
| *Time trials* | | |
| Time increases until correct answer | The timer will keep increasing until 1/3/5 boards are completed correctly | 04:00 |
| Timer continues when skip is clicked | The timer does not stop/reset when skip is clicked | 03:32 |
| Time is added to database when done | If time trial, the times should be added to the database. If fast enough, the attempt will also be added to the leaderboards. | 08:16 |
| Picture – Before | Screen Clipping | |
| Picture - After | Screen Clipping | |

# Evaluation

## Requirements Met

Below I have compared the final system with the original objectives that I set out in the analysis.

* Visual display
  + The graph’s edges should not cross
    - I met this requirement since there is validation in the program to ensure that the lines will never cross. The number of nodes overall, the number of nodes in each column and the locations of the nodes is limited to ensure this.
  + The user should be able to switch between inputs quickly and easily due to time limit
    - I met this requirement since you can tab through the text boxes, starting from node 1. You will go to the working box first, then the order box before finally the final value. this means the user does not need to click on each individual text box.
  + Inputs need to have validation so the system does not crash
    - I met this requirement since the program will not crash if the user’s username or password is entered incorrectly. There is a try, catch to prevent this. Later in the program, the input is taken as a string and converted to an integer. There is a try, catch around this as well. Also, the program checks to make sure the user is correct so they cannot enter any value and still move on to the next question.
  + The text should fit in the box regardless of the number
    - I met this requirement since is a minimum of 4 edges to get from the first node to the last. Also, it can always be reached in 4 edges. With a maximum weight of 20, the largest number is 80 and with a minimum of 5, the smallest number is 20 (technically 0 in node 1). The program can take a 1 digit input and a 2-digit input.
  + Your time should be visible at all times
    - I met this requirement. If you choose time trial from the main menu the time will be displayed in the top right of the screen. This will not stop if skip is clicked however it will stop if quit is clicked. The timer will also keep counting if more than 1 graph is chosen and they have not all been completed.
  + The core information in the problem should be quick and easy to spot
    - I met this requirement since at the bottom of the screen there is a brief explanation of what to do. Also, if the help button is clicked, the user will be taken to a help screen that will explain what to do in more detail.
  + There should not be a delay between boards unless incorrect answer
    - I partly met this requirement. If the user gets the answer incorrect, a message box will appear informing them of their mistake. If the user is correct a message box will appear informing them they are correct. If going for a fast time, enter can be pressed to make the box go away instantly however the timer does continue while the box is open. This means there could be a delay between boards if the user does not know what to do. Fortunately, there is a help page. This explains everything.
  + The system will be easy to navigate
    - I met this requirement since the buttons are big and the font is easy to read. The text is not too small and you can tab in-between text boxes and buttons instead of having to click.
  + There will be a menu when you log in showing the leaderboards
    - I met this requirement since from the main menu you can view the leaderboard form. Then you can load the correct leaderboard from the database based on what set of data you are interested in.
  + There will be a help screen to explain how to use the new system
    - I met this requirement since there is a help screen. The data is loaded from a text file so it can be edited easily. The help screen clearly explains how to get the most out of this program.
  + You can Quit at any time
    - I met this requirement. On all forms, there is a quit button that will exit the application.
  + It will be easy to choose how many questions you wish to complete
    - I met this requirement since on the main menu, there are clear buttons for a practice question, and 1, 3, or 5 time trialled questions.
* Timer
  + All times should be input to the database
    - I met this requirement. So, long as all the questions are finished, the timer will stop and the user’s ID and time will be added to the database.
  + The leaderboards should update real time
    - I met this requirement. As soon as the user finishes, the times are added to the database. When the leaderboards are refreshed, the new times will be in the leaderboard, so long as they were fast enough.
  + There should be different leaderboards based on the number of shortest paths found
    - I met this requirement since there are 3 leaderboards, one for 1 board, one for 3 boards and one for 5 boards. The correct leaderboard is displayed when the button is clicked.
  + You should be able to see how far your attempt was behind first place and top 10
    - I met this requirement. You can see what time you got when you answer the last question correct. If you look at the leaderboard you can see the top 10 times for the number of questions you answered. This will show you how far of first you are.
* Database
  + The database should not be able to crash due to irregular or extreme inputs
    - I met this requirement. Nothing input into the database is input by the user so none of the inputs can crash the leaderboard. The database will be supplied by the teacher with all their student’s info in so the database ‘student’s creation does not affect my program.
  + The student should not be allowed to enter invalid information
    - I met this requirement since if the user is incorrect on any input, the program will inform them of their mistake.
  + The database should be in 3rd normal form
    - I tried to meet this requirement.
  + The student should not be able to crash the database through irregular input
    - I met this requirement by default since the student does not get to input any data into the database
  + The student should not be able to add themselves to the database
    - I met this requirement since the student does not have access to the student database only the institution does.
* User Login
  + The student's username should be displayed upon login
    - I have met this requirement. Once the user has logged in; their user ID is displayed in the top right of the screen. The user cannot log out until the program closes so the user ID will not change.
  + It should be easy to login.
    - I met this requirement. As soon as the user runs the program, they are forced to login.
  + You should be able to close the program at any time.
    - I met this requirement since there are many quit buttons that will close the application. There is one on every form.
  + The password should be encoded
    - I have met this requirement. It is encoded using one of VB’s inbuilt functions. It cannot be decoded.
  + Solving the problem to check answer
    - I have met this requirement. The algorithm I created to solve Dijkstra’s works fully and I can check the users input from this. Through testing I never discovered any errors in this algorithm. No inputs make the program crash and every distance is calculated following Dijkstra’s algorithm: The way the user will solve it.
  + The program should be able to solve the problem efficiently and check if the students working is correct
    - I have met this requirement since the users’ inputs are all validated and the user is only correct if both the order number and the final value is correct. There are certain selection statements making the code more efficient. i.e. when iterating through the matrix; I only iterate through half since it is symmetrical.
  + There should be a very small chance of repeated graphs/answers.
    - I met this requirement. There are 12 different node patterns the graph can take and certain lines can join to one, another or both. Each weight has a random, equal chance of being from 5 to 20. There is a very slim chance of the user getting an identical graph.
  + The program should be able to check if there are multiple shortest paths.
    - I have met this requirement. If the distance to a node is the same. It does not matter which order the user writes the order number. There is validation, however, to check if the user put the same order number in both boxes.
  + The student should not be able to get the fastest time by guessing.
    - The final answer (node 12) can range from 20 to 80. Both the order box and final value are checked for every box. The user will not get the right answer from guessing. It is extremely improbable.

Overall I met most of my requirements. The delay in-between graphs are insignificant. It is required in order to inform the user of their mistakes/congratulate the user. Also, I tried to get the database in third normal form however with I knowledge I am uncertain if I succeeded. There are no errors in the program and it does not crash. The graph is visually pleasing (in my opinion) and the main menu is easy to navigate. I have used complex structures in the code to make the program robust and run efficiently.

## User feedback

I game my program to Tim Hills. Below is the transcript of what he said:

Tim Hills: “

I am extremely pleased with the final project since it allows me to view the students work ethic and it allows them to practice this topic in a much more efficient manner. The menu is easy to navigate and the login process is very straight forward.

I like how easy it is to change the text displayed on the help screen. This makes the program extremely user friendly. The database is also easy to interpret.

I would have liked to have seen some sort of difficulty scaling so those who are more capable can do harder maths. Maybe you could allow the user to define the range they wanted the lengths to be.

Thank you very much for the program from me and all my students

Tim Hills

“

## Analysis of end user feedback

Overall I am very happy with Tim’s feedback. He liked the display which I put a lot of effort into. I am glad that the majority of the review was positive.

I do agree with the constructive criticism he gave however I never planned for this since he did not mention it in the original interview. Neither did he message me throughout the project about further changes even though I encouraged him to do so.

I am pleased with how the solution came out, even though my plan did vary slightly from the original design due to unforeseen circumstances. Overall it was a very successful solution!

## Future improvements

If I had more time I would add an extra tab to the main menu allowing the user to pick a difficulty. I would call them easy, medium, hard and insane. Each would have a different range for the weights. I would also like to add more nodes to the graph to make it more complex.

If I had unlimited time I would create another option where the user could drag and drop nodes, draw lines in-between the nodes, input the weights and then the algorithm would solve it, showing all the steps. This would act as a tutorial mode and it would allow the user to learn how to do Dijkstra’s from the program rather than just using it for practice. To do this I would have to use a GUI to create the board and the curser would have to be limited to a grid. This would stop nodes being placed overtop each other. There would need to be a lot more validation since every node may not be connected and the user would have to define the start and end node. Also, the number of nodes would not be defined so the matrix size couldn’t be static.