Symbiosis Institute of Technology

(A constituent member of Symbiosis International University)

A REPORT

ON

ROUTING SERVICE FOR SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Submitted by:

NAME	PRN
Himank Jain	18070122027
Ayush Tiwari	18070122013

Faculty Mentor: Rupali Gangarde

TABLE OF CONTENTS

1. ABSTRACT

Routing is the process of selecting a path for traffic in a network, or between or across multiple networks. It can be performed in public and private transportation, such as system of streets, roads and highways. It involves path selection applying a routing metric to multiple routes to select the best route.

Our project i.e. **Routing Services** aims to find the minimum distance and routes to the major landmarks of Symbiosis Institute of Technology (SIT), Pune. The main targets are the students and individuals who are new to the campus and do not know the path to various locations of SIT like Hostel, Canteen, Academic Block etc. It also provides a brief description about the source and destination that the user has entered.

2. INTRODUCTION

The core of our project is based on the Dijkstra's Algorithm which makes use of the Graph Data Structure.

2.1 DIJKSTRA'S ALORITHM

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later. The algorithm exists in many variants.

2.2 EXISTING METHOD

For path selection, we can use following instruments like ranging rod and measuring tapes. But these methods are not feasible, since:

- They cannot be conducted in large areas.
- They are subject to several chances of errors which may be caused by problem of chains. This can result in inaccurate data.
- It is time consuming
- It may not be conducted in areas with steep slopes or water logged areas.

2.3 SOLUTION (INNOVATION)

To facilitate the routing service, we have mapped the entire region of the Institute into the system in the form of an adjacency matrix, where each landmark serves as a vertex. The matrix contains the vertices as well as the distance between all of them.

This matrix is used to implement the Dijkstra's Algorithm. The user will be able to browse all the locations in the campus, and will also be able to find the shortest path and distance from the source node to the destination where the user wants to travel.

When the Source and Destination Nodes are entered, the algorithm calculates the shortest distance which is stored in a distance array and the path is returned through a character array or string.

In this way, a static map alongside Dijkstra's Algorithm is used to simplify the process of routing which eliminates the use of time consuming and redundant methods.

Our project also incorporates File Handling for an articulate description of the landmarks as well as a clear overview of the routing service.

2.4 FILE HANDLING

File Handling concept in C language is used for store a data permanently in computer. Using this concept, we can store our data in Secondary memory (Hard disk). All files related function are available in **stdio.h** header file.

To achieve a better understanding of the landmarks, present in the campus, we create a file containing a brief description about them. Each location has a unique description stored in the file.

The file is accessed whenever the user enters the source and the destination, and the data about these locations is searched and retrieved from the file, which is then displayed on the user screen. Each location has special character assigned to it, which helps the system to search for its description.

Once, the output is displayed, the file is rewound and closed.

By this virtue, our project also acts a guide to the Symbiosis Institute of Technology.

3. LITERATURE SURVEY

The shortest path algorithm is one of the best choices to find the shortest distance between the source and destination. The shortest path (SP) problem involves the problem of finding a suitable path between two vertices or nodes in a graph in such a way that the sum of the weights of its component edges is minimal. There are many theories for solving this problem one of the widely used way solution for solving this problem is Dijkstra's algorithm (DA). [1]

A ***linear graph or a graph*** G=(V,E) consists of a set of objects V={v1, v2, …..} called vertices, and another set E={e1,e2,……}, whose elements are called edges, such that each edge ek, is identified with an unordered pair (vi,vj) of vertices.
The most popular form in which a graph or digraph is fed to computer is its
Dijkstra's algorithm labels the vertices of the given digraph. At each stage in the algorithm some vertices have permanent labels and others temporary labels. The algorithm begins by assigning a permanent label 0 to the starting vertex s, and a temporary label □ to the remaining n-1 vertices. From then on, in each iteration another vertex gets a permanent label, according to the following rules:
1. Every vertex j that is not yet permanently labelled gets a new temporary label whose value is given by Min [old label of j, (old label of i+dij)], Where i is the latest vertex permanently labelled, in the previous iteration, and dij is the direct distance between vertices i and j. If i and j are not joined by and edge, then dij=∞.
2. The smallest value among all the temporary labels is found, and this becomes the permanent label of the corresponding vertex. In case of a tie, select any one of the candidates for permanent labelling.
Steps 1 and 2 are repeated alternately until the destination vertex t gets a permanent label.

Function: Shortestpath: int Short Path(a, n, s, t, path, dist)
Input:
a-Adjacency Matrix describing the graph
n-Number of Nodes in the graph
s-Source Node-Target Node or Sink Node
Output: Path - list of optimal paths from source to sink
dist - Minimum Distance between source and sink
Returns: 0 - if there is no path count indicating the number of nodes along the optimal path.
adjacency matrix: After assigning a distinct number to each of the n vertices of the given graph G, the n by n binary matrix X(G) is used for representing G during input, storage, and output. [2]

Why Use Adjacency Matrix-
Adjacency matrix is very convenient to work with. Adding or removing an edge can be done in O(1) time, the same time is required to check, if there is an edge between two vertices. Also, it is very simple to program and in all our graph tutorials we are going to work with this kind of representation. But Adjacency matrix consumes huge amount of memory for storing big graphs. Next drawback of the adjacency matrix is that in many algorithms you need to know the edges, adjacent to the current vertex. To draw out such an information from the adjacency matrix you have to scan over the corresponding row, which results in O(|V|) complexity. [3]

Dijkstra's Algorithm to find the shortest routes between two locations or from their current production plant to any delivery store of choice now and in the nearest future to help them

- Minimize the distance of transporting the goods.
- Save time in transporting the products profit of the company.
- Minimize the cost of running the transportation of goods to maximize profits of the company. [4]

3.1 GAPS
Today, many applications use this shortest path algorithm such as Google Maps, Uber cabs, Ola cabs etc. to determine the most optimal and shortest path for its users. But no such application exists for Symbiosis Institute of Technology, Pune. Every year there is a batch that is new to the college. Parents and students often visit the college for different purposes. It is important that they know what is the shortest path to different locations within the college. Thus, our application guides them with the most optimal path from the source to destination

3.2 NOVELTY OF WORK
The main idea behind application is to help the visitors, faculties and students to determine the shortest path from the entered source to destination in the campus. For the weights between the edges we took the coordinates of all the building through google map and distance measuring apps.
By having an idea of the positions of all the buildings, a rough graph was created. From the rough graph adjacency matrix was created in which all the measured distances were fed.

Also using file handling a file was created which contained the information about all the locations to help the user get a basic idea about that particular site. So, when the user enters the source and destination the information about both the locations is displayed to him.
The Dijkstra algorithm was then implemented to find the shortest path and distance and display it to the user.
If the user selects academic block as his destination, he is given an option whether he wants to enter inside the campus and if the answer is "YES" then again, the user is asked for source and destination to guide him to his desired location in the short span of time.

4. MODULES

Our project is combination of following modules:

1. **Main()-** This method interacts with user by allowing the user to enter source and destination node. Call to modules like function_handling, sourceval, destionationval, Dijkstra, exitservice etc takes place from this method.

2. **Sourceval()-**This method is used for source nodes. When user enters source location, that location is compared with locations in this method and if found a value is assigned to that particular location.

3. **Destinationval()-** This method is used for destination nodes. When user enters destination location, that location is compared with locations in this method and if found a value is assigned to that particular location.

4. **SIT_landmarks()-**This method displays all the locations which are part of the project.

5. **SIT_Academic_landmarks()-**This method displays all the locations inside the academic block which are considered for this project.

6. **SIT_Distances()-**This method contains adjacency matrix which contains all distances between each location considered for this project. This data is required by Dijkstra method to calculate shortest path.

7. **Init()-**This method is used to initialize the adjacency matrix to infinity.

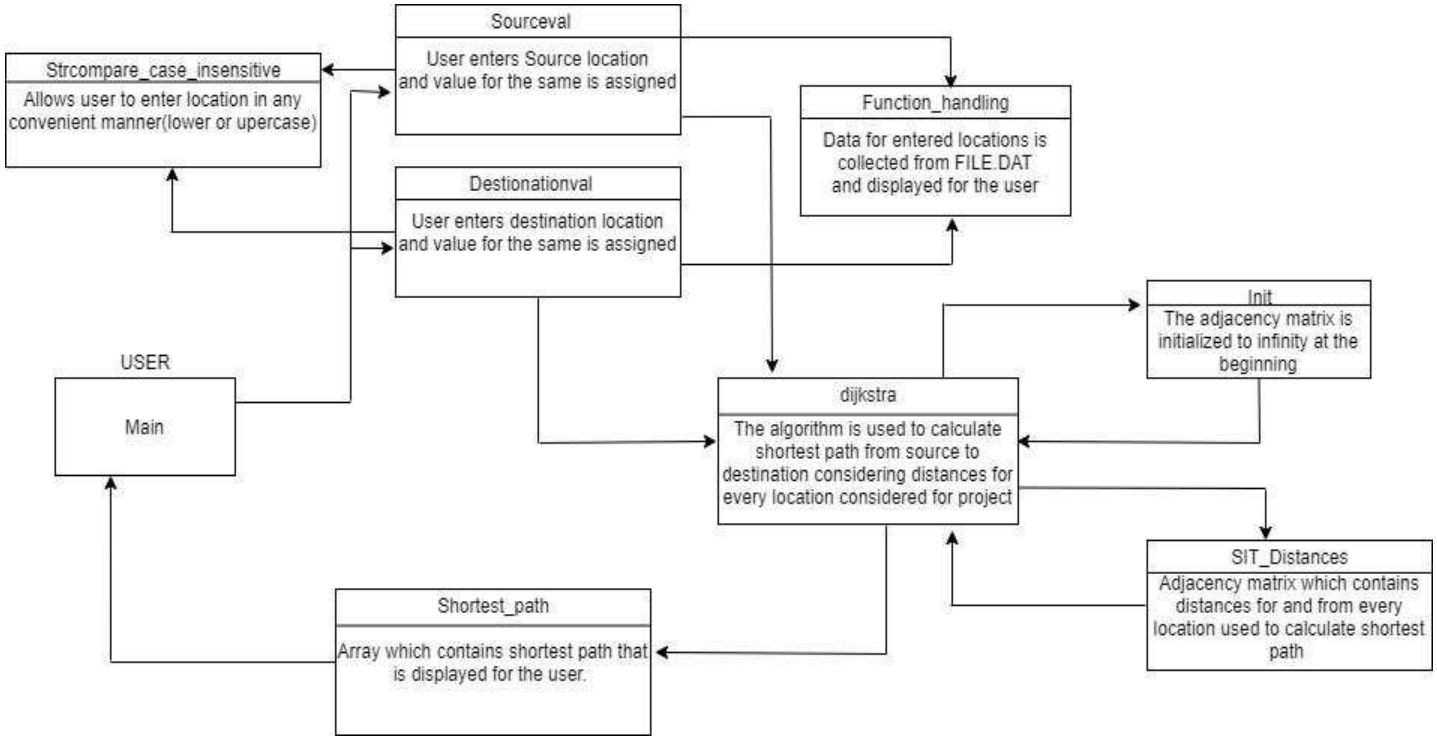
8. **Strcompare_case_insensitive()-**This method is used to make our project case insensitive. This feature is helpful for the users as input can be in lowercase or uppercase.

9. **Dijkstra()-**This is the most important module of the project. Here the shortest distance between source and destination is calculated from the data provided by adjacency matrix.

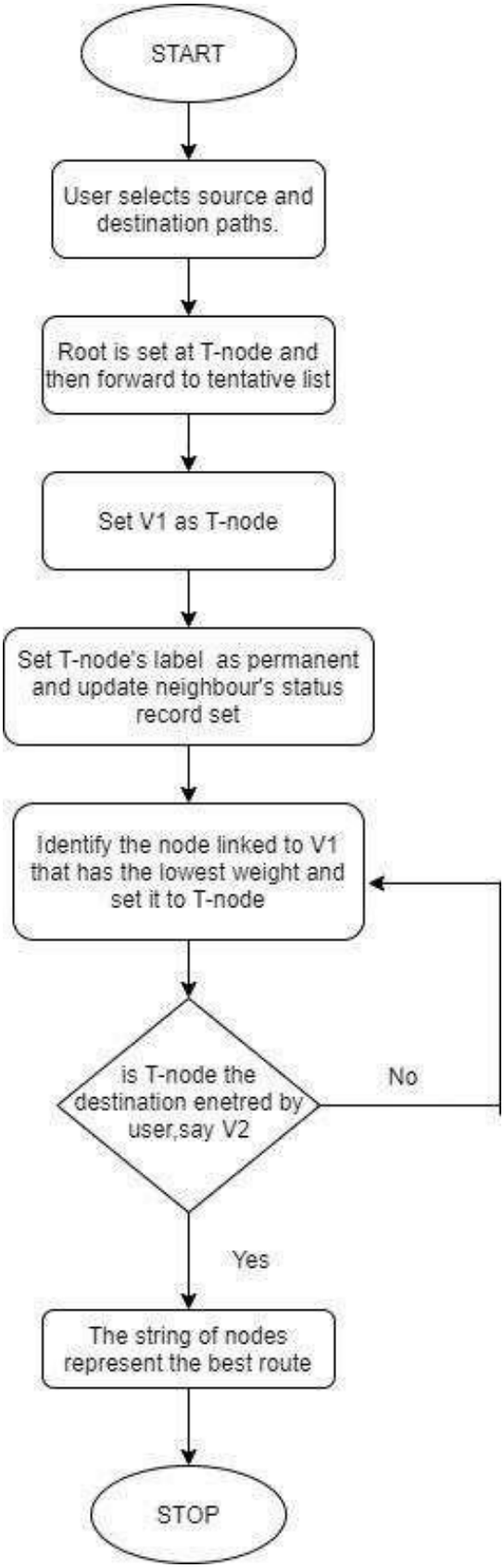
10. **Shortest_path()-** This module contains an array which has the shortest path from source to destination. The shortest path from the array is displayed with help of switch case.

11. **function_handling()-**This module deals with files and its methods. A file which contains a description of all the locations considered for project. When user enters source and destination, a description of entered locations is displayed from the file.

5. BLOCK DIAGRAM



FLOWCHART FOR DIJKSTRA ALGORITHM



6. REQUIREMENTS

There are several requirements that take part in the process of creating and implementing the service. The list of all software and hardware involve in the development process are:

6.1 SOFTWARE REQUIREMENTS

Software type	**Description/Version**
Operating System	Windows XP / Vista / 7 / 10
Code Editing	DEV-C++, Notepad, Notepad++
Web Browser (Information)	Google Chrome, Mozilla Firefox, Microsoft Edge
Presentation and Report Processing	Microsoft Word, Microsoft PowerPoint

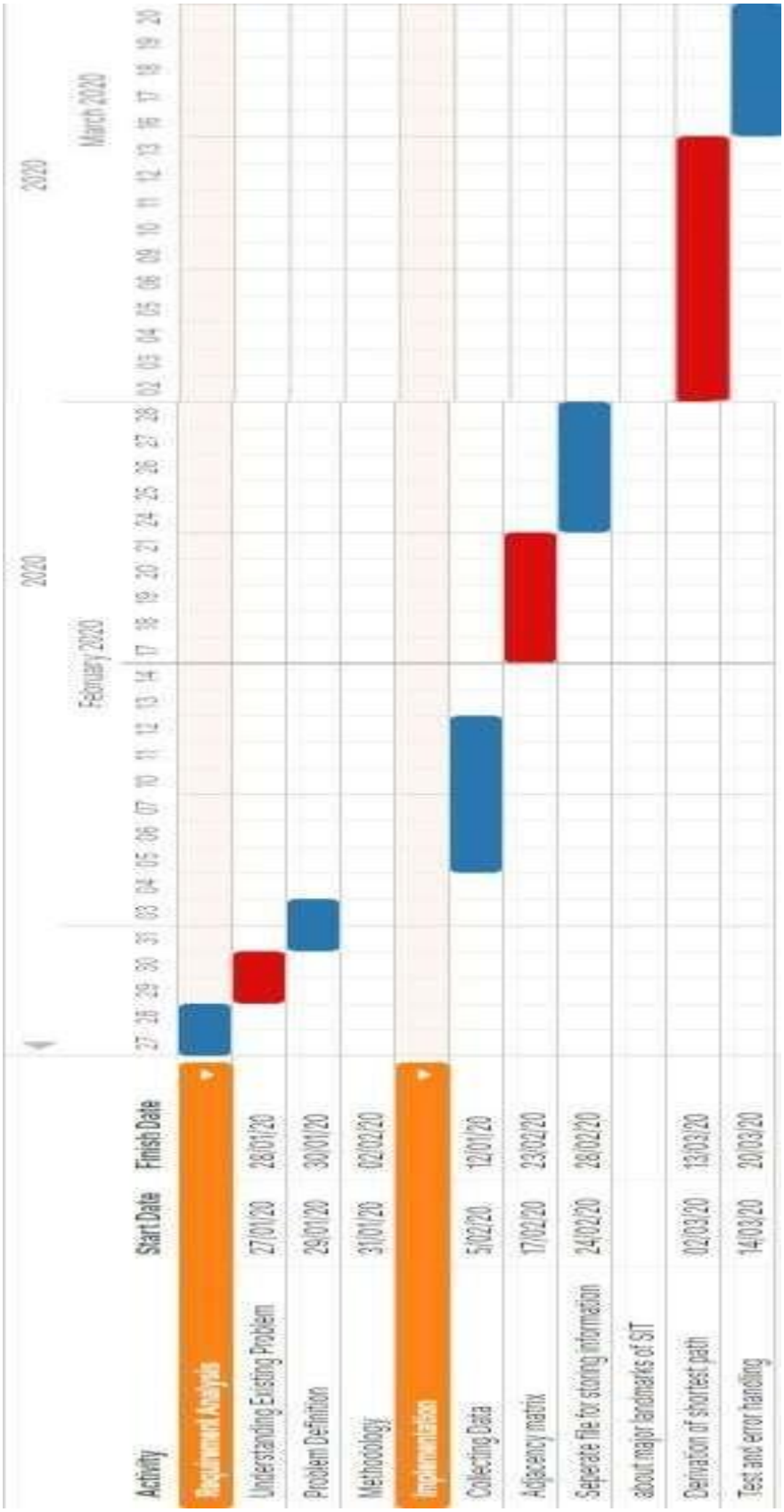
6.2 HARDWARE REQUIREMENTS

Item Name	**Description/Version**
Laptop	Development Specification • Intel Pentium IV or Higher • RAM 256 MB • Memory 1 GB

7. TEAM CONTRIBUTION

Members	Contribution
Ayush Tiwari(13)	File handling, Shortest path(),Case Insensitive()
Himank Jain(27)	Advanced Dijkstra algorithm, Adjacency matrix,Main ()

GANTT CHART (Please rotate the screen to view horizontally)



8.RESULTS AND SCREENSHOTS

1)Displaying list of locations:

```
        ROUTING SERVICE FOR SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE (SIT)


List of major landmarks of SIT to visit:
1) Hostel-A
2) Hostel-B
3) Hostel-C
4) Hostel-D
5) Hostel-E
6) Hostel-F
7) Hostel-G
8) Hostel-H
9) Hostel-S
10) Xerox-Shop
11) Symbus
12) Mess
13) Rangoli
14) Coffee-Stop
15) Academic-Block
16) Main-Gate
17) SUHRC-Boys-Hostel
18) SUHRC-Girls-Hostel
```

2)User input:

```
Enter the source node: Hostel-A

Hostel A
        A Girls Hostel for
        Symbiosis School of Culinary Arts
        Symbiosis and School of Biomedical Sciences`

Enter the destination node: Suhrc-boys-hostel

SUHRC BOYS
        A New Hostel for Boys: Located Within Symbiosis Institute of Medical Sciences`


Shortest Path:

        Start-> Hostel-A-> Mess-> Hostel-F-> SUHRC Boys Hostel-> End

The Shortest Distance between the nodes: 620 metres (approx)
```

3) When Academic-Block is the destination:

```
Enter the source node: Suhrc-girls-hostel

SUHRC GIRLS
        A New Hostel for Girls: Located within Symbiosis Institute of Medical Sciences`

Enter the destination node: Academic-block

Academic Block
        Heart of the College
        WiFi enabled
        Let the Lectures and Labs Begin`


Shortest Path:

        Start-> SUHRC Girls Hostel-> SUHRC Boys Hostel-> Academic-Block-> End

The Shortest Distance between the nodes: 701 metres (approx)

Want to go inside? (Y/N):
```

4) Inside the Academic-block:

```
Want to go inside? (Y/N): Y


2nd Floor of Academic Block of SIT :
1) Academic Block gate
2) S1
3) S8
4) Ds-lab
5) MT-lab
6) Amphithreatre
```

5) User input inside the Academic-block:

Academic-block-gate ---------> S8

```
Want to go inside? (Y/N): Y


2nd Floor of Academic Block of SIT :
1) Academic Block gate
2) S1
3) S8
4) Ds-lab
5) MT-lab
6) Amphithreatre

Enter the source node: academic-block-gate
Enter the destination node: S8

Shortest Path:

        Start-> Academic Block Gate-> Amphithreatre-> Stairs to Amphithreatre -> S7 / S8 / S9-> End

The Shortest Distance between the nodes: 30 metres (approx)
```

Academic-block-gate ---------> Ds-lab

```
2nd Floor of Academic Block of SIT :
1) Academic Block gate
2) S1
3) S8
4) Ds-lab
5) MT-lab
6) Amphithreatre

Enter the source node: academic-block-gate
Enter the destination node: Ds-lab

Shortest Path:

       Start-> Academic Block Gate-> Amphithreatre-> Stairs to Amphithreatre -> S7 / S8 / S9-> Ds-lab / Java-lab / OS-lab-> End

The Shortest Distance between the nodes: 40 metres (approx)
```

Academic-block-gate ---------> S1

```
Want to exit? (Yes/No): no


2nd Floor of Academic Block of SIT :
1) Academic Block gate
2) S1
3) S8
4) Ds-lab
5) MT-lab
6) Amphithreatre

Enter the source node: Academic-block-gate
Enter the destination node: S1

Shortest Path:

        Start-> Academic Block Gate-> Stairs-> S1 / S2 / S3-> End

The Shortest Distance between the nodes: 28 metres (approx)
```

9. REFERENCES

1) G. Deepa, Priyank Kumar, A. Manimaran, K. Rajakumar, V. Krishnamoorthy, Dijkstra Algorithm Application: Shortest Distance between Buildings, *International Journal of Engineering & Technology.*

2) P. Manikandan*, S.Yuvarani*, Implementation Of Shortest Path Algorithm Using In C, *International Journal of Computers & Technology, Vol. 2 Issue 6.*

3) Harmanjit Singh, Richa Sharma, Role of Adjacency Matrix & Adjacency List in Graph Theory, *International Journal of Computers & Technology, Volume 3, No. 1.*

4) Ojekudo, Nathaniel Akpofure (PhD) & Akpan, Nsikan Paul (PhD), An application of Dijkstra's Algorithm to shortest route problem, *IOSR Journal of Mathematics (IOSR-JM).*