# MINOR-2 PROJECT

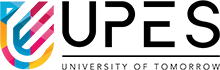
**FINAL REPORT**

## For

**COMPUTER BASED SIMULATION OF RAIL NETWORK TO REALIZE SHORTEST PATH ALGORITHM**

Submitted By

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **SAP ID** | **ROLL NO** | **SPECILIZATION & BATCH** |
| ROHAN BAKSHI | 500096302 | R2142211066 | BTECH CSE CCVT B-6 |
| VILISH KUMAR | 500095673 | R2142211034 | BTECH CSE CCVT B-6 |
| ASHISH KUKRETI | 500096132 | R2142211009 | BTECH CSE CCVT B-6 |



## Department of Systemics School of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

## DEHRADUN- 248007. Uttarakhand

Mr. Alok Jhaldiyal Dr. Hitesh Kumar

**Project Guide Cluster Head**

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Title** | **Page No.** |
| 1 | Abstract | 3 |
| 2 | Introduction | 4 |
| 3 | Literature Review | 4 |
| 4 | Problem Statement | 4 |
| 5 | Objectives | 4 |
| 6 | Methodology | 5 |
| 7 | Algorithm | 5-6 |
| 8 | SWOT Analysis | 6-7 |
| 9 | PERT chart | 8 |
| 10 | Result | 9-11 |
| 11 | References | 11 |

**Project Title**

Computer Based Simulation of Rail Network to Realize shortest Path Algorithm

### Abstract

The increasing complexity of modern rail networks demands efficient route planning systems to enhance transportation logistics and passenger experience. This project focuses on developing a computer-based simulation that models a rail network, with railway stations as nodes and the distances between stations as edges. The primary objective is to implement Dijkstra's algorithm to determine the shortest path between two railway stations.

The simulation begins by representing the rail network as a graph, where each node corresponds to a railway station and the edges represent the distances between stations. Real-world data, such as track lengths or travel times, can be utilized to accurately reflect the rail network's characteristics. The Dijkstra's algorithm is then applied to find the shortest path from a source station to a destination station, considering the cumulative distances along the railway edges.

Three Categories of it’s are:

1. **Passenger Travel and Commuting:** It helps in enhancing the efficiency of public transportation systems, reducing travel times, and improving the overall experience for passengers.
2. **Freight Logistics and Transportation:** This is crucial for minimizing transportation costs, ensuring timely deliveries, and optimizing the use of freight rail networks, supply chain by identifying the shortest and most cost-effective paths for transporting goods between production facilities, distribution centers, and markets.
3. **Railway Operations and Planning:** It helps in minimizing travel time, reducing congestion, and improving overall efficiency in the movement of trains.

Problems in managing are:

1. **Dynamic Network Changes:** Managing real-time updates and adjustments in the simulation to account for dynamic changes in the rail network, such as maintenance or unexpected closures.
2. **Data Accuracy and Reliability:** Ensuring the precision and dependability of simulation outcomes by regularly validating and updating input data to reflect the current state of the rail network.
3. **Scalability and Computational Efficiency:** Optimizing the simulation's ability to handle larger and more complex rail networks by improving computational efficiency and scalability.

### Introduction

Computer-based simulation, employing a graph-based model of the rail network, where nodes signify railway stations and edges denote the distances between stations. Utilizing real-world data, such as track lengths and travel times, the simulation leverages the attributes of the actual rail network. The focal point of this endeavor is the implementation of Dijkstra's algorithm, aimed at determining the shortest path between any two railway stations. This simulation not only represents the rail network's topology but also calculates optimal routes by considering cumulative distances along railway edges, contributing to the advancement of efficient and effective rail transportation systems.

### Literature Review

Computers can simulate rail networks, researchers highlight the importance of using graph-based models to accurately depict the details of railways. These models use nodes to represent stations and edges to show distances between them. Scholars are keen on using real-world data, like track lengths and travel times, to make simulations more realistic and reflective of actual rail systems. A main focus in this research is on Dijkstra's algorithm, a powerful tool for finding the best paths between stations in simulations. By combining this algorithm with real-world data, researchers aim to make rail transportation systems more efficient. Real-life examples, like [specific examples or case studies], illustrate how these simulations can optimize routes and improve overall network efficiency. Overall, the literature emphasizes the importance of using computer-based simulations, especially those incorporating Dijkstra's algorithm, to enhance and shape modern rail transportation networks.

### Problem Statement

Rail transportation currently struggles with inefficient route planning and suboptimal resource use due to the limitations of traditional methods. The proposed solution involves a computer-based simulation using Dijkstra's Algorithm for optimal route planning, aiming to overcome these challenges. However, developing and implementing this simulation present their own complexities, necessitating a well-defined problem statement.

### 5.Objectives

* + Computer based simulation of rail network in which (Railway Stations are my Node and distance is edge)
  + Implementation of algorithms to find shortest path.
  + Comparison of implemented algorithms

### Methodology

### Graph Representation: The program represents the metro stations and connections as a graph, where each station is a vertex and each metro line between stations is an edge. It uses a hashmap to store vertices and their adjacent stations.

### Graph Operations: It supports various operations like adding and removing stations, adding and removing connections between stations, checking for the existence of stations and connections, and calculating the number of stations and connections.

### Path Finding Algorithms: The program implements Dijkstra's algorithm to find the shortest path between two stations based on either distance or time. It uses a priority queue (heap) to efficiently find the shortest path.

### User Interface: The application provides a user-friendly interface through the console. It allows users to view the metro map, list all stations, find the shortest distance or time between stations, and find the shortest path between stations, displaying the interchanges and relevant information.

### Error Handling: The program includes error handling mechanisms to deal with invalid inputs and ensure the robustness of the application.

### Modularity: The code is structured into methods and classes, promoting modularity and reusability. Each method or class performs a specific task, enhancing code readability and maintainability.

### Dynamic Updates: The application allows dynamic updates to the metro map, enabling addition, removal, or modification of stations and connections without compromising functionality.

### ALGORITHM

### Graph Traversal (DFS):

### The hasPath method implements Depth-First Search (DFS) to determine whether a path exists between two given stations. It recursively explores adjacent stations to find a path.

### Dijkstra's Algorithm:

### The dijkstra method implements Dijkstra's algorithm to find the shortest path between two stations based on either distance or time. It uses a priority queue (heap) to efficiently select the next station with the minimum cost.

### Shortest Path Retrieval:

### The Get\_Minimum\_Distance and Get\_Minimum\_Time methods utilize Dijkstra's algorithm to find the shortest path between two stations based on either distance or time. They also retrieve the path (sequence of stations) with minimum distance or time.

### Interchange Identification:

### The get\_Interchanges method identifies interchanges (stations where passengers need to change lines) along a given path. It parses the path string and identifies stations where different lines intersect.

### User Interface Handling:

### The main method implements an interactive console-based user interface to interact with the Metro Map application. It handles user inputs, displays menu options, and executes corresponding functionalities based on the selected option.

### 8.1 AREA OF ALGORITHM

* **Graph Theory:**

The code implements graph data structures and algorithms to represent and traverse the metro network. It utilizes adjacency lists to store station connections and performs graph traversal operations such as DFS (Depth-First Search) to check path existence.

* **Shortest Path Algorithms:**

Dijkstra's algorithm is employed to find the shortest path between two stations in terms of either distance or time. This algorithm efficiently computes the shortest path by greedily selecting the next station with the minimum cost.

* **Path Retrieval:**

The code includes functionality to retrieve the shortest path between stations. This involves tracing back from the destination to the source using pointers to reconstruct the path.

* **User Interface Design:**

While not strictly algorithmic, the code includes features for user interaction and input handling. It provides a console-based menu system for users to choose various actions, input stations, and view results.

* **String Manipulation:**

The code utilizes string manipulation techniques to parse and format station names and paths. This includes splitting strings based on delimiters and concatenating strings to represent paths.

### SWOT Analysis

### A SWOT analysis is a strategic planning tool used to assess the strengths, weaknesses, opportunities, and threats associated with a project, business, or situation.

### 8.1 Strengths:

### Modular Design: The code is well-structured into classes and methods, enhancing readability and maintainability.

### Efficient Algorithms: Implementation of Dijkstra's algorithm ensures efficient computation of shortest paths, crucial for metro navigation.

### Graph Representation: Effective use of graph data structures facilitates the representation of metro stations and their connections.

### User Interface: The console-based user interface provides a straightforward interaction method, allowing users to input their queries easily.

### Error Handling: The code includes error-checking mechanisms to handle invalid inputs and ensure robustness.

### 8.2 Weaknesses:

### Limited Scalability: The code may face challenges in handling large-scale metro networks efficiently due to its reliance on adjacency lists for graph representation.

### Single Threaded: The application appears to be single-threaded, potentially limiting its performance for concurrent user requests in real-world scenarios.

### Input Validation: While basic error handling is implemented, more comprehensive input validation could enhance the robustness of the application further.

### 8.3 Opportunities:

### Graph Optimization: Exploring advanced graph optimization techniques could further improve the efficiency of pathfinding algorithms, enhancing the overall performance.

### GUI Implementation: Developing a graphical user interface (GUI) could enhance user experience and make the application more intuitive to navigate.

### Integration with Real-time Data: Integrating with real-time metro data APIs could provide users with up-to-date information on schedules, delays, and station statuses.

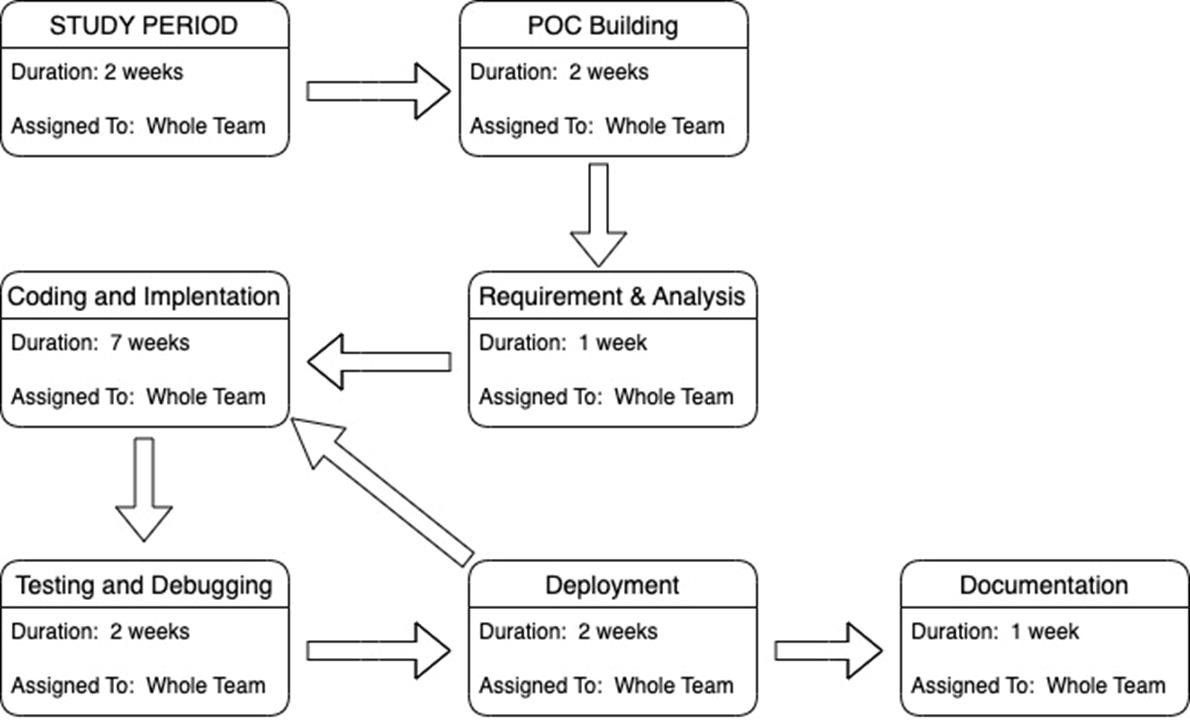
### 8.4 Threats:

### Competitive Apps: Competition from existing metro navigation applications with more features and established user bases could pose a threat to user adoption.

### Technological Changes: Rapid advancements in technology could necessitate updates and adaptations to ensure compatibility and performance on new platforms and environments.

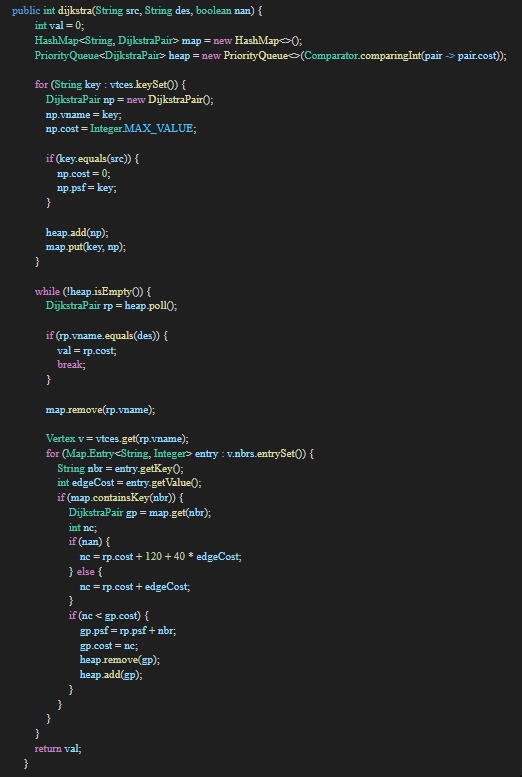
### Security Concerns: Potential security vulnerabilities in the application, such as input validation flaws or data leakage, could pose risks to user data and privacy.

### PERT Chart

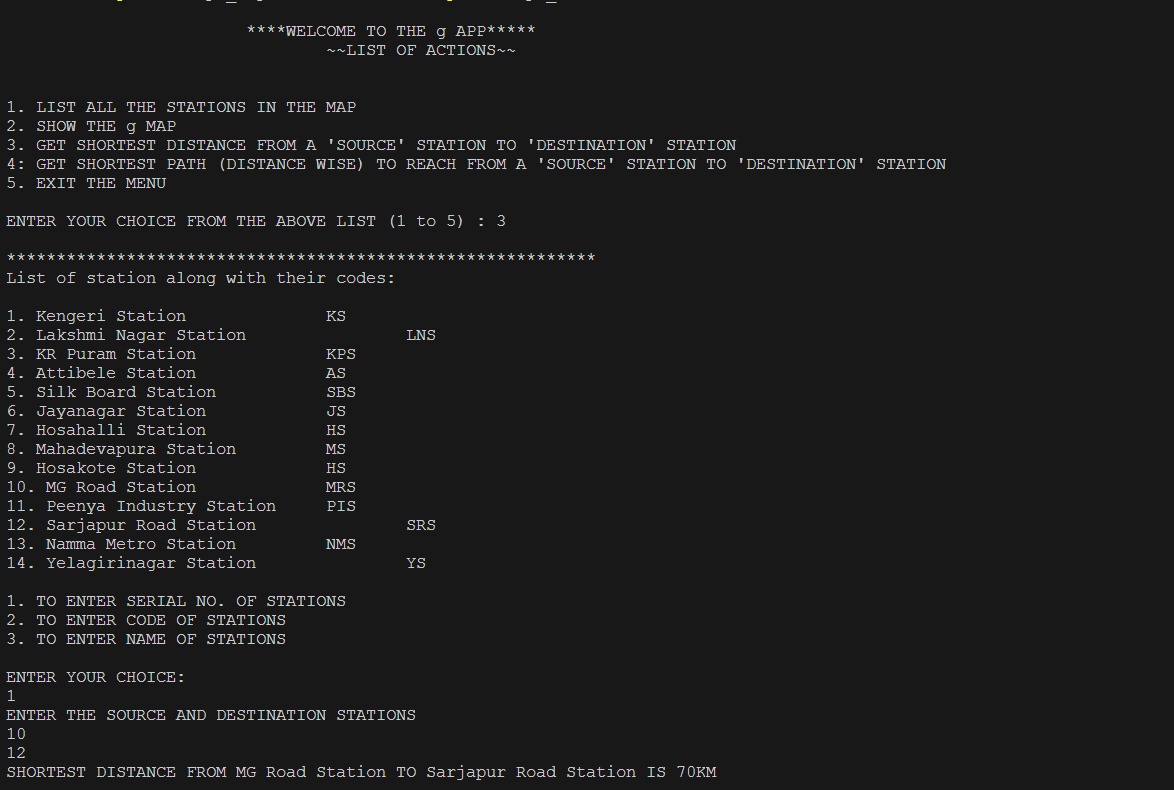


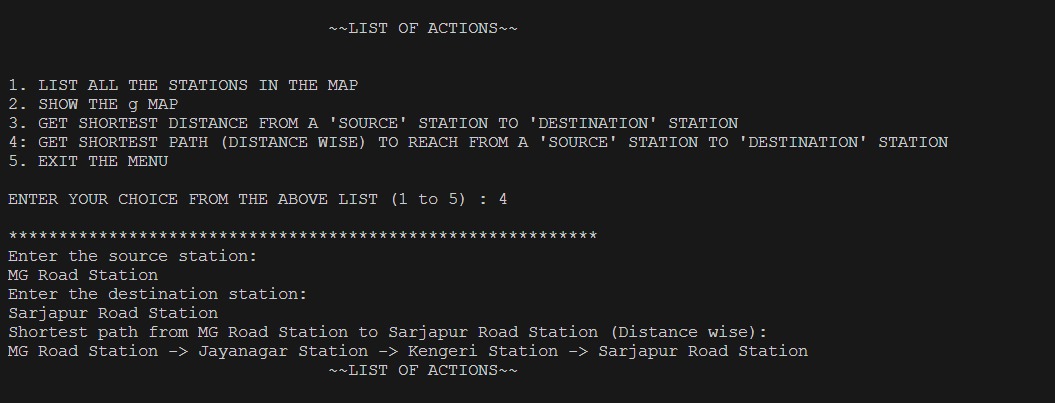
1. **Result**

**Figure 1:**

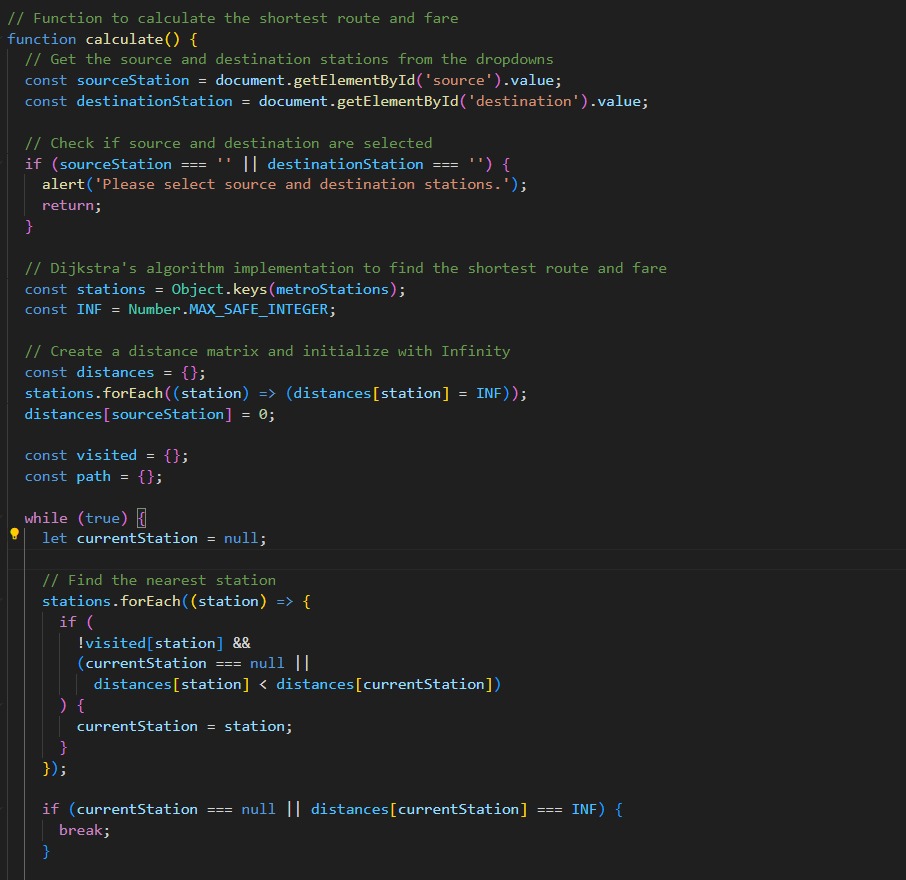
****

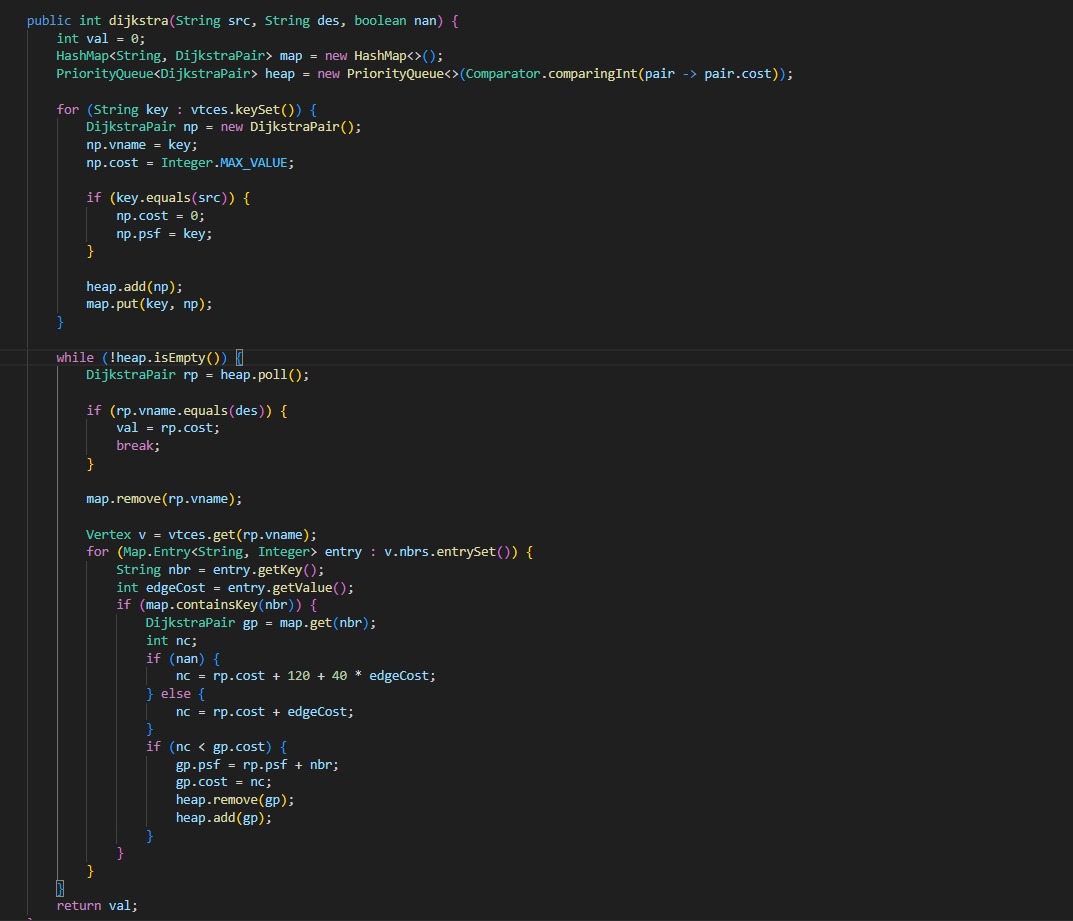
**Output:**



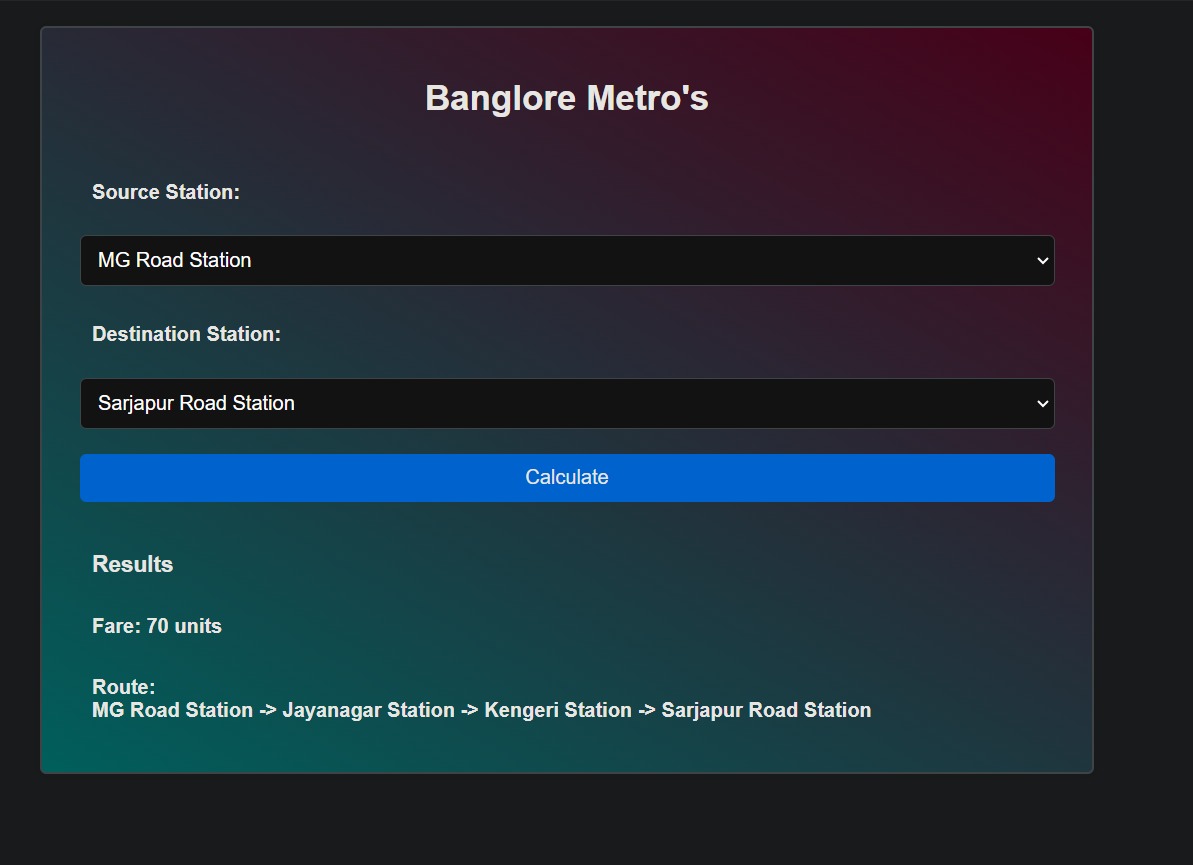


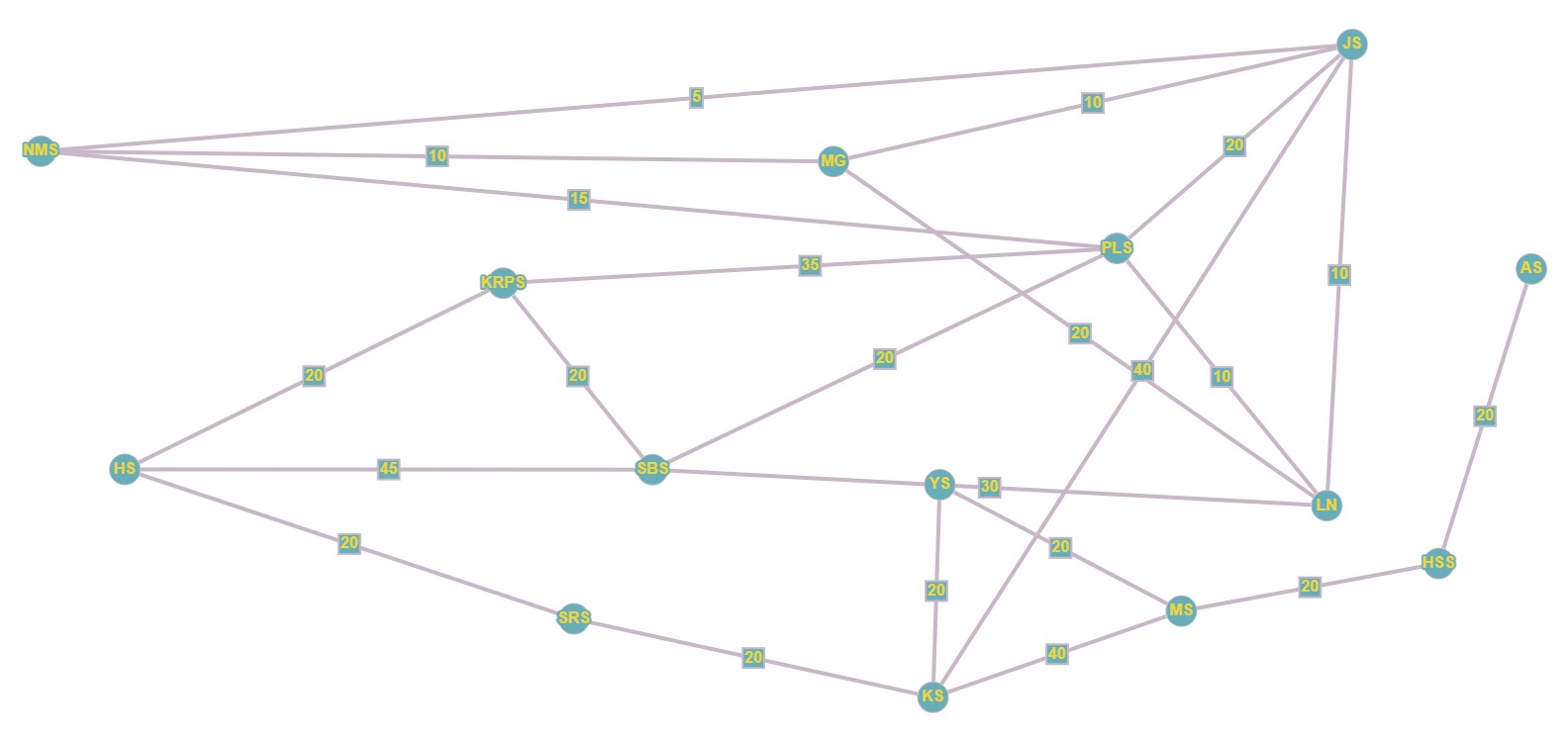
**Figure 2:**



****

**Output:**

****

****

1. **References**
2. Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269-271.
3. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
4. Federal Railroad Administration (FRA). (2021). Rail Network Operations. Retrieved from <https://railroads.dot.gov/>
5. Association of American Railroads (AAR). (2021). Railroads and Intermodal Transportation. Retrieved from <https://www.aar.org/>
6. Transport Canada. (2021). Rail Transportation. Retrieved from https://tc.canada.ca/en/transportation-rail