CSE-AI (B) Group 8: ADS Mid-Sem

**Indian Map Navigator with Interactive Map using Dijkstra's Algorithm**

## **Abstract**

This report presents the development and implementation of an Indian Map Navigator with an interactive map interface, utilizing Dijkstra's algorithm for optimal route finding. The project demonstrates the practical application of graph theory and advanced data structures in solving real-world problems, specifically in the domain of route planning and navigation systems.

## **Problem Statement**

This project focuses on implementing Dijkstra's algorithm, a fundamental graph traversal algorithm, to find the shortest path between cities in India. The implementation is enhanced with an interactive web-based interface, providing users with a visual representation of the map and the calculated routes.

Develop a web-based application that allows users to:

1. Visualize major Indian cities on an interactive map
2. Select start and end cities directly on the map
3. Calculate and display the optimal route between selected cities
4. Provide a visual representation of the calculated route on the map

The core challenge lies in efficiently implementing Dijkstra's algorithm to handle route calculations and integrating it with a user-friendly web interface.

## **Objectives**

1. Implement Dijkstra's algorithm for finding the shortest path in a weighted graph representing Indian cities and their connections
2. Develop a backend server to handle route calculations and city data management
3. Create an interactive frontend with a map interface for user interactions and route visualization
4. Demonstrate the practical application of graph theory and advanced data structures in solving real-world problems

## **Methodology**

### **Data Structure Design**

The project utilizes the following key data structures:

1. **Graph Representation**: An adjacency list structure to represent the network of Indian cities and their connections.
2. **Priority Queue**: Implemented implicitly in Dijkstra's algorithm for efficient selection of the next node to process.
3. **Hash Table**: Used for quick lookup of city data and maintaining visited nodes during algorithm execution.

### **Algorithm Implementation**

Dijkstra's algorithm is implemented with the following steps:

1. Initialize distances to all nodes as infinite, except the start node (distance = 0)
2. Create a priority queue and add the start node
3. While the priority queue is not empty:

a. Extract the node with the minimum distance

b. For each neighbor of the extracted node:

* + Calculate the distance through the current node
  + If this distance is less than the previously known distance, update it
  + Add the neighbor to the priority queue

1. Reconstruct the path from the start node to the end node

### 

### **Frontend Development**

The frontend is implemented using HTML, CSS, and JavaScript, featuring:

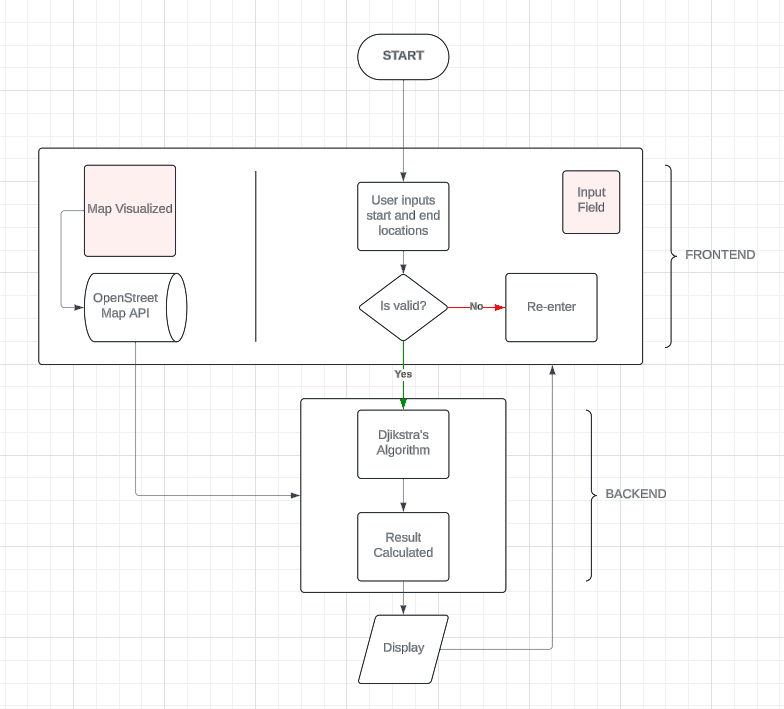
1. An interactive map using the Leaflet.js library and OpenStreetMap API
2. City markers for user selection
3. Route visualization on the map
4. A sidebar for displaying route information

### **Backend Development**

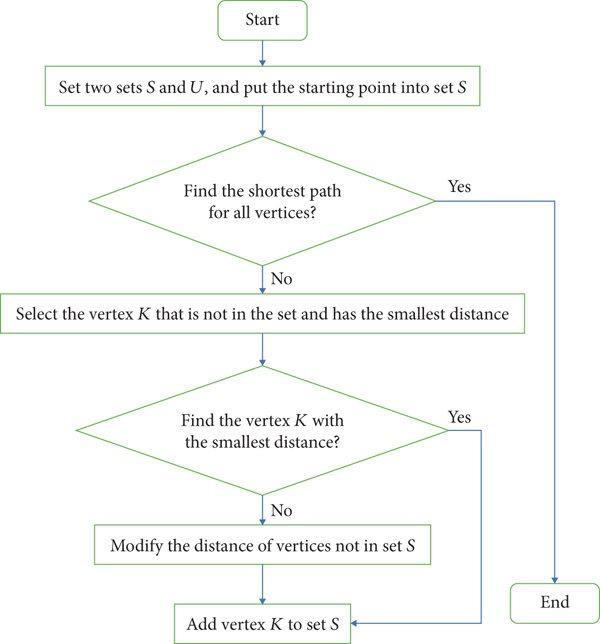
The backend server is developed using Node.js and Express.js, providing RESTful API endpoints for:

1. Fetching city data (including coordinates)
2. Calculating optimal routes between two cities

**Process Flow Diagram**



**Dijkstra's algorithm flowchart :**



**LITERATURE REVIEW**

**[1] “Graph based navigation system”** : This paper presents the development of a web-based navigation application for SRMIST Chennai campus, leveraging GPS, compass, and accelerometer technologies. It integrates campus floor plans and uses the Open Street Map API for precise navigation. Developed with an agile approach, the app helps users find campus locations and engage with the community through real-time updates and discussions.

**[2] “Map Matching Algorithm:** Trajectory and Sequential Map Analysis on Road Network” : GPS tracking data is vital for systems like traffic assessment, routing, and fleet management. However, GPS accuracy can be compromised by weak signals and obstacles, especially in India. Map matching improves road data accuracy by minimizing Frechet distance. This work focuses on enhancing pathfinding by locating Frechet distance in free space, addressing trajectory data errors, and improving road network mapping with sequential computation.

**[3] “A Comparative Study on Shortest Path Visualization using Artificial Intelligence”**  : This study presents a GUI-based tool for finding the shortest path using Dijkstra and A\* algorithms, developed with Java AWT and SWING. The tool visualizes the results and stores output in a database. Comparative analysis showed that A\* is more efficient and faster than Dijkstra in pathfinding. This research highlights the tool's potential applications in gaming, robotics, logistics, and crowd simulation.

**[4] : “Advanced traveler information system for Hyderabad City”** : This paper presents a GIS-based Advanced Traveler Information System (ATIS) for Hyderabad City, India, developed using ArcView GIS. The system offers comprehensive travel-related information, including road networks, hospitals, offices, and tourist spots, aiding in route selection and trip planning. The user-friendly ATIS can be utilized in public transport hubs and personal computers to assist travelers.

**[5] : “The Algorithms Behind The Working Of Google Maps” :** Google Maps, launched in 2005, has grown into a widely-used service with over a billion monthly users. Built with C++, JavaScript, XML, and Ajax, the app uses Dijkstra's and A\* algorithms to find the shortest path between two points. These graph-based algorithms optimize route calculations, with Dijkstra's being a well-known greedy algorithm for finding the shortest paths in a graph.