RapidResponse:
Optimized Emergency Routing System

PROJECT SYNOPSIS

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY
MASTER OF COMPUTER APPLICATIONS

Submitted by:

Abhinav Negi, Rishav Raj and Daksh Rautela

GRAPHIC ERA UNIVERSITY,

DEHRADUN

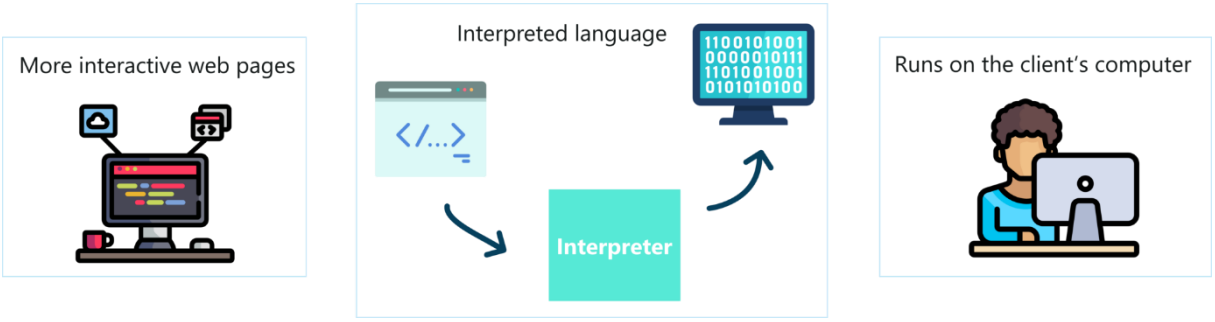
1. Introduction:

RapidResponse:

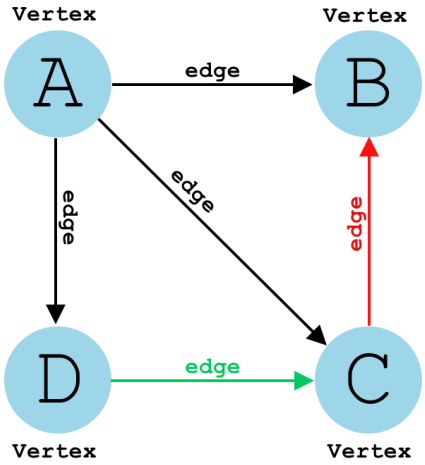
- RapidResponse is a Data Structures and Algorithms (DSA) project aimed at designing an efficient pathfinding system for emergency vehicles.
- The goal is to provide the shortest and fastest route to a destination, ensuring that emergency services such as ambulances, fire trucks, and police vehicles can reach their target locations as quickly as possible.
- This project leverages graph-based algorithms and real-time map data to optimize route planning.
- The project is currently under implementation and is being developed by a team of three collaborators: Abhinav, Daksh, and Rishav.

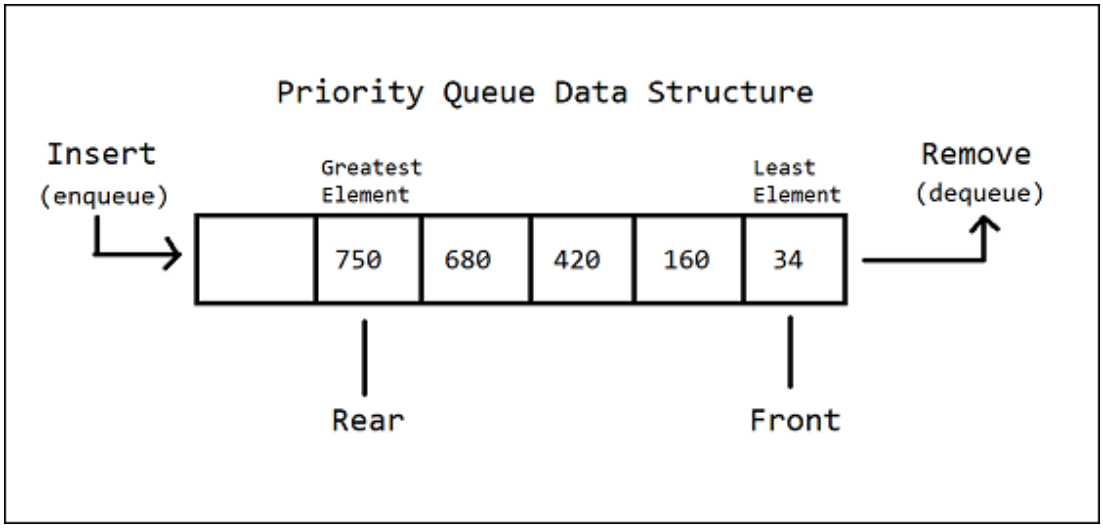
2. Requirements Analysis:

- **Programming Language: JavaScript**
  JavaScript is a high-level, versatile programming language primarily used for creating interactive and dynamic web content. It's supported by all modern browsers and is essential for client-side web development. JavaScript also powers server-side applications via environments like Node.js.



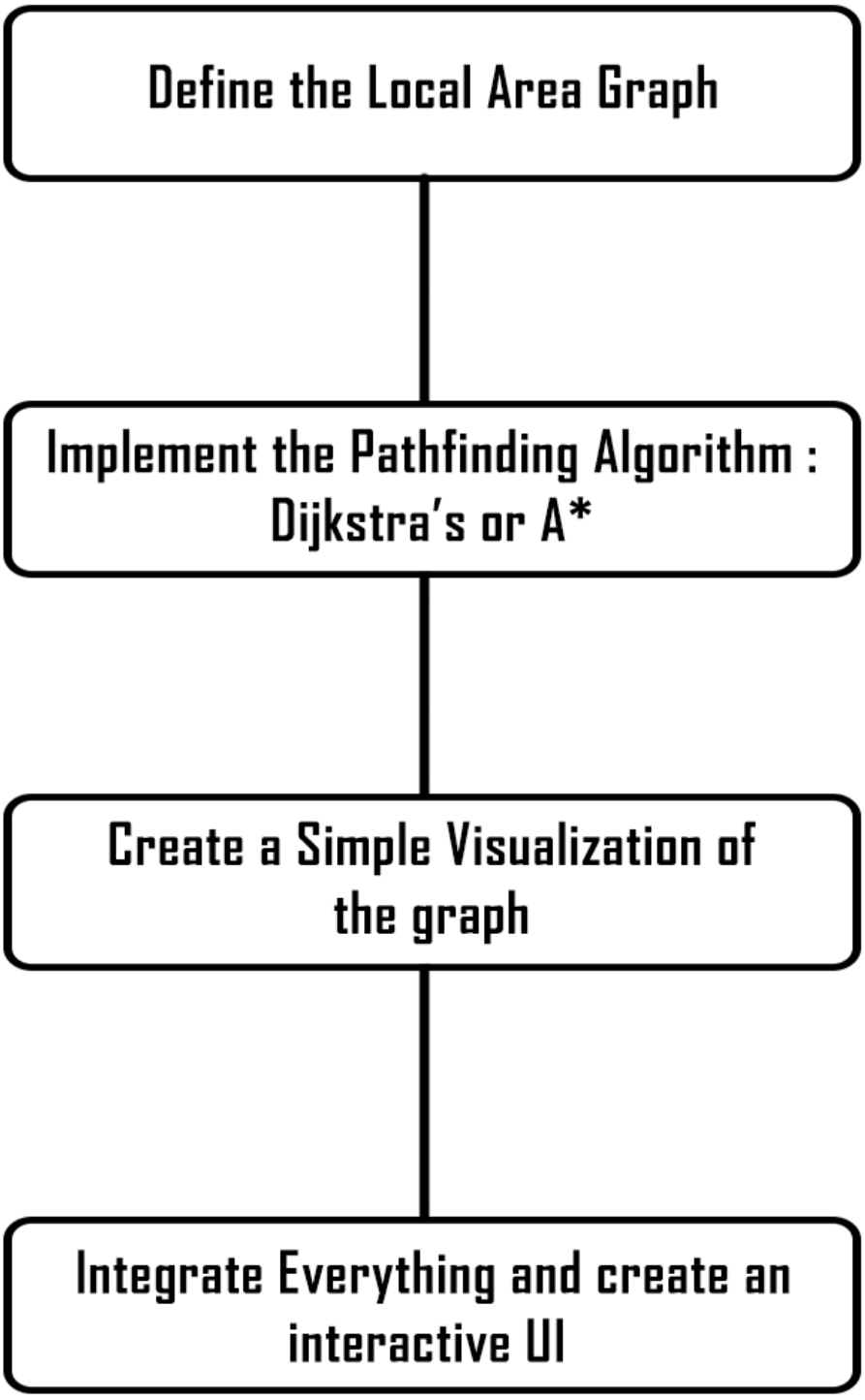
- **Data Structures:**
  - **Graphs (Adjacency List/Matrix):**
    Graphs represent relationships between entities (nodes/vertices). An adjacency list uses a list where each node stores its neighbors, making it space-efficient for sparse graphs. An adjacency matrix is a 2D array indicating edge presence and weights between nodes, better for dense graphs.
  - **Priority Queues:**
    A priority queue is a special type of queue where elements are dequeued based on priority rather than insertion order. It's commonly implemented with heaps and is key in algorithms like Dijkstra's.





- **Algorithms:**
  - **Dijkstra's Algorithm:**
    Dijkstra's algorithm finds the shortest path from a starting node to all other nodes in a weighted graph with non-negative edges. It uses a priority queue to explore the nearest unvisited node first.
  - **A* Search:**
    A* (A-star) is an informed search algorithm that finds the shortest path by combining the actual cost to a node with a heuristic estimate to the goal. It's widely used in pathfinding for games and robotics.

3. Methodology:



Define the Local Area Graph

- What to do:
    - Manually create a simplified graph of your local area. For example, pick 10-20 key intersections or landmarks (nodes) and the roads connecting them (edges). Assign weights to edges based on distance or estimated travel time.

Implement the Pathfinding Algorithm

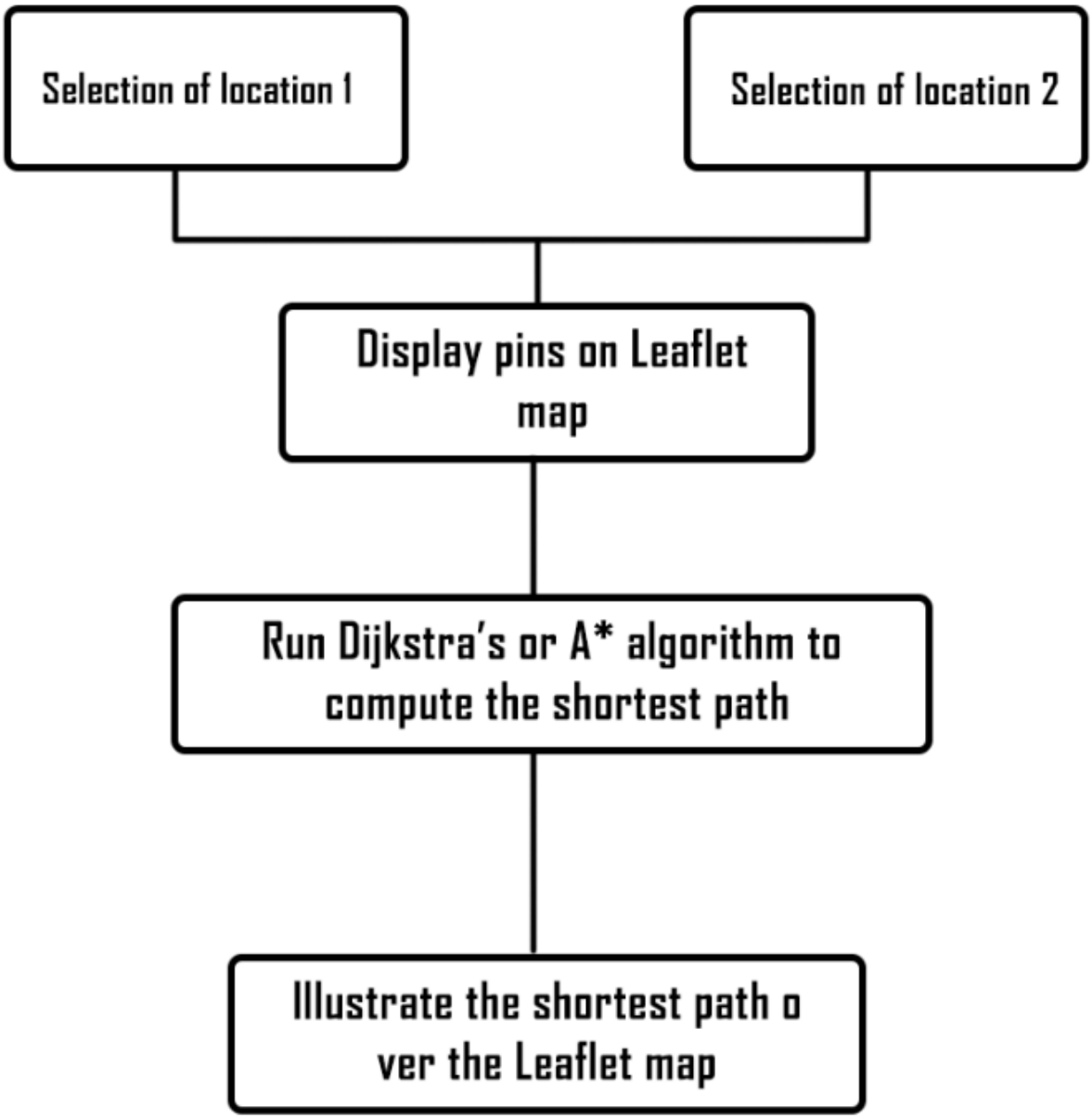
- What to do:
    - Choose and code a shortest-path algorithm. Dijkstra's is a good starting point since it's simpler and guarantees the shortest path. If you want to optimize further, try A* with a heuristic (e.g., straight-line distance).

Create a Simple Visualization

- What to do:
    - Display the graph and the computed route on a 2D map. Since we can't use Google APIs, we will be using a basic canvas or SVG to draw nodes and edges.

Integrate Everything & create and interactive UI

- What to do:
    - Combine the graph, algorithm, and visualization into a working prototype. Allow users to select a start and end point (e.g., via dropdowns or clicks) and show the route.



4. Facilities required for proposed work:

1. Development Environment
    a. Code Editor:
        i. Visual Studio Code (most popular choice)
        ii. Alternatives: Sublime Text, Atom, WebStorm
    b. Node.js:
        i. To run JavaScript outside of the browser.

2. Database
    a. MongoDB:
        i. it is a NoSQL document database primarily used for storing and managing large volumes of data in JSON-like formats. It's known for its scalability, flexibility, and ease of use, making it suitable for various applications, including web applications, mobile applications, and IoT devices.

3. Map Visualization
    a. Leaflet:
        i. it is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 42 KB of JS, it has all the mapping features most developers ever need.

5. BIBLIOGRAPHY:

A list of sources that provided crucial information and helped with the creation of this project:

- Dijkstra's algorithm
  - https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php
- A* algorithm
  - https://www.datacamp.com/tutorial/a-star-algorithm
- Priority queue
  - https://www.programiz.com/dsa/priority-queue
- Leaflet
  - https://leafletjs.com/