Department of Computer Science and Engineering

Course: CSE 246 (Algorithms)

Section - 02

Academic Session: Fall 2025

Project Report

Disaster Response & Rescue System

Submitted to

Dr. Md. Tauhid Bin Iqbal

Assistant Professor

Department of Computer Science and Engineering

East West University

Submitted by

Name	ID
Hafsa Ferdousi	2023-3-60-321
Nafisa Naowar	2023-3-60-326
Md. Jafor Sadik	2023-3-60-372
Israt Jahan Naima	2023-3-60-512

Date of Submission

December 17, 2025

1. Introduction

- ## Problem statement:

A major disaster creates urgent and interconnected challenges for emergency response: assessing and prioritizing victims, selecting appropriate hospitals, deploying limited rescue teams and vehicles, and navigating disrupted traffic networks. Time-sensitive, suboptimal decisions in this dynamic environment can lead to avoidable loss of life and inefficient use of resources.

This "Disaster Response & Rescue System" project tackles these issues by creating an algorithmic simulation to optimize emergency logistics. Using core data structures and graph algorithms, the system will automate victim triage, calculate the fastest viable routes to hospitals, and intelligently allocate all available resources—rescuers, ambulances, and hospital beds—while accounting for real-world constraints like road closures, transport speeds, and medical capacity. The system must also process live updates (e.g., newly blocked roads) and validate all inputs to ensure reliability.

The fundamental goal is to minimize response time and maximize lives saved through the efficient coordination of multiple, resource-constrained decisions within a changing environment. This project illustrates how classical algorithms—including Priority Queues, Dijkstra's Algorithm, Floyd–Warshall, and Greedy approaches—can be combined into a C++ system to model and address critical real-world problems in disaster management.

- ## Motivation and objectives:

The development of the "Disaster Response & Rescue System" is directly motivated by the urgent, real-world crisis caused by the devastating earthquake that struck Bangladesh on November 21, 2025. This catastrophic event, with its epicenter near Dhaka, tragically resulted in at least 10 fatalities and left over 600 people injured. Witnessing the immediate chaos—from building damage and blocked roads to overwhelming demand on hospitals—revealed a stark reality: our current disaster response systems are dangerously unprepared for such rapid-onset emergencies.

This project emerged from witnessing those critical shortcomings firsthand: the agonizing delays in ambulance dispatch, the inefficient routing around collapsed infrastructure, the chaotic triage of victims, and the desperate shortage of coordinated resources. As a densely populated, seismically vulnerable region, Bangladesh represents the urgent need for a smarter, algorithmic approach to emergency management.

Our motivation is to transform this tragic experience into a practical solution. By applying core computer science algorithms, this system aims to simulate how technology can save lives—by optimizing victim prioritization, calculating the fastest safe routes, and dynamically allocating limited rescue teams, vehicles, and hospital capacity. This is not merely an academic exercise; it is a direct response to a national tragedy, with the goal of building a more resilient and responsive framework for future disasters

This Disaster Response & Rescue System has a three-tiered set of goals.

Its primary objectives focus on optimizing four critical emergency functions:

1. Victim Triage: Using a priority queue to instantly classify victims by injury severity.

2. Route Planning: Employing algorithms like Dijkstra's to find the shortest, safest paths, adapting in real-time to road blockages.

3. Resource Allocation: Intelligently assigning limited rescue teams, vehicles (ambulances/vans), and hospital beds to maximize efficiency.

4. Hospital Matching: Creating a smart system to send victims to the most appropriate hospital based on specialization, capacity, and distance.

The secondary objectives aim to make the system robust and insightful:

- **Resilience:** Handling live updates (e.g., new road closures) without performance loss.
- **Awareness:** Providing a comprehensive, real-time dashboard of all resources and operations for coordinators.
- **Extensibility:** Building a modular foundation for future upgrades and demonstrating the value of an algorithmic approach.

Ultimately, the project's broader impact goals are to save lives by minimizing critical response times, support emergency coordinators with data-driven decisions, and advocate for systematic technological preparedness in disaster-prone regions like Bangladesh.

2. Literature Review / Background Study

2.1 Relevant theories, methods, or existing works

1.Relevant Theories

a) Priority Scheduling Framework

The system employs priority scheduling, processing tasks (victims) based on their urgency level. A heap-based priority queue guarantees that those with life-threatening injuries receive immediate attention, mirroring triage protocols in emergencies that prioritize resource use to optimize survival rates.

b) Graph and Shortest Path Algorithms

The disaster zone is represented as a weighted undirected graph, with zones as nodes and roads as edges weighted by distance. Algorithms like Dijkstra's and Floyd-Warshall calculate the fastest rescue paths, adapting to evolving road blockages for minimal travel time.

c) Greedy Algorithm Approach

Greedy strategies enable quick choices in resource-limited settings. The system picks the nearest available rescue team, suitable vehicle, and closest appropriate hospital first, delivering swift responses essential for effective disaster operations.

2. Methods Used

a) Priority Management Algorithm

A tailored comparator in a binary heap (priority queue) dynamically ranks victims by injury level and arrival time, enabling efficient dispatching with O(log n) time complexity.

b) Adaptive Path Optimization

A dual-algorithm setup combines real-time Dijkstra's for handling road obstructions with precomputed Floyd-Warshall all-pairs paths for rapid lookups amid changes, striking a balance between precision and speed.

c) Resource Matching Strategy

Hospital selection uses a weighted score factoring in distance, bed capacity, and specialized units. Greedy rules assign vehicles and teams via straightforward nearest-available logic, creating a practical, tiered process for rescue coordination.

3. Existing Works

a) Triage and Emergency Dispatch Tools

Real-world triage systems prioritize patients by injury severity, much like this project. Priority queues power ambulance dispatching and ER workflows globally.

b) Graph-Driven Routing in Disasters

Disaster studies commonly model urban areas as graphs, leveraging Dijkstra's or A* for route planning. Tools like emergency GPS and crisis navigation apps adopt these graph-based methods.

c) Heuristic Allocation Platforms

Disaster software relies on greedy or heuristic techniques for assigning ambulances, teams, and hospitals under tight timelines. This project's scoring for hospitals echoes contemporary logistics decision aids in emergencies.

2.2 Gap or limitations in existing approaches

1. Weak Handling of Changing Road Networks

Most disaster response platforms depend on fixed maps or pre-set routes. Sudden road closures from disasters overwhelm them, as they struggle to recalculate efficient paths on the fly, causing delays in critical rescues.

2. Suboptimal Victim Triage Processes

Conventional methods often stick to first-in-first-out queues or manual assessments, failing to prioritize the most severely injured. This oversight lowers survival chances in mass casualty events.

3. Overly Simplified Resource Choices

Many tools pick hospitals or assets solely by distance, overlooking essentials like open beds, specialized trauma care, or emergency departments. Consequently, close facilities get swamped while superior options sit idle.

4. Ineffective Resource Distribution

Current systems rarely optimize rescue team and vehicle assignments intelligently, leading to idle assets alongside overburdened ones. This mismatch hampers efficiency amid high-demand crises.

5. Fragmented Decision Pipelines

Disaster tools typically manage routing, prioritization, and allocation in isolation. Without a unified algorithmic framework, this fragmentation slows responses and complicates coordination.

6. Insufficient Error Handling and Reliability

Certain platforms mishandle faulty or unforeseen inputs, risking crashes or flawed outputs—especially problematic in high-stakes scenarios demanding rapid, precise data processing.

7. Challenges with Scaling in Live Scenarios

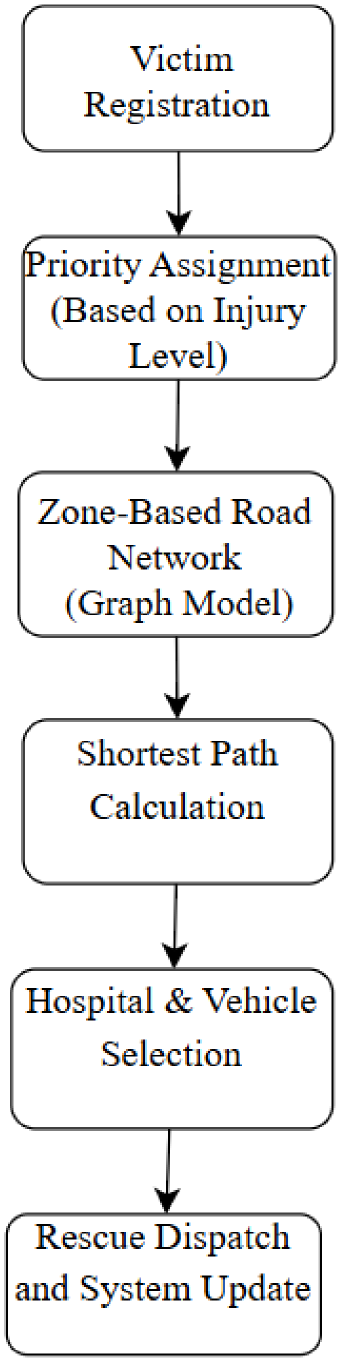
Real-time systems shy away from compute-heavy algorithms like all-pairs shortest paths due to speed issues. They either compromise on precision or buckle under growing loads from more zones, victims, or resources.

3. Proposed Method / Approach

3.1 System Design & Methodology

The Disaster Response & Rescue System uses a priority-driven, graph-based approach to deliver fast and effective emergency response during disaster situations. Victims are managed using a priority queue that ensures those with critical injuries are treated first. The disaster area is modeled as a weighted, undirected graph, allowing the system to dynamically recalculate optimal routes when roads are blocked or reopened. Hospital selection is handled through a multi-criteria weighted scoring method that considers distance, bed availability, and specialized medical facilities to ensure appropriate care. Rescue teams and vehicles are assigned using simple greedy strategies to reduce decision time in urgent scenarios. To support efficient routing, the system combines precomputed all-pairs shortest paths using the Floyd–Warshall algorithm with real-time shortest path computation using Dijkstra’s algorithm, enabling both rapid planning and adaptive response to changing road conditions.

Overall System Workflow:


Victim Registration
Priority Assignment (Based on Injury Level)
Zone-Based Road Network (Graph Model)
Shortest Path Calculation
Hospital & Vehicle Selection
Rescue Dispatch and System Update

The modular design ensures that changes in road conditions, hospital availability, or resource status dynamically affect the final rescue decision

3.2 Algorithms, models, or frameworks used

The system enhances emergency response by combining specialized algorithms and data structures for efficiency. It utilizes Dijkstra's and Floyd-Warshall algorithms for real-time routing, even around blocked roads; a priority queue for quickly triaging critical patients; and greedy heuristics for optimal resource distribution. Together, these components ensure rapid dispatch of rescue teams, immediate care for urgent cases, and overall maximization of lives saved during a disaster.

3.2.1 Victim Prioritization

Problem Description

A key challenge in disaster response is triage: deciding who to rescue first when there are more victims than resources, using injury severity as the sole criterion.

Algorithm Used

Priority Queue (Greedy Algorithm)

Model / Framework

Urgency-Based Scheduling Model

Analytical Justification

This approach is analytically justified because the problem is urgency-driven, and optimal results are achieved by consistently selecting the most severe local case. The greedy algorithm is suitable as prioritizing the highest-risk victim directly maximizes the likelihood of survival. A priority queue provides the necessary data structure to maintain this order, enabling real-time updates and immediate access to the next critical victim.

Why Alternative Algorithms Were Not Chosen

Algorithm	Used in System	Time Complexity	Space Complexity	Advantages	Limitations / Problems
Priority Queue (Greedy Severity)	Yes	O(log N)	O(N)	Prioritizes critical victims; fast for real-time emergencies	Ignores distance, vehicle availability; may not always optimize overall response time
FIFO	No	O(1)	O(N)	Simple, very easy to implement	Ignores severity; critical victims may be delayed
Round-Robin	No	O(1)	O(N)	Extremely simple; fair allocation across victims	Ignores severity; critical victims may wait longer; not optimal for emergencies

1. Priority Queue with Greedy Severity (Used in the System)
The system prioritizes victims based on injury severity using a priority queue. Critical cases are handled first, followed by moderate and minor injuries, with lower IDs breaking ties. This ensures urgent cases are attended quickly. Time complexity is O(log N) per insertion and space complexity is O(N). A limitation is that it does not consider distance or vehicle availability, which may sometimes delay transport.

2. First-Come, First-Served (FIFO)

Victims are attended in the order they are reported, ignoring severity. This method is simple with O(1) insertion time and minimal space use, but it may delay care for critical patients and is less suitable for emergencies.

3.2.2 Route Selection

Problem Description

After selecting a victim, the system must determine the shortest and safest route to a hospital while accounting for blocked or unavailable roads.

Algorithm Used

Dijkstra's Shortest Path Algorithm

Model / Framework

Graph-Based Road Network Model

Analytical Justification

The road network contains weighted edges representing distances, and all weights are non-negative. Dijkstra's algorithm is optimal for this scenario and efficiently recalculates routes when road conditions change, making it suitable for dynamic disaster environments.

Why Alternative Algorithms Were Not Chosen

In the disaster rescue management system, route selection is a critical real-time operation where injured victims must be transported to hospitals in the shortest possible time. Although several path-finding algorithms exist, not all are suitable for this problem due to performance, accuracy, and practical constraints.

1.Breadth-First Search (BFS) was not selected because it assumes all edges in the graph have equal weight. In this system, roads have different distances, and BFS may choose a route with fewer edges but longer total distance, leading to inefficient rescue operations.

2.Bellman–Ford Algorithm, while capable of handling weighted graphs, was also avoided because of its high computational cost. Its time complexity of O(V×E)O(V \times E)O(V×E) makes it unsuitable for real-time emergency systems where routing decisions must be made /quickly and repeatedly.

List	Algorithms	Algorithms which can be used	Time Complexity & Space complexity	Why Not Chosen / Limitations
1	Dijkstra	Best Choice	O((V+E) log V) O(V + E)	Limitation : It cannot handle negative weights, must be re-run for each query
2	BFS	Not Suitable	O(V+E) O(V)	It cannot handle weighted roads, may select longer paths
3	Bellman–Ford	Not Ideal	O(V×E) O(V)	It is too slow for real-time emergency systems

In the route allocation Dijkstra is the best algorithm which can be used,However there are some limitations or negative side of using Dijkstra as well which has been shown below :

It Cannot Handle Negative Edge Weights

Dijkstra assumes that all roads or edges have non-negative distances. If there is a negative weight (for example, a penalty or shortcut), the algorithm may calculate an incorrect shortest path, because it never revisits nodes it has already processed.

Recalculation Needed for Each Source

The algorithm computes shortest paths from a single source node. If multiple victims are in different zones, Dijkstra must be run separately for each source, which increases computational time and can slow down real-time rescue operations.

Does Not Consider Dynamic Factors

Dijkstra only considers static distances and does not account for real-world conditions such as traffic, road congestion, or vehicle locations. Therefore, the chosen path may be shortest in distance but not necessarily the fastest in actual travel time.

3.2.3 Hospital Allocation

Problem Description

The traditional approach of sending disaster victims to the nearest hospital often results in inefficient and dangerous outcomes, because it ignores real-time hospital capacity and whether facilities have the necessary specialized equipment. This mismatch—where patients arrive at hospitals that are already full or lack the required resources—leads to life-threatening secondary transfers and overcrowding. There is a clear need for an intelligent system that dynamically balances proximity, medical capabilities, and real-time bed availability to ensure victims are sent to the most appropriate hospital.

Algorithm Used

Custom Weighted Scoring Algorithm (Greedy Selection)

Model / Framework

Constraint-Based Decision Model

Analytical Justification

The system uses a weighted scoring model to balance two critical objectives: medical appropriateness and operational efficiency. It prioritizes hospitals that have the required specialized care—assigning heavy penalties to those that lack necessary units—so that each victim is matched to a facility equipped for their injuries. At the same time, the model accounts for travel distance and real-time bed availability to prevent overcrowding and ensure efficient resource use. This approach ensures victims are not just sent to the nearest or best-equipped hospital, but to the most viable option that supports regional healthcare stability during a disaster.

Why Alternative Algorithms Were Not Chosen

A custom weighted scoring algorithm—using a greedy selection strategy—was developed to overcome the critical shortcomings of pure pathfinding methods like Dijkstra's and A*. While those algorithms reliably find the nearest hospital by calculating optimal travel distance, they are fundamentally limited because they ignore essential medical and capacity factors in allocation decisions. Here's why:

1. Dijkstra's Algorithm (The "Nearest Hospital" Approach)

This method operates by calculating the shortest possible travel distance from the victim's location to every hospital in the area. It systematically explores all road connections, guaranteeing it will always identify the nearest hospital. Performance: It has a time complexity of $O((V + E) \log V)$ and a space complexity of $O(V)$, where V (nodes) are road intersections and E (edges) are road segments. Use Case: It is a classic, reliable choice for finding the closest geographic facility. Key Limitation: It makes decisions based on distance alone, completely ignoring critical factors like a hospital's bed capacity, its specialized medical units, and the victim's injury severity, which can lead to overcrowding and poor patient-care matching.

2. A* Algorithm (Efficient, Heuristic-Based Pathfinding)

A* is an enhanced pathfinder that speeds up the search for a single destination by using an informed guess (heuristic), like straight-line distance, to prioritize which roads to explore. It combines the actual travel cost so far with an estimate of the remaining distance, focusing the search efficiently toward the target hospital. Performance: While its worst-case complexity is $O((V + E) \log V)$, it is almost always significantly faster than Dijkstra for point-to-point routing. It guarantees the optimal path if the heuristic is *admissible* (never overestimates distance). Use Case: It is highly effective for fast, optimal geographic routing. Key Limitation: Like Dijkstra, A* is purely a routing algorithm. It finds the fastest route but does not evaluate whether the destination hospital has space, proper equipment, or is appropriate for the victim's needs, making it insufficient for holistic allocation decisions.

Algorithm	Used in System	Time Complexity	Space Complexity	Advantages	Limitations / Problems
Custom Weighted Scoring (Greedy Selection)	Yes	O(n) (n = number of hospitals)	O(n)	Multi-factor, dynamic load-balancing for efficient and equitable victim distribution	Limited by data accuracy, tuning effort, and local—not global
Dijkstra's Algorithm (Nearest Hospital)	No	O((V + E) log V)	O(V)	Reliable shortest-path routing, ideal for sparse networks, using a well-established algorithm	Prioritizes distance alone, leading to overcrowded hospitals and poor matching of victims to required care.
A*	No	O((V + E) log V)	O(V)	Delivers optimal paths faster than Dijkstra by intelligently narrowing the search, ideal for maps.	A complex routing tool that, without integration, fails to address the core allocation challenges of capacity and medical need.

3.2.4 Vehicle Allocation

Problem Description

To reduce critical risks like delays and resource mismatches, the system uses an automated, severity-based method to assign transport. It guarantees that high-priority patients receive ambulances first, while less urgent cases are allocated to vans, optimizing limited resources. The process also adapts dynamically—automatically switching to alternative vehicles when needed—which prevents bottlenecks and operational delays. This fast, medically informed approach improves rescue efficiency and survival rates in high-pressure disaster scenarios.

Algorithm Used

Greedy Algorithm

Model / Framework

Resource Suitability Model

Analytical Justification

The analytical justification for the vehicle allocation logic relies on the efficiency of a rule-based greedy algorithm, which mirrors the rapid-fire decision-making required in field triage. By prioritizing the immediate matching of high-acuity victims to advanced life-support units, the model ensures that specialized medical resources are not prematurely exhausted on non-critical cases. This approach provides a computationally lightweight yet robust framework that maintains operational momentum; it guarantees that the most equipped vehicles are mathematically reserved for those with the highest injury severity, while simultaneously providing a fallback logic to utilize alternative transport when primary assets are depleted.

Why Alternative Algorithms Were Not Chosen

Greedy allocation prioritizes medical appropriateness by directly matching victim severity to vehicle type for fast dispatch. For comparison, the following two strategies illustrate different trade-offs in how vehicles can be assigned during a rescue operation.

1. Priority Queue-Based Allocation

In a priority queue-based method, vehicles are stored by priority according to type, availability, or speed. The highest-priority vehicle is assigned to each victim, ensuring ambulances serve critical cases first. Time complexity is O(log V) per allocation, and space complexity is O(V). This method is more efficient for larger fleets and supports dynamic updates, but it is slightly more complex to implement than the greedy approach.

2. Round-Robin / FIFO Allocation

The Round-Robin or FIFO approach assigns vehicles in the order they become available, cycling through the fleet without considering victim priority or vehicle type. It is very simple and fast, with O(1) time complexity and minimal space usage. However, it ignores patient severity, which can delay transport for critical victims, making it less suitable for time-sensitive emergencies.

Algorithm	Used in System	Time Complexity	Space Complexity	Advantages	Limitations / Problems
Greedy	Yes	O(V)	O(1)	Fast, simple, respects victim priority	Not globally optimal; may assign best vehicle to less critical victim
Priority Queue / Heap	No	O(log V)	O(V)	Fast for large fleets; can prioritize dynamically	More complete. overhead for small fleets
Round-Robin	No	O(1)	O(1)	Extremely simple	Ignores priority, it may be inefficient for emergencies

Overview

Feature	Algorithm Used	Purpose
Victim Priority	Priority Queue + Custom Comparator	Ensure critical victims are handled first
Shortest Path	Dijkstra’s Algorithm	Find fastest route to hospital in real-time
All-Pairs Shortest Path	Floyd–Warshall Algorithm	Precompute shortest paths between all zones
Team Selection	Greedy	Pick the first available rescue team
	Rule-Based Greedy	Assign ambulance/van based on injury level
Hospital Selection	Weighted Scoring Algorithm	Choose hospital considering distance, capacity, and special units
Travel Time	Basic Math	Estimate time = distance / vehicle speed
Input Validation	Loop + Safe Parsing	Prevent wrong input and program crash

3.3. Tools, technologies, or datasets applied

Programming Language: C++

Standard Template Library (STL) Components:

- **Vector –** Serves as the primary container for storing all core entities—including zones, hospitals, rescue teams, vehicles, and graph edges—enabling flexible and efficient data management.
- **Priority Queue –** Automatically and continuously sorts victims based on their medical urgency, ensuring that the most critical patients are always processed first.
- **Unordered Map** – Provides constant-time O(1) lookup for accessing individual victim records instantly using their unique ID as a key.
- **Set –** Maintains a real-time, unique list of blocked roads, allowing for quick existence checks to inform immediate routing and pathfinding decisions**.**

Dataset:

Zones (6 locations in Dhaka):

- Dhanmondi
- Gulshan
- Uttara
- Mirpur
- Banani
- Motijheel

Hospitals:

- City Hospital
- General Hospital
- Emergency Medical Center
- Burn & Trauma Center

4. Implementation

4.1. Description of project modules

1.Input Helper Functions

getInt() – Safely takes integer input and handles invalid input.
getString() – Safely takes string input and avoids empty input.

2.System Initialization

System() – Constructor that initializes the entire disaster response system.
intData() – Initializes zones, roads, teams, vehicles, hospitals, and default data.

3.Graph & Path Computation

addRoad() – Adds a bidirectional road between two zones with distance.
computeFloydWarshall() – Precomputes all-pairs shortest paths between zones.
dijkstra() – Computes shortest paths from a given source zone considering blocked roads.

4.Road Management

showRoadStatus() – Displays all open and blocked roads in the system.
blockRoadInteractive() – Blocks a road between two zones based on user input.
unblockRoadInteractive() – Unblocks a previously blocked road.

5.Victim Handling

addVictim() – Adds a new victim and inserts them into the priority queue. dispatchVictim() – Assigns team, vehicle, and hospital to the highest-priority victim.

_**6.Resource Selection Logic**

pickTeam() – Selects the first available rescue team. pickVehicleForVictim() – Selects the most suitable vehicle based on injury severity. pickHospitalEnhanced() – Selects the best hospital using distance, bed availability, and emergency facilities.

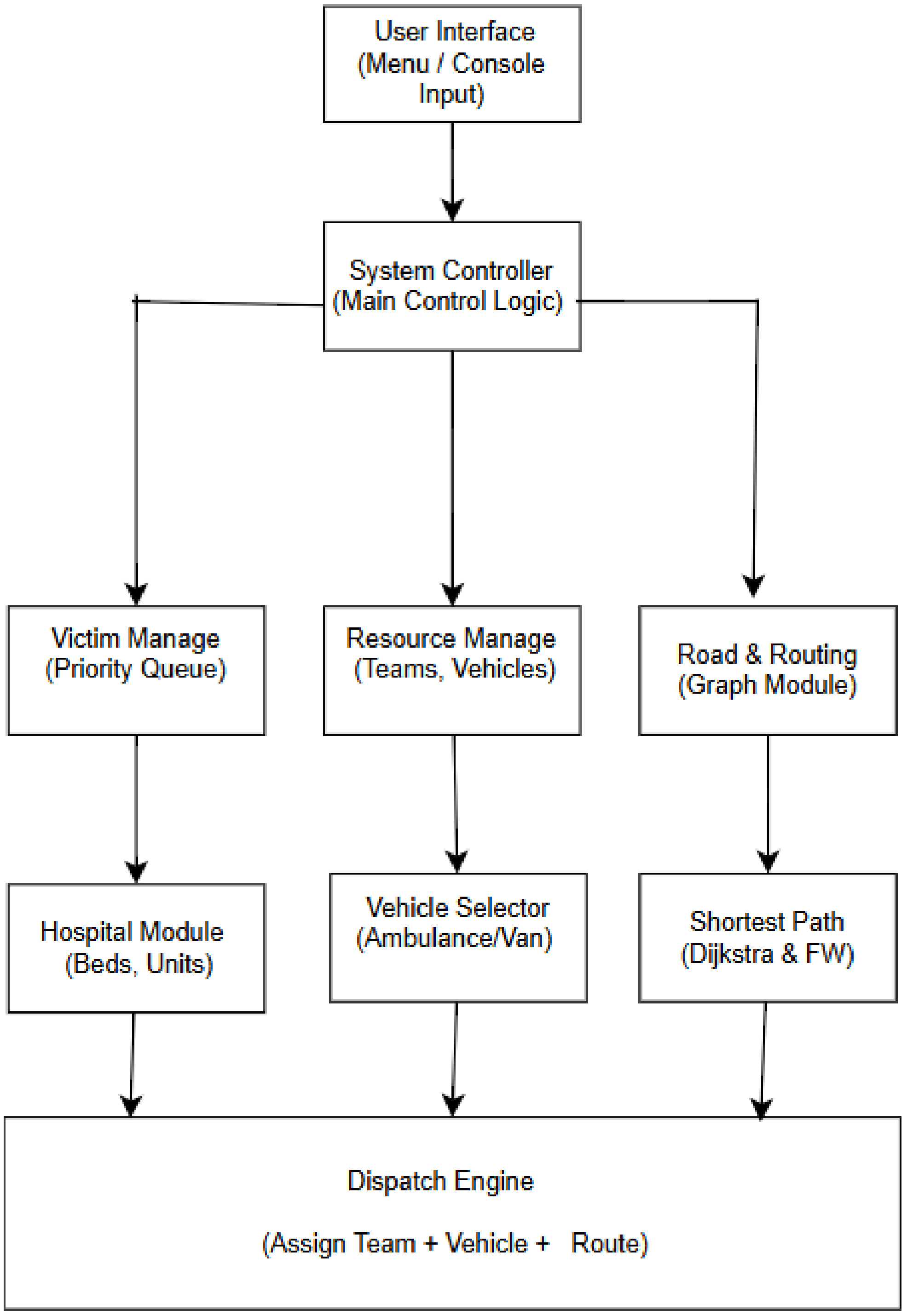
7.System Monitoring

showStatus() – Displays the current status of teams, vehicles, hospitals, roads, and victims.

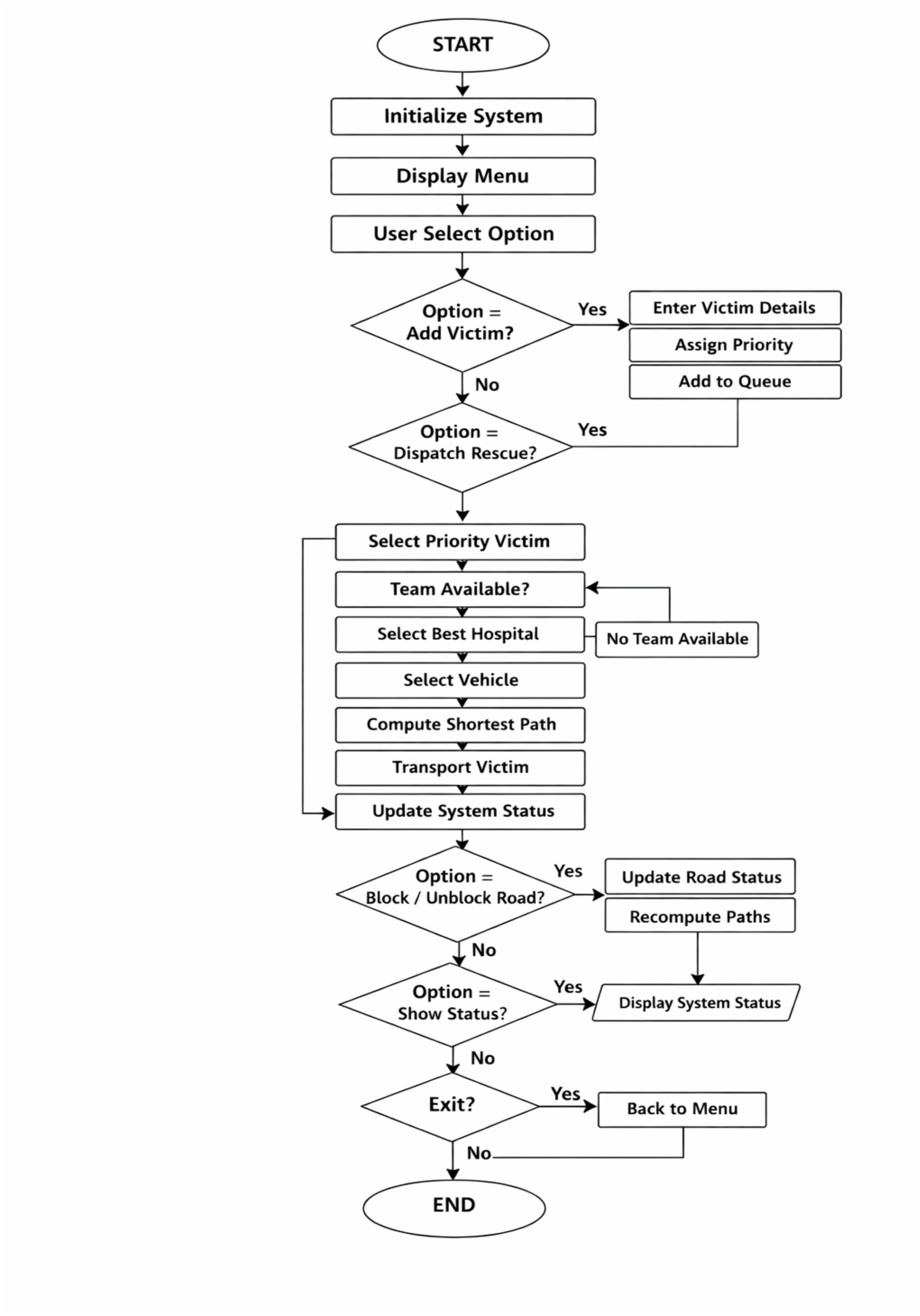
8.Program Control

run() – Runs the menu-driven disaster rescue system. main() – Entry point of the program that starts the system.

4.2 Architecture diagrams


User Interface (Menu / Console Input)
System Controller (Main Control Logic)
Victim Manage (Priority Queue)
Resource Manage (Teams, Vehicles)
Road & Routing (Graph Module)
Hospital Module (Beds, Units)
Vehicle Selector (Ambulance/Van)
Shortest Path (Dijkstra & FW)
Dispatch Engine
(Assign Team + Vehicle + Route)

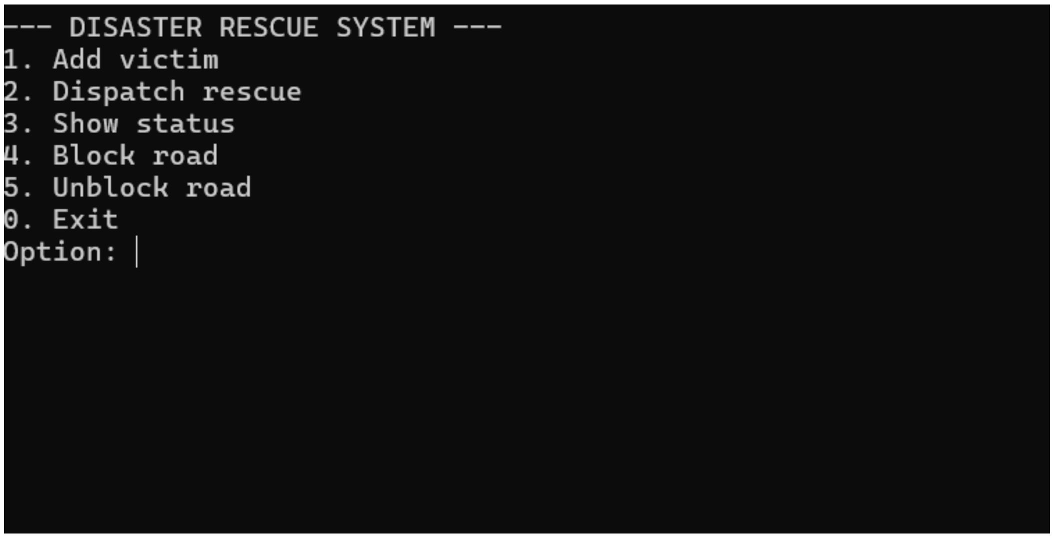
Flow Chart:


START
Initialize System
Display Menu
User Select Option
Option = Add Victim?
Yes
Enter Victim Details
Assign Priority
Add to Queue
No
Option = Dispatch Rescue?
Yes
Select Priority Victim
Team Available?
Select Best Hospital
No Team Available
Select Vehicle
Compute Shortest Path
Transport Victim
Update System Status
Option = Block / Unblock Road?
Yes
Update Road Status
Recompute Paths
No
Option = Show Status?
Yes
Display System Status
No
Exit?
Yes
Back to Menu
No
END

5. Results and Discussion

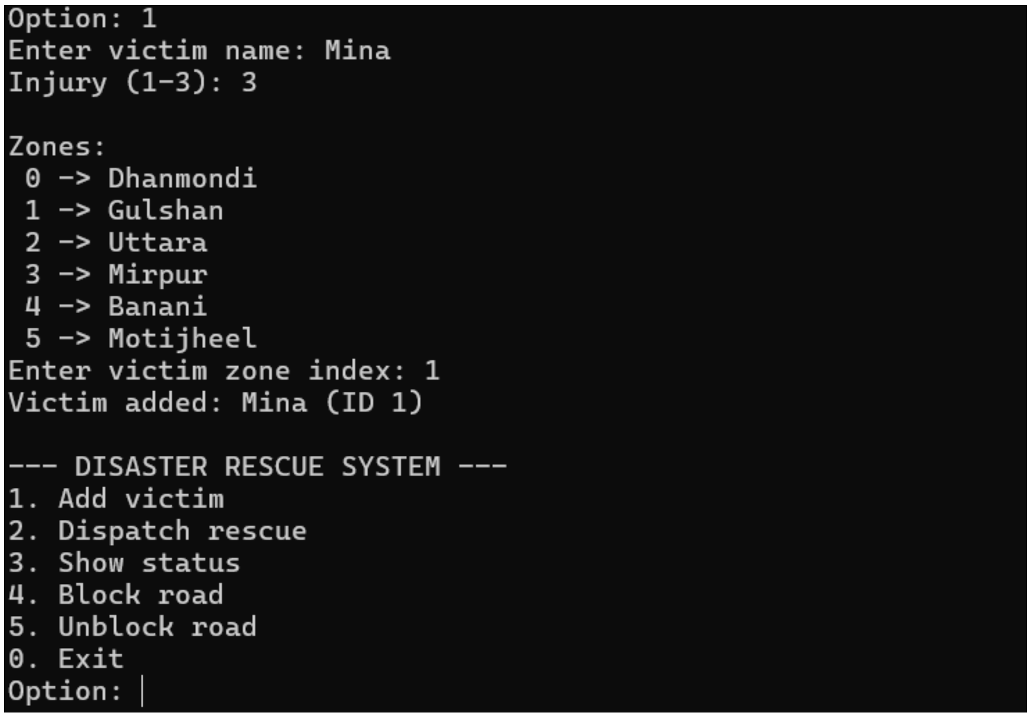
5.1 Output screenshots / performance analysis

Main Menu



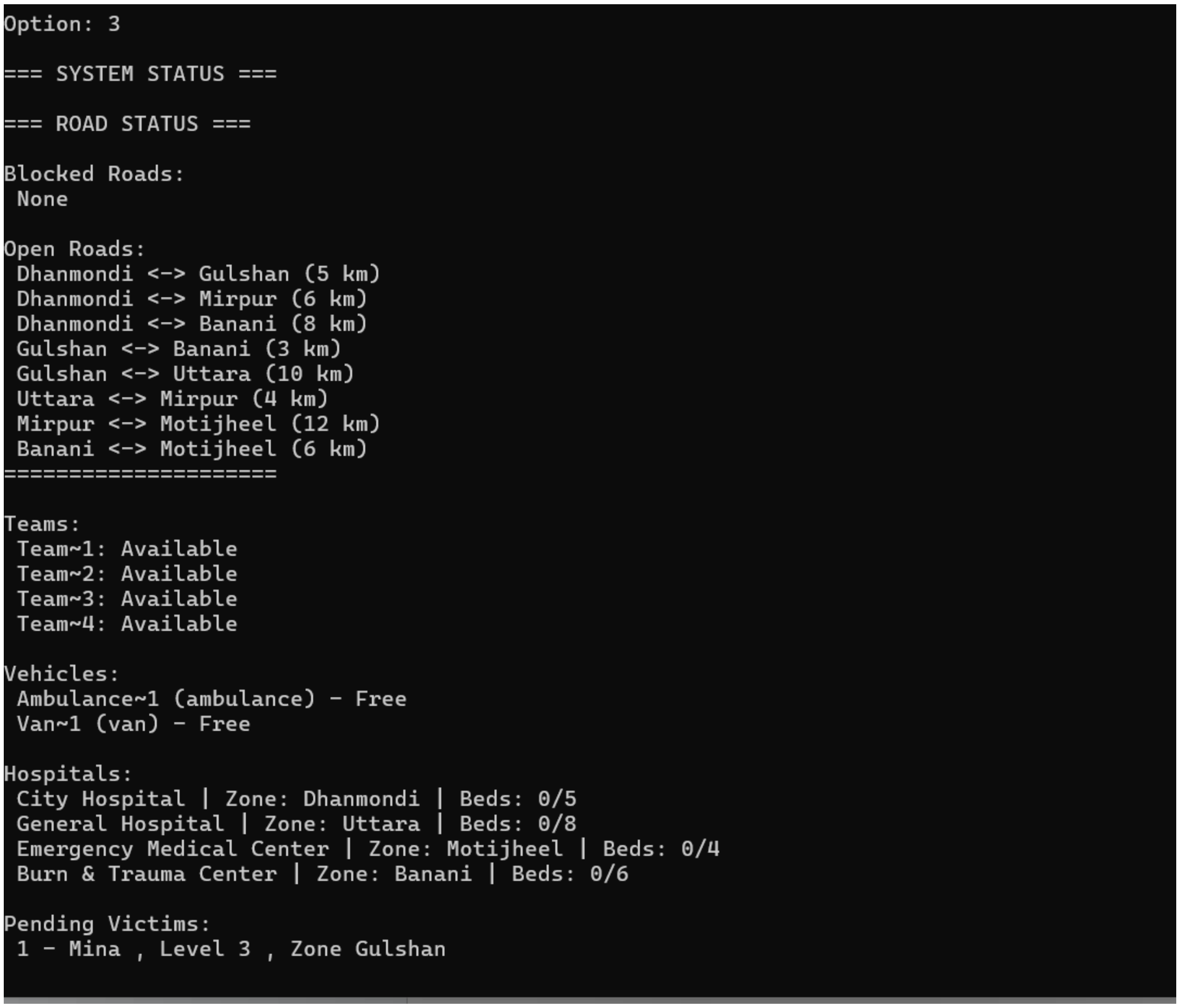
```
--- DISASTER RESCUE SYSTEM ---
1. Add victim
2. Dispatch rescue
3. Show status
4. Block road
5. Unblock road
0. Exit
Option: 
```

Adding Victim



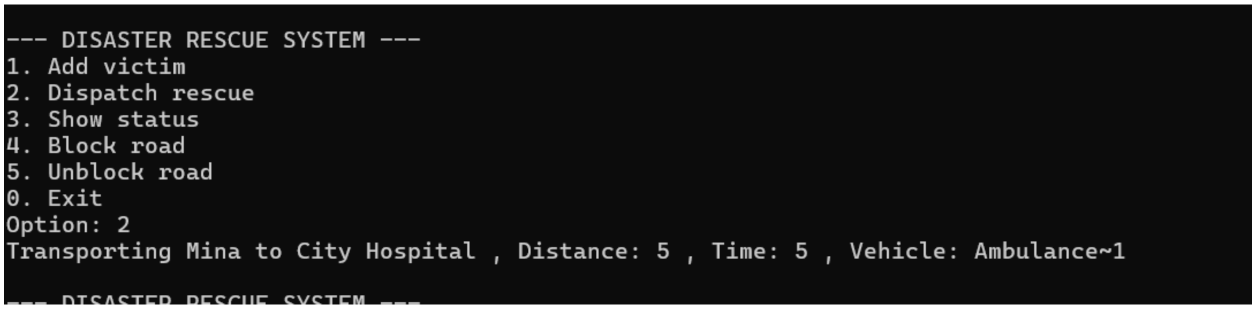
```
Option: 1
Enter victim name: Mina
Injury (1-3): 3

Zones:
 0 -> Dhanmondi
 1 -> Gulshan
 2 -> Uttara
 3 -> Mirpur
 4 -> Banani
 5 -> Motijheel
Enter victim zone index: 1
Victim added: Mina (ID 1)

--- DISASTER RESCUE SYSTEM ---
1. Add victim
2. Dispatch rescue
3. Show status
4. Block road
5. Unblock road
0. Exit
Option: 
```

After Adding Victim Show Status



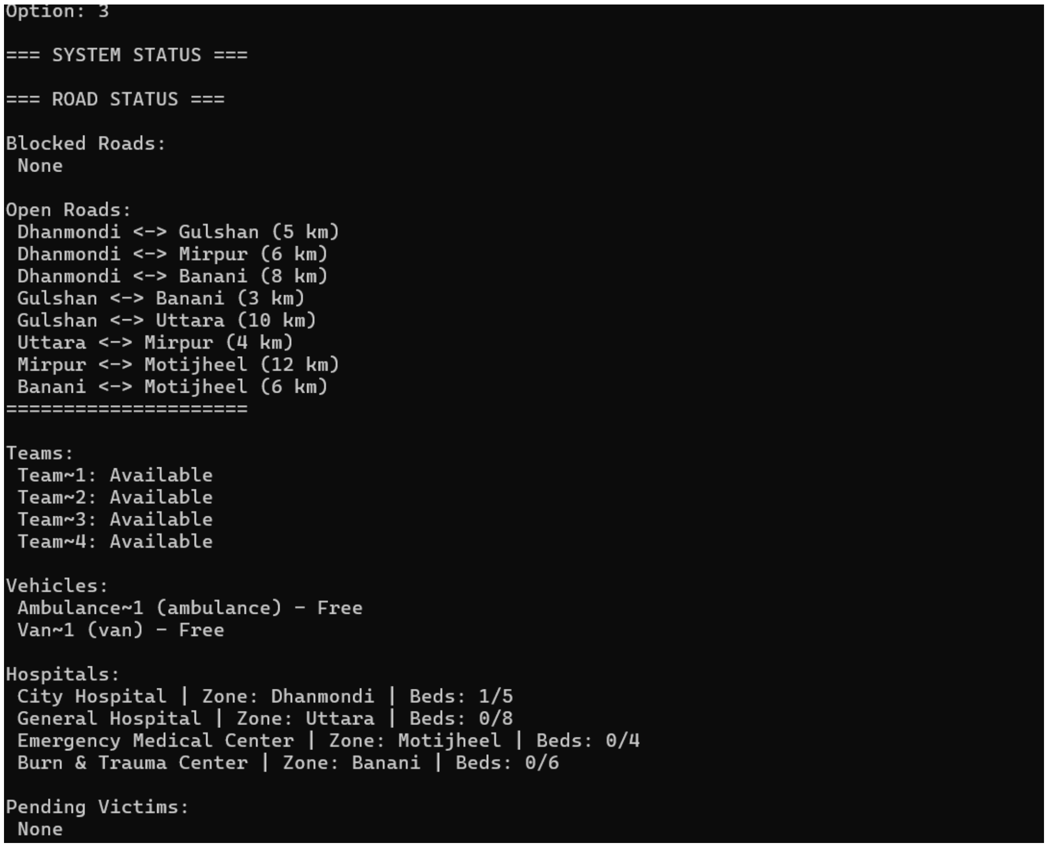
```
Option: 3

=== SYSTEM STATUS ===

=== ROAD STATUS ===

Blocked Roads:
 None

Open Roads:
 Dhanmondi <-> Gulshan (5 km)
 Dhanmondi <-> Mirpur (6 km)
 Dhanmondi <-> Banani (8 km)
 Gulshan <-> Banani (3 km)
 Gulshan <-> Uttara (10 km)
 Uttara <-> Mirpur (4 km)
 Mirpur <-> Motijheel (12 km)
 Banani <-> Motijheel (6 km)
=====================

Teams:
 Team~1: Available
 Team~2: Available
 Team~3: Available
 Team~4: Available

Vehicles:
 Ambulance~1 (ambulance) - Free
 Van~1 (van) - Free

Hospitals:
 City Hospital | Zone: Dhanmondi | Beds: 0/5
 General Hospital | Zone: Uttara | Beds: 0/8
 Emergency Medical Center | Zone: Motijheel | Beds: 0/4
 Burn & Trauma Center | Zone: Banani | Beds: 0/6

Pending Victims:
 1 - Mina , Level 3 , Zone Gulshan
```

Dispatch rescue



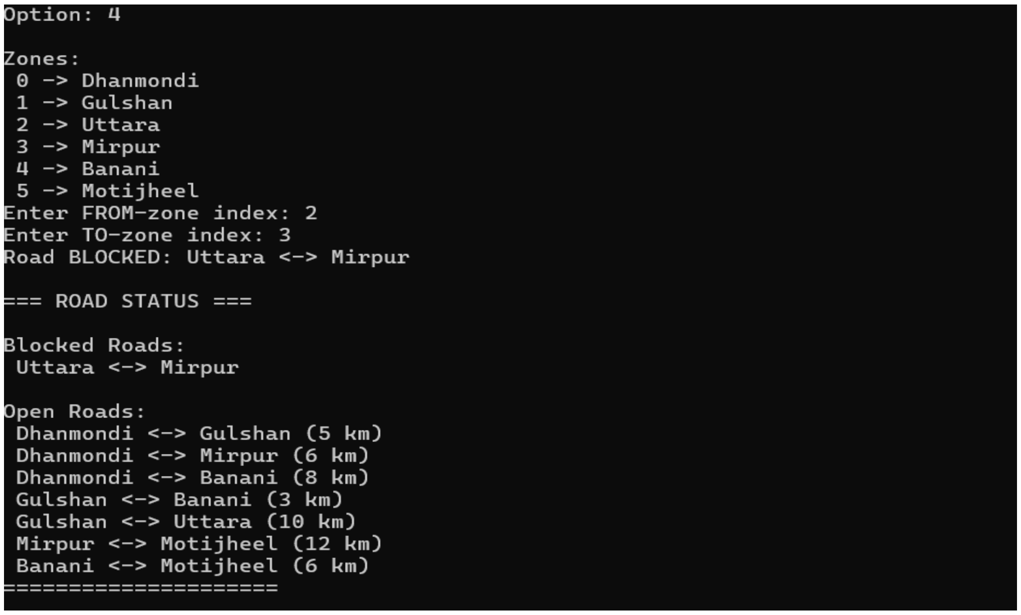
```
--- DISASTER RESCUE SYSTEM ---
1. Add victim
2. Dispatch rescue
3. Show status
4. Block road
5. Unblock road
0. Exit
Option: 2
Transporting Mina to City Hospital , Distance: 5 , Time: 5 , Vehicle: Ambulance~1
```

After Dispatch rescue Show status



```
Option: 3

=== SYSTEM STATUS ===

=== ROAD STATUS ===

Blocked Roads:
 None

Open Roads:
 Dhanmondi <-> Gulshan (5 km)
 Dhanmondi <-> Mirpur (6 km)
 Dhanmondi <-> Banani (8 km)
 Gulshan <-> Banani (3 km)
 Gulshan <-> Uttara (10 km)
 Uttara <-> Mirpur (4 km)
 Mirpur <-> Motijheel (12 km)
 Banani <-> Motijheel (6 km)
=====================

Teams:
 Team~1: Available
 Team~2: Available
 Team~3: Available
 Team~4: Available

Vehicles:
 Ambulance~1 (ambulance) - Free
 Van~1 (van) - Free

Hospitals:
 City Hospital | Zone: Dhanmondi | Beds: 1/5
 General Hospital | Zone: Uttara | Beds: 0/8
 Emergency Medical Center | Zone: Motijheel | Beds: 0/4
 Burn & Trauma Center | Zone: Banani | Beds: 0/6

Pending Victims:
 None
```

Block Road



```
Option: 4

Zones:
 0 -> Dhanmondi
 1 -> Gulshan
 2 -> Uttara
 3 -> Mirpur
 4 -> Banani
 5 -> Motijheel
Enter FROM-zone index: 2
Enter TO-zone index: 3
Road BLOCKED: Uttara <-> Mirpur

=== ROAD STATUS ===

Blocked Roads:
 Uttara <-> Mirpur

Open Roads:
 Dhanmondi <-> Gulshan (5 km)
 Dhanmondi <-> Mirpur (6 km)
 Dhanmondi <-> Banani (8 km)
 Gulshan <-> Banani (3 km)
 Gulshan <-> Uttara (10 km)
 Mirpur <-> Motijheel (12 km)
 Banani <-> Motijheel (6 km)
=====================
```

6. Conclusion and Future Work

This project successfully creates a Disaster Response System that effectively applies classical algorithms to solve key emergency management challenges. By integrating priority-based victim triage, graph-based route planning, and fast resource allocation, the system ensures timely and logical rescue operations. Its ability to adapt to dynamic road conditions adds practical realism. Ultimately, it demonstrates how algorithmic design can build an efficient, scalable, and reliable framework for emergency response.

Although the system excels in algorithmic efficiency, it faces limitations like reliance on small-scale simulated data, single-threaded console interface, static operational assumptions (e.g., fixed speeds and capacities), and no machine learning for predictions. Future enhancements could address these by integrating live GPS/traffic data for realistic city-scale testing, developing GUI/web dashboards with parallel/cloud processing, enabling dynamic updates via A* routing, and adding ML models to predict victim severity, hospital loads, and optimal allocations—transforming it into a deployable disaster response solution.