PROJECT TITLE :

DISASTER RESPONSE AND RESCUE PATH FINDING SYSTEM

PROJECT REPORT

DEPARTMENT OF CSE/IT
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

SUBMITTED BY:

TARISHE PANDITA	2401030006
MISHTHI ABROL	2401030025
ANUSHKA GARG	2401030029

FACULTY MEMBERS:

DR. DHANALEKSHMI G.
DR. VARUN SRIVASTAVA
DR. ANUPAMA PADHA

TABLE OF CONTENTS

SUMMARY

The "Disaster Response and Rescue Path Finding System" is a C++ based simulation designed to address the complex logistical challenges faced by emergency response teams during natural or man-made disasters. By modeling a city as a dynamic graph and utilizing advanced data structures, the system effectively prioritizes victims based on injury severity, manages the availability of limited rescue teams, and calculates the safest, shortest paths through a road network that may be compromised by blockages.

The primary objective of this project is to demonstrate the practical application of core Data Structures—specifically Graphs, Min-Heaps, Binary Search Trees, Linked Lists, Stacks, and Queues—in solving a real-world optimization problem. The system aims to minimize response time, maximize rescue efficiency, and provide a transparent audit trail of all emergency operations.

INTRODUCTION

Disasters such as earthquakes, floods, and fires create chaotic environments where infrastructure is often compromised. Roads may become impassable, communication lines may fail, and the number of victims often outstrips the available medical resources. In such high-stakes scenarios, human decision-making can be clouded by stress or incomplete information. Efficient, algorithmic decision-making becomes a matter of life and death.

PROBLEM STATEMENT

Emergency teams face three critical, interrelated challenges:

1. Prioritization (Triage): In a mass-casualty event, treating patients based solely on "first-come, first-served" is inefficient and potentially fatal for critical cases.
2. Dynamic Navigation (Pathfinding): Rescue teams would not be able to reach victims if the primary routes are blocked. So, teams need instant recalculation of routes based on current road conditions.
3. Resource Management (Allocation): The team which is closest, has the right equipment, and is currently available would be allocated. Misallocation of resources leads to wasted time and fuel.

OBJECTIVES

- To model a city map using Graph data structures where nodes represent zones and edges represent roads.
- To implement a Min-Heap for prioritizing victims, ensuring that patients with Severity Level 1 (Critical) are processed before Level 10 (Minor).

- To use Breadth-First Search (BFS) for finding the shortest path in an unweighted graph, capable of bypassing blocked nodes.
- To manage rescue teams using a Binary Search Tree (BST) for O(log n) retrieval and status updates.
- To maintain an immutable audit trail of system events using a Singly Linked List.

SCOPE OF THE PROJECT

The current scope is a console-based simulation application. It accepts user inputs for victim reports, road blockages, and team registrations. It outputs the optimal path and allocation status textually. The system is designed to be modular, allowing for future integration with GUI front-ends, real-time GPS data, or persistent database storage.

SYSTEM REQUIREMENTS

Hardware Requirements:

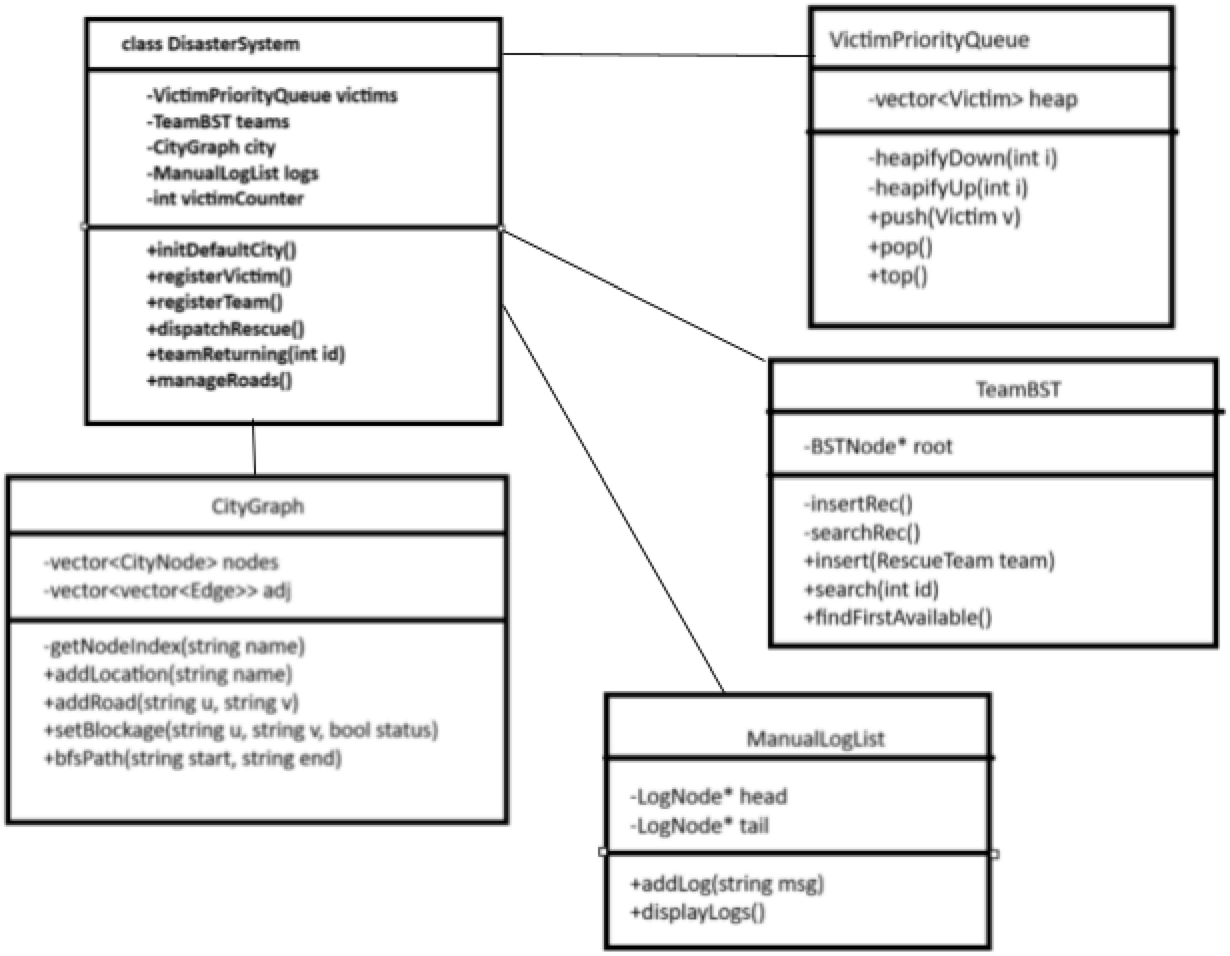
- Processor: Intel Core i3 (Gen 5+) or AMD Ryzen 3 equivalent.
- RAM: 4 GB minimum (8 GB recommended for large dataset simulations).
- Storage: 100 MB free space for source code and compiled binaries.

Software Requirements:

- Operating System: Windows 10/11, Linux (Ubuntu 20.04+), or macOS.
- Programming Language: C++ (Standard C++11 or higher).
- Compiler: GCC (g++) version 9.0+, Clang, or MSVC.
- IDE: Visual Studio Code, DevC++, or CLion.
- Tools: Git for version control.

DESIGN AND IMPLEMENTATION

CLASS DIAGRAM



DATA STRUCTURE IMPLEMENTATIONS

- **Classes:** Uses classes like RescueTeam, Victim, and CityMap to organize data and functions cleanly.
- **Arrays:** Store availability of resources such as vehicles and supplies. Access is very fast (constant time). **– O(1).**
- **Queue:** Manages rescue requests in the order they arrive (first-in, first-out), ensuring fairness and order. **– O(1)**.
- **Hash Table:** Quickly checks if roads are blocked during rescue planning, enabling fast decision updates – **O(1)** average.
- **Min Heap:** Priority Queue for victims. A Min-Heap provides **O(1)** access to the highest priority element (most critical victim) and O(log n) insertion/deletion. This is far superior to sorting an array **O(n log n)** every time a new victim is added.
- **Graphs:** Models the city map. An Adjacency List is space-efficient **O(V+E)** for sparse graphs like city roads, compared to an Adjacency Matrix **O(V^2)**. It allows efficient traversal of neighbors.
- **Binary Search Tree:** Rescue Team Database. A BST allows for efficient searching **O(log n)** of teams by ID to update their status. It is faster than a linear search through an array for large datasets.
- **Stack:** Path Reconstruction. Used within the pathfinding logic to backtrack from the destination to the source. The LIFO nature of the stack naturally reverses the path to the correct order Source to Destination.
- **Linked List:** System Logs. A Linked List allows for **O(1)** insertion of new logs at the tail. Since logs are append-only and rarely searched, a Linked List is the optimal choice for an audit trail.
- **Heap Operations:** Priority Queue. Used to maintain the victim list sorted by severity.

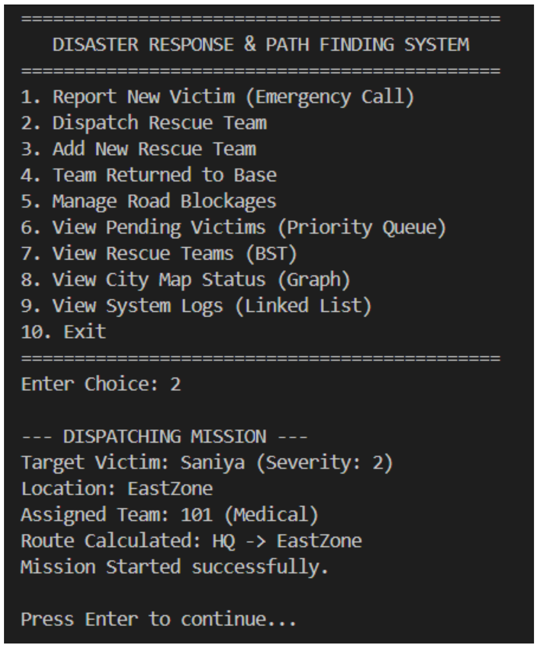
ALGORITHMS AND THEIR COMPLEXITIES

Algorithms	Purpose	Complexities
Breadth-First Search (BFS)	Finds the shortest path from Headquarters to the Victim in an unweighted graph.	O(V + E)
Min-Heapify Up (Bubble Up)	Used during Insertion of a new victim.	O(log n)
Min-Heapify Down (Bubble Down)	Used during Deletion (Dispatch) of the top victim.	O(log n)
BST Insertion	Adds a new rescue team to the TeamBST.	O(log n)
BST Search	Finds a specific team by ID to update its status (Busy/Available).	O(log n)
In-Order Traversal (DFS)	Traverses the tree	O(n)
Linear Search	Look up the index associated with a city location name.	O(n)
Path Backtracking	Reconstructs the path once BFS finds the destination.	O(k), where k is path length
Swapping	Basic utility to exchange two elements in memory.	O(1)

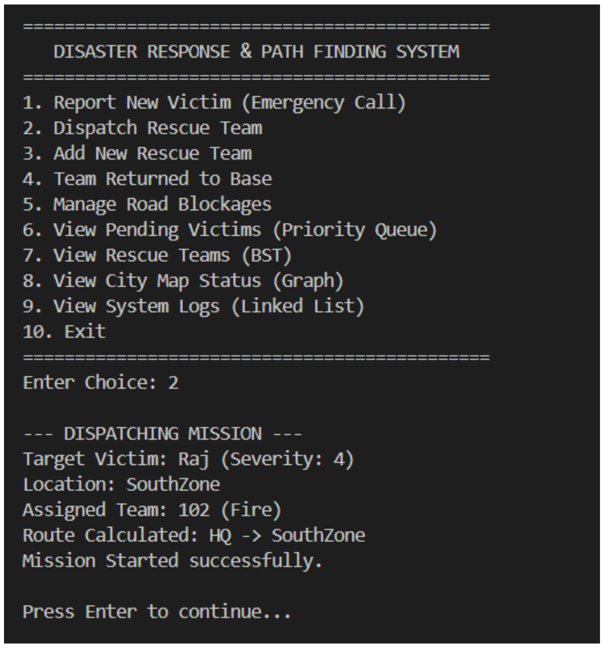
CODE SNIPPETS

1)

```
vector<string> bfsPath(string start, string end) {
while(head < q.size())
{
int u = q[head++];
 if (u == e)
 {
 found = true;
break;
}
for (int i = 0; i < adj[u].size(); i++)
{
int v = adj[u][i].toIndex;
if (!visited[v] && !adj[u][i].isBlocked)
 {
 visited[v] = true;
 parent[v] = u;
 q.push_back(v);
 }
 }
 }
}
```



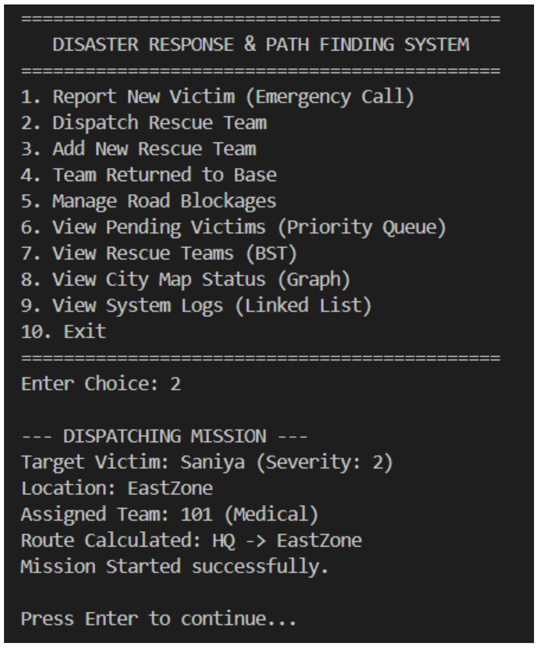
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 2

--- DISPATCHING MISSION ---
Target Victim: Saniya (Severity: 2)
Location: EastZone
Assigned Team: 101 (Medical)
Route Calculated: HQ -> EastZone
Mission Started successfully.

Press Enter to continue...
```



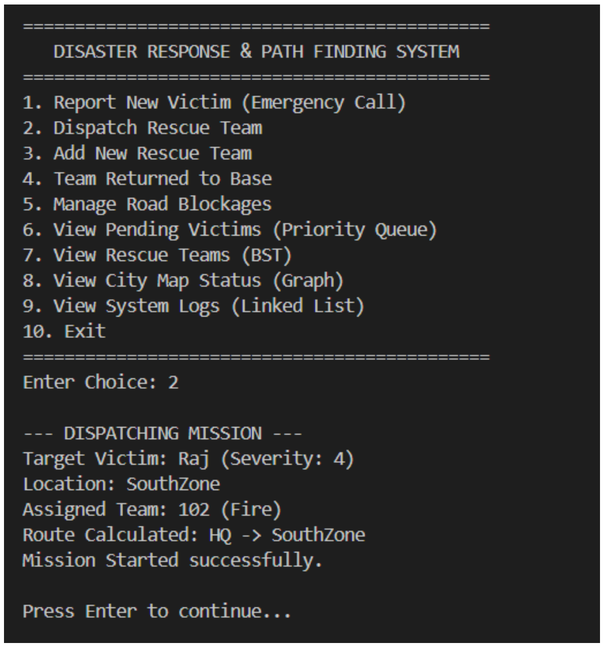
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 2

--- DISPATCHING MISSION ---
Target Victim: Raj (Severity: 4)
Location: SouthZone
Assigned Team: 102 (Fire)
Route Calculated: HQ -> SouthZone
Mission Started successfully.

Press Enter to continue...
```

2)

```
void heapifyUp(int i)
{
if (i && heap[parent(i)].severity > heap[i].severity)
{
mySwap(heap[i], heap[parent(i)]);
 heapifyUp(parent(i));
 }
 }
victims.pop();
```



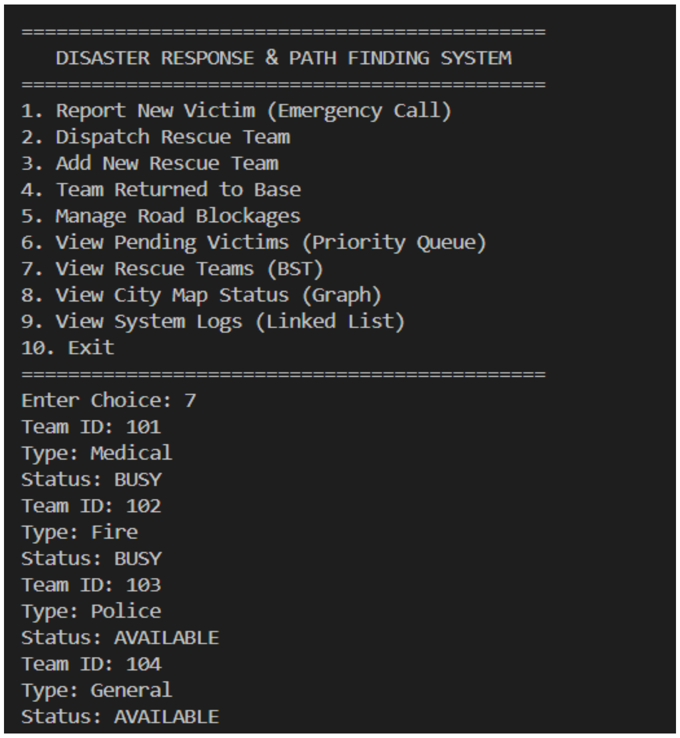
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 2

--- DISPATCHING MISSION ---
Target Victim: Saniya (Severity: 2)
Location: EastZone
Assigned Team: 101 (Medical)
Route Calculated: HQ -> EastZone
Mission Started successfully.

Press Enter to continue...
```



```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 2

--- DISPATCHING MISSION ---
Target Victim: Raj (Severity: 4)
Location: SouthZone
Assigned Team: 102 (Fire)
Route Calculated: HQ -> SouthZone
Mission Started successfully.

Press Enter to continue...
```

3)

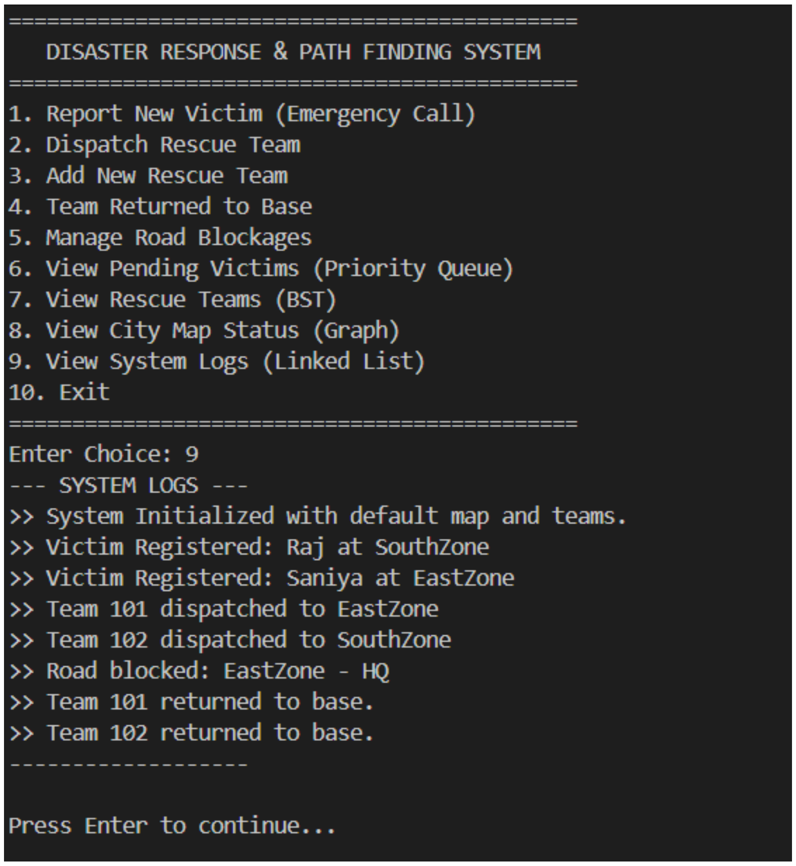
```
void inorderRec(BSTNode* node)
{
    if (node != nullptr)
    {
        inorderRec(node->left);
        cout << "Team ID: " << node->data.teamID << endl;
        cout << "Type: " << node->data.type << endl;
        cout << "Status: " << (node->data.isBusy ? "BUSY" : "AVAILABLE") << endl;
        inorderRec(node->right);
    }
}
```



```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 7
Team ID: 101
Type: Medical
Status: BUSY
Team ID: 102
Type: Fire
Status: BUSY
Team ID: 103
Type: Police
Status: AVAILABLE
Team ID: 104
Type: General
Status: AVAILABLE
```

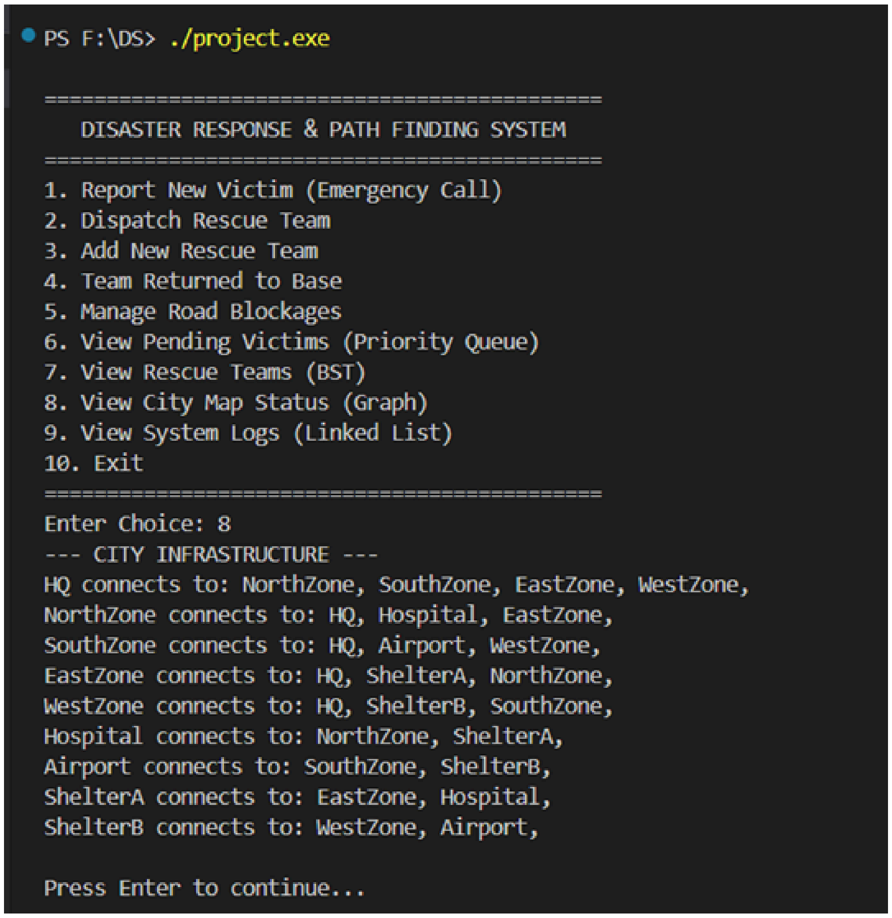
4)

```
void addLog(string msg)
{
LogNode* newNode = new LogNode;
 newNode->message = msg;
newNode->next = nullptr;
 if (head == nullptr)
 {
head = newNode;
 tail = newNode;
 }
else
{
 tail->next = newNode;
 tail = newNode;
}
}
```



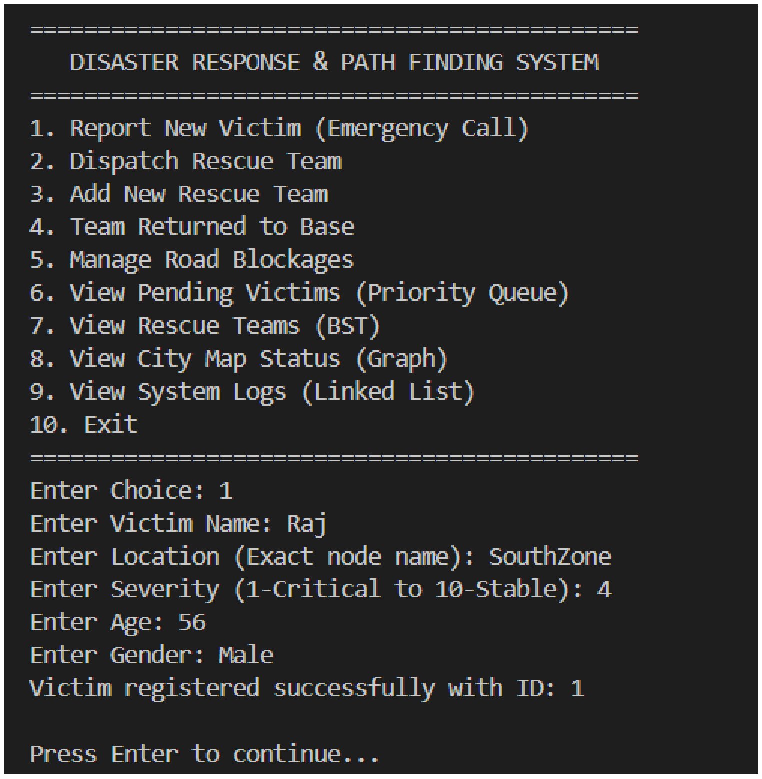
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 9
--- SYSTEM LOGS ---
>> System Initialized with default map and teams.
>> Victim Registered: Raj at SouthZone
>> Victim Registered: Saniya at EastZone
>> Team 101 dispatched to EastZone
>> Team 102 dispatched to SouthZone
>> Road blocked: EastZone - HQ
>> Team 101 returned to base.
>> Team 102 returned to base.
-------------------

Press Enter to continue...
```

OUTPUTS



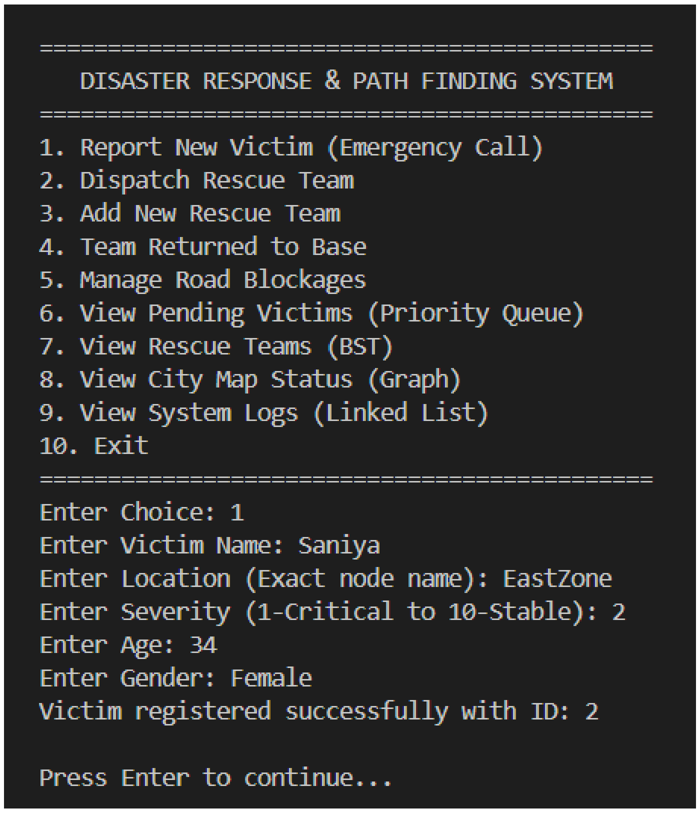
```
PS F:\DS> ./project.exe

 =============================================
    DISASTER RESPONSE & PATH FINDING SYSTEM
 =============================================
 1. Report New Victim (Emergency Call)
 2. Dispatch Rescue Team
 3. Add New Rescue Team
 4. Team Returned to Base
 5. Manage Road Blockages
 6. View Pending Victims (Priority Queue)
 7. View Rescue Teams (BST)
 8. View City Map Status (Graph)
 9. View System Logs (Linked List)
 10. Exit
 =============================================
 Enter Choice: 8
 --- CITY INFRASTRUCTURE ---
 HQ connects to: NorthZone, SouthZone, EastZone, WestZone,
 NorthZone connects to: HQ, Hospital, EastZone,
 SouthZone connects to: HQ, Airport, WestZone,
 EastZone connects to: HQ, ShelterA, NorthZone,
 WestZone connects to: HQ, ShelterB, SouthZone,
 Hospital connects to: NorthZone, ShelterA,
 Airport connects to: SouthZone, ShelterB,
 ShelterA connects to: EastZone, Hospital,
 ShelterB connects to: WestZone, Airport,

 Press Enter to continue...
```

The user selects 8. "View City Map Status", displaying the graph's adjacency list where nodes like HQ are shown connected to neighbors such as NorthZone, SouthZone, EastZone, and WestZone.



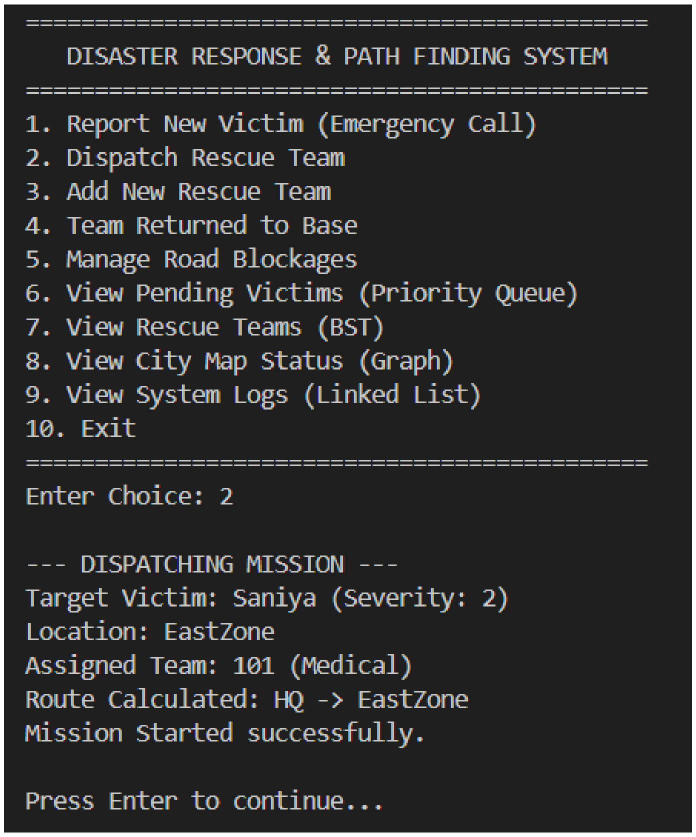
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 1
Enter Victim Name: Raj
Enter Location (Exact node name): SouthZone
Enter Severity (1-Critical to 10-Stable): 4
Enter Age: 56
Enter Gender: Male
Victim registered successfully with ID: 1

Press Enter to continue...
```

The user selects Option 1 to report a new victim named Raj (Age 56, Male) located at SouthZone with a severity of 4.



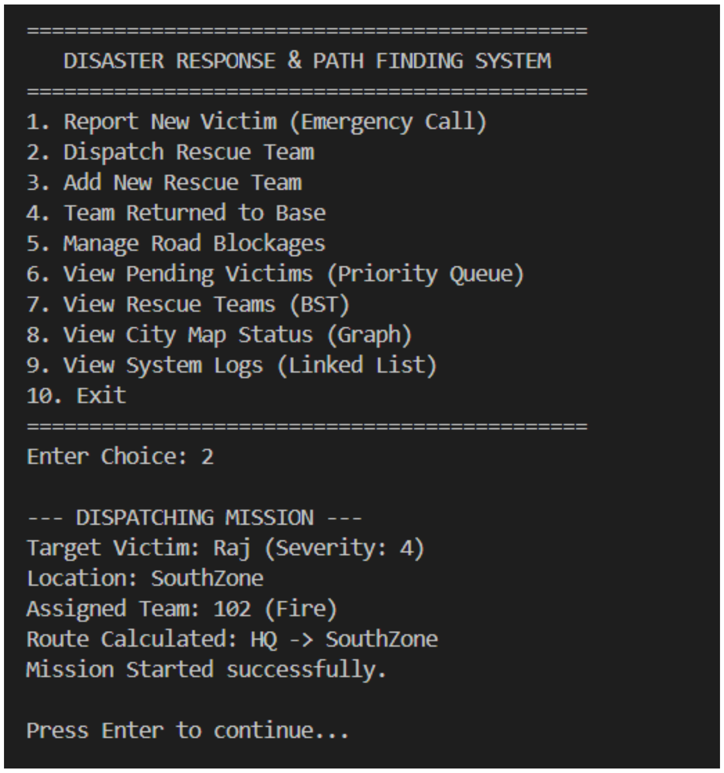
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 1
Enter Victim Name: Saniya
Enter Location (Exact node name): EastZone
Enter Severity (1-Critical to 10-Stable): 2
Enter Age: 34
Enter Gender: Female
Victim registered successfully with ID: 2

Press Enter to continue...
```

The user registers a second victim named Saniya (Age 34, Female) located at EastZone with a more critical severity of 2.



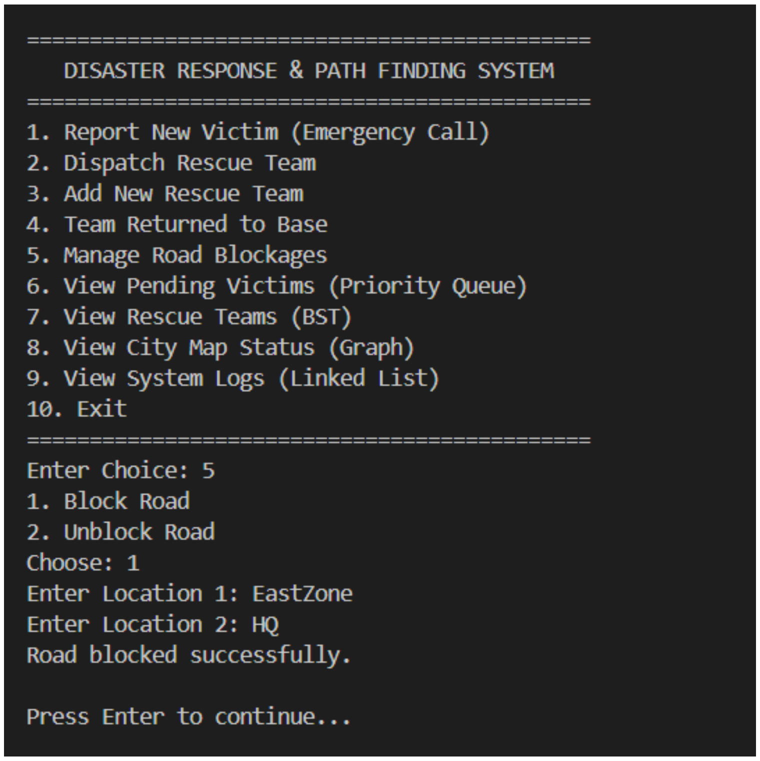
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 2

--- DISPATCHING MISSION ---
Target Victim: Saniya (Severity: 2)
Location: EastZone
Assigned Team: 101 (Medical)
Route Calculated: HQ -> EastZone
Mission Started successfully.

Press Enter to continue...
```

The user selects Option 2 "Dispatch Rescue Team". The system prioritizes the more critical victim (Saniya, Severity 2), assigns Team 101 (Medical), and calculates the path HQ -> EastZone.



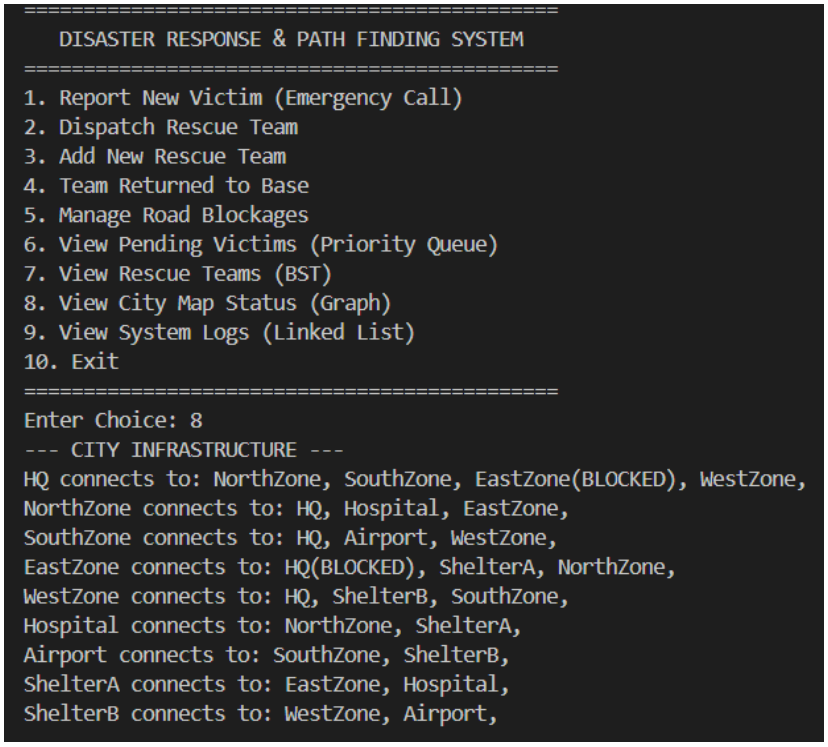
```
==============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
==============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
==============================================
Enter Choice: 2

--- DISPATCHING MISSION ---
Target Victim: Raj (Severity: 4)
Location: SouthZone
Assigned Team: 102 (Fire)
Route Calculated: HQ -> SouthZone
Mission Started successfully.

Press Enter to continue...
```

The user selects Option 2 again. The system assigns Team 102 (Fire) to the remaining victim, Raj (Severity 4), and calculates the path HQ -> SouthZone.



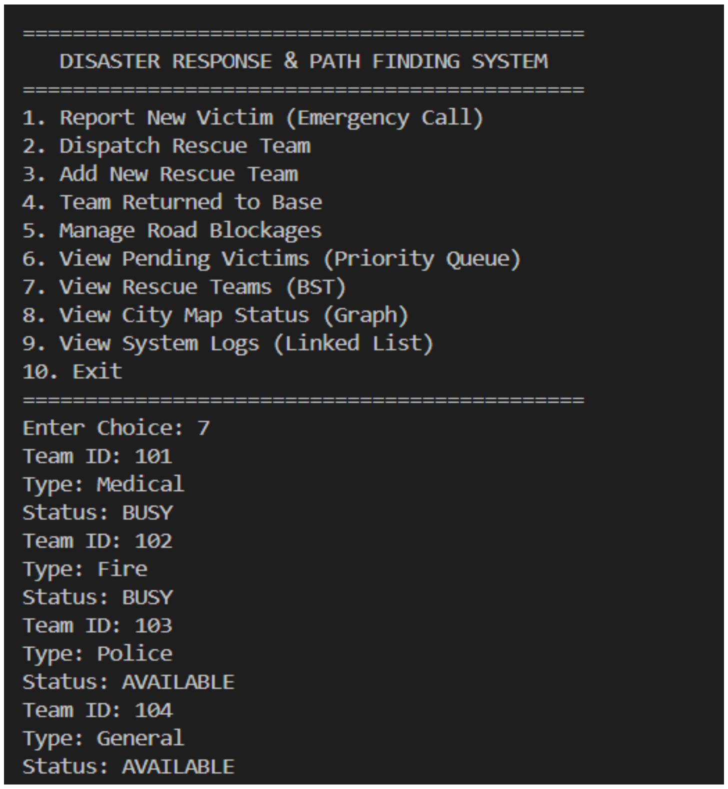
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 5
1. Block Road
2. Unblock Road
Choose: 1
Enter Location 1: EastZone
Enter Location 2: HQ
Road blocked successfully.

Press Enter to continue...
```

The user selects Option 5 "Manage Road Blockages", chooses to "Block Road," and successfully disconnects the route between EastZone and HQ.



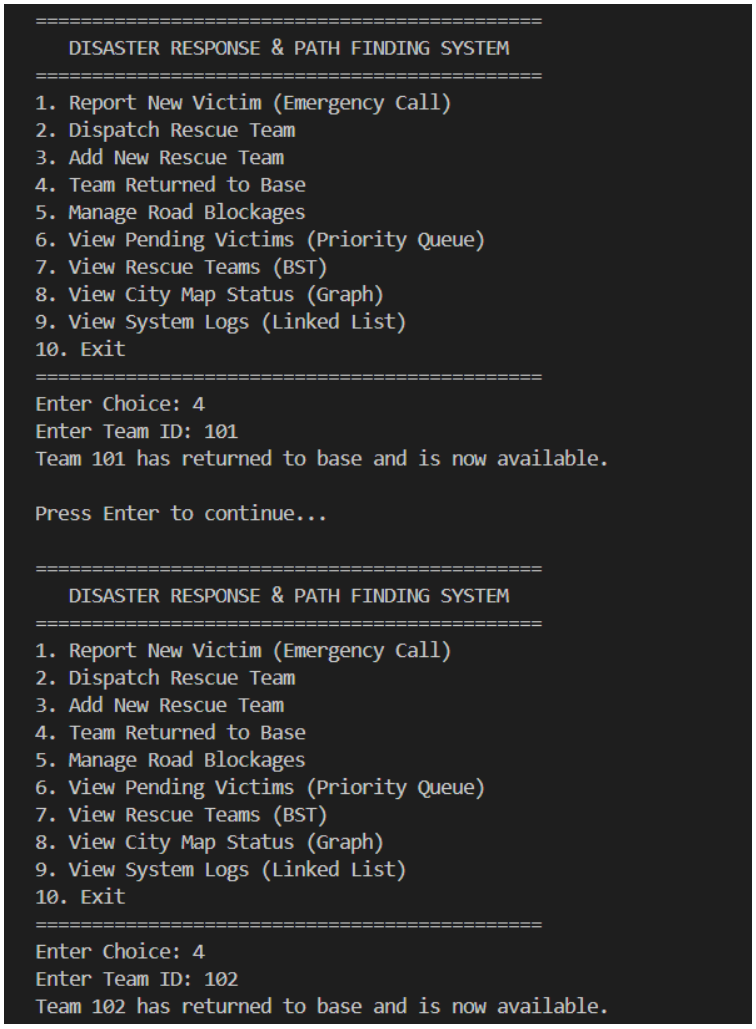
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 8
--- CITY INFRASTRUCTURE ---
HQ connects to: NorthZone, SouthZone, EastZone(BLOCKED), WestZone,
NorthZone connects to: HQ, Hospital, EastZone,
SouthZone connects to: HQ, Airport, WestZone,
EastZone connects to: HQ(BLOCKED), ShelterA, NorthZone,
WestZone connects to: HQ, ShelterB, SouthZone,
Hospital connects to: NorthZone, ShelterA,
Airport connects to: SouthZone, ShelterB,
ShelterA connects to: EastZone, Hospital,
ShelterB connects to: WestZone, Airport,
```

The user views the map status Option 8 again, which now visually indicates that the connection between HQ and EastZone is (BLOCKED).



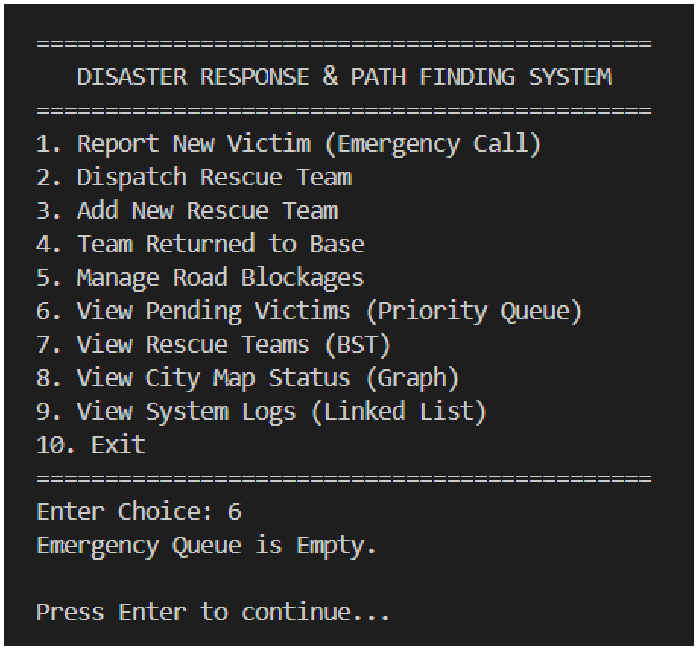
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 7
Team ID: 101
Type: Medical
Status: BUSY
Team ID: 102
Type: Fire
Status: BUSY
Team ID: 103
Type: Police
Status: AVAILABLE
Team ID: 104
Type: General
Status: AVAILABLE
```

The user views the Rescue Teams Option 7, which confirms that Team 101 and Team 102 are currently BUSY, while Teams 103 and 104 are AVAILABLE.



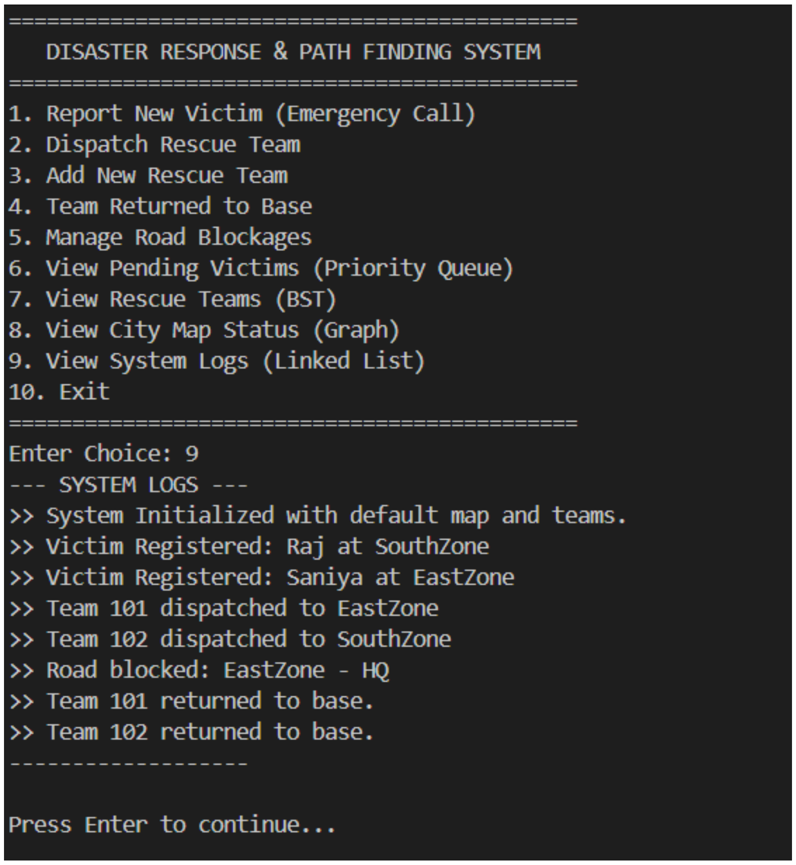
```
============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
============================================
Enter Choice: 4
Enter Team ID: 101
Team 101 has returned to base and is now available.

Press Enter to continue...

============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
============================================
Enter Choice: 4
Enter Team ID: 102
Team 102 has returned to base and is now available.
```

The user selects Option 4 "Team Returned to Base" and enters Team ID 101, successfully marking the team as available again.



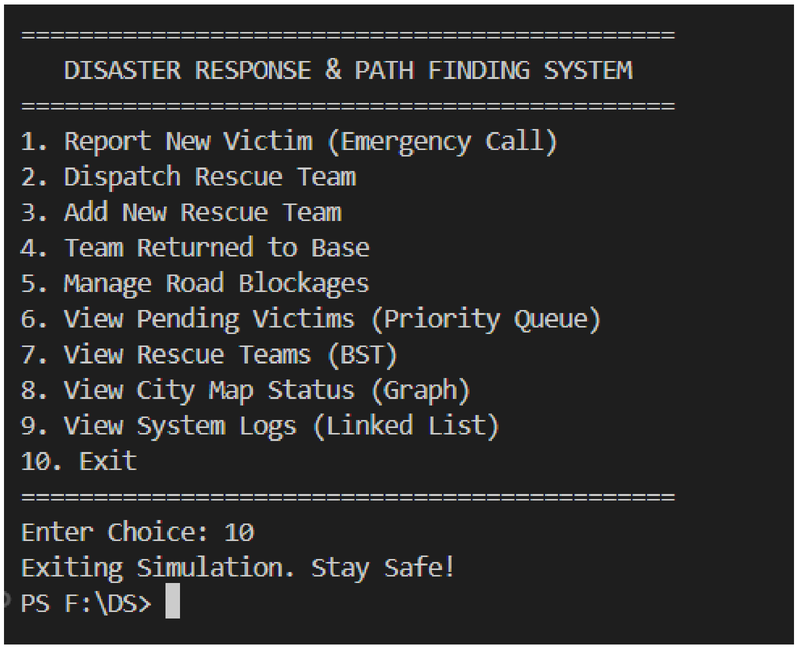
```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 6
Emergency Queue is Empty.

Press Enter to continue...
```

The user checks the Pending Victims (Option 6), and the system confirms that the Emergency Queue is Empty.



```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 9
--- SYSTEM LOGS ---
>> System Initialized with default map and teams.
>> Victim Registered: Raj at SouthZone
>> Victim Registered: Saniya at EastZone
>> Team 101 dispatched to EastZone
>> Team 102 dispatched to SouthZone
>> Road blocked: EastZone - HQ
>> Team 101 returned to base.
>> Team 102 returned to base.
-------------------

Press Enter to continue...
```

The user selects Option 9 "View System Logs", displaying a linked list history of all actions performed, including registrations, dispatches, and road blockages.



```
=============================================
   DISASTER RESPONSE & PATH FINDING SYSTEM
=============================================
1. Report New Victim (Emergency Call)
2. Dispatch Rescue Team
3. Add New Rescue Team
4. Team Returned to Base
5. Manage Road Blockages
6. View Pending Victims (Priority Queue)
7. View Rescue Teams (BST)
8. View City Map Status (Graph)
9. View System Logs (Linked List)
10. Exit
=============================================
Enter Choice: 10
Exiting Simulation. Stay Safe!
PS F:\DS> 
```

The user selects Option 10 to exit the system, displaying the closing message "Exiting Simulation. Stay Safe!".

CONCLUSION

The project successfully implements a robust, console-based Disaster Response System. By integrating custom implementations of Graphs, Heaps, and BSTs, we achieved an efficient mechanism for prioritizing life-saving operations without relying on standard library containers. The system correctly identifies the shortest path even when roads are dynamically blocked, fulfilling the core requirement of disaster resilience. The codebase demonstrates a strong understanding of memory management, pointers, and algorithmic logic.

Future Work and Recommendations:

To enhance the system, weighted graphs can be used by incorporating Dijkstra's Algorithm to manage roads with varying lengths or traffic costs instead of treating all routes equally. A graphical user interface can be built and visually display the city map and allow users to interact with it, such as clicking on roads to block them. Integrating GPS capabilities through real-time APIs like the Google Maps API would enable the system to fetch live traffic data for actual cities. Additionally, transitioning from in-memory storage to SQL or NoSQL databases would ensure that victim and team data persist across system restarts.

REFERENCES

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
2. Stroustrup, B. (2013). *The C++ Programming Language* (4th ed.). Addison-Wesley.
3. GeeksforGeeks. (2024). *Graph Data Structure and Algorithms*. Retrieved from https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
4. Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley