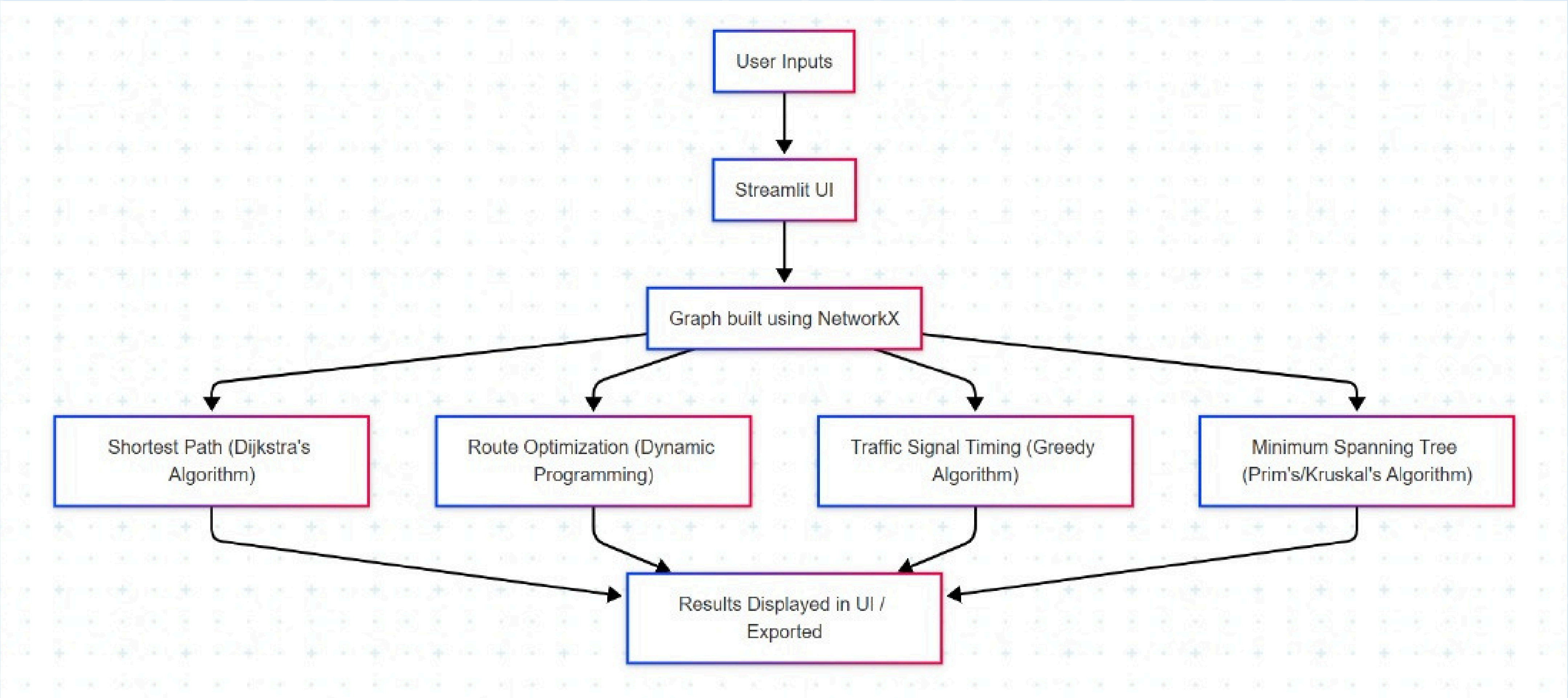
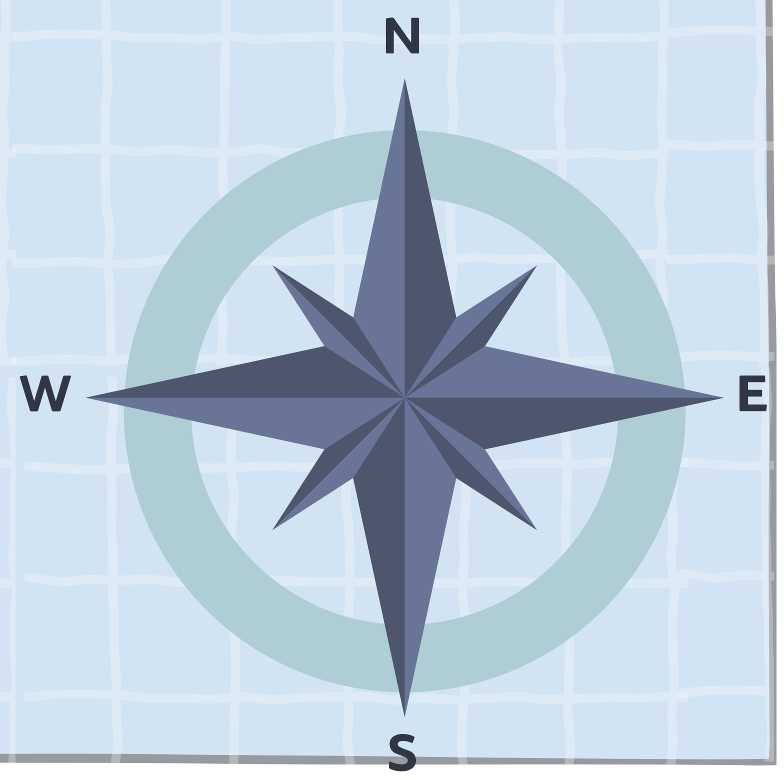
RUNTIME TERROR

MAPS

SYSTEM ARCHITECTURE AND DESIGN





MST

1.System Design:

- Loads four datasets: new_roads, neighborhoods, facilities, and roads.
- Converts all IDs to strings for consistency.
- Builds a list of potential roads, each with a dynamically adjusted cost based on:
- High population areas (population_factor)
- Facility connectivity (facility_factor)
- If apply_priority is True, those factors are applied to reduce cost for more important roads.
- Final MST is built from these roads with optional budget limits.

2. Algorithm Implementation

- kurskal Uses a Disjoint Set (Union-Find) class to avoid cycles and build the MST.
- Sorts proposed roads by adjusted cost.
- Iteratively adds roads that connect disjoint components.
- Stops adding new roads when max_budget is reached (if limit_budget = True).
- After MST construction, ensures full connectivity by rechecking components.

3. Performance and Results

- The function returns:
  - final_mst: all selected roads (new and reused old roads)
  - total_cost: total cost of the selected roads
  - len(extra_edges): number of old roads added to complete connectivity
- The result balances cost-efficiency and critical priorities like population and infrastructure.

4. Challenges and Solutions

- Inconsistent data formats → Solved by converting IDs to strings
- Priority weight tuning → Handled via population_factor and facility_factor
- Connectivity gaps after MST → Solved by reintegrating essential old roads
- Budget control → Implemented by stopping additions once max_budget is reached

SHORTEST PATH

1. System Design

- Loads three datasets: Existing_Roads.csv, Facilities.csv, and Neighborhoods.csv.
- Constructs a directed graph using Network with real distances as weights.
- Incorporates traffic multipliers and location coordinates.
- Supports both normal and emergency pathfinding.

2. Algorithm Implementation

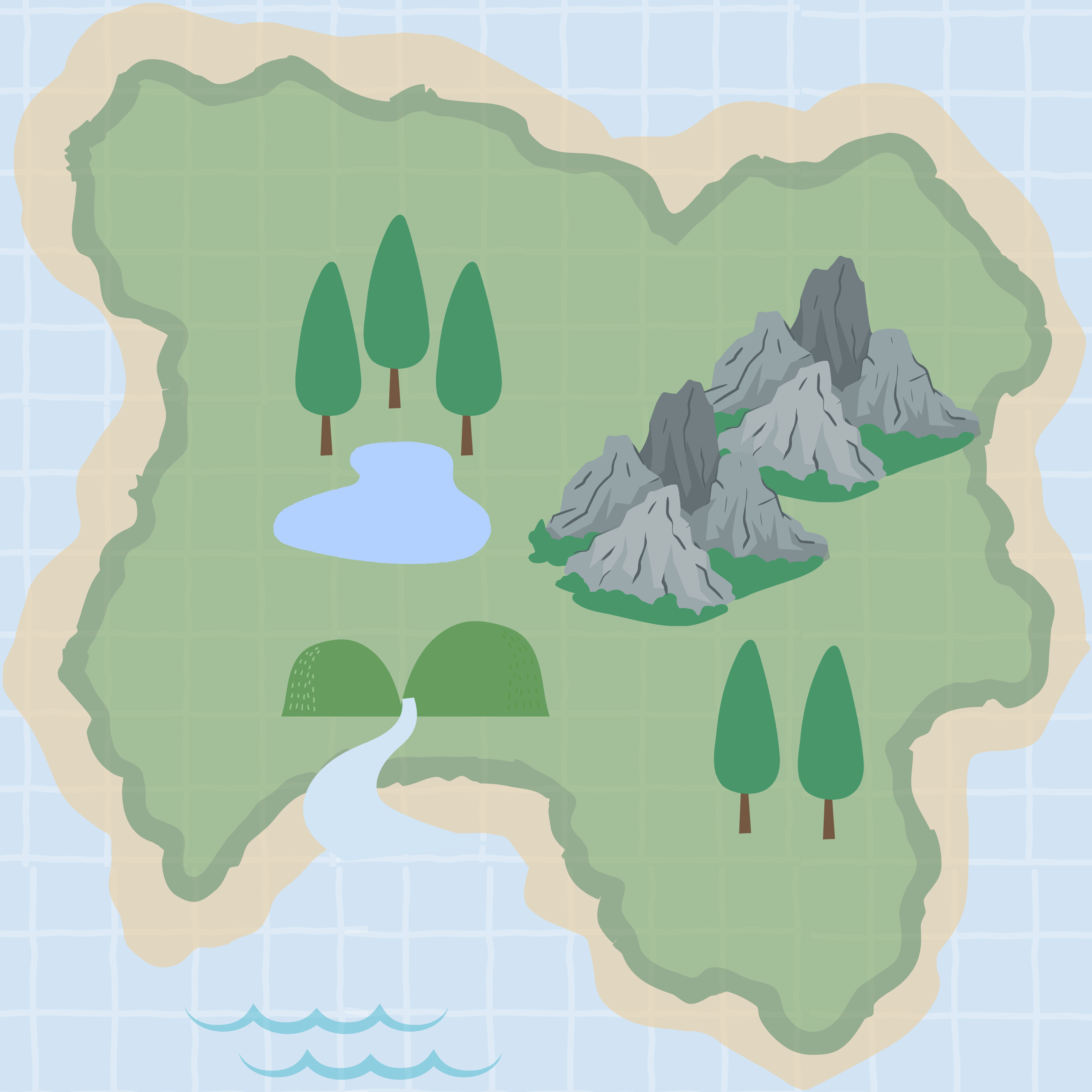
- Implements:
    - Dijkstra’s Algorithm for general shortest path.
    - A* algorithm for emergency routing to the nearest hospital.
- Heuristic in A* uses Euclidean distance between nodes (Latitude, Longitude).
- Graph weights are dynamically adjusted based on traffic time (e.g., morning).

4. Challenges and Solutions

- Missing coordinates → Used default (0, 0) when unavailable.
- Dynamic weights → Handled via traffic multipliers per road segment.
- Multiple targets (A*) → Compared all hospitals to pick the nearest.
- Fallbacks on failure → Try/except used to avoid crashes if paths are not found.

3. Performance and Results

- Dijkstra: Finds the shortest route from source to target (N1 → N5).
- A*: Finds nearest hospital to a source (N3) with minimum travel cost.
- Traffic-aware routing: Dijkstra with adjusted weights reflects peak hour delays.
- Prints: Path + distance (in kilometers) for each case.



GREEDY

The system is designed to optimize traffic signal directions by analyzing traffic volume and emergency route data. It processes inputs using a modular structure: data loading, direction estimation based on coordinates, and output generation per intersection per time period.

1. System Design

2. Algorithm Overview

A greedy logic is applied:
*If an emergency vehicle route is detected at a specific intersection and time period, the green light is assigned in that direction.
 If no emergency, the direction with the highest vehicle count is selected.
Coordinate-based estimation (east/west/north/south) is used to infer directions.
Time complexity is approximately O(N + I*T), where:

- N = number of traffic records
- I = number of intersections
- T = 4 fixed time periods

3. Performance Summary

The output greedy_signal_results.csv contains the optimal green light direction for each intersection.

- Ensures emergency access when needed.
- Reduces congestion by favoring busy directions.

4. Challenges and Solution

Missing coordinates:handled with "unknown" fallback.
Period format mismatch:solved via mapping dictionary.

DYNAMIC PROGRAMMING

1. System Design

- The system consists of three integrated modules:
- Signal Control Module:Determines optimal green light directions based on traffic volume and emergency routing.
- *Maintenance Planning Module:Uses dynamic programming (Knapsack) to select critical roads for repair under budget constraints.
- Transit Scheduling Module: Allocates buses and metro trains efficiently across routes to maximize service coverage.

2. Algorithm &Techniques Used

- Greedy Signal Optimization: Emergency routes are prioritized; otherwise, highest traffic direction is selected.
- Dynamic Programming (0/1 Knapsack):
- Road Maintenance: Chooses high-priority roads based on condition and traffic impact.
- Bus Scheduling: Distributes buses to routes based on passenger count and stops.
- Metro Scheduling: Allocates trains based on daily ridership and station count.

Results and Evaluation

Signal Optimization: CSV output provides direction per intersection and time period (greedy_signal_results.csv).
Maintenance Plan: Ensures cost-effective selection of roads within budget.
Transit Allocation:Boosts transit coverage while respecting vehicle constraints.

Key Challenges and Resolutions

Data Gaps: Handled missing coordinates and unmatched time formats using default logic and mapping.
Resource Limitation:Resolved via optimal selection using dynamic programming.