Key Algorithms and Implementation Details

A. Nearest Neighbor Search Algorithm

The nearest neighbor search algorithm is implemented in the function `findClosestResponder`, located in `src/components/dispatch-dashboard.tsx`. Its primary purpose is to identify the geographically closest available responder unit for a newly reported incident. The algorithm iterates through the list of all available responder units, computing the distance from each unit to the incident location using the Haversine formula,

The Haversine formula calculates the great-circle distance (the shortest distance over the surface of a sphere) between two points on a sphere, given their latitude and longitude implemented via the geolib library. This approach provides accurate great-circle distance calculations between two geographical points on the Earth's surface. The unit with the minimum computed distance is selected for dispatch.

B. AI-Powered Text Classification and Entity Extraction

The text classification and entity extraction process is implemented in `src/ai/flows/analyze-report.ts`. This algorithm performs two critical tasks using a Large Language Model (LLM):

1. **Classification: The algorithm classifies free-text emergency reports submitted by callers into predefined categories such as "Cardiac Arrest" or "Fire". It employs few-shot prompting, wherein a set of representative examples is provided in the model prompt, enabling the model to perform accurate classification even with limited training data.**

2. **Entity Extraction: The algorithm also extracts key phrases from the report that are most influential in the classification decision. These extracted entities are later utilized to enhance system transparency and decision support.**

C. AI-Powered Recommender Systems

Two key flows utilize the LLM as a sophisticated constraint-based recommender engine:

1. **Dispatch Package Recommendation (`src/ai/flows/get-dispatch-package.ts`): Given a complex incident, the system recommends the optimal number and types of response vehicles based on incident severity and requirements.**

2. **Hospital Recommendation (`src/ai/flows/recommend-hospital.ts`): The system recommends the most appropriate hospital by analyzing multiple factors, including required medical capability, bed availability, distance, and live traffic conditions. The recommendation provides decision support to dispatchers for optimal patient routing.**

D. Routing and Pathfinding Algorithm

Implemented in the `updateRoute` function within `src/components/map-layout.tsx`, the routing functionality leverages the Open Source Routing Machine (OSRM) as an external service. Upon dispatching a unit, the system performs an API call to OSRM, which computes the fastest driving route between the responder’s current location and the incident location. OSRM internally utilizes advanced algorithms such as Contraction Hierarchies, an optimized variant of Dijkstra's algorithm specifically designed for road networks. The resulting sequence of coordinates is used to animate the vehicle's movement on the system map in real time

---

✅ Nearest Neighbor Search Algorithm

Location: `src/components/dispatch-dashboard.tsx` (in the `findClosestResponder` function)
Purpose:
Finds the geographically closest available responder to a new incident by:

- Iterating through all available units.
- Calculating the distance from each unit to the incident using the **Haversine formula** (via the **geolib library**) for accurate great-circle distance calculation.
- Selecting the unit with the **minimum distance**.

✅ AI-Powered Text Classification & Entity Extraction

Location: `src/ai/flows/analyze-report.ts`
Purpose:
Analyzes free-text emergency reports from callers through:

1. **Classification**
   Uses a Large Language Model (LLM) with **few-shot prompting** to classify the report into predefined incident types (e.g., "Cardiac Arrest," "Fire"). Few-shot prompting improves accuracy by giving rich examples in the prompt.

2. **Entity Extraction**
   Extracts key phrases from the report that influenced the classification decision, supporting downstream tasks.

✅ AI-Powered Recommender Systems

Location:

- `src/ai/flows/get-dispatch-package.ts`

- `src/ai/flows/recommend-hospital.ts`
  Purpose:
  Leverages LLM as an advanced **constraint-based recommender system** for:

1. **Dispatch Packages**
   Recommends the optimal number and types of vehicles required for complex incidents.

2. **Hospital Recommendation**
   Recommends the most suitable hospital based on multiple factors:

    - Medical capability required.
    - Bed availability.
    - Distance from incident.
    - Live traffic conditions.

✅ Routing & Pathfinding Algorithm

Location: `src/components/map-layout.tsx` (in the `updateRoute` function)
Purpose:
Uses **Open Source Routing Machine (OSRM)** to calculate efficient driving routes by:

- Making API calls to OSRM.
- OSRM employs optimized routing algorithms such as **Contraction Hierarchies** (an efficient variant of Dijkstra's algorithm for road networks).
- Receives a sequence of coordinates representing the route.
- Animates the vehicle movement on the map based on the route.

.