SOURCE CODE DOCUMENTATION

1. Project Overview

The **AI-Powered Command Center** is designed to optimize emergency response management through intelligent automation, real-time tracking, and AI-driven decision-making. The system integrates data from multiple sources, classifies incidents, recommends dispatch units and hospitals, and visualizes routes dynamically.

---

2. Major System Components

Major System Components

A. AI Subsystem (src/ai/flows/)

Handles all AI-driven decision-making tasks using Google **Genkit** integration.
Modules:

- **analyze-report.ts** – Classifies emergency reports using LLM (few-shot prompting) and extracts key entities.
- **get-dispatch-package.ts** – Suggests the optimal type and number of responder vehicles.
- **recommend-hospital.ts** – Suggests the most suitable hospital (based on capacity, distance, and traffic).
- **get-protocol.ts** – Generates procedural checklists.
- **get-traffic-report.ts** – Provides live traffic analysis for routing.
- **summarize-incident.ts / debrief-incident.ts** – Create post-incident summaries.
- **genkit.ts** – Manages communication with Genkit AI API.

B. Dashboard Components (src/components/dashboard/)

Provide interfaces for monitoring and control:

- **Analytics Dashboard** – Displays KPIs and trends.
- **Dispatch Dashboard** – Central hub for live incident tracking and unit management.
- **Fleet Status Monitor** – Tracks responders in real time.
- **Incident Summary & Logging** – Maintains event history and audit trail.
- **New Incident Form** – For manual or AI-assisted report submission.

C. Incident Management (src/components/incident/)

Handles lifecycle management:

- **Incident Card/List** – Visual incident summary.
- **Incident Details & Debrief** – AI-generated post-response evaluations.

D. Mapping Components (src/components/map/)

- Uses **MapLibre GL JS** and **React Map GL**.
- **map-layout.tsx** integrates **OSRM API** for optimized routing.
- **map-route.tsx** visualizes computed driving routes.

E. UI Components (src/components/ui/)

- Built using **ShadCN UI** and **Tailwind CSS**.
- Provides reusable design elements (buttons, forms, modals, etc.).

F. Utility Libraries

- **Theme management**, **device detection**, and **toast notifications**.
- Includes **types.ts**, **data.ts**, and **utils.ts** for type safety and reusability.

---

3. Key Algorithms & Implementation Details

Key Algorithms and Implementati…

A. Nearest Neighbor Search

- **Location:** `src/components/dispatch-dashboard.tsx`
- **Purpose:** Find the geographically closest available responder to a new incident.
- **Logic:**
    - Iterates through available units.
    - Calculates great-circle distance using **Haversine formula** via **geolib**.
    - Selects the unit with the minimum distance.

B. AI-Powered Text Classification & Entity Extraction

- **Location:** `src/ai/flows/analyze-report.ts`
- **Purpose:** Classify emergency reports and extract influential entities.
- **Implementation:**
  - Uses **LLM with few-shot prompting**.
  - Identifies report categories (e.g., “Cardiac Arrest”, “Fire”).
  - Extracts key text entities for decision transparency.

C. AI-Powered Recommender Systems

- **Dispatch Package Recommendation:** `src/ai/flows/get-dispatch-package.ts`
- **Hospital Recommendation:** `src/ai/flows/recommend-hospital.ts`
- **Logic:**
  - Uses LLM as a **constraint-based recommender**.
  - Considers medical capacity, bed status, distance, and traffic.

D. Routing & Pathfinding

- **Location:** `src/components/map-layout.tsx`
- **Purpose:** Compute optimal driving routes via **Open Source Routing Machine (OSRM)**.
- **Algorithm: Contraction Hierarchies** (optimized **Dijkstra’s algorithm**).
- **Output:** Sequence of coordinates used to animate responder movement.

---

4. Source Code Documentation Enhancements

Source Code Details

A. map-layout.tsx — Simulation Engine

- Handles **real-time simulation** of responder movements.
- Includes **finite state machine logic** for unit transitions:
  - Available → Enroute → On Scene → Transporting.
- Documents **state variables**, **transition logic**, and **path updates**.

B. dispatch-dashboard.tsx — Main UI Controller

- Central control interface for dispatchers.
- Documents:
  - **Component props** (data and callbacks).
  - **View management** (switch between forms, analytics, lists).
  - **Core functions** (e.g., `createNewIncident` for initiating workflows).

C. new-incident-form.tsx — Human-AI Interaction

- Documents:
  - **Form state management** for user input.
  - **Asynchronous AI flow triggers** (integration with `analyze-report.ts`).
  - **Response handling** and **confirmation logic** for dispatch initiation.

D. analyze-report.ts — Backend AI Logic

- Documents:
  - **Input processing** for unstructured, multi-language text.
  - **Prompt design** for LLM interaction.
  - **Output structuring** into standardized JSON responses.

E. Documentation Purpose

- Improves **readability**, **maintainability**, and **onboarding efficiency**.
- Establishes transparent logic flow and structured commenting across modules.

5. Technical Stack Summary

Layer	Technology
Frontend	React, TypeScript, TailwindCSS, ShadCN UI
AI Subsystem	Google Genkit (LLM), Few-shot prompting
Mapping	MapLibre GL, OSRM API
Algorithms	Haversine (Geolib), Dijkstra (Contraction Hierarchies)
Backend	Node.js / API Integration
Utilities	Hooks, Type Definitions, Toast Notifications

6. Conclusion

The codebase is now **fully documented**, **modular**, and **AI-integrated**.
Each subsystem — AI, Mapping, and Dashboard — has been explicitly commented for future scalability. The system enables **intelligent dispatch**, **real-time route visualization**, and **human-AI collaboration** for improved emergency response efficiency.