Aegis: Intelligent Mission Control for Paramedics

> **Mission:** Faster Response. Smarter Support. Safer Lives.
>
> **Category:** Healthcare • Re-engineering
>
> **Sponsor Alignment (demo narrative):** York Region (EMS / Fire / Community Safety)
>
> **Core Demo Claim:** Real road-law routing + GPS-style simulation + live navigation telemetry + AI triage assistant + algorithmic routing telemetry (Dijkstra vs Duan–Mao BM-SSSP).

> **Safety / Compliance Disclaimer (read this first):**
>
> Aegis is a **hackathon prototype** for demonstration and research exploration. It is **not** a medical device, **not** certified clinical decision support, and **must not** be used to make real patient-care decisions or live dispatch decisions. AI outputs are **advisory only** and must be validated against official protocols.

---

1) Executive Summary

The problem

Emergency crews often operate with **fragmented tooling**:

- Consumer GPS: routing and ETA, but not dispatch-aware and not built for mission workflows
- Radio/dispatcher updates: not integrated into route logic or a single UI
- Separate systems for patient notes, protocols, and equipment checks

This fragmentation increases **avoidable cognitive load** and can contribute to **delays**, exactly when time and attention are most scarce.

The solution: Aegis

Aegis is a **single-screen in-vehicle command dashboard** concept that fuses:

- **Real routing** on a directed road network (one-ways respected)
- **Turn-by-turn navigation** with live ETA and “next maneuver in X meters”

- **GPS-style simulation** (vehicle icon follows the route, follow-camera)
- **AI clinical copilot** for concise protocol guidance and summaries (voice-capable)

Demo KPI (York Region alignment)

Our demo uses a **6-minute response-time target** as a **north-star KPI** for high-priority calls. Aegis is built around two practical levers that can support that goal:

1. reducing wrong turns / missed maneuvers under stress
2. reducing re-route latency when disruptions occur

> Note: This KPI is used for demo framing and sponsor alignment; operational targets vary by call type and policy.

---

2) What’s Real Today vs What’s Next

What’s real today

- **Routing is real**: OpenStreetMap drive network routing via OSMnx + NetworkX (directed graph)
- **Vehicle simulation is real**: marker follows the route polyline and stays on-road (snapped)
- **Navigation telemetry is real**: remaining distance, remaining ETA, next instruction, current street
- **AI assistant endpoints exist**:
  - Gemini for responses
  - optional ElevenLabs TTS
  - demo audio fallbacks included
- **Algorithmic Race is real**:
  - minimap replay loop of Dijkstra vs Duan–Mao BM-SSSP exploration + final path
  - expanded Bloomberg-style telemetry (KPIs, trend lines, histograms, benchmark runner)

What’s next (roadmap)

- Dispatch feed integration (WebSocket + incident updates)

- Incident/closure integration (public data feeds) → automated reroute logic
- “Black box” mission logging (audit + replay) for compliance and training

3) Interface Layout (Single-Screen Grid)

Aegis uses a **glanceable dashboard layout** around a live map canvas.

Design principles

- Glanceable + motion-safe: large type, few focal points, progressive disclosure
- Two visual modes (concept):
  - **Cruise Mode:** low clutter, “glassy” panels
  - **Red Alert Mode:** high contrast, larger nav instructions, fewer distractions, more audio cues

Zone	Panel	What it shows	Data source today	Planned expansion
Center	**Live Map (Canvas)**	2D/3D map, route polyline, vehicle icon, follow camera	MapLibre GL	incident overlays, more map layers
Top Left	**Live Updates**	dispatch notes, scene changes, unit status	mock/static	WebSocket dispatch + alerts
Mid Left	**AI Assistant**	protocol guidance & concise summaries (optional voice)	Gemini + (optional ElevenLabs)	structured outputs + tool use
Bottom Left	**Equipment Diagnostics**	O2/defib/supplies/vehicle health (demo feed)	mock feed	IoT/BLE integrations
Top Right	**Nav-Com**	next maneuver, ETA, distance remaining, current street	**real route metadata**	auto reroute, “nearest ER”
Mid/Bottom Right	**Patient / Mission**	patient summary, vitals, timeline	mock/placeholder	mission event log + replay

4) Navigation: What Makes It “Real”

Road-law routing

- Routes run on the **OSM “drive” network** (directed graph): one-ways are respected.

- Start/end are **snapped** to the nearest drivable node to avoid “inside buildings” artifacts.

Live navigation telemetry

The backend returns:

- `path_coordinates` (route polyline)
- `cum_distance_m[]` and `cum_time_s[]` aligned to polyline points
- `steps[]` (maneuvers derived from bearings + street-name changes)
- totals: `total_distance_m`, `total_time_s`

The frontend displays:

- **Next instruction** and **distance to next maneuver**
- **Current street**
- **ETA remaining**
- **Remaining distance**

Simulation profiles (demo mode)

Simulation pacing + UI emphasis can change by scenario:

- Routine
- Trauma
- Cardiac Arrest

(In production, this could be derived from dispatch type + policy constraints.)

---

5) Algorithmic Core: Duan–Mao (BM-SSSP) vs Dijkstra

Baseline: Dijkstra

Dijkstra’s algorithm is the reliable workhorse for non-negative edge weights and is excellent in practice on many graphs.

Experimental accelerator: Duan–Mao BM-SSSP (“Breaking the Sorting Barrier”)

Recent research introduces a deterministic directed SSSP algorithm with improved asymptotic runtime in certain models, often described as “breaking the sorting barrier.”

How Aegis uses this responsibly

- **Default:** Dijkstra (robust and predictable)
- **Optional:** BM-SSSP runner (Node/TypeScript) as an accelerator path
- **Fallback:** if BM-SSSP fails or overhead dominates, Aegis falls back automatically

Why this matters for emergency response workflows
Routing is rarely “compute once.” In an operational setting, you can have:

- closures/incidents forcing **reroute**
- repeated “nearest facility” queries
- multi-unit planning extensions

A faster recompute path can reduce time-to-care when routing triggers frequently.

> Realism note: On small subgraphs, asymptotic wins may not appear due to constants/overhead.
>
> That’s why Aegis ships both and benchmarks them visibly.

---

6) Algorithmic Race: Telemetry + Benchmarking

Aegis includes an **Algorithmic Race** mini-map that visually compares:

- explored “footprint” (what the search touches)
- final path
- speed and consistency

Expanded overlay includes

- KPI header strip:
  - Winner
  - Speedup (×)
  - Explored Δ
  - ETA Δ
- Trend lines:
  - explored edges over time
  - completion % over time

- Histograms:
  - route segment-length distribution
  - benchmark exec-time distributions (RUN 20×)

This is intentionally designed as a “Bloomberg-style” evidence display when asked:

> “What is this algorithm?”
>
> “Does it actually help?”
>
> “When is it better?”

---

7) Tech Stack & Architecture

Frontend

- Vite + React + TypeScript
- TailwindCSS
- MapLibre GL (OSM-friendly, open source)
- Recharts for telemetry charts

Backend

- FastAPI (Python) with interactive docs at `/docs`
- OSMnx + NetworkX for graph retrieval + routing
- Optional BM-SSSP runner (Node/TS) for Duan–Mao path
- AI endpoints:
  - Gemini (assistant)
  - optional ElevenLabs (TTS)

Data layer (future-proofing)

The prototype runs without persistent storage. Planned sponsor-friendly expansions:

- immutable mission logs (“black box” replay/audit)
- operational JSON stores (incident payloads, patient packets, device telemetry)
- event streaming (WebSockets now; later Kafka/PubSub)

---

8) Repository Structure

```
.
├─ docs/
│  └─ algorithm_for_map.pdf
├─ backend/
│  ├─ .env.example
│  ├─ requirements.txt
│  ├─ bmssp-runner/
│  │  ├─ package.json
│  │  ├─ run.mjs
│  │  └─ server.mjs
│  └─ app/
│     ├─ main.py
│     ├─ services/
│     │  ├─ gemini.py
│     │  └─ voice.py
│     └─ algorithm/
│        └─ router.py
└─ frontend/
   ├─ package.json
   ├─ vite.config.ts
   └─ src/
      ├─ App.tsx
      ├─ components/
      │  ├─ Map.tsx
      │  ├─ AlgoRaceMiniMap.tsx
      │  ├─ AlgoRaceCharts.tsx
      │  ├─ AlgoBenchmarkCharts.tsx
      │  └─ panels/...
      └─ constants/...
```

---

9) Setup & Commands

> Full step-by-step instructions live in `README.md`.
>
> This section is a concise doc-friendly reference.

Backend

```
cd backend
python -m venv .venv
# macOS/Linux:
source .venv/bin/activate
# Windows PowerShell:
# .venv\\Scripts\\Activate.ps1
pip install -r requirements.txt
uvicorn app.main:app --reload --port 8000
```

Frontend

```
cd frontend
npm install
npm run dev
```

Optional: BM-SSSP runner

```
cd backend/bmssp-runner
npm install
```

10) Demo Script

Global notes

- Default start location is pre-set (demo-friendly).
- Judges can type a destination → Aegis computes a real route and simulates movement.
- Aegis should fail gracefully (fallback routing, safe UI).

Stage 0 — IDLE (Standby)

Goal: prove "real navigation loop."

1. open Aegis
2. enter destination (hospital/POI)
3. GO → route draws → vehicle follows polyline

4. toggle follow camera

Suggested judge prompts

- “Summarize route status in one sentence.”
- “What’s the next maneuver and ETA?”

Scenario 1 — Cardiac Arrest (High Priority)

Goal: show urgency UI + AI protocol bullets.

- Red Alert style (high contrast, bigger nav)
- Patient/vitals panel enabled (clearly labeled simulated)

Suggested AI prompts

- “20-second CPR checklist. Bullet points only.”
- “Adult cardiac arrest epi dosing guidance (concise).”
- “Provide ED handoff summary template.”

Scenario 2 — MVA Trauma (Scene → Load → Transport + Reroute)

Goal: show state transitions + disruption reroute.
Timeline:

1. Navigate to scene
2. On-scene pause (“loading patient”)
3. Patient loaded → trauma mode
4. Inject road block → reroute computed
5. show updated ETA and nav steps

Suggested AI prompts

- “ABCDE rapid trauma assessment checklist.”
- “Hemorrhagic shock warning signs during transport.”

---

11) Professional Disclaimers

- **Prototype only**: not a certified system for clinical decision-making or dispatch operations.
- **AI outputs**: informational; users must follow official protocols.

- **No PHI**: do not input real patient identifiers into AI prompts.
- **Map/data reliability**: OpenStreetMap/Overpass/Nominatim are community services; production use requires offline caching and/or enterprise geodata.

12) Next Additions

- offline cached York Region graph (demo reliability + speed)
- closure/incident ingestion + auto reroute
- mission event logging + replay for training/compliance
- multi-unit selection (best unit by travel time)
- protocol cards + structured handoff generator

Appendix: Team Instigate Cafe

- **Sukesan** | Systems / Backend / AI integration
- **Yazanth** | Map / Routing / Simulation
- **Sanchit** | UI/UX / Frontend
- **Nithursan** | Data / Logging / Infrastructure