

NAVAL INTELLIGENCE PLATFORM

A Real-Time Maritime Surveillance Web Application



Agenda

What we'll walk through

01 Project Overview & Motivation

02 Project Objectives

03 System Architecture

04 Technology Stack

05 Milestone 1 — Vessel Tracking

06 Milestone 2 — Environmental Intelligence

07 Milestone 3 — System Health Monitoring

08 Milestone 4 — Fleet Analytics

09 Challenges & Engineering Decisions

10 Conclusion

Project Overview

Why this project

THE TRADEOFF

Maritime operations centers depend on aggregating live vessel positions and environmental conditions into a single situational picture.

Commercial platforms like MarineTraffic are closed-source and expensive to license.

Building from raw feeds means stitching together unstable government APIs and managing real-time rendering at scale.

This project: a focused, engineering-rigorous demonstration of how those tradeoffs can be solved with modern web technologies.

1,400

Vessels tracked simultaneously

4

Milestones — all complete

3

External APIs integrated

<100m

S

Marker update cycle at peak load

Project Objectives

Four objectives, four application modules



OBJECTIVE 01

Real-Time Vessel Tracking

Live AIS positions on an interactive Leaflet map with custom heading-rotated SVG markers, marker clustering, and click-to-inspect popups.



OBJECTIVE 02

Environmental Intelligence

Wave height and wind direction overlays from Open-Meteo, plus a six-control filter panel covering category, motion, speed, destination, status, and region.



OBJECTIVE 03

System Operations Monitoring

API status cards, live category bar chart, and a 150-entry circular-buffer log feed — polled every 5 seconds.



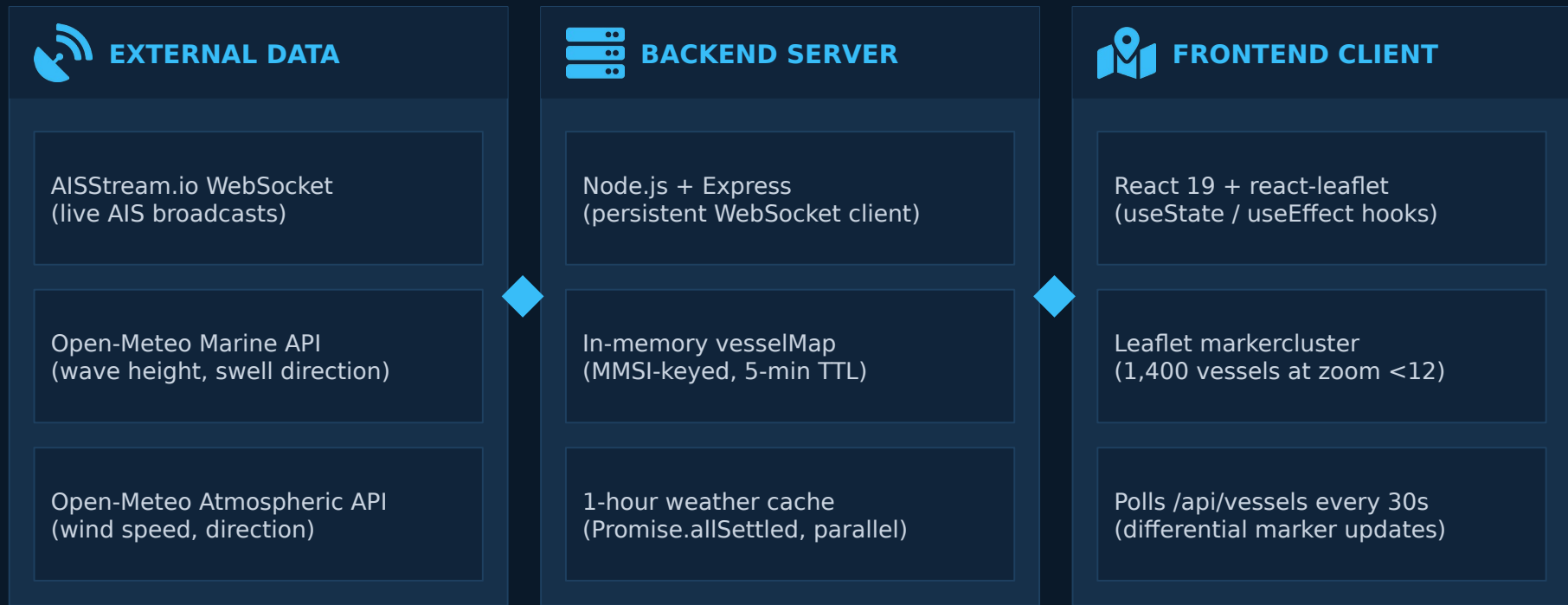
OBJECTIVE 04

Fleet Analytics Dashboard

Ten sections of live statistics: composition, speed distribution, regional breakdown, top destinations, weather summaries — all computed client-side.

System Architecture

Three-tier pipeline — WebSocket ingestion, REST polling, in-memory state



Data flow: AISStream.io → Node.js WebSocket → in-memory state → REST poll every 30s → React + Leaflet re-render

Technology Stack

Selected for real-time, event-driven workloads

Frontend

React 19 (CRA)

Functional components + hooks for live state

Leaflet + react-leaflet

Interactive map, layered overlay system

leaflet.markercluster

1,400 vessels rendered without freezes

OpenStreetMap + CARTO

Free global tiles, dark/light themes

Backend

Node.js

Event-driven, non-blocking I/O

Express.js

REST API: /api/vessels, /weather, /system

ws

Persistent WebSocket client to AISStream.io

dotenv

API key isolation, never in client bundle

APIs & Tooling

AISStream.io

Real-time AIS via WebSocket

Open-Meteo Marine

Wave height, swell direction

Open-Meteo Atmospheric

10 m wind speed and direction

GitHub

PR-based feature workflow

Milestone 1 — Core Infrastructure & Vessel Tracking

Target: March 14, 2026



COMPLETE

DELIVERABLES

- Interactive Leaflet map with light/dark CARTO basemaps
- Custom heading-rotated SVG vessel markers (true heading, COG fallback)
- Marker clustering with leaflet.markercluster (zoom-12 cutoff)
- Persistent AISStream.io WebSocket with auto-reconnect
- In-memory vesselMap, MMSI-keyed, 5-minute TTL cleanup
- Differential marker diffing — no flicker on 30s polls
- Vessel data panel with CSV export
- System health status bar with live log feed

HOW IT WORKS

- 1 AISStream.io** broadcasts vessel updates
- 2 Node.js backend** parses, merges by MMSI
- 3 GET /api/vessels** returns latest snapshot
- 4 React polls 30s** diffs MMSI sets, updates
- 5 Leaflet re-renders** only changed markers

Milestone 2 — Environmental Intelligence

Open-Meteo overlays + six-control vessel filter



COMPLETE



Wave Height Overlay

20

ocean waypoints

Color-coded L.circle markers at 800 km radius. Six conditions: Calm → Dangerous. Hover for height, condition, swell direction.



Wind Direction Overlay

90

grid points (5°×5°)

Animated rotating SVG arrows with staggered CSS pulse. Color-coded by wind speed: green ≤ 8 m/s, orange to 15, red above.



Six-Control Filter Panel

6

independent filters

Category (19 AIS types), motion, speed range, destination with port chips, navigational status, geographic region.

Milestone 3 — System Health & Live Monitoring

Live observability into the platform itself



COMPLETE



API Status Cards

One card per service (AIS Stream, Weather API). Colored status dot — green / pulsing yellow / red. Vessel count, message count, last fetch time, data point counts.



Live Category Bar Chart

Horizontal bar chart of vessel count by category, sorted, color-matched to map markers. Updates every 5-second poll cycle as vessels arrive, depart, or are reclassified.



Live Log Feed

150-entry server-side circular buffer in a 12-line logger.js module. Color-coded by source (AIS / Weather) and level (info / warn / error). Auto-scrolls newest-first.

Endpoint: [GET /api/system/health](#) | Polled every 5 seconds | Returns full health snapshot + 150-entry log buffer

Milestone 4 — Fleet Analytics Dashboard

Ten sections of live statistics, computed entirely client-side



COMPLETE

01

Fleet Overview Cards

6 summary cards

02

Composition by Type

Horizontal bar chart

03

Speed Distribution

6-bucket histogram

04

Navigational Status

10 AIS status codes

05

Regional Distribution

Pacific / Gulf / Carib / Atl

06

Speed Intelligence

Avg, median, max, % underway

07

Top 10 Fastest Vessels

With medal indicators

08

Top 10 Destinations

Port pills by category

09

Wave Height Stats

Avg / max / min, 20 points

10

Wind Speed Stats

Avg / max / min, 90 points

Architecture: *all stats computed in the browser from raw `/api/vessels` and `/api/weather/data` — server stays stateless. **~5 ms** to compute all ten groupings on 1,400 vessels.*

Challenges & Engineering Decisions

Real constraints, pragmatic tradeoffs

Open-Meteo Replaced NOAA

Three NOAA endpoints failed: NWS lacks bulk queries, ERDDAP returns 302→404, NDBC needs 50 sequential fetches (15–30s). Open-Meteo: batched, 2s, no key.

MMSI-Keyed Marker Diffing

Naive recreation froze the browser for 30s on every poll. Fix: diff MMSI sets, update markers in place, leverage React reconciliation. Result: <100 ms cycles.

L.circle Beat leaflet.heat

Heatmap canvas clipped at viewport boundary, producing a quarter-circle artifact over the Atlantic. Switching to L.circle with 800 km geographic radius fixed it cleanly.

Cut Historical Tracking

Required a paid AISStream.io tier outside project budget. Dropped the feature entirely and reinvested time into polishing real-time. M1 shipped cleaner because of it.

WebSocket Reconnect Guards

Concurrent reconnect timers spawned duplicate connections. Fix: OPEN-state guard at startAISStream(), single module-scoped timeout handle cleared before each schedule.

Stateless Server Choice

Polling chosen over server-sent events. Backend tracks no clients, no WebSocket-to-browser lifecycles. Stateless property makes horizontal scaling trivial if ever needed.

Live System — All Milestones Complete

What is built, working, and demoed today

LIVE



Vessel Tracking

1,400 live vessels, clustered, color-coded

LIVE



AIS WebSocket

Persistent, auto-reconnecting

LIVE



Weather Overlays

Wave (20 pts) + wind (90 pts)

LIVE



Vessel Filtering

Six independent map filters

LIVE



System Monitor

Status cards, log feed, category chart

LIVE



Fleet Analytics

Ten sections, computed in browser

Conclusion

Engineering takeaways from a real-time defense-adjacent system

01

Scope management is a technical skill

Cutting historical tracking, swapping NOAA for Open-Meteo, replacing leaflet.heat with L.circle — each was a deliberate deviation from the spec to deliver something that actually worked.

02

Stable keys are foundational to real-time UI

MMSI as React's reconciliation key was the single highest-leverage performance fix in the project — taking marker updates from 30 s freezes to under 100 ms.

03

Event-driven architecture matches the data

Node.js's non-blocking I/O fits AIS naturally — one persistent WebSocket, concurrent REST polling, no thread management. Same pattern command centers actually use.

04

Independent rendering layers protect each source

Vessels refresh every 30 s, weather every hour. Leaflet's layer model lets each own its lifecycle without React orchestration — directly applicable to multi-source operational displays.