P&O Maritime Logistics, DP World Group Coding Challenge: Vessel Tracker Mobile App (MarineTraffic / Pole Star Inspired)

Assignment Objective: Vessel Tracker App
Duration: 3 Days
Target Platforms: React Native / Flutter / Kotlin / Swift
Focus: Mapping, geospatial data, filtering, real-time simulation

---

📋 Background

Vessel tracking platforms like **MarineTraffic** and **Pole Star Global** offer advanced marine intelligence, combining satellite data, AIS feeds, and analytics to help track maritime vessels across the globe. These systems play a critical role in port operations, security compliance, fleet efficiency, and maritime logistics.

This coding challenge simulates a simplified version of such a platform and is designed to evaluate your mobile development skills in mapping, UI/UX design, data handling, filtering logic, and optionally, simulation and risk analysis. You'll create a mobile app that can display vessels on a map, provide contextual information, and offer filtering and sorting capabilities to improve maritime situational awareness.

The app will be focused on representing vessels in a particular region (e.g., Arabian Gulf, Singapore Strait, or Red Sea). Interns are encouraged to explore how vessel information is used in logistics, emergency response, port call planning, or safety audits. While the use of actual marine APIs (like MarineTraffic) is out of scope due to licensing, this project will use free AIS data sources like AISstream.io and NOAA for vessel positioning and metadata.

🧭 Purpose of the App

The goal of the Vessel Tracker App is to allow port operators, shipping line managers, or maritime inspectors to visualize ships in real-time (or near real-time) on a digital map. In the app, users should be able to:

- Instantly identify vessels in their region of interest.
- Retrieve basic operational metadata (speed, heading, status).
- Filter ships based on activity, size, or type.
- Detect possible anomalies like anchored ships outside designated zones.
- View past travel routes and risk indicators.

- Forecast the vessel’s estimated route and arrival at future waypoints.
- Configure alerts based on location, ETA, or vessel characteristics.

Such functionality helps stakeholders make faster and more informed decisions regarding berthing, cargo readiness, compliance inspections, and coordination with logistics companies.

🧰 System Features Overview

The app will be built to support:

- Interactive GIS/mapping interface.
- Visual ship markers using lat/lon data.
- Information overlays/modal views with full vessel metadata.
- Filters and sorting logic to reduce data clutter.
- Historical tracking of vessel paths (breadcrumb trails).
- Support for offline or simulated data scenarios.
- Integration of mock geofencing zones (e.g., around port boundaries).
- Forecast routing and ETA calculations for upcoming ports or checkpoints.
- Custom alert settings for user-defined triggers.
- Configuration options for UI preferences and fleet monitoring filters.
- Scalability to handle hundreds or thousands of vessels in fleet view.

Each of the above features will map directly to a module or screen in your application. Proper componentization and use of state management are expected.

🔑 Required User Stories

1. As a user, I want to see vessels displayed on a map.

- So that I can understand their positions and operational areas.

2. As a user, I want to click/tap a vessel icon to view more details.

- So that I can determine where it is going, how fast it's moving, and what type of vessel it is.

3. As a user, I want to filter vessels by type or operational status.

- So that I can focus only on tankers or vessels currently underway.

4. As a user, I want to see the trail/history of a selected vessel.

- So that I can validate its recent route and behavior.

5. As a user, I want to identify high-risk or non-compliant vessels.

- So that I can alert port security or make further inquiries.

6. As a user, I want to search for a vessel by name or IMO.

- So that I can quickly locate a specific ship.

7. As a user, I want to view estimated arrival times at ports or waypoints.

- So that I can anticipate docking and cargo handling schedules.

8. As a user, I want to define geofenced areas and receive alerts.

- So that I'm notified when a vessel enters or exits restricted waters.

9. As a user, I want to customize which data fields and filters are visible.

- So that I can tailor the interface for specific operations.

10. As a user, I want the app to handle large fleets without performance issues.

- So that I can use it across multiple regions or vessel types.

📱 App Screens & Navigation Structure

The app must have at least the following main screens:

1. **Map Screen (Main)**
    - Interactive map with vessel markers.
    - Zoom controls and recentering functionality.
    - Floating filter/search panel or bottom sheet.
    - Layer toggle for trails, forecasts, and zones.
2. **Vessel Detail Modal**
    - Triggered when a marker is tapped.
    - Shows extended metadata (IMO, MMSI, ETA, Speed, Destination).
    - Button to view vessel trail and forecasted route.
    - Option to set alerts (e.g., ETA threshold, zone breach).

3. **Vessel List/Directory**
    - Sorted list of vessels (name, type, ETA, speed).
    - Filter controls (type, risk, port zone).
    - Search bar.
    - Infinite scroll or pagination support for large fleets.
4. **Settings or Configuration Screen**
    - Toggle simulation.
    - Map theme (dark/light).
    - Refresh interval.
    - Fleet segmentation (e.g., by region, size, flag state).
5. **Voyage Planner Screen** *(Advanced)*
    - Shows planned routes and estimated arrival at each waypoint.
    - Editable waypoints for simulated planning.
    - Integrated with map overlays and detail modals.
6. **Geofence Manager** *(Bonus)*
    - UI for defining and editing virtual port zones.
    - Alert rules per zone.
    - List of active zone alerts.
7. **Secure Profile Screen** *(Bonus)*
    - Detailed history and inspection records.
    - Flag state, previous ports of call.
    - Risk explanation.

Navigation should be managed using best practices for the chosen framework (e.g., react-navigation, go_router, or Android Jetpack Navigation).

📐 Data Format, Integration Options & Handling Best Practices

- Use a repository or service pattern to handle data loading and caching.
- On load, parse data into model classes (e.g., Vessel, VesselTrackPoint, ForecastLeg).

- Use mock delay (Future.delayed) to simulate network latency.
- Support updates every 10 seconds for position if simulating movement.

🌐 AIS Data Integration Options

To enhance realism and simulate real-time maritime tracking, you may optionally integrate open AIS (Automatic Identification System) data sources. The following are **free AIS providers** that can be used to enrich the application or replace mock JSON where feasible:

1. AISstream.io (Recommended for Simulation)

- Offers a **free WebSocket API** that streams live AIS messages in JSON format.
- Ideal for rendering real-time vessel locations and updates without needing an onboard feed.
- Low setup barrier and includes documentation for integration.
- Recommended for powering animated markers and live ETA recalculations.

2. NOAA (U.S. Coast Guard Historical AIS Data)

- Provides free access to **historical AIS data** in CSV format.
- Coverage includes U.S. coastal waters with precise timestamps.
- Useful for offline playback or route history simulation.

Integration Tips:

- Use AISstream WebSocket in development mode to drive animation.
- Store NOAA CSV files locally to enable route trail overlays.
- Build a middleware parser for JSON/NMEA if using raw feeds.
- Respect data licensing terms — these sources are for educational/demo purposes only.
- Forecast paths should include estimated timestamps and optional reasons (e.g., traffic, weather).

Data Optimization Tips:

- Normalize types and status enums.
- Minimize re-renders by memoizing static lists.

- Use flatList optimizations or ListView.builder for large datasets.
- Use background threads or debouncing for filter-heavy operations.
- Split heavy components (e.g., route planning logic) into lazy-loaded chunks.

🔒 Confidentiality & Intellectual Property

All code, documentation, designs, and materials submitted as part of this coding challenge are considered **confidential** and the **intellectual property of P&O Maritime Logistics**. By participating in this challenge, you agree that any and all output produced—whether partially complete, fully functional, or in draft form—is owned exclusively by P&O Maritime Logistics, DP World Group and may be used internally for evaluation, implementation, or further development.

Participants must not share, distribute, publish, or reuse the content of this challenge submission for other purposes without written permission from P&O Maritime Logistics, DP World Group.