Survale System Architecture & Data Model – v1.0

Overview:
The Survale MVP system enables small law enforcement teams (8–10 users) to coordinate in real time using live map updates, in-app chat, and target/staging management. The stack is optimized for reliability, privacy, and cost efficiency.

System Architecture:
Frontend: iOS (Swift/SwiftUI) – handles map display, chat, background GPS tracking, and PDF export requests.
Backend: Supabase (self-hosted PostgreSQL + Realtime) – stores operations, users, messages, and location data with row-level security (RLS).
Custom Service: Node.js/Express – handles PDF export rendering, auto-ending operations, proximity alerts, and scheduled purges.
Push Notifications: APNs or OneSignal for invites, proximity alerts, and background chat notifications.
Hosting: Docker Compose on a small cloud VM (Hetzner/Linode/AWS Lightsail) with TLS via Let's Encrypt.

Data Flow:
1. Case Agent creates an operation in Supabase and invites team members.
2. Members join and begin transmitting GPS updates every 3–5 seconds.
3. Supabase Realtime distributes updates over WebSockets to all connected clients.
4. Chat messages and media uploads are stored in Supabase Storage; direct messages remain ephemeral.
5. The Node service monitors disconnects, proximity between tenants, and 7-day purge timers.
6. After operations end, Case Agents can replay movements and export a PDF summary.

Core Entities (ERD Summary):
- agencies(id, name)
- teams(id, agency_id, name, active_user_cap)
- users(id, email, full_name, callsign, vehicle_type, vehicle_color, team_id, agency_id)
- operations(id, agency_id, team_id, case_agent_id, incident_number, name, status, timestamps)
- operation_members(operation_id, user_id, role, joined_at, left_at)
- operation_invites(operation_id, invitee_user_id, status, expires_at)
- targets(id, operation_id, type[person/vehicle/location], title, notes, created_by)
- staging_areas(id, operation_id, name, lat, lon, notes)
- locations_stream(id, operation_id, user_id, ts, lat, lon, accuracy_m, speed_mps, heading_deg)
- locations_archive(id, operation_id, user_id, ts, lat, lon, accuracy_m, speed_mps, heading_deg)
- op_messages(id, operation_id, sender_user_id, body_text, media_path, media_type, created_at)
- exports(id, operation_id, requested_by, status, storage_path)

Security & Retention:
- RLS ensures data isolation by agency and team.
- All tracking is disabled outside active operations.
- Operations auto-end 10 minutes after all users disconnect.
- Data purge occurs 7 days after operation end (DB + media).

Realtime Channels:
- op_{operation_id}_locations (GPS updates)
- op_{operation_id}_chat (operation-wide chat)

- dm_{userA_userB} (ephemeral direct messages)

Node Service Jobs:
- invite-expiry (1 hour)
- auto-end (10 min after disconnect)
- purge (7 days)
- proximity (1km cross-tenant alert)

Performance Notes:
- GiST indexes on geography columns (lat/lon).
- Materialized view for latest location per user.
- Stream retention limited to ~2 hours.
- Archive table used for replay and export.

Next Deliverable:
API & Backend Spec (Supabase RPC functions, Node endpoints, and data contracts).