SOLYRA: TECHNICAL BLUEPRINT & DEVELOPER BRIEF

A Probabilistic Geospatial Deduplication and Triage Framework
Based on the White Paper by Sohaib Khan (May 2026)

---

1. Project Concept & Innovation: Why Solyra is Different

In high-density refugee settlements and humanitarian crises, reporting is often fragmented, overlapping, and delayed. Multiple actors report the same issue, while critical incidents remain underreported, leading to a misuse of limited resources.

Existing tools (like KoboToolbox or ArcGIS) rely on manual deduplication, which introduces delays. **Solyra is not a replacement for existing GIS systems; it is a decision-support augmentation layer.** It shifts humanitarian crisis mapping from *deterministic mapping* (assuming every report is a unique fact) to *probabilistic, uncertainty-aware decision support*.

The Core Innovation: Solyra treats humanitarian deduplication as a probabilistic entity resolution problem under spatiotemporal uncertainty. It explicitly represents uncertainty rather than hiding it behind deterministic labels.

---

2. System Architecture (The 4-Layer Blueprint)

Solyra is designed for low-bandwidth deployment scenarios and supports asynchronous synchronization where persistent connectivity is unavailable. The architecture is divided into four primary layers:

Layer 1: Ingestion Layer

- **Channels:** SMS, Mobile Apps, Web Interfaces, and API integrations.

- **Environment:** Built for low-connectivity environments. Must support offline-first data capture and asynchronous syncing.

Layer 2: Preprocessing Layer

- **Function:** Data cleaning, normalization, and feature extraction.
- **Data Parsing:** Extracts location data (coordinates), timestamps, and semantic attributes from incoming unstructured or semi-structured reports.

Layer 3: Deduplication & Similarity Engine (Core Logic)

- **Function:** Performs probabilistic linkage and graph-based clustering.
- **Continuous Evaluation:** Reports are continuously evaluated and merged into evolving "Event Nodes" based on similarity estimates and confidence thresholds.

Layer 4: Analysis & Visualization Layer

- **Function:** Trend analysis, recurrence detection, and operational simulation.
- **Output:** A dynamic, uncertainty-aware **"Living Map"** that supports situational awareness for field operators.

---

3. Core Algorithms: The Deduplication Framework

Developers must implement the following mathematical models to drive the Similarity Engine.

A. Data Representation

Each incoming report (

$$R$$

) is modeled as a structured tuple:
r_i = (x_i, y_i, t_i, a_i, s_i)

- **(x_i, y_i):** Spatial coordinates
- **(t_i):** Timestamp
- **(a_i):** Event attributes and severity indicators (Semantic)
- **(s_i):** Source metadata (Reliability)

B. Multi-Signal Similarity Model

The objective is to estimate

$$P(r_i \sim r_j)$$

— the probability that two reports correspond to the same underlying incident. The model combines four dimensions into a unified scoring function, transformed using a logistic function (

$$\sigma$$

):

P(r_i \sim r_j) = σ(α*d_{ij} + β*Δt_{ij} + γ*m_{ij} + δ*q_{ij})

- d_{ij}

(Spatial Distance): Geographic distance between reports (accounts for GPS noise).

- Δt_{ij}

(Temporal Difference): Accounts for reporting delays and event persistence.

- m_{ij}

(Semantic Similarity): Based on event type, severity, and descriptive features.

- q_{ij}

(Source Reliability): Weights reports based on reporter credibility, historical consistency, or cross-validation.

- **α, β, γ, δ:** Calibrated weights for each feature.

C. Graph Construction & Clustering

- **Nodes:** Each report is a node.
- **Edges:** Established between pairs of reports where the estimated probability exceeds a defined threshold: P(r_i \sim r_j) > θ.
- **Event Formation:** Event clusters are identified as *connected components* within this graph. This avoids rigid assumptions about cluster shapes.

---

4. Design Instructions for Developer: UI & Triage Logic

The frontend UI must revolve around the **"Living Map"**. Unlike traditional maps that show static pins, Solyra's map visualizes *Event Nodes* characterized by Confidence Scores, Source Diversity, Temporal Evolution, and Uncertainty Bounds.

The Triage Engine (State Transitions)

Event clusters are categorized into three reversible, evidence-driven states:

- 🔴 **RED (Actionable):** High-confidence incidents supported by strong multi-source evidence. Triggers operational alerts.
- 🟡 **YELLOW (Uncertain):** Partially validated incidents requiring further verification or partial evidence.
- 🟢 **GREEN (Resolved):** Resolved incidents retained for historical continuity. Can be reactivated (moved back to Yellow) if new data arrives.

UI/UX Mandate: State transitions must be reversible. Confidence scores and uncertainty representations must be exposed explicitly to users. Do not conceal uncertainty behind definitive markers.

---

5. Scalability & Performance Strategy

High-density refugee camps generate massive amounts of overlapping data. Comparing every report to every other report creates an

$$O(n^2)$$

computational bottleneck.

- **Developer Requirement:** Implement **Approximate Nearest-Neighbor (ANN)** techniques.
- **Goal:** Restrict pairwise comparisons to spatially relevant candidates only.
- **Target Complexity:** Reduce computational complexity from

$$O(n^2)$$

to approximately

$$O(n\log n)$$

, making the framework feasible for large-scale, real-time deployments.

---

6. Global Standard: Ethical & Governance Framework

Technology in a crisis must be responsible. Solyra must be engineered with the following strict compliance and ethical standards:

- **Privacy-by-Design:** The core functionality *does not require* Personally Identifiable Information (PII). Data collection must be minimized to operationally relevant fields only.

- **Accountable Decision-Making:** The system is a *decision-support tool*, not an authoritative autonomous system. Human operators must retain final responsibility for interpretation and intervention prioritization.
- **Epistemic Risk Management:** Aggressive merging of reports can obscure distinct incidents. The system must balance automation with transparency.
- **Compliance Alignment:** The architecture must align conceptually with:
    - UN OCHA Data Responsibility Guidelines
    - IASC Operational Guidance
    - GDPR principles of proportionality and minimization

---

7. Recommended Tech Stack (Developer Addendum)

(Note: While the white paper defines the logic, the following is a standard modern stack recommendation to achieve Solyra's goals)

- **Frontend / Mobile (Ingestion & Living Map):** React Native or Flutter (for cross-platform, offline-first mobile capabilities), Mapbox GL JS or Leaflet (for dynamic, uncertainty-aware map rendering).
- **Backend & API:** Node.js or Python (FastAPI/Django) to handle asynchronous data ingestion and API integrations.
- **Database & Spatial Querying:** PostgreSQL with **PostGIS** (crucial for spatial distance calculations and Approximate Nearest-Neighbor indexing).
- **Graph & Clustering Engine:** Python (NetworkX or similar graph libraries) for connected component analysis, or a dedicated graph database like Neo4j if scaling globally.
- **Machine Learning / Math:** Python (Scikit-learn/SciPy) for the logistic probability model and temporal decay modeling.