Proposal for CrisisEcho: LLM-Augmented Early Detection of Local Emergencies from Social Media

Samuel Enam Zih
CS, Virginia Tech samuelez@vt.edu

***Abstract*—CrisisEcho is a consumer-facing platform that applies Retrieval-Augmented Generation (RAG) and multi-step LLM agents to detect, classify, and summarize hyperlocal emergencies from social media in near real-time. Official emergency channels routinely lag events by minutes or hours; CrisisEcho closes this gap by ingesting candidate social and official feeds, storing data across three MongoDB Atlas databases (document, geospatial, vector) plus Redis and AWS S3, and running a three-step LangChain agent chain that clusters posts, scores severity, and generates plain-language alerts. Alerts are delivered via a Next.js frontend with a Leaflet.js interactive map and WebSocket notifications, backed by a Go Fiber API. Exact data sources, backend service count, and non-core libraries are not fixed and will be finalized based on API accessibility and deployment resources available during development.**

***Index Terms*—emergency detection, social media mining, RAG, LLM agents, MongoDB Atlas Vector Search, real-time systems, crisis informatics**

I. Introduction

Official channels — 911 dispatch, government sensors, police scanners — frequently lag fast-moving crises by minutes or hours. Ordinary people post hyperlocal signals on social media well before any official report: "Water rising fast on my street!" or "Heard gunshots near the mall." Existing tools are keyword-based and fail on indirect language, sarcasm, and regional idioms. CrisisEcho replaces rigid keyword matching with semantic RAG-powered reasoning, demonstrating four LLM-driven paradigm shifts: text to semantics, retrieval to reasoning, vertical to multi-domain, and closed-world to open-world generalization.

II. Related Work

Imran et al. [1] and Olteanu et al. [2] established Twitter as a crisis sensor and built foundational lexicons, while Middleton et al. [4] showed geotagged posts correlate with ground-truth disaster events. Lewis et al. [5] introduced RAG to ground generative models in retrieved evidence, reducing hallucination. Yao et al. [6] demonstrated chained LLM reasoning steps outperform single-prompt approaches, and Karpukhin et al. [7] showed dense vector retrieval outperforms BM25 — motivating our Atlas Vector Search hybrid retrieval design.

III. Data Sources

Note: The sources below are candidates. The final integrated set will be determined during development based on API rate limits, access restrictions, and available deployment resources.

Social (candidates): Twitter/X, Reddit, Bluesky, Mastodon, Nextdoor, Telegram, Facebook Groups. *Official (candidates):* NWS, USGS Earthquake, FEMA, PulsePoint, Waze Connected Citizens. *News/Crowdsourced (candidates):* GDELT, Patch.com RSS, NewsAPI, Ushahidi, Citizen App. *Training/Evaluation:* HumAID [3] (77k tweets), CrisisNLP [1], CrisisLex [2], Kaggle Disaster Tweets, CrisisMMD, FireFlood Dataset.

IV. System Architecture

CrisisEcho runs as a five-layer pipeline (Fig. 1). **Layer 1** platform-specific Python workers publish raw JSON to Aiven for Apache Kafka across four topics (`social_raw`, `official_alerts`, `news_feed`, `citizen_reports`). **Layer 2** a Python preprocessor applies spaCy cleaning, a cascading geocoder (Carmen → spaCy NER → location DB priors), MinHash LSH deduplication, a DistilBERT relevance filter (removes ≈70% of posts), Voyage AI text embedding (1024-dim), CLIP image embedding (512-dim), and S3 image upload. **Layer 3** stores data across five backends (Section V). **Layer 4** hybrid retrieval feeds a three-step LangChain LLM agent. **Layer 5** a Go Fiber API and Next.js frontend deliver alerts.

V. Database Architecture

MongoDB Atlas M0 (free tier) serves as the unified data platform because a single Atlas cluster natively supports document storage, 2dsphere geospatial indexing, and Atlas Vector Search — no separate vector or geospatial service required. The free tier is generous for a course project, and Atlas Vector Search supports multi-modal vector dimensions per collection.

DB 1 — Main (documents + geospatial). Collections: `posts` (unified), per-source raw collections (`twitter_posts`, `reddit_posts`, etc.), `clusters`, `crises`, `alerts`, `subscriptions`, `official_alerts`. All `location` fields carry a `2dsphere` index. Key `posts` fields: `post_id`, `source`, `raw_text`, `cleaned_text`, `timestamp`, `location` (GeoJSON Point), `location_confidence`, `text_embedding_id`, `image_urls` (S3 links), `cluster_id`, `crisis_type`, `severity_score`, `metadata.username_hash` (SHA-256).

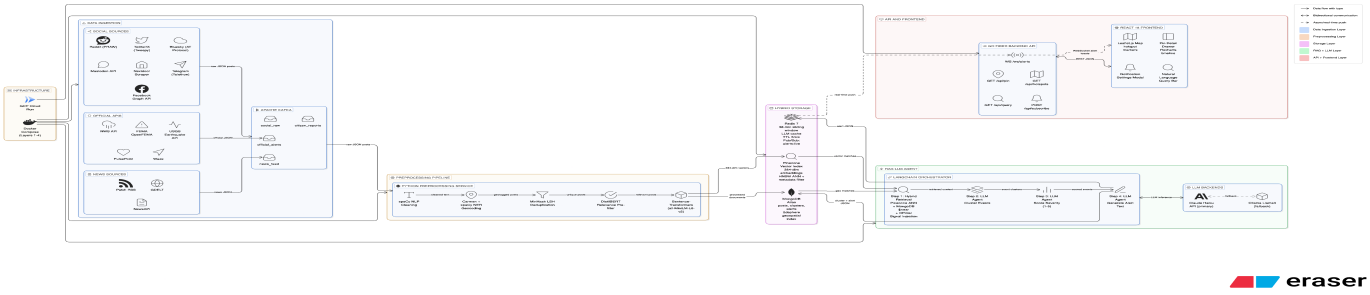

Fig. 1: CrisisEcho System Architecture

DB 2 — Location (geocoding support). Collections: `geo_priors`, `place_index`, `location_cache` (TTL-indexed, keyed by SHA-256 text hash). Enables the preprocessing cascade to avoid re-geocoding identical text.

DB 3 — Vector (Atlas Vector Search). Collections: `text_embeddings` (1024-dim cosine) and `image_embeddings` (512-dim cosine). Each document carries `post_id`, `timestamp`, `lat`, `lng`, `crisis_type`, `is_relevant` as filterable metadata inside `$vectorSearch`.

DB 4 — AWS S3 (media). Raw images are uploaded to S3 (`bucket/<source>/<post_id>/`); only S3 URLs are stored in MongoDB, keeping documents lean.

DB 5 — Redis 7 (cache + Pub/Sub). 30-minute sliding post window, 5-minute LLM summary TTL, and `alerts:live` Pub/Sub channel relaying alerts to the Go WebSocket server.

VI. RAG Pipeline and Evaluation

A. Hybrid Retrieval

Each 60-second trigger issues three concurrent queries: (1) Atlas Vector Search on `text_embeddings` (top 50 by cosine similarity, filtered by time window and bounding box); (2) MongoDB `$near` on `posts` (geographic proximity regardless of semantics); (3) `official_alerts` lookup injected as high-authority context. Unresolved posts are enriched via location DB geo priors. Results are merged, deduplicated, and ranked: $0.5\times\text{vec} + 0.3\times\text{recency} + 0.2\times\text{authority}$.

B. Three-Step LLM Agent

Claude claude-haiku-4-5 (primary) or Ollama Llama3 (fallback) executes: **Step 1** cluster posts into distinct events (JSON output); **Step 2** score severity 1–5 with rationale; **Step 3** generate a 2–3 sentence public alert with no PII. Publish threshold: severity ≥ 3, ≥ 3 independent contributors from ≥ 2 platforms, geocoding confidence above minimum.

C. Evaluation

Compared against a CrisisLex keyword baseline on the same corpus. Targets: Precision@K > 0.75, Recall > 0.70, end-to-end latency < 90 s, FPR < 0.20, geocoding accuracy > 0.80, F1 improvement > 15 pp. Ground truth: HumAID/CrisisNLP test splits (offline) + PulsePoint/NWS active alerts (online).

VII. Technology Stack

Table I lists the confirmed core stack. Non-core libraries and auxiliary tooling are subject to change during development. Backend service count may be adjusted based on implementation decisions.

TABLE I: CrisisEcho Core Technology Stack

Component	Technology	Rationale
Backend	Go (Fiber)	High throughput; goroutine WebSocket delivery
AI Pipeline	Python + LangChain	LLM/NLP ecosystem; LCEL agent chains
Frontend	Next.js + Leaflet.js	SSR support; geospatial map
Main DB	MongoDB Atlas M0	2dsphere geospatial; free cloud tier
Location DB	MongoDB Atlas M0	Geocoding cache + geo priors
Vector DB	Atlas Vector Search	Multi-modal ANN; no extra service
Media	AWS S3	Image storage; URLs only in MongoDB
Cache	Redis 7	Post window; LLM TTL; WS relay
Queue	Aiven Kafka	Managed Kafka; decouples ingestion
LLM	Claude Haiku / Llama3	Cost-efficient; free offline fallback
Deploy	GCP Cloud Run	Serverless; free-tier prototype

References

[1] M. Imran, P. Mitra, C. Castillo, "Twitter as a lifeline," *LREC*, 2016.
[2] A. Olteanu et al., "CrisisLex," *ICWSM*, 2014.
[3] F. Alam et al., "HumAID," *ICWSM*, 2021.
[4] S. Middleton et al., "Real-time crisis mapping," *IEEE Intell. Syst.*, vol. 29, no. 2, 2014.
[5] P. Lewis et al., "RAG for knowledge-intensive NLP," *NeurIPS*, 2020.
[6] S. Yao et al., "ReAct," *ICLR*, 2023.
[7] V. Karpukhin et al., "Dense passage retrieval," *EMNLP*, 2020.

Appendix

All architectural and design decisions were made by me.

TABLE II: Responsibilities

Member	Responsibilities
S. Zih	Backend: AI/DB/API/Docker, Go Fiber, MongoDB, Vector Search, LangChain agent, Redis, WebSocket, NLP pipeline, evaluation; Frontend: map, UI/UX, dashboard, notifications, query bar; Report: all sections

TABLE III: Project Schedule (13 Weeks)

Weeks	Milestone
1–2	Literature review; provision Atlas, Aiven Kafka, Redis; define schemas
3–4	Ingestion workers; spaCy + geocoding; DistilBERT fine-tune; embeddings to Atlas Vector Search
5–6	Hybrid retrieval; LangChain 3-step agent; Haiku + Llama3 fallback
7–8	Cluster persistence; Redis Pub/Sub; Go Fiber REST + WebSocket API
9–10	Next.js frontend; evaluation suite; keyword baseline comparison
11–12	Stretch features; system testing; latency and threshold tuning
13	Prompt refinement; documentation; demo; final report and submission