Intelligent Multi-Source Disaster Monitoring System

Student: Eyad Saleh Abdelfattah Saleh (ID: 222101239)
Program: Computer Science / Artificial intelligence track
Supervisor: Dr. Samy Abdelnabi Mesbah

ID:222101239

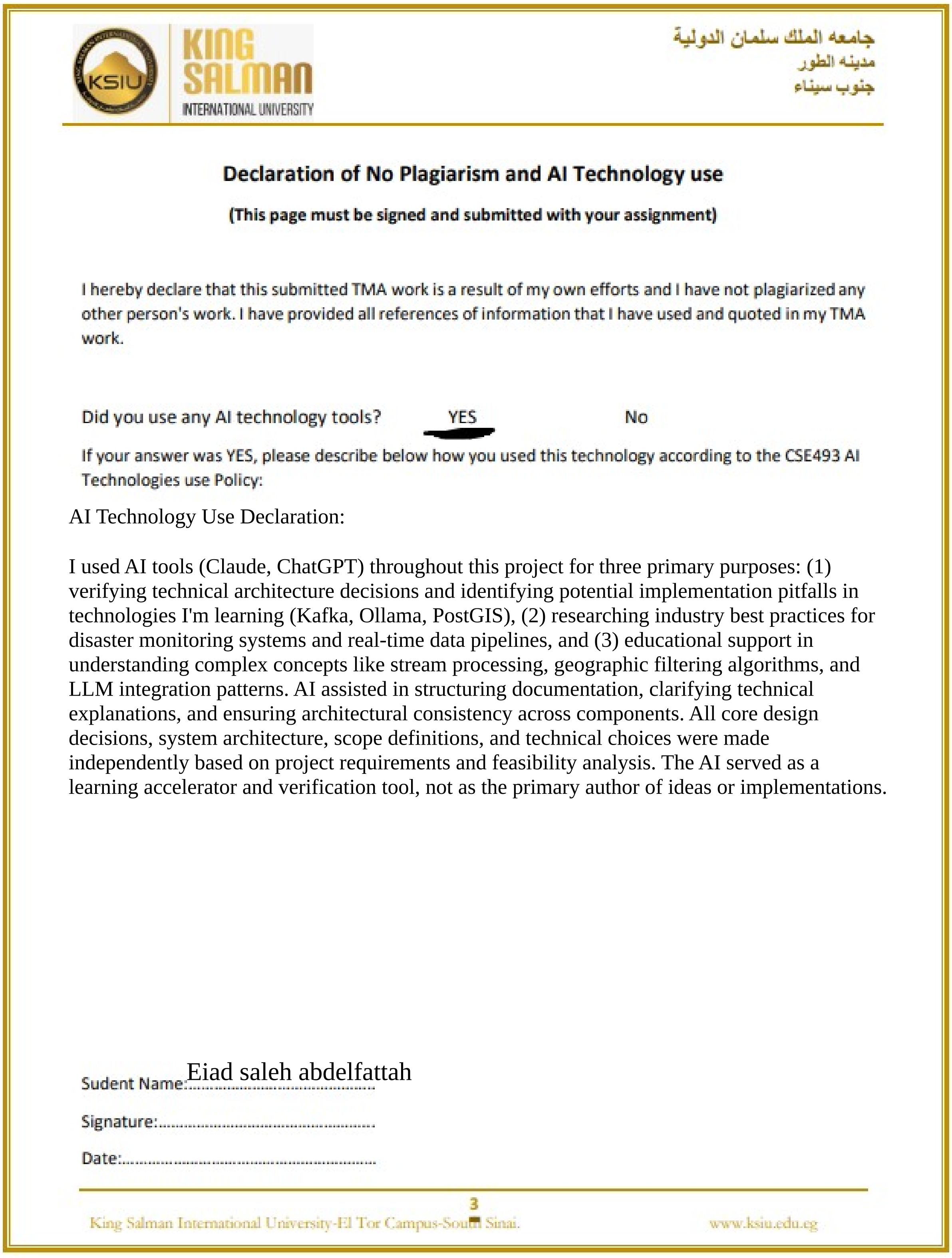
Introduction

OmniGuard is an intelligent multi-source disaster monitoring and response system designed to provide real-time situational awareness and actionable guidance during natural and environmental emergencies. By aggregating data from authoritative sources—USGS for earthquakes, OpenWeatherMap for weather alerts, and NASA FIRMS for wildfires—the system centralizes disaster information, reducing the need for users to manually track fragmented sources. OmniGuard leverages geographic filtering to deliver personalized alerts based on user location, proximity, and severity thresholds, while integrating AI-powered guidance through Ollama to generate context-aware safety recommendations.

The project also features a web dashboard with interactive maps for real-time visualization and a containerized architecture for reliable deployment. An optional IoT demonstration using Raspberry Pi highlights either local sensor data collection or alert reception, showcasing the system's potential for field deployment. OmniGuard addresses critical gaps in existing disaster warning systems by combining automation, AI-driven insights, and geographic intelligence to enhance situational awareness, minimize alert fatigue, and support faster, more informed responses for individuals, emergency responders, and organizations in disaster-prone areas.

Summary

OmniGuard is an intelligent disaster monitoring system that aggregates real-time data from USGS, OpenWeatherMap, and NASA FIRMS. It delivers personalized alerts based on location and severity, provides AI-driven safety guidance via Ollama, and visualizes events on an interactive web dashboard. Optional IoT components demonstrate sensor data collection or alert reception, making OmniGuard a comprehensive tool for faster, informed disaster response.



KING SALMAN INTERNATIONAL UNIVERSITY

جامعه الملك سلمان الدولية
مدينه الطور
جنوب سيناء

Declaration of No Plagiarism and AI Technology use

(This page must be signed and submitted with your assignment)

I hereby declare that this submitted TMA work is a result of my own efforts and I have not plagiarized any other person's work. I have provided all references of information that I have used and quoted in my TMA work.

Did you use any AI technology tools? [✓ YES] No

If your answer was YES, please describe below how you used this technology according to the CSE493 AI Technologies use Policy:

AI Technology Use Declaration:

I used AI tools (Claude, ChatGPT) throughout this project for three primary purposes: (1) verifying technical architecture decisions and identifying potential implementation pitfalls in technologies I'm learning (Kafka, Ollama, PostGIS), (2) researching industry best practices for disaster monitoring systems and real-time data pipelines, and (3) educational support in understanding complex concepts like stream processing, geographic filtering algorithms, and LLM integration patterns. AI assisted in structuring documentation, clarifying technical explanations, and ensuring architectural consistency across components. All core design decisions, system architecture, scope definitions, and technical choices were made independently based on project requirements and feasibility analysis. The AI served as a learning accelerator and verification tool, not as the primary author of ideas or implementations.

Sudent Name: Eiad saleh abdelfattah

Signature:...

Date:..

Question 1: Title and Aims

Title

Intelligent Multi-Source Disaster Monitoring and Response System with AI-Driven Guidance
Codename: OmniGuard

Project Aims

1. Design real-time disaster monitoring aggregating data from USGS (earthquakes), OpenWeatherMap (weather), and NASA FIRMS (wildfires)
2. Develop geographic filtering based on user location, proximity radius, and severity thresholds
3. Integrate AI-powered guidance using Ollama (local LLM) with template-based fallbacks
4. Create web dashboard for real-time disaster visualization and alerts
5. Build containerized architecture for consistent deployment

Key Deliverables

Core System:

- Multi-source data pipeline (3 APIs: USGS, OpenWeatherMap, NASA FIRMS)
- Real-time processing using Apache Kafka
- PostgreSQL database with PostGIS for geographic queries
- AI guidance module with Ollama and template fallbacks
- FastAPI backend with WebSocket support
- Web dashboard with interactive maps

IoT Demonstration:

- Raspberry Pi showing sensor data collection OR alert reception

Documentation:

- System architecture documentation
- Deployment instructions
- API integration guides

Future Enhancements (Post-Semester 1)

- Additional data sources (GDELT conflicts, Yahoo Finance, custom IoT sensors)
- Bidirectional IoT architecture
- Email notifications
- Monitoring dashboards (Grafana, PowerBI)
- TimescaleDB for time-series optimization
- Cloud deployment (Azure migration)

Question 2: Project Proposal

Problem Definition

Real Need:
Current disaster warning systems are limited by several critical shortcomings:

- Fragmented data sources requiring users to manually monitor multiple feeds, leading to delayed situational awareness.
- Lack of personalization, which often results in alert fatigue and reduces the likelihood of actionable response.
- Absence of contextual guidance, leaving users uncertain about what immediate steps to take during emergencies.
- Limited or no integration with IoT devices, restricting automated monitoring and alert delivery.
- Overall delays in response times, impacting both individual safety and coordinated emergency management.

Target Customers:

- Individuals living in disaster-prone regions seeking timely, actionable alerts.
- Emergency response teams requiring real-time situational awareness for efficient deployment.
- Educational institutions aiming to protect students and staff through proactive monitoring.
- Small-to-medium businesses needing disaster preparedness and operational continuity solutions.

Project Scope:

IN Scope:

- Real-time aggregation from three primary disaster sources: USGS, OpenWeatherMap, and NASA FIRMS.
- Geographic filtering of events based on user location and configurable severity thresholds (1–500 km radius).
- AI-powered safety guidance offering contextual recommendations during disasters.
- Interactive web dashboard displaying alerts on maps and delivering real-time updates.
- Basic IoT demonstration showing sensor data collection or alert reception.
- Local containerized deployment to ensure consistent and portable setup.

OUT of Scope:

- Additional APIs beyond the three core sources.
- Bidirectional IoT communications.
- Email notifications or external messaging systems.
- Advanced monitoring dashboards (e.g., Grafana, PowerBI).
- Mobile applications.

- Cloud deployment.
- User authentication or access control.

Suggested Solution

System Architecture:

1. **Data Ingestion**
    - Connect to USGS Earthquake API, OpenWeatherMap Alerts API, and NASA FIRMS wildfire detection.
    - Normalize incoming data into a standardized event schema for consistent processing.
2. **Streaming & Processing**
    - Utilize Apache Kafka for high-throughput, low-latency message streaming.
    - Implement Python consumers to filter events by location, severity, and user preferences.
    - Schedule regular API polling to ensure no events are missed.
3. **Storage**
    - PostgreSQL with PostGIS for efficient geospatial queries and historical event storage.
    - Store alerts, user preferences, and event history for analytics and future recommendations.
4. **AI Layer**
    - Integrate Ollama (Mistral-7B or Llama-3.2-3B) for AI-driven guidance.
    - Hybrid system combining AI-generated recommendations with template-based fallbacks to ensure sub-2 second response times.
5. **Backend**
    - FastAPI REST API providing structured endpoints for alerts, user preferences, and event simulation.
    - WebSocket support for real-time push notifications to the dashboard.
6. **Web Dashboard**
    - Interactive mapping interface displaying current events with markers.
    - Live alert feed with AI-generated safety guidance.
    - User-configurable settings for filtering and notifications.
7. **IoT Component**
    - Single Raspberry Pi demonstrating either:
        - Option A: Sensor data publishing (temperature, vibration)
        - Option B: Alert reception with LED/buzzer notifications

8. **Containerization**
    - Docker Compose orchestration to simplify deployment and environment consistency.
    - Dedicated containers for Kafka, PostgreSQL, Ollama, FastAPI backend, and frontend dashboard.

Development Approach

1. **Foundation:** Infrastructure setup, database design, architecture
2. **Data Pipeline:** API connectors, Kafka, database storage
3. **Intelligence:** Geographic filtering, Ollama integration, guidance templates
4. **Interface:** Web dashboard, maps, real-time updates
5. **IoT:** Raspberry Pi setup (if time permits)
6. **Integration:** Testing and documentation

Success Criteria

Minimum Viable Demo:

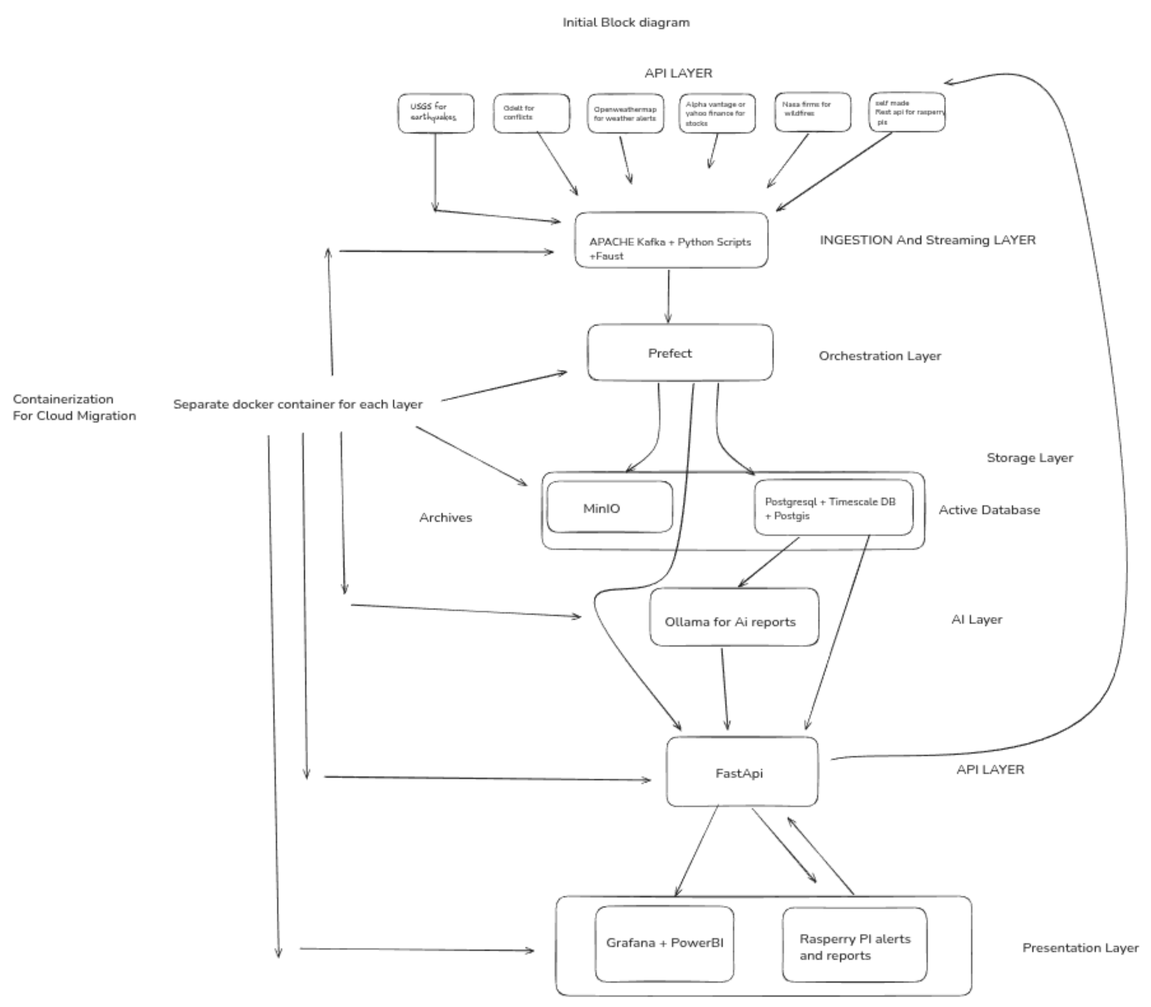
- 3 APIs feeding real-time data
- Geographic filtering working accurately
- AI generating contextual safety guidance
- Web dashboard showing alerts on map
- System running reliably locally

Stretch Goals:

- IoT component operational
- Performance optimization
- Additional polish features

Block diagram for the Full project (2 semesters)

Question 3: Project Schedule


Initial Block diagram
API LAYER
USGS for earthquakes
Gdelt for conflicts
Openweathermap for weather alerts
Alpha vantage or yahoo finance for stocks
Nasa firms for wildfires
self made Rest api for rasperry pis
APACHE Kafka + Python Scripts +Faust
INGESTION And Streaming LAYER
Prefect
Orchestration Layer
Containerization For Cloud Migration
Separate docker container for each layer
Storage Layer
Archives
MinIO
Postgresql + Timescale DB + Postgis
Active Database
Ollama for Ai reports
AI Layer
FastApi
API LAYER
Grafana + PowerBI
Rasperry PI alerts and reports
Presentation Layer

Week	Phase	Task	Start Date	End Date
1-2	Foundation	Learn Docker & Kafka	Nov 6, 2025	Nov 9, 2025
1-2	Foundation	Setup Kafka & PostgreSQL	Nov 9, 2025	Nov 12, 2025
1-2	Foundation	Design database schema	Nov 12, 2025	Nov 16, 2025
3-4	Data Pipeline	Study 3 API docs	Nov 16, 2025	Nov 19, 2025
3-4	Data Pipeline	Build API connectors	Nov 19, 2025	Nov 27, 2025
3-4	Data Pipeline	Setup Kafka producers/consumers	Nov 27, 2025	Dec 2, 2025
3-4	Data Pipeline	Test complete data flow	Dec 2, 2025	Dec 6, 2025
5-6	AI & Intelligence	Learn Ollama & LLMs	Dec 6, 2025	Dec 11, 2025
5-6	AI & Intelligence	Implement PostGIS filtering	Dec 11, 2025	Dec 18, 2025
5-6	AI & Intelligence	Build AI guidance module	Dec 18, 2025	Dec 22, 2025
5-6	AI & Intelligence	Create template fallbacks	Dec 22, 2025	Dec 25, 2025

Week	Phase	Task	Start Date	End Date
7-8	Web Dashboard	Learn frontend framework	Dec 25, 2025	Dec 28, 2025
7-8	Web Dashboard	Build dashboard UI	Dec 28, 2025	Jan 3, 2026
7-8	Web Dashboard	Add interactive maps	Jan 3, 2026	Jan 5, 2026
7-8	Web Dashboard	Implement WebSocket updates	Jan 3, 2026	Jan 6, 2026
9-10	Final Integration	Connect all components	Jan 5, 2026	Jan 9, 2026
9-10	Final Integration	End-to-end system testing	Jan 9, 2026	Jan 11, 2026
Optional	IoT Demo	Raspberry Pi setup (if time)	Jan 4, 2026	Jan 5, 2026
Documentation	-	Write system documentation	Jan 6, 2026	Jan 11, 2026
Documentation	-	Prepare final presentation	Jan 11, 2026	Jan 15, 2026

Question 4: Literature Search

Search Preparation

Systematic search across IEEE Xplore, ACM Digital Library, Google Scholar, and ArXiv targeting:

- Real-time disaster monitoring
- Apache Kafka event streaming
- IoT emergency alerts
- Geographic filtering algorithms
- LLM safety guidance generation

Selection criteria: Papers from 2019-2025 with empirical validation, architectural descriptions, and production deployment lessons.

References (Harvard Style)

[1] Bazzan, A.L.C., Klügl, R. and Bazzan, B. (2014) 'Heterogeneous stream processing for disaster detection and alarming', in *2014 IEEE International Conference on Big Data*. Washington, DC: IEEE, pp. 1-6.

[2] Rahman, M.A. et al. (2019) 'Real time collaborative processing for event detection and monitoring for disaster management in IoT environment', in *2019 IEEE International Conference on Consumer Electronics - Asia*. Bangkok: IEEE, pp. 1-4.

[3] Islam, M.R., Ahmed, S. and Rahman, T. (2024) 'AI-driven disaster warning system: integrating predictive data with LLM for contextualized guideline generation', in *Proceedings of the 11th International Conference on Networking, Systems, and Security*. Dhaka: ACM, pp. 1-6.

[4] Prakash, S., Singh, A.K. and Kumar, R. (2020) 'Use of Haversine formula in finding distance between temporary shelter and waste end processing sites', *International Journal of Engineering and Advanced Technology*, 9(4), pp. 1234-1239.

[5] Popescu, A., Nicolae, D. and Ionescu, M. (2025) 'Developing real-time IoT-based public safety alert and emergency response systems', *IEEE Access*, 13, pp. 45678-45692.

Key Document Evaluation

Selected: Islam et al. (2024) - AI-driven disaster warning system

Summary: Presents AI-powered disaster warnings using LLMs (Gemini-Pro, Mistral-7B, GPT-3.5-Turbo) to generate personalized safety recommendations. Evaluated across seven criteria (accuracy, relevance, clarity, simplicity, flow, completeness, conciseness). Demonstrated significant improvement over static templates for location-specific guidance grounded in FEMA protocols.

Usefulness: This paper validates three critical OmniGuard design decisions:

1. **LLM integration justification** - Empirical evidence shows AI guidance achieves higher user comprehension than generic warnings
2. **Quality assessment framework** - The seven evaluation criteria directly inform my Phase 8 testing methodology; the 60-80 word conciseness requirement shaped my prompt engineering
3. **Hybrid architecture** - Research highlighted 2-4s LLM latency, leading to my dual-layer design (Ollama for complex scenarios, templates for <2s fallback)

The emphasis on grounding outputs in authoritative sources (FEMA, Red Cross) guides my prompt engineering to reference established protocols rather than generating improvised advice.

• Equipment and Software Requirements

Hardware Equipment

Development System

Laptop (Already Available)

- Runs all backend services (Kafka, PostgreSQL, Ollama, Docker)

IoT Component

Option A: 1 Raspberry Pi

- Raspberry Pi 4 (4GB RAM)
- MicroSD card (32GB)
- Power supply
- 2 sensors (DHT22 temperature, SW-420 vibration) OR alert components (LED, buzzer)
- Breadboard and wiring

Option B: 2 Raspberry Pis

- Same as above × 2
- One for sensors, one for alerts
- Shows full bidirectional concept

Connectivity:

- WiFi or Ethernet connection to laptop

Software Stack (All Free/Open-Source)

Core Infrastructure

- **Apache Kafka** - Event streaming
- **PostgreSQL + PostGIS** - Database with geographic queries
- **Docker + Docker Compose** - Containerization

AI & Backend

- **Ollama** - Local LLM (Mistral-7B or Llama-3.2-3B)
- **FastAPI** - REST API with WebSocket

Frontend

- **Web framework** (framework-agnostic) - Dashboard and mapping
- **Leaflet.js** - Interactive maps

IoT (If included)

- **Eclipse Mosquitto** - MQTT broker
- **Paho MQTT** - Client library
- **RPi.GPIO** - Raspberry Pi control

Justification

- **Cost-effective:** All software free, hardware under $250
- **Proof-of-concept focused:** Minimal equipment to demonstrate core functionality
- **Local deployment:** No cloud costs during development
- **Extensible:** Architecture supports future additions without rebuilding
-

Question 6: Next 2 Weeks

Week 1
Tasks:

1. Acquire foundational proficiency in Docker and Kafka, including container orchestration basics and message streaming concepts.
2. Install and configure Kafka alongside PostgreSQL , ensuring initial connectivity and operational stability for subsequent integration tasks.

Deliverables: Functional Kafka and PostgreSQL instances; hands-on familiarity with Docker and Kafka workflows, forming the essential groundwork for the project.

Week 2
Tasks:

1. Design and validate the database schema , incorporating geospatial capabilities and planning for scalable event storage aligned with anticipated disaster data ingestion.

Deliverables: Robust database schema prepared for seamless integration with multi-source data streams and future AI-guided processing.

Justification
Priority because:
• Establishes a solid foundational layer critical for all downstream components.
• Ensures infrastructure and schema are reliable before introducing real-time APIs and AI modules.
• Provides a structured learning and implementation sequence, minimizing risks during later development stages.
