Team 5

May 9, 2025

DisasterLens

APPENDIX

About Us

Our Project

AI Model

Backend

Frontend

Meet the Team

Ashish Das

Team Lead & Scrum Master

Phong

AI Development

Jiwoo

Front-end developer

Meet the Team

Anish
Front-end Developer

Uday
Backend Developer

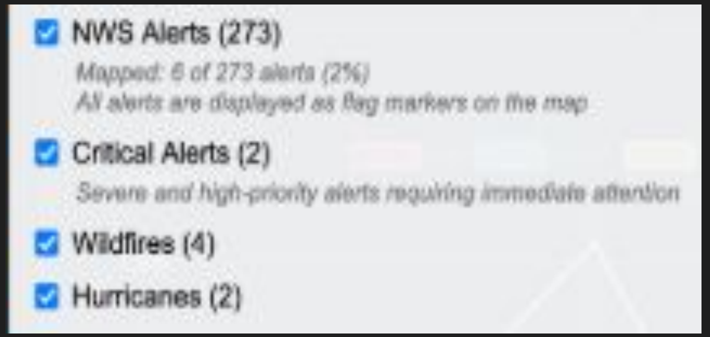
Antony
Backend Developer

Business Context

- **Business Problem:**

  - Disaster events evolve in real time, yet traditional news outlets deliver delayed, fragmented reports.

  - Emergency teams lack timely, consolidated information, impeding rapid decision-making and resource allocation.


☑ NWS Alerts (273)

Mapped: 6 of 273 alerts (2%)
All alerts are displayed as flag markers on the map

☑ Critical Alerts (2)

Severe and high-priority alerts requiring immediate attention

☑ Wildfires (4)

☑ Hurricanes (2)

- **Objective:**

  - Deploy a fully automated platform analyzing live Bluesky data.

  - Instantly deliver precise disaster alerts and visualizations to support swift, data-driven crisis response.

Project Setup & Features

- Real-time data ingestion
- Advanced NLP classification
- Geospatial & urgency extraction
- Interactive visualization

Project Key Risks

- Data quality & noise
- API limitations
- Scalability & performance
- Regulatory & privacy
- Model generalization

Team 5

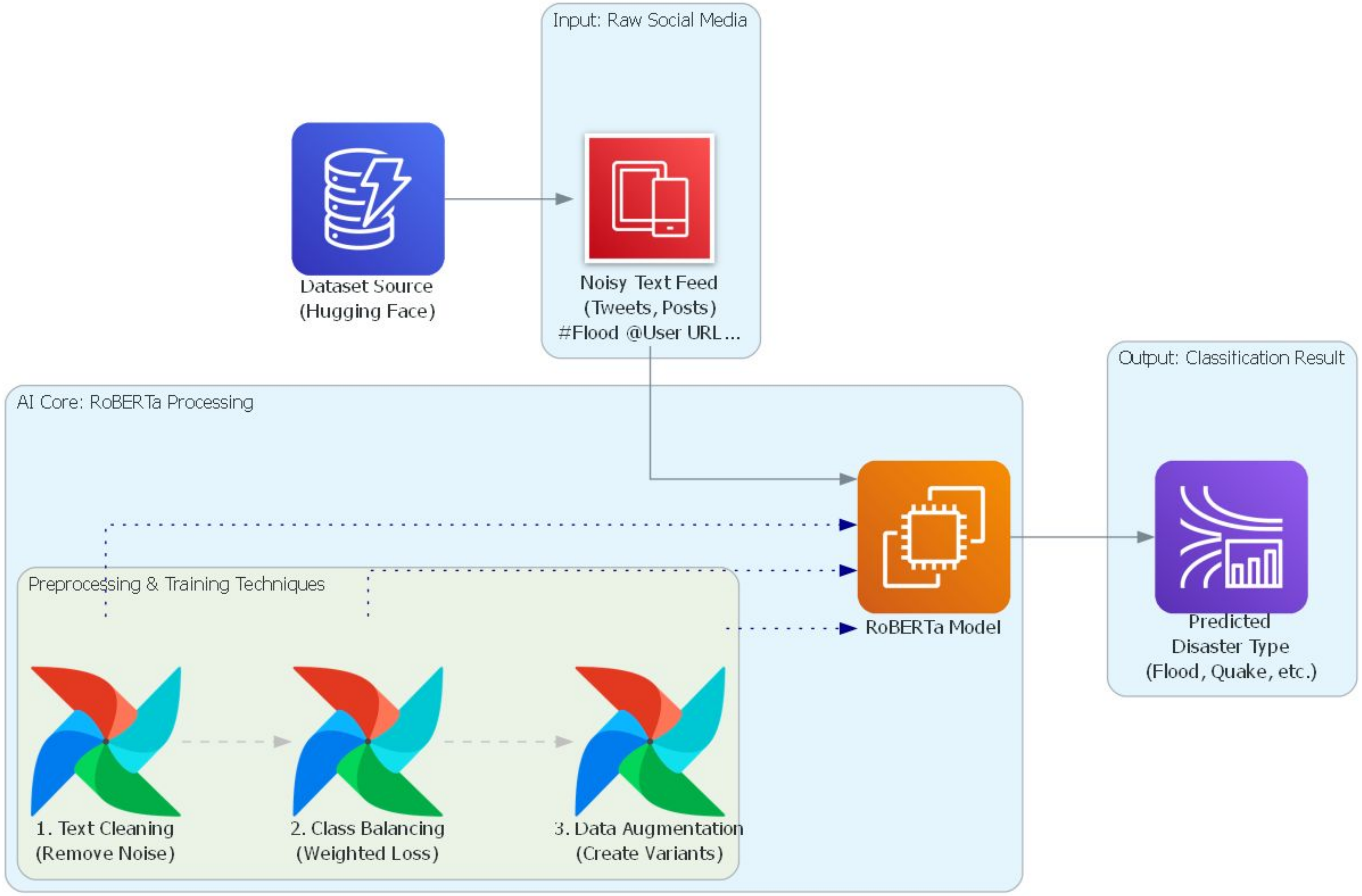
May 9, 2025

The Core AI

Team 5

AI Model Overview

This diagram shows the overall workflow: We take noisy social media text, process it using RoBERTa enhanced with key techniques, and output a classified disaster type.

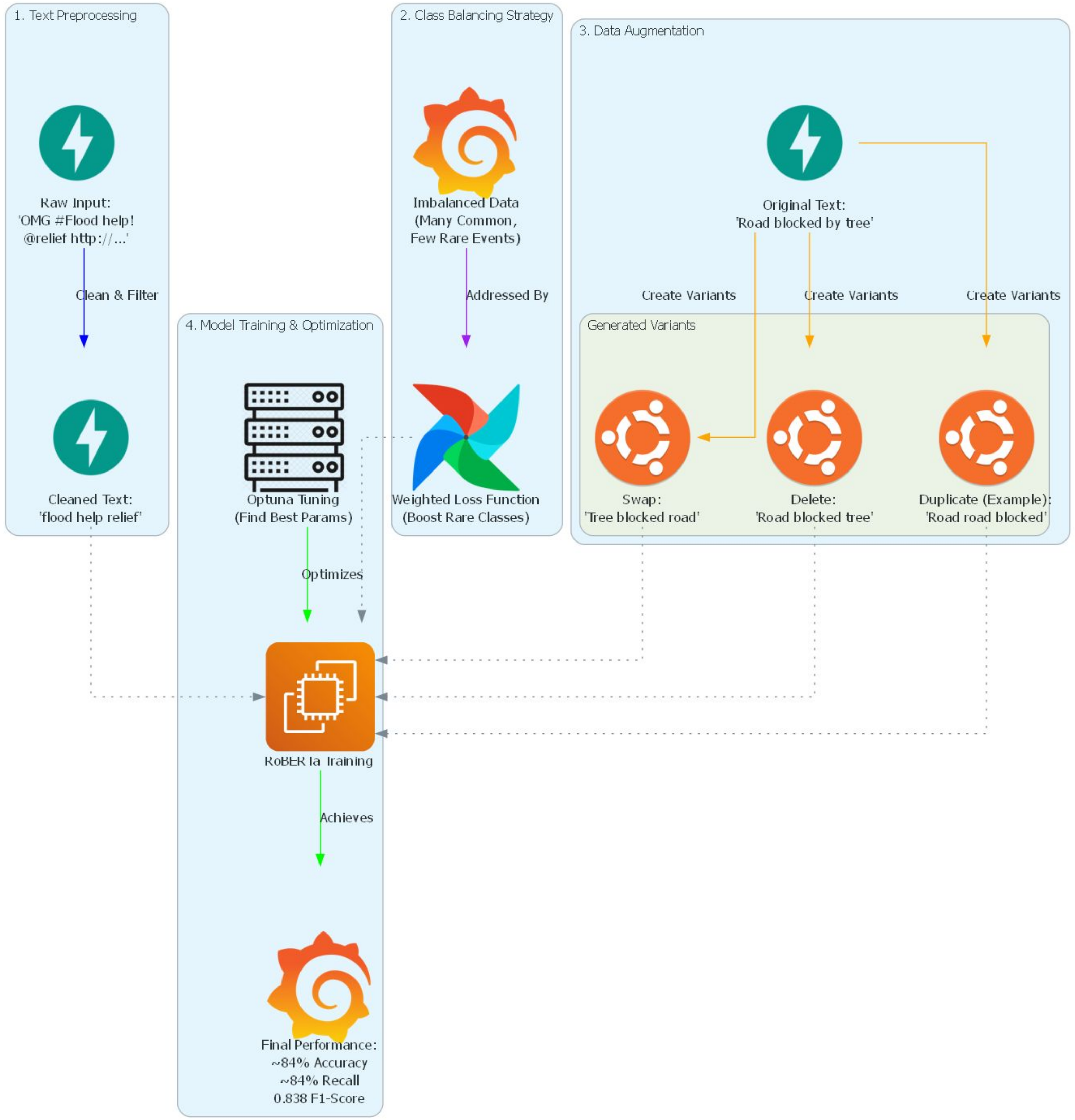


AI Model Overview - Phong Dau

Team 5

Structure & Algorithm

Here we detail the core techniques: Text preprocessing cleans the data, weighted loss addresses class imbalance, data augmentation increases robustness, and Optuna optimizes the final RoBERTa model training.



1. Text Preprocessing
Raw Input: 'OMG #Flood help! @relief http://...'
Clean & Filter
Cleaned Text: 'flood help relief'
2. Class Balancing Strategy
Imbalanced Data (Many Common, Few Rare Events)
Addressed By
Weighted Loss Function (Boost Rare Classes)
3. Data Augmentation
Original Text: 'Road blocked by tree'
Create Variants
Create Variants
Create Variants
Generated Variants
Swap: 'Tree blocked road'
Delete: 'Road blocked tree'
Duplicate (Example): 'Road road blocked'
4. Model Training & Optimization
Optuna Tuning (Find Best Params)
Optimizes
RoBERTa Training
Achieves
Final Performance: ~84% Accuracy ~84% Recall 0.838 F1-Score

Structure & Algorithm - Phong Dau

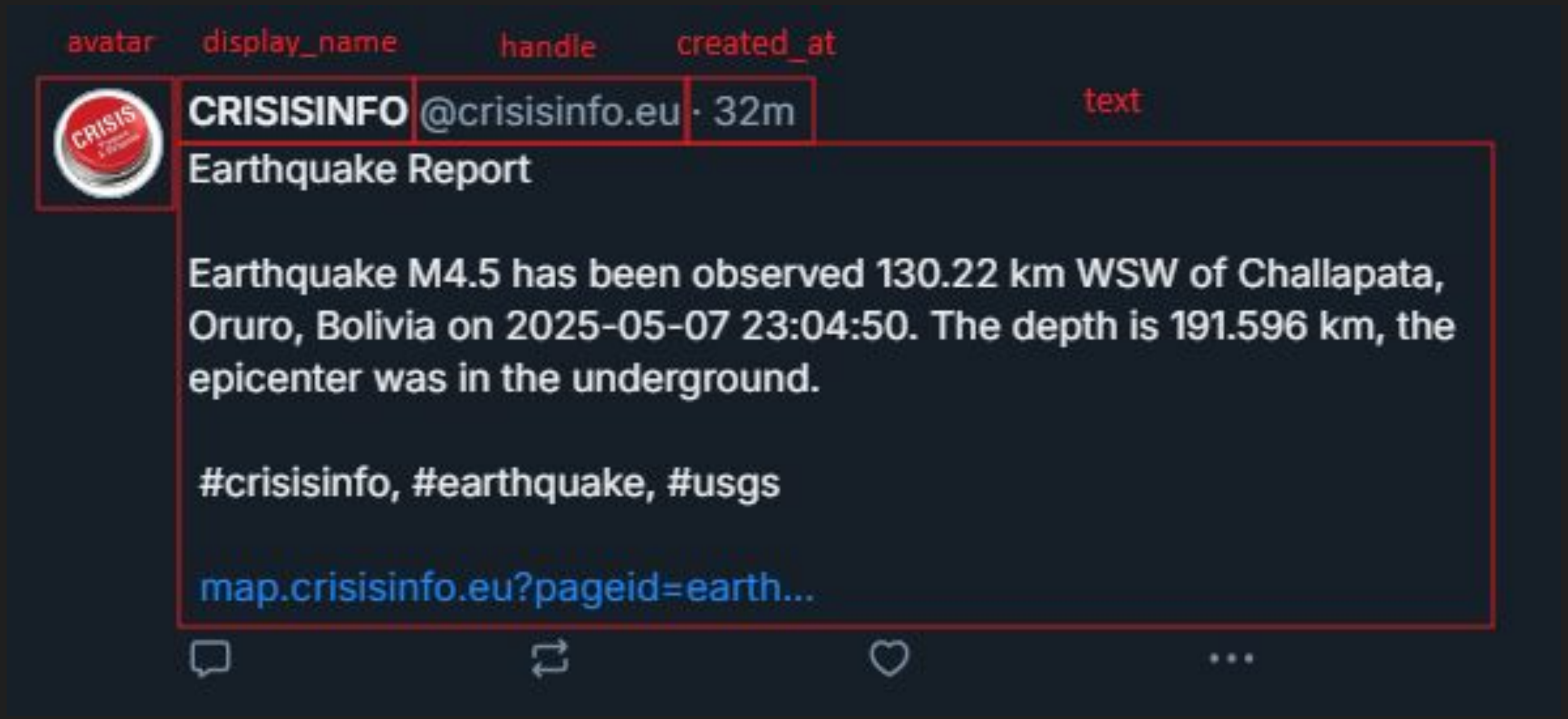
Team 5

May 9, 2025

Backend Architecture & Data

Backend Core

- **Bluesky API Integration**
  - Uses atproto python library to take advantage of API calls
  - Post retrieval with search_posts() call
  - Attributes retrieved with post.”attribute”() calls
- **Main Script Functions**
  - Text pre-processing for model
  - Uses model for disaster post classification
  - Stores posts and user data in database
  - Post retrieval done in batches



CRISISINFO @crisisinfo.eu · 32m

Earthquake Report

Earthquake M4.5 has been observed 130.22 km WSW of Challapata, Oruro, Bolivia on 2025-05-07 23:04:50. The depth is 191.596 km, the epicenter was in the underground.

#crisisinfo, #earthquake, #usgs

map.crisisinfo.eu?pageid=earth...

Database

- **Database Used: DynamoDB**
  - a fully managed NoSQL database provided by AWS, Chosen for seamless integration with other AWS services
  - Avoided setting up and managing separate database servers
  - Enabled direct backend API interaction without complex drivers
- **Schema overview**
  - DynamoDB database had two key tables:
    1. Users Table: stores user ID, handle, display name, avatar URL, created_at
    2. Posts Table: stores post ID, timestamp, user ID, disaster type, confidence score, original text

Database

- **Storage Workflow:**
  - Disaster posts retrieved from Bluesky API → preprocessed → passed to RoBERTa classification model
  - Once the predicted disaster type is assigned, the post is stored in DynamoDB (Posts Table)
  - Users are stored (if new) in the Users Table
- **Storage Workflow:**
  - Backend API exposes endpoints like /api/posts for frontend
  - API supports filtering by disaster type, date, language, confidence score
  - Once, the API retrieves filtered disaster data → served to frontend maps/charts

Team 5

May 9, 2025

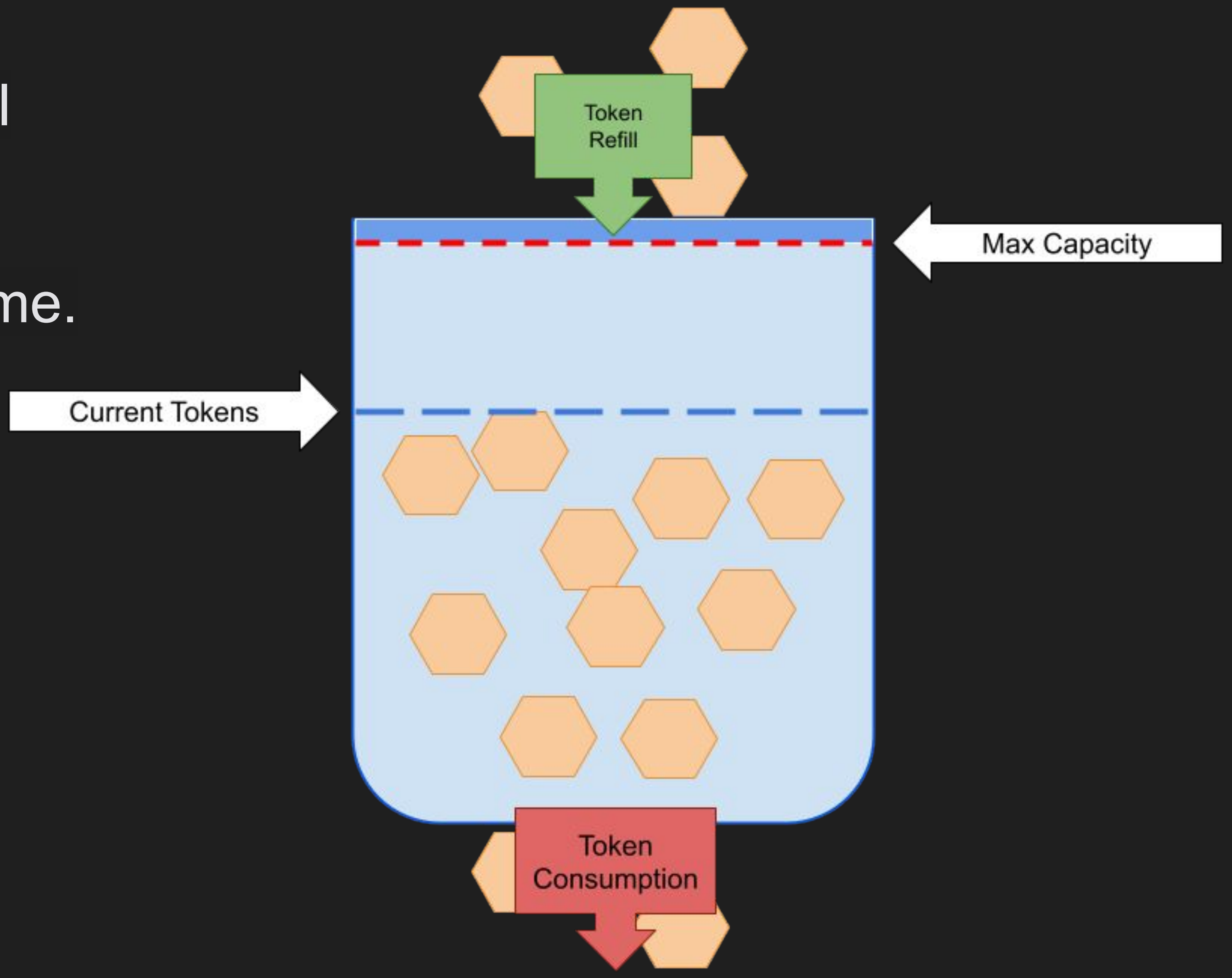
Backend Integrations & Reliability

Team 5

May 9, 2025

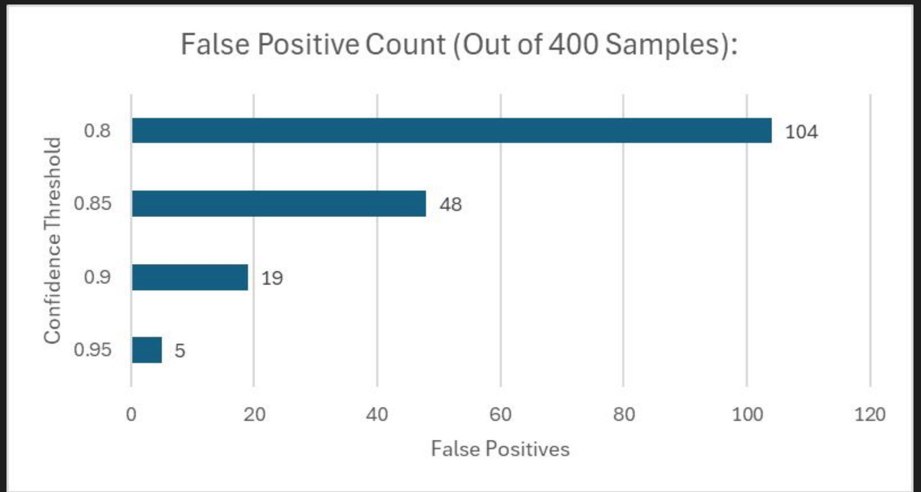
Rate-Limiting

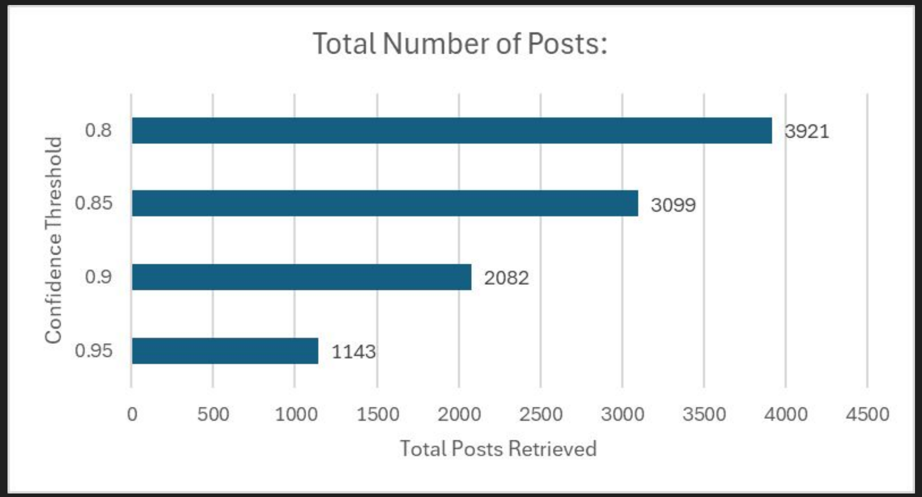
- **Bluesky Overall API Request Limits:**
  - 3000 requests per 5 minutes
- **Solution: Token Bucket Algorithm**
  - Tokens are allotted and consumed per API call
  - If tokens are exhausted, the system waits
  - Tokens are refilled proportionally to elapsed time.



Backend Testing

- **Feed vs. Keyword Searching**
  - Feeds search limited by curation
  - Keyword search retrieved 6.5x more posts than curated feeds.





- **Confidence Threshold Sampling**
  - Challenge: Limiting non-disaster posts
  - Solution:Sampled 400 posts in ranges from 0.80 to 0.95+
  - 0.95 threshold continues to yield enough posts

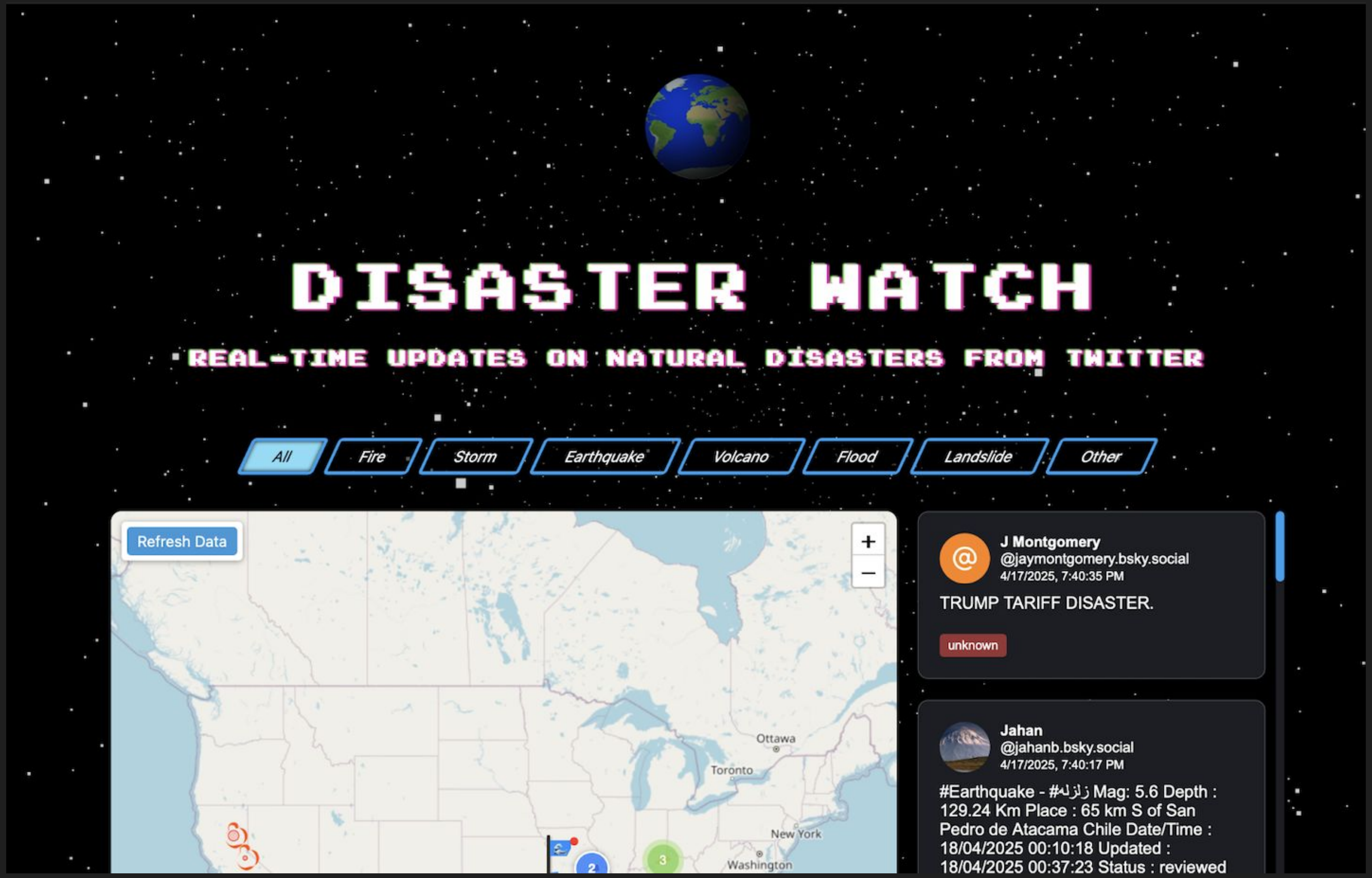
Team 5

May 9, 2025

Frontend Development

Frontend Foundation

- Space Theme: Tweet as our satellite orbiting earth



J Montgomery
@jaymontgomery.bsky.social
4/17/2025, 7:40:35 PM

TRUMP TARIFF DISASTER.

unknown

Jahan
@jahanb.bsky.social
4/17/2025, 7:40:17 PM

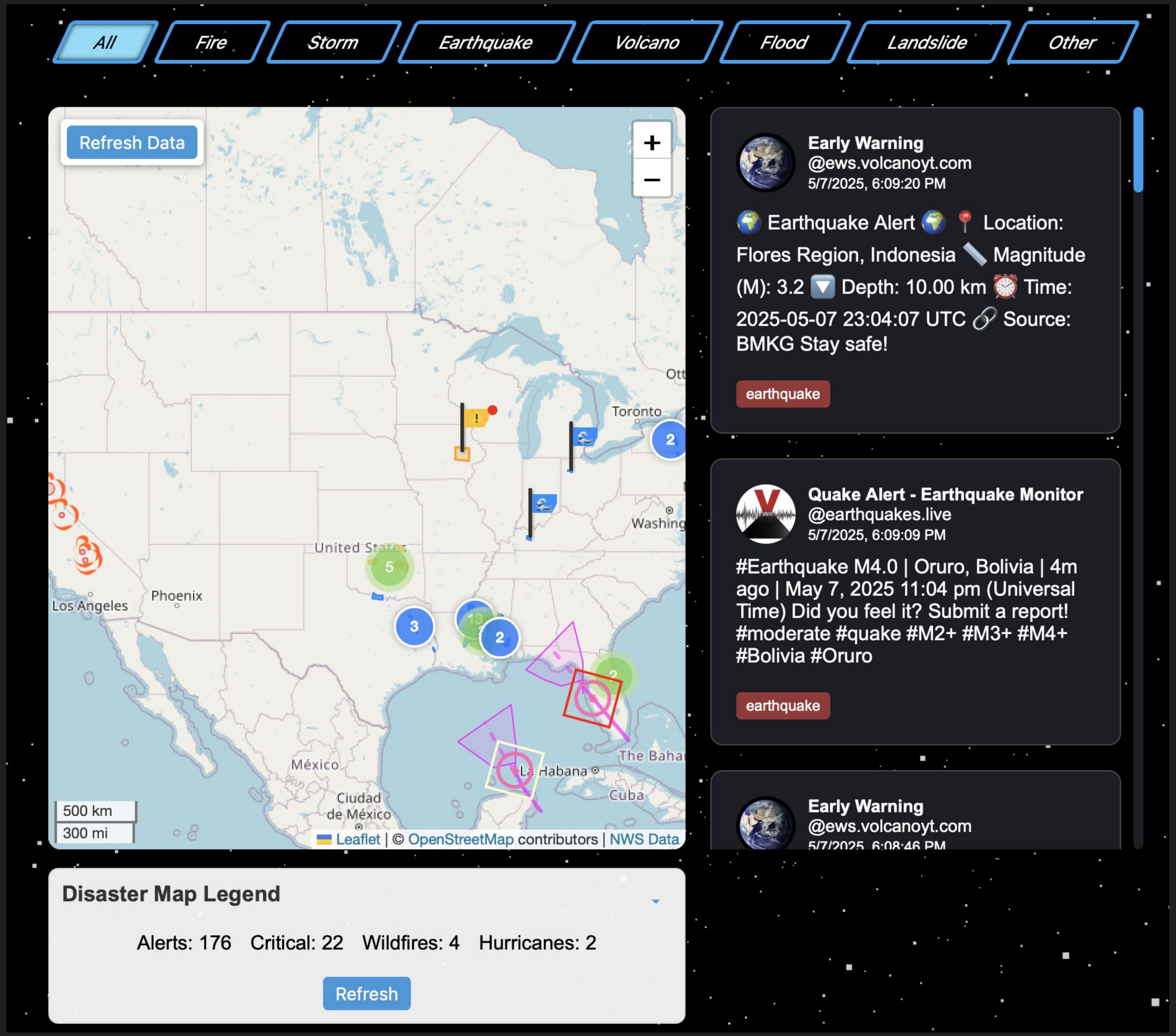
#Earthquake - #زلزله Mag: 5.6 Depth : 129.24 Km Place : 65 km S of San Pedro de Atacama Chile Date/Time : 18/04/2025 00:10:18 Updated : 18/04/2025 00:37:23 Status : reviewed

Team 5

May 9, 2025

Component development

- Category button
- Map & Tweet list



Early Warning
@ews.volcanoyt.com
5/7/2025, 6:09:20 PM

🌍 Earthquake Alert 🌍 📍 Location: Flores Region, Indonesia 📏 Magnitude (M): 3.2 🔽 Depth: 10.00 km ⏰ Time: 2025-05-07 23:04:07 UTC 🔗 Source: BMKG Stay safe!

earthquake

Quake Alert - Earthquake Monitor
@earthquakes.live
5/7/2025, 6:09:09 PM

#Earthquake M4.0 | Oruro, Bolivia | 4m ago | May 7, 2025 11:04 pm (Universal Time) Did you feel it? Submit a report! #moderate #quake #M2+ #M3+ #M4+ #Bolivia #Oruro

earthquake

Early Warning
@ews.volcanoyt.com
5/7/2025, 6:08:46 PM

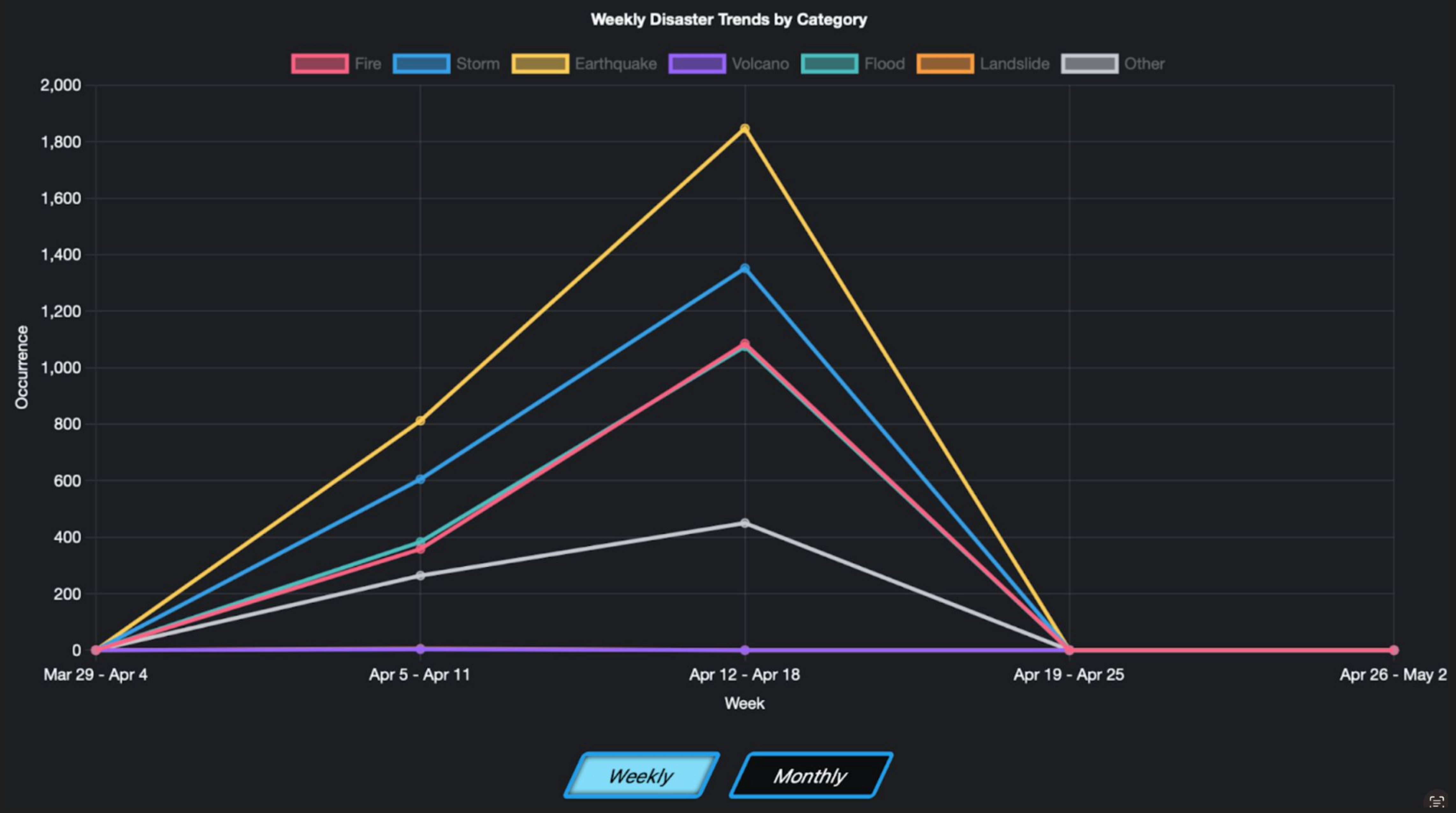
Disaster Map Legend

Alerts: 176 Critical: 22 Wildfires: 4 Hurricanes: 2

Refresh

Component development

- Time chart

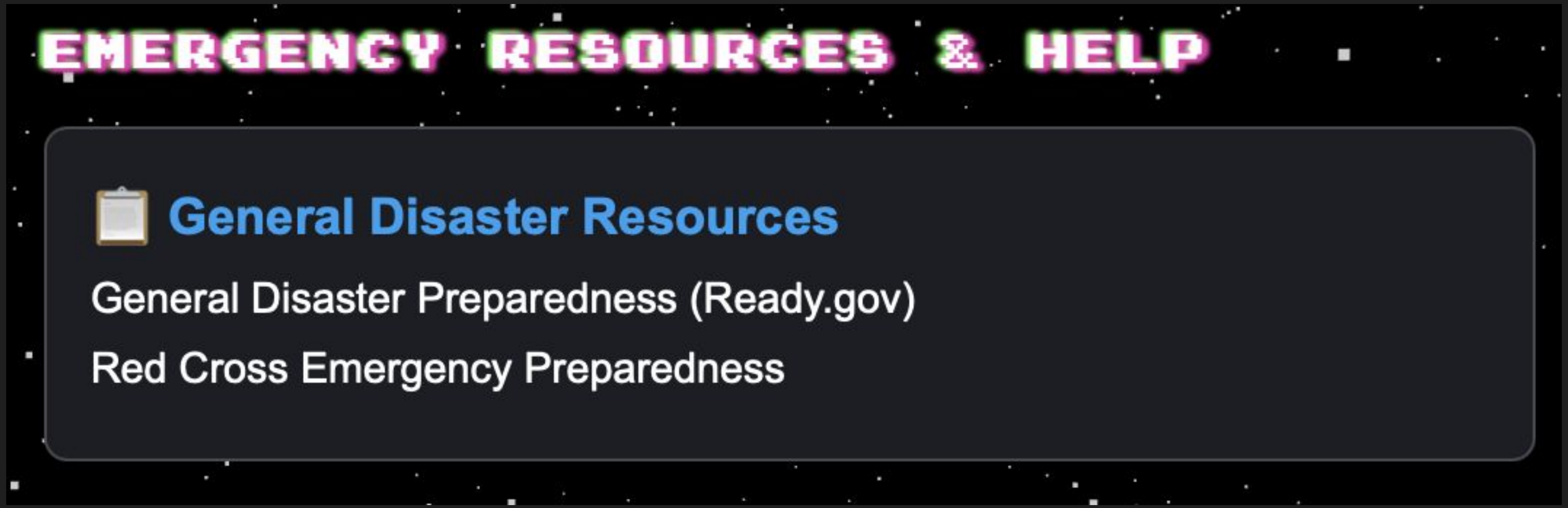


Team 5

May 9, 2025

Component development

- Help Section



EMERGENCY RESOURCES & HELP

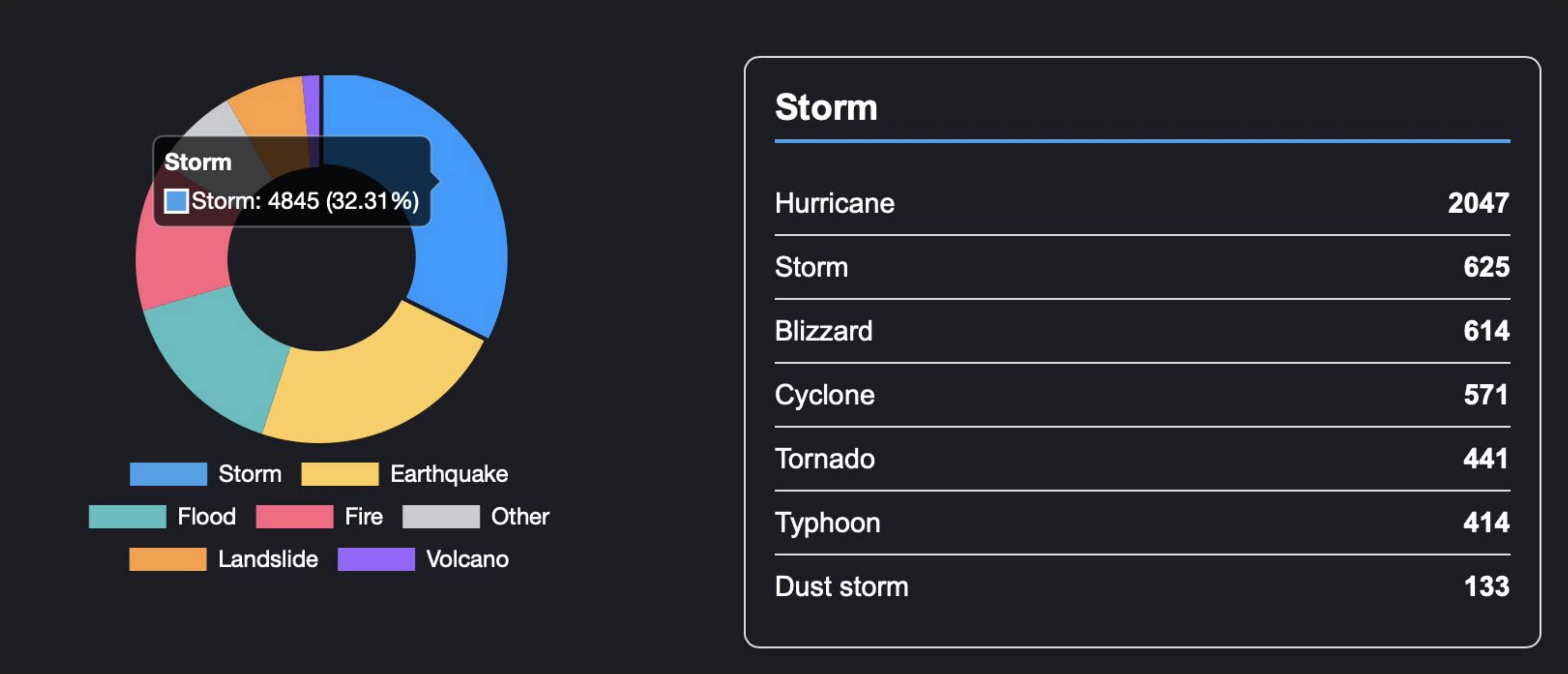
📋 General Disaster Resources

General Disaster Preparedness (Ready.gov)

Red Cross Emergency Preparedness

Component development

- Donut chart



Storm

Hurricane	**2047**
Storm	**625**
Blizzard	**614**
Cyclone	**571**
Tornado	**441**
Typhoon	**414**
Dust storm	**133**

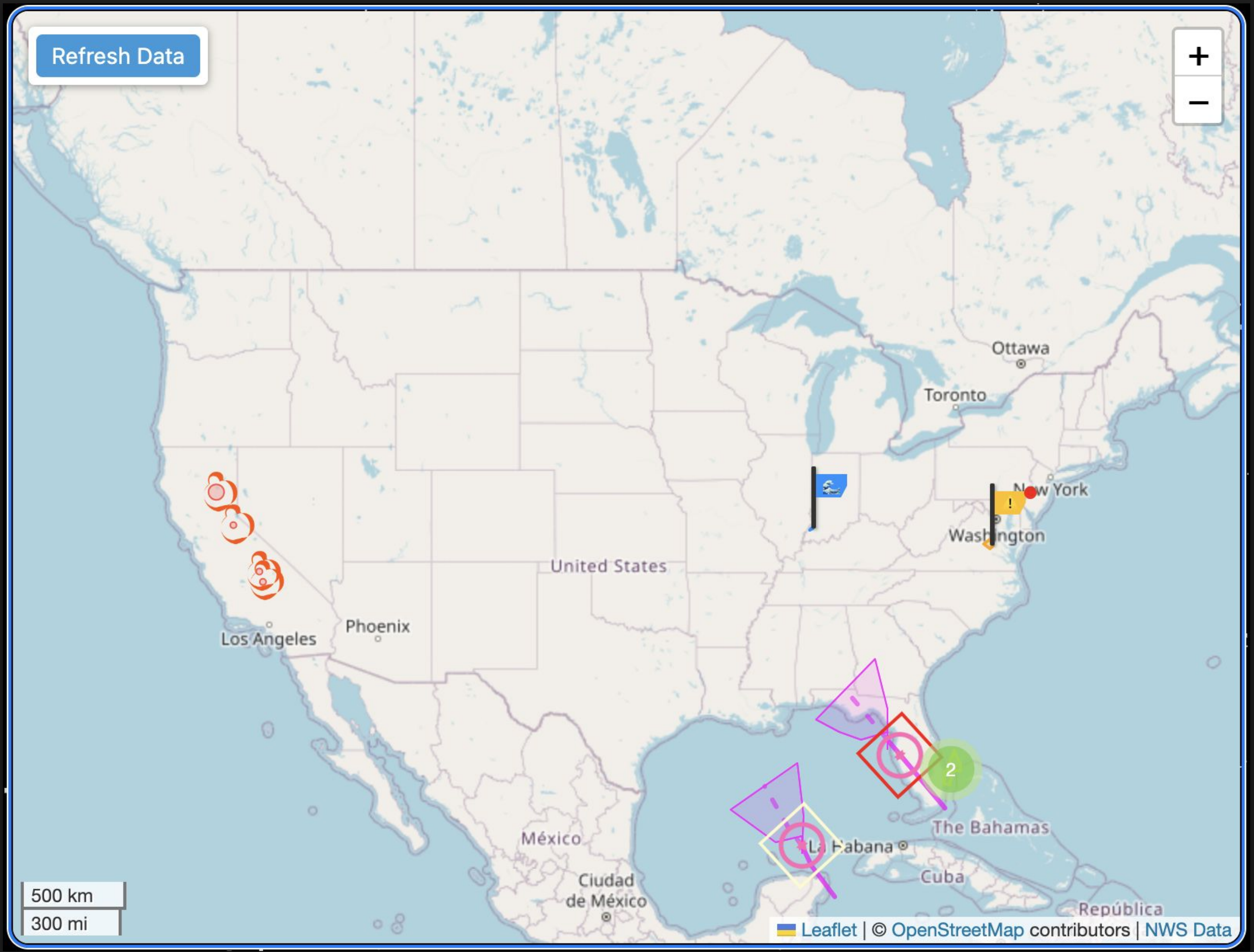
Team 5

May 9, 2025

Frontend - Mapping & Data Visualization

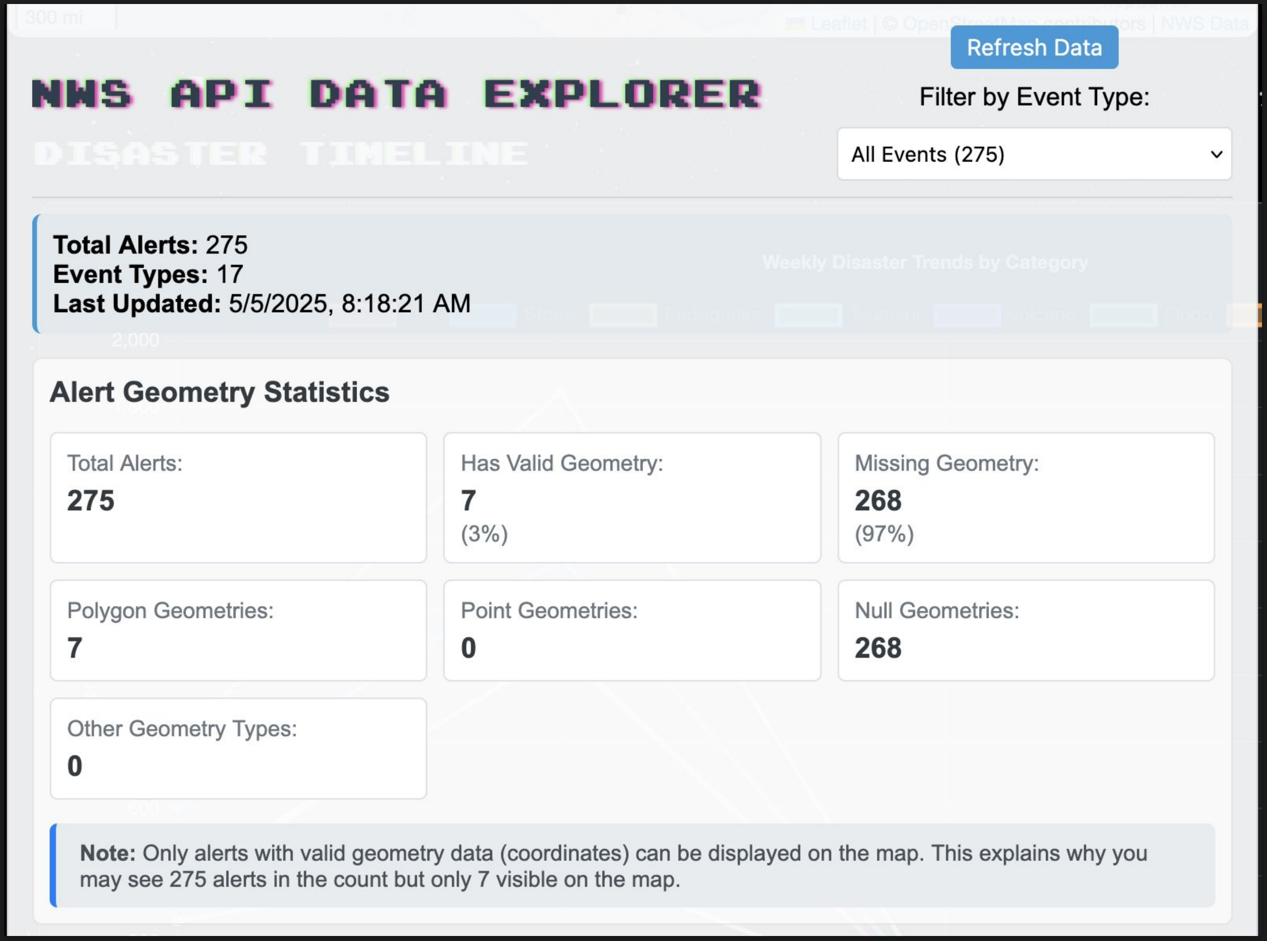
Map Interface Development

- **Core mapping library: Leaflet.js**
  - Dynamic loading of Leaflet with integrity checks for security
  - Responsive map interface with zoom, pan, and layer controls
  - Custom marker system using flag icons to represent alerts
- **Key map functionalities**
  - Interactive zoom and pan controls
  - Custom map markers for different disaster types
  - Toggle-able data layers (NWS Alerts, Wildfires, Hurricanes)
  - Auto-refresh capability (data updates every 5 minutes)
  - Responsive design works across desktop and mobile devices



NWS Data Integration & Display

- **NWS data fetching:**
  - Real-time data fetched from the National Weather Service API
  - URL: https://api.weather.gov/alerts/active
  - Filters for active and actual alerts only
- **NWS Data Viewer component:**
  - Comprehensive view of all NWS alerts with filtering options
  - Event type filtering to focus on specific disaster categories
  - Alert geometry statistics to track mappable vs. unmappable alerts
  - Expandable alert details with severity, urgency, and expiration information
  - Auto-refresh functionality to ensure data currency



NWS API DATA EXPLORER

Total Alerts: 275
Event Types: 17
Last Updated: 5/5/2025, 8:18:21 AM

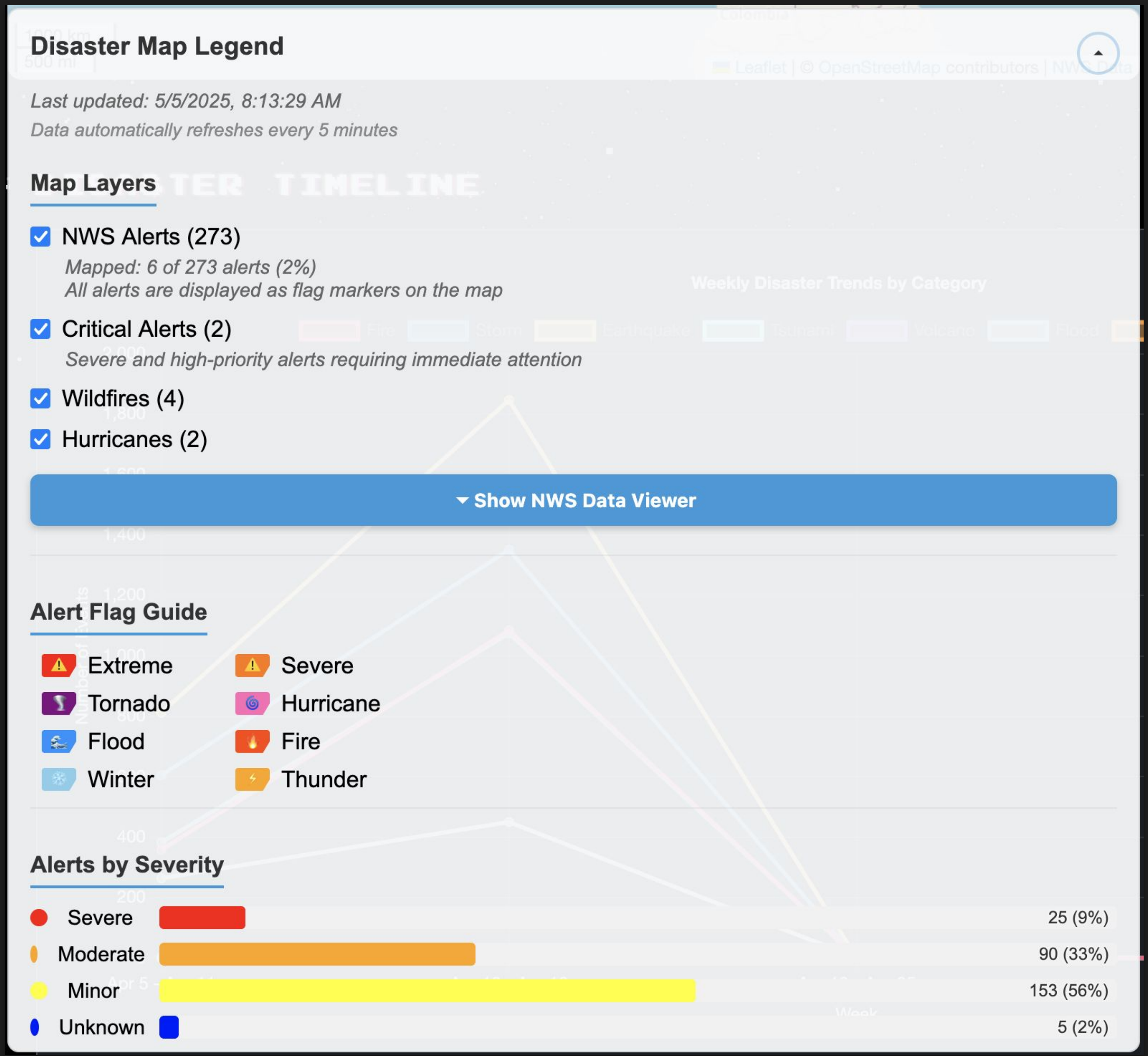
Alert Geometry Statistics

Total Alerts: **275**	Has Valid Geometry: **7** (3%)	Missing Geometry: **268** (97%)
Polygon Geometries: **7**	Point Geometries: **0**	Null Geometries: **268**
Other Geometry Types: **0**		

Note: Only alerts with valid geometry data (coordinates) can be displayed on the map. This explains why you may see 275 alerts in the count but only 7 visible on the map.

Map Legend

- **Purpose and design:**
  - Interactive legend with expandable/collapsible interface
  - Color-coded indicators for different alert severities and types
  - User-toggleable map layers for customized viewing
- **How it helps users interpret map data:**
  - Visually categorizes alerts by severity (Extreme, Severe, Moderate, Minor)
  - Color-coding system for different disaster types:
    - Floods: Blue (#1e90ff)
    - Tornadoes: Purple (#800080)
    - Wildfires/Smoke: Orange-red (#ff4500)
    - Winter/Snow/Ice: Light blue (#87ceeb)
    - Hurricanes/Tropical: Pink (#ff69b4)
    - Heat: Crimson (#dc143c)
  - Shows distribution percentages of each alert type
  - Includes timestamp information for data currency
  - Displays mapping statistics (how many alerts have coordinates vs. total)



Disaster Map Legend

Last updated: 5/5/2025, 8:13:29 AM
Data automatically refreshes every 5 minutes

Map Layers

☑ NWS Alerts (273)
Mapped: 6 of 273 alerts (2%)
All alerts are displayed as flag markers on the map

☑ Critical Alerts (2)
Severe and high-priority alerts requiring immediate attention

☑ Wildfires (4)

☑ Hurricanes (2)

Alert Flag Guide

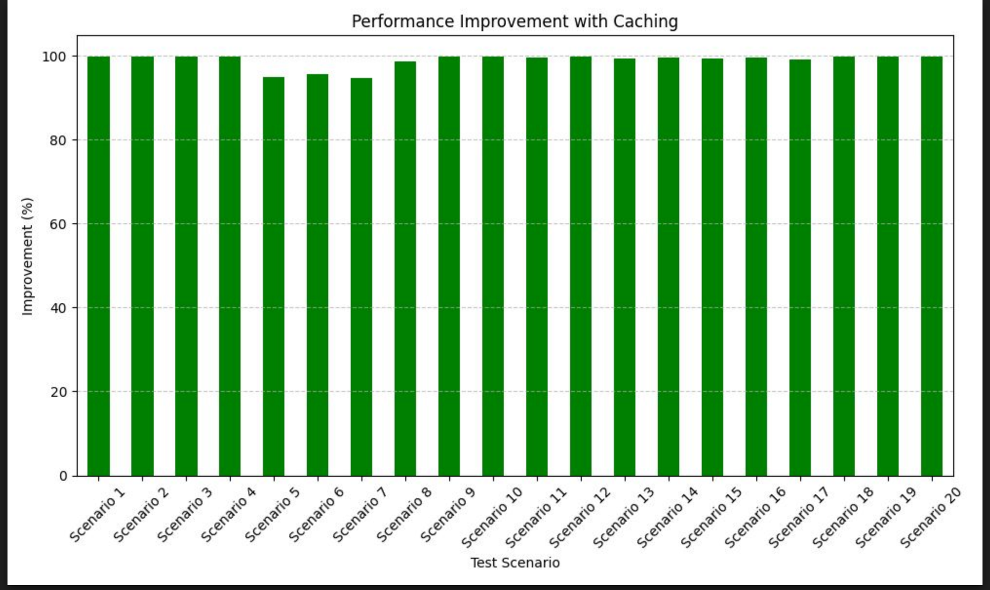
⚠ Extreme	⚠ Severe
Tornado	Hurricane
Flood	Fire
Winter	Thunder

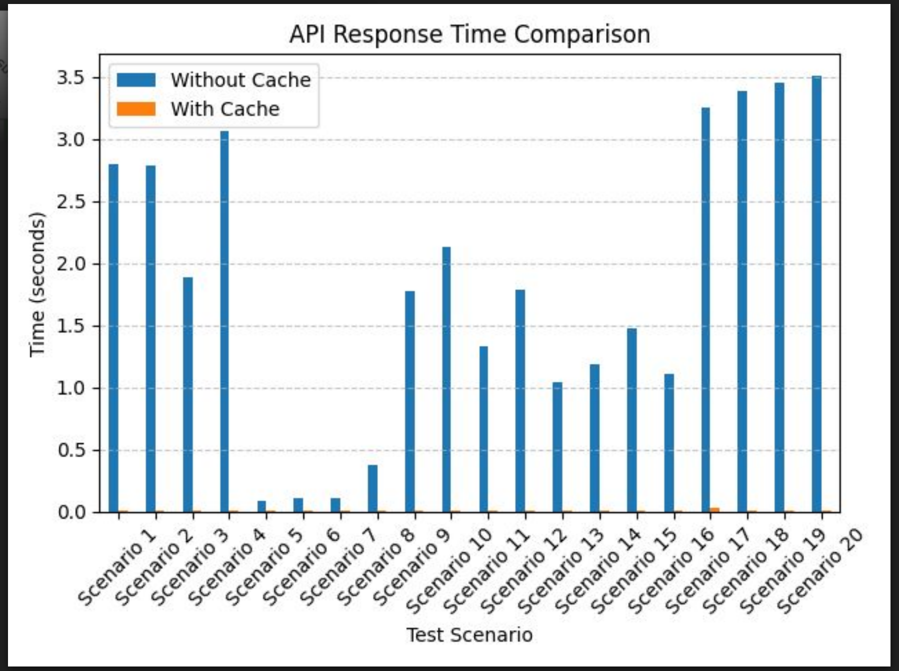
Alerts by Severity

● Severe	25 (9%)
● Moderate	90 (33%)
● Minor	153 (56%)
● Unknown	5 (2%)

Caching Implementation:

- **Purpose**
  - Performance Improvement: 94% faster response times for repeated requests
  - Cost Reduction: Minimizes intensive database queries
  - User Experience: Delivers chart data with minimal loading times
- **Where Caching is Implemented**
  - Disaster Timeline Endpoint with parameterized caching
  - Daily and weekly time-based aggregations
  - Filtered disaster type views (fire, flood, earthquake, hurricane)
  - Various time ranges (7-day, 30-day spans)
- **Why It Is Needed**
  - Handles complex timeline calculations efficiently
  - Maintains consistent performance during peak usage
  - Reduces server load for common visualizations
  - Improves overall application responsiveness





Team 5

May 9, 2025

Infrastructure & Deployment

Cloud Services & Hosting

- We developed a full-stack disaster management web application and deployed it on Amazon Web Services (AWS). The goal was to make the application publicly accessible, reliable, and easy to use, with a custom domain name.
- **Cloud Platform Selection: AWS**
  - It allowed easy integration with DynamoDB, enabling seamless backend database management without setting up servers.
  - AWS provided flexible hosting options (Lambda, S3, EC2), letting us choose the best fit for our application.
- **Deployment Architecture:**
  - Amazon EC2 → Hosted frontend + backend on one server.
  - Amazon Route 53 → Managed twdisasterwatch.com.
  - Amazon Linux OS for secure, optimized server.

EC2 Setup & Configuration

- Instance Selection: We deployed a t3.medium EC2 instance (2 vCPUs, 4 GB RAM) in the us-east-1 region to ensure sufficient resources for running both frontend and backend.
- Security Group Configuration: Opened ports 22 (SSH), 80 (HTTP), 443 (HTTPS), 3000 (frontend), and 5000 (backend) to enable secure access and public traffic to the application.
- Project Code Deployment: Cloned the project repository from GitHub into the EC2 instance and organized it under /home/ec2-user/team5disasteranalysis for deployment.
- Public IP Configuration: Assigned and used the EC2 instance's public IPv4 address to make the frontend and backend accessible over the internet.
- Port Testing & Validation: Verified public access to the frontend and backend by testing connections to ports 3000 (React) and 5000 (Flask) via browser.
  - http://52.23.167.1:3000 (frontend)
  - http://52.23.167.1:5000 (backend)

Domain & DNS Setup (Route 53)

- Domain Registration with Route 53: We registered the custom domain twdisasterwatch directly through AWS Route 53, simplifying domain management within the AWS ecosystem.
- Hosted Zone Creation: Created a hosted zone in Route 53 to manage all DNS records associated with the domain, enabling complete control over domain resolution.
- A Record Configuration: Added an A Record to point the domain to the EC2 instance’s public IP, making the application accessible by domain name.
- CNAME Record for Subdomain: Configured a CNAME Record for the domain to redirect the subdomain to the main domain, ensuring both addresses lead to the same site.
- Domain Resolution Testing: Tested domain resolution by accessing the application via
  - http://twdisasterwatch.com
  - http://www.twdisasterwatch.com

DEMO