**Catastrophe Alert & Response Microservice – Technical Overview**

1. Purpose

The Catastrophe Alert & Response Microservice detects, validates, and disseminates alerts for natural and human-made disasters (earthquakes, floods, wildfires, storms, etc.). Its goal is to provide structured, reliable, and real-time information to end-users, authorities, and connected systems while ensuring data integrity and traceability.

2. Scope

* Ingest event data from multiple sources: sensors, external APIs, user reports.
* Validate input fields: coordinates, event type, severity, timestamp.
* Store events in structured format (DB or JSON).
* Publish alerts to downstream systems: mobile apps, dashboards, emergency response.
* Provide a queryable API for historical and active events.

3. Microservice Architecture

| Layer | Responsibility |
| --- | --- |
| Input Layer | Accepts event reports via REST API, sensor streams, or external APIs. |
| Validation Layer | Checks coordinates, dates, catastrophe types, and severity thresholds. |
| Processing Layer | Deduplicates events, applies geofencing, enforces severity thresholds. |
| Storage Layer | Stores events with metadata: eventId, type, location, time, severity, reporter, status. |
| Output Layer | REST endpoints and event broker integration (Kafka/RabbitMQ) to distribute alerts in real-time. |

4. API Endpoints

| Method | Endpoint | Description |
| --- | --- | --- |
| POST | /catastrophe/report | Submit a new catastrophe event |
| GET | /catastrophe/active | Fetch active events (filterable by country/region) |
| GET | /catastrophe/history | Retrieve historical events |

5. Data Requirements

Events are stored in structured JSON or relational DB with the following fields:

{

"eventId": "string",

"type": "string",

"location": {

"country": "string",

"region": "string",

"lat": "float",

"long": "float"

},

"time": "ISO8601 datetime",

"severity": "float",

"reporter": "string",

"status": "string",

"createdAt": "datetime",

"updatedAt": "datetime"

}

Constraints & Notes:

* eventId is unique and generated per type-region-date.
* severity is mandatory and numeric.
* Valid types: earthquake, flood, wildfire, landslide, storm, volcano, tsunami.

6. Validation & Processing Rules

1. Event Type: Must match predefined types; otherwise rejected.
2. Coordinates: Latitude must be [-90, 90], longitude [-180, 180].
3. Severity: Must be numeric; reject missing values.
4. Date/Time: Must be valid ISO8601 string.
5. Deduplication: Combine identical events occurring within geospatial/time threshold.
6. Cleanup: Automatically remove events older than 14 days.

7. Use Cases & Expected Output

Use Case 1 – Valid Earthquake Report

Input:

{

"type": "earthquake",

"location": { "country": "Turkey", "region": "Istanbul", "lat": 41.01, "long": 28.95 },

"time": "2025-10-05T08:30:00Z",

"severity": 6.5

}

Output:

{

"status": "accepted",

"eventId": "EQ-20251005-IST-001"

}

Use Case 2 – Missing Severity

Input:

{

"type": "flood",

"location": { "country": "Egypt", "region": "Cairo", "lat": 30.0444, "long": 31.2357 },

"time": "2025-10-06T10:00:00Z"

}

Output:

{

"error": "Severity level is required"

}

Use Case 3 – Invalid Type

Input:

{

"type": "alien-attack",

"location": { "country": "USA", "region": "Nevada", "lat": 39.5, "long": -116.0 },

"time": "2025-10-07T12:00:00Z",

"severity": 10

}

Output:

{

"error": "Invalid catastrophe type"

}

Use Case 4 – Query Active Events

GET /catastrophe/active?country=Egypt

Output:

[

{

"eventId": "FL-20251006-CRO-001",

"type": "flood",

"severity": 4.2,

"location": "Cairo, Egypt",

"time": "2025-10-06T10:00:00Z"

}

]

8. Non-Functional Requirements

* Performance: ≥1000 reports/sec ingestion.
* Scalability: Horizontally via Docker/Kubernetes.
* Reliability: 99.9% uptime SLA.
* Security: Authenticated reporters; TLS encryption.
* Auditability: Every report stored with traceable metadata.

9. Future Enhancements

* ML-based severity scoring and prediction.
* Multi-source data fusion (satellite, social media, sensors).
* Map-based visualization of active and historical events.
* Predictive alerts based on historical patterns.