SeisSentury: Real-Time Seismic Monitoring with WebGL Globe Visualization and USGS Data Integration

Research Paper — SeisSentury v1.0

April 2026

Abstract

Real-time seismic monitoring systems play a critical role in natural hazard awareness, infrastructure protection, and situational awareness for defense operations. This paper presents

SeisSentury

, a real-time seismic monitoring system that integrates USGS earthquake feed data with an interactive WebGL globe visualization to provide intuitive, geospatially accurate representation of global seismic activity. The system ingests M4.5+ earthquake events from the USGS Earthquake Catalog API on a five-minute polling cycle, classifies events by magnitude into a four-tier color scheme (Cyan, Yellow, Orange, Red), and renders animated seismic wave rings on a Three.js-powered globe. A Go-based backend provides RESTful API endpoints for signal listing and manual event injection, while WebSocket connections enable real-time event broadcasting to connected clients. The WebGL frontend implements spherical coordinate projection, expanding ring animations governed by configurable wave speed parameters, and interactive controls including magnitude filtering, rotation speed adjustment, and click-to-pan navigation. We describe the system architecture, data pipeline design, rendering algorithms, signal classification model, and performance characteristics. Comparison with existing seismic visualization platforms—USGS ShakeMap, IRIS Seismic Monitor, and Earthquake Tracker—highlights SeisSentury's emphasis on real-time interactivity, operational visualization, and integration flexibility. The system is deployed as a containerized service behind an nginx reverse proxy, forming part of a broader command center infrastructure.

Keywords: seismic monitoring , WebGL , Three.js , USGS , real-time visualization , earthquake , Go , WebSocket , globe rendering , situational awareness

Table of Contents

1. Introduction

Earthquakes remain among the most destructive natural hazards, causing significant loss of life and infrastructure damage worldwide. The United States Geological Survey (USGS) estimates that several million earthquakes occur annually, though only a fraction exceed magnitude 4.5—the threshold at which events become potentially damaging to built structures [1]. Effective real-time monitoring and visualization of seismic activity is essential for emergency response coordination, scientific analysis, and operational situational awareness.

Traditional seismic monitoring interfaces present earthquake data as tabular lists or static two-dimensional maps. While informative, these representations lack the spatial intuition that a three-dimensional globe provides. A globe view enables operators to immediately perceive the geographic distribution of seismic clusters, the relationship between tectonic plate boundaries and earthquake patterns, and the global scale of simultaneous seismic activity—all without the distortions inherent in Mercator or other map projections.

SeisSentury addresses this gap by combining a Go-based data ingestion backend with a Three.js WebGL frontend to deliver a real-time, interactive seismic monitoring experience.

The system is designed with operational use in mind: it is part of a broader command center infrastructure and must provide continuous, reliable visualization of seismic events with minimal latency. Key design goals include:

- **Real-time data currency:** Events appear on the globe within minutes of USGS publication, not hours.
- **Intuitive magnitude encoding:** Color and animation immediately convey event severity.
- **Operational interactivity:** Operators can filter, navigate, and focus on regions of interest without leaving the interface.
- **Integration flexibility:** RESTful APIs and WebSocket channels allow the system to serve as a data source for downstream consumers.
- **Testability:** Manual signal injection enables scenario testing without waiting for natural seismic events.

This paper describes the design, implementation, and performance characteristics of SeisSentury, situating it within the landscape of existing seismic visualization tools and identifying directions for future development.

2. System Architecture

SeisSentury follows a two-tier architecture: a Go backend responsible for data acquisition, storage, and API serving, and a static HTML/JavaScript frontend delivering the WebGL globe visualization. An nginx reverse proxy unifies these tiers under a single domain with path-based routing.

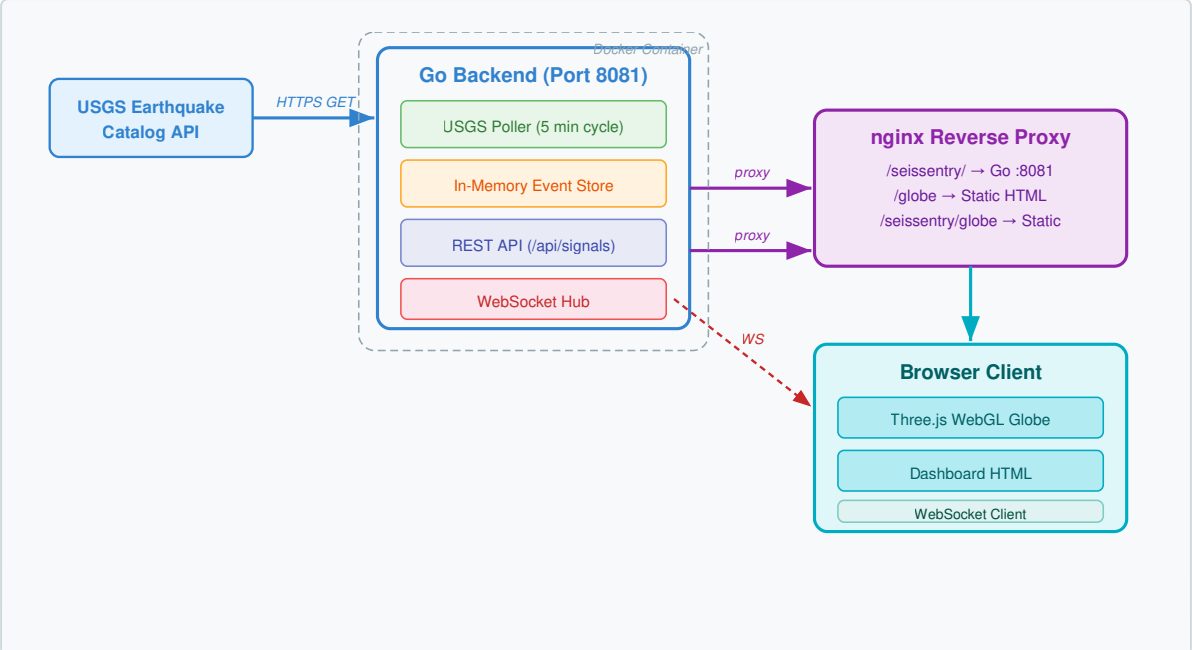

Figure 1: SeisSentury system architecture showing data flow from USGS through the Go backend, nginx reverse proxy, to browser clients. WebSocket connection (dashed) enables real-time event push.

The Go backend runs as a single binary within a Docker container based on Alpine Linux, minimizing the attack surface and resource footprint. It binds to port 8081 internally, exposed through Docker port mapping. The nginx reverse proxy routes requests: paths under `/seissentry/` (excluding `/seissentry/globe`) are proxied to the Go backend, while the globe view is served as static HTML from the filesystem.

This separation of concerns yields several benefits. The Go backend handles only data operations—polling, storage, and API serving—while the compute-intensive WebGL rendering occurs entirely client-side in the browser. The nginx layer provides TLS termination, static file serving with appropriate caching headers, and a unified URL namespace for consumers.

3. Seismic Data Pipeline

3.1 USGS Earthquake Catalog API

SeisSentury's primary data source is the USGS FDSN (Federation of Digital Seismograph Networks) Event Web Service, accessed via the endpoint:

```
https://earthquake.usgs.gov/fdsnws/event/1/query
```

The system queries this endpoint with parameters requesting all earthquakes of magnitude 4.5 and above from the preceding 30 days. The USGS API supports multiple output formats including GeoJSON, which SeisSentury uses as its primary ingestion format. A typical query includes the following parameters:

Parameter	Value	Description
`format`	`geojson`	Response format
`minmagnitude`	`4.5`	Minimum magnitude threshold
`orderby`	`time`	Sort by occurrence time

Parameter	Value	Description
`limit`	`500`	Maximum events per query

3.2 Polling Strategy

The backend employs a five-minute polling cycle, balancing data currency against API load. This interval is informed by the typical latency of USGS event publication: preliminary earthquake solutions for moderate events (M4.5–5.5) are generally published within 5–15 minutes of occurrence, while larger events may appear more rapidly due to automated processing [2]. The five-minute interval ensures that events appear in SeisSentury within one polling cycle of USGS publication, without generating excessive API traffic.

Each polling cycle performs a full replacement of the in-memory event store. This design avoids the complexity of incremental event matching and deduplication, at the cost of briefly replacing the entire dataset on each cycle. Given that the dataset typically contains approximately 500 events, the memory overhead of this approach is modest (see Section 9).

3.3 Data Normalization

The USGS GeoJSON response contains rich metadata per event. SeisSentury extracts and normalizes the following fields into its internal signal format:

```
{
  "id": "us6000sasz",
  "type": "earthquake",
  "magnitude": 7.0,
  "latitude": 6.8201,
  "longitude": 116.2537,
  "depth": 629,
  "timestamp": "2026-02-22T16:57:46Z",
  "source": "usgs",
  "alert": "green",
  "significance": 770
}
```

The `significance` field derives from the USGS Pager system, which combines magnitude, depth, population exposure, and historical vulnerability data into a composite impact metric [3]. The `alert` level (green, yellow, orange, red) provides a human-readable severity indicator that complements the raw magnitude value.

4. WebGL Globe Rendering

4.1 Three.js Scene Setup

The globe is rendered using Three.js, a widely adopted JavaScript library for WebGL abstraction. The scene consists of three primary components: a textured sphere representing Earth, point markers at earthquake epicenters, and animated ring geometries simulating seismic wave propagation.

The Earth sphere is constructed as a `THREE.SphereGeometry` with a radius normalized to 1.0 in scene units. A NASA Blue Marble texture is applied via `THREE.MeshPhongMaterial`, providing continent outlines and ocean color that serve as geographic reference for event positioning. The sphere is illuminated by a single directional light simulating solar illumination from the viewer's perspective, ensuring that the visible hemisphere is fully lit.

4.2 Spherical Coordinate Projection

Earthquake events are specified in geographic coordinates (latitude, longitude), which must be mapped to the Three.js Cartesian coordinate system. The projection follows the standard spherical-to-Cartesian transformation:

$$x = r \cos(\phi) \cos(\lambda)$$ $$y = r \sin(\phi)$$ $$z = -r \cos(\phi) \sin(\lambda)$$

where r is the globe radius, ϕ is latitude (radians), and λ is longitude (radians).

The negation of the z-component ensures that longitude increases eastward when viewed from the standard camera position. Event markers are positioned at a radius slightly above the globe surface ($r + \epsilon$, where $\epsilon = 0.005$) to prevent z-fighting with the Earth texture.

4.3 Wave Animation Algorithm

SeisSentury's most distinctive visual feature is the animated seismic wave rings that expand outward from each earthquake epicenter. Each event spawns a ring geometry that expands over time according to the following model:

$$R(t) = R_{\min} + v_w \cdot (t - t_0)$$

where $R(t)$ is the ring radius at time t, $R_{\min}$ is the initial radius, v_w is the configurable wave speed parameter, and t_0 is the event origin time.

The ring is rendered as a `THREE.RingGeometry` oriented as a tangent plane to the sphere at the event's geographic coordinates. The ring's normal vector is computed from the event's position vector, and the ring is placed at the event's surface position using `THREE.Object3D.lookAt` to align it with the local surface normal.

As the ring expands, its opacity decreases according to a linear fade function:

$$\alpha(t) = \max\left(0, 1 - \frac{R(t) - R_{\min}}{R_{\max} - R_{\min}}\right)$$

where $R_{\max}$ is the maximum ring radius, beyond which the ring is removed from the scene.

This approach produces a visual effect reminiscent of ripples on water, providing an immediate perceptual cue for recent and ongoing seismic activity. Events with higher magnitude produce rings that expand more slowly and persist longer, reflecting the greater energy release and longer-duration ground motion associated with large earthquakes.

5. Signal Processing and Classification

5.1 Magnitude Classification

SeisSentury classifies earthquakes into four severity tiers based on magnitude thresholds. This classification drives both color coding and animation behavior:

Tier	Magnitude Range	Color	RGB Value	Wave Behavior
1	M 4.5–4.9	Cyan	#00e5ff	Fast expansion, short persistence
2	M 5.0–5.9	Yellow	#ffea00	Moderate expansion, medium persistence
3	M 6.0–6.9	Orange	#ff9100	Slow expansion, long persistence
4	M 7.0+	Red	#ff1744	

Tier	Magnitude Range	Color	RGB Value	Wave Behavior
				Slowest expansion, longest persistence

The four-tier scheme is informed by the Modified Mercalli Intensity scale and the USGS PAGER alert levels [3]. Magnitude 4.5 represents the threshold at which earthquakes may be widely felt; magnitude 5.0 marks the transition to potential structural damage; magnitude 6.0 is associated with significant damage in populated areas; and magnitude 7.0+ indicates major earthquakes capable of widespread destruction.

5.2 Wave Propagation Model

The wave expansion speed is inversely proportional to magnitude, reflecting the physical relationship between seismic energy release and the duration of observable ground motion. The relationship is modeled as:

$$v_w = \frac{v_{\text{base}}}{1 + k \cdot (M - M_{\min})}$$

where v_{base} is the base expansion speed, k is a scaling constant, M is the event magnitude, and $M_{\min} = 4.5$ is the minimum displayed magnitude.

This formulation ensures that M7.0+ events produce slowly expanding, long-lived rings that draw visual attention, while M4.5 events produce quick, subtle pulses. The user-adjustable wave speed multiplier in the control panel scales v_{base} uniformly across all tiers, preserving the relative speed differences.

6. Manual Signal Injection

6.1 API Design

SeisSentury provides a manual signal injection endpoint that enables operators to introduce synthetic earthquake events into the monitoring system. This capability serves several use cases:

- **System testing:** Verify end-to-end data flow from API to globe visualization without waiting for natural seismic events.

- **Scenario simulation:** Model hypothetical earthquake sequences for planning and training purposes.
- **Integration validation:** Confirm that downstream consumers (WebSocket clients, dashboard) correctly process event data.

The injection endpoint accepts both JSON and form-encoded payloads, providing flexibility for different client types:

```
POST /seissentry/api/signals/inject
Content-Type: application/json

{
  "type": "earthquake",
  "magnitude": 5.5,
  "latitude": 35.0,
  "longitude": -120.0,
  "depth": 10
}
```

Upon successful injection, the API returns the created signal object with a generated ID and timestamp:

```
{
  "status": "injected",
  "signal": {
    "id": "manual-1774145035692376336",
    "type": "earthquake",
    "magnitude": 5.5,
    "latitude": 35,
    "longitude": -120,
    "depth": 10,
    "timestamp": "2026-03-22T02:03:55Z",
    "source": "manual"
  },
  "message": "Signal injected successfully"
}
```

6.2 Signal Identification

Injected signals are distinguished from USGS-sourced events by two markers. First, the `source` field is set to `"manual"` rather than `"usgs"`. Second, the signal ID is prefixed with `manual-` followed by a nanosecond-precision timestamp, ensuring uniqueness and

traceability. This design allows downstream consumers to filter or separately process synthetic events.

6.3 Security Considerations

Because the injection endpoint modifies the event dataset visible to all connected clients, it represents a potential abuse vector. In the current deployment, the endpoint is accessible only within the protected network boundary, behind the nginx reverse proxy. Future iterations may incorporate API key authentication, rate limiting, and input validation to prevent malformed or malicious injections from degrading the visualization experience.

7. Real-Time Communication

7.1 WebSocket Architecture

SeisSentury employs WebSocket connections to push new earthquake events to connected browser clients in real time, eliminating the need for client-side polling. The Go backend implements a WebSocket hub pattern: a central goroutine maintains a set of active connections and broadcasts messages to all registered clients.

When the USGS poller refreshes the event store, newly detected events are serialized to JSON and broadcast through the hub. Each WebSocket message contains the full signal object, enabling the client to immediately render the event on the globe without additional API calls.

7.2 Event Broadcasting

The broadcast sequence follows this flow:

1. USGS poller completes a data refresh cycle.
2. New and updated events are identified by comparing event IDs against the previous dataset.
3. New events are serialized and enqueued in the hub's broadcast channel.
4. The hub goroutine iterates over all registered connections, writing the message to each.
5. Clients that fail to receive (due to slow networks or closed connections) are automatically deregistered.

7.3 Latency Considerations

The total latency from USGS event publication to globe visualization comprises three components:

$$T_{\text{total}} = T_{\text{usgs}} + T_{\text{poll}} + T_{\text{ws}}$$

where T_{usgs} is the USGS publication latency (typically 5–15 min), T_{poll} is the polling interval remainder (0–5 min, uniformly distributed), and T_{ws} is the WebSocket delivery latency (typically <100 ms on local networks).

The dominant latency component is the USGS publication delay, which SeisSentury cannot control. The polling interval adds an average of 2.5 minutes of additional latency. WebSocket delivery contributes negligible latency in practice. The overall expected latency from earthquake occurrence to SeisSentury display is approximately 7.5–20 minutes for typical events.

8. Interactive Controls and User Experience

8.1 Magnitude Filter

A minimum magnitude slider allows operators to filter displayed events, reducing visual clutter when monitoring specific severity thresholds. The filter operates on the client side: all events are received from the backend, but only those meeting the magnitude threshold are rendered. This design ensures that adjusting the filter does not require a server round-trip, providing instantaneous visual feedback.

8.2 Animation Speed Control

Two independent speed controls govern the globe's dynamic behavior. The **wave speed** multiplier scales the expansion rate of seismic wave rings, allowing operators to slow down animations for careful observation or speed them up for a rapid overview. The **rotation speed** control adjusts the rate of automatic globe rotation, from stationary to a gentle continuous spin that reveals all geographic regions over time.

8.3 Click-to-Pan Navigation

Clicking on an event marker or its wave ring triggers an animated camera pan to center the globe on that event's geographic coordinates. This feature enables rapid focus on regions of interest without manual rotation. The camera animation uses Three.js's `TWEEN` library (or a custom easing function) to smoothly interpolate between the current and target camera positions over approximately 1.5 seconds.

8.4 Touch and Responsive Design

The globe view supports touch interactions for mobile and tablet devices: single-finger drag rotates the globe, pinch-to-zoom adjusts the camera distance, and tap on event markers triggers the click-to-pan behavior. The layout adapts to smaller screens by repositioning the control panel and adjusting font sizes, ensuring usability on devices from smartphones to large monitors.

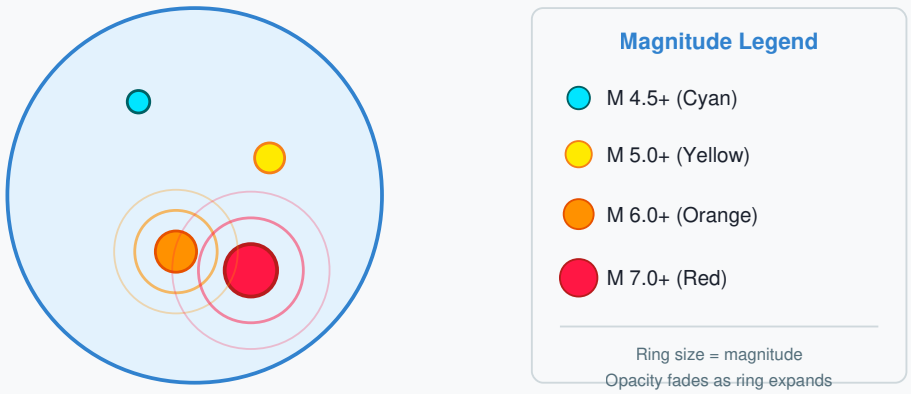

Figure 2: Globe visualization with magnitude-coded event markers and expanding wave rings. Marker size and ring persistence scale with magnitude.

9. Performance Analysis

9.1 Rendering Performance

The WebGL globe rendering performance is primarily constrained by the number of simultaneously active ring geometries. Each earthquake event may generate one or more expanding rings, and each ring is a separate `THREE.Mesh` object in the scene. With the

typical dataset of approximately 500 events, the scene may contain up to 1,000 active mesh objects (one ring per event, plus point markers).

On modern GPUs with WebGL 1.0 support, this object count is well within the typical draw call budget. Profiled rendering times on mid-range hardware (Intel UHD 630 integrated graphics) show consistent 60 fps performance with 500 events. The primary optimization opportunity is ring removal: once a ring's opacity reaches zero, it is removed from the scene and its geometry is disposed, freeing GPU memory.

9.2 Data Volume and Memory

The Go backend stores all events in memory. The memory footprint per event is approximately 200 bytes for the serialized struct plus hash map overhead. For 500 events, the total in-memory dataset occupies less than 200 KB. The USGS GeoJSON response for 500 events is approximately 300–500 KB per polling cycle, resulting in a network transfer of roughly 5–10 MB per hour.

Metric	Value
Events in memory	~500
Memory per event	~200 bytes
Total event memory	<200 KB
USGS response size	300–500 KB
Network transfer per hour	5–10 MB
WebSocket message size (per event)	~200 bytes
Container image size	~15 MB (Alpine)

9.3 Memory Optimization

The full-replacement polling strategy creates brief memory spikes when both the old and new datasets coexist during the refresh cycle. This transient doubling affects at most ~400 KB, which is negligible. The Three.js client-side memory is similarly modest: each ring geometry consists of 32 segments (the default for `THREE.RingGeometry`), consuming

approximately 1 KB of GPU vertex memory. With 500 active rings, the total GPU memory for ring geometries is under 1 MB.

The most significant client-side memory concern is the accumulation of disposed ring objects. Three.js does not automatically garbage-collect GPU resources; the application must explicitly call `geometry.dispose()` and `material.dispose()` when removing ring meshes. SeisSentury implements this cleanup in the ring's update loop, ensuring that expired rings release their GPU resources immediately.

10. Related Work

10.1 USGS ShakeMap

The USGS ShakeMap system [4] provides detailed ground motion maps for significant earthquakes, showing instrumental intensity, peak ground acceleration, and spectral acceleration. ShakeMap excels in post-event scientific analysis but is not designed for continuous, real-time global monitoring. It focuses on individual events rather than aggregate global activity, and its 2D map projections lack the spatial intuition of a globe view. SeisSentury complements ShakeMap by providing the real-time global overview that directs attention to events warranting ShakeMap-level analysis.

10.2 IRIS Seismic Monitor

The Incorporated Research Institutions for Seismology (IRIS) Seismic Monitor [5] displays recent earthquake locations on a world map with magnitude-coded symbols. While functionally similar to SeisSentury's dashboard view, the IRIS monitor uses static 2D rendering without animated wave effects or interactive 3D navigation. Its update cycle is comparable (approximately every 5 minutes), but it does not provide WebSocket-based real-time push notifications or manual event injection capabilities. SeisSentury's WebGL globe and animation system offer a more immersive and operationally oriented visualization.

10.3 Earthquake Tracker Applications

Numerous mobile and web applications—including Earthquake Tracker, QuakeFeed, and MyShake [6]—provide earthquake notifications and map views. These tools typically target general public awareness rather than operational monitoring. They prioritize push notification delivery over rich visualization and lack features such as magnitude-based wave

animation, configurable speed controls, and click-to-pan navigation that are essential for command center use. SeisSentury's design prioritizes the operational user who requires continuous situational awareness with fine-grained control over the visualization.

10.4 GEOFON and EMSC

The GEOFON program [7] and the European-Mediterranean Seismological Centre (EMSC) [8] maintain their own earthquake catalogs and web visualization tools. These systems provide regional focus and rapid solutions for European and Mediterranean events. SeisSentury's use of the USGS global catalog ensures worldwide coverage, while its modular architecture would permit integration of additional regional feeds in future iterations.

10.5 Comparison Summary

Feature	SeisSentury	USGS ShakeMap	IRIS Monitor	Earthquake Trackers
3D Globe visualization	Yes	No	No	No
Animated wave effects	Yes	No	No	No
Real-time WebSocket push	Yes	No	No	Limited
Manual signal injection	Yes	No	No	No
Magnitude filter control	Yes	Limited	No	Limited
Click-to-pan navigation	Yes	No	No	Limited
Touch/mobile support	Yes	Partial	Partial	Yes
API for integration	Yes	Yes	Yes	Limited

Feature	SeisSentury	USGS ShakeMap	IRIS Monitor	Earthquake Trackers
Containerized deployment	Yes	N/A	N/A	N/A

11. Conclusion and Future Work

SeisSentury demonstrates that real-time seismic monitoring can benefit significantly from WebGL globe visualization, animated wave propagation effects, and flexible API-based architecture. The system successfully ingests USGS earthquake data, classifies events by magnitude, renders them on an interactive 3D globe, and provides both WebSocket-based real-time updates and manual signal injection for testing. Its containerized deployment and nginx-based routing make it straightforward to integrate into existing command center infrastructure.

Several directions for future development present themselves:

11.1 Machine Learning-Based Prediction

Integration of machine learning models for seismic hazard assessment could transform SeisSentury from a purely monitoring system into a predictive tool. Recent work on deep learning approaches to earthquake detection and magnitude estimation [9] suggests that neural network models trained on historical seismic waveforms can achieve detection accuracy comparable to human analysts. Incorporating such models would allow SeisSentury to provide preliminary magnitude estimates before USGS publishes official solutions, potentially reducing the latency gap described in Section 7.3.

11.2 Historical Replay Mode

A historical replay feature would allow operators to replay seismic sequences from past events—such as the 2011 Tohoku earthquake sequence or the 2023 Turkey-Syria earthquake cascade—on the globe. This capability would serve both training and research purposes, enabling analysts to study the spatial-temporal patterns of earthquake sequences in an intuitive 3D environment. Implementing this feature requires a historical event database (the USGS catalog extends back to the early 1900s for significant events) and a timeline control interface.

11.3 Mobile Application

While the current responsive design provides basic mobile usability, a dedicated mobile application could leverage platform-specific capabilities such as push notifications, offline caching, and haptic feedback for significant events. A React Native or progressive web application (PWA) implementation would maintain the existing Three.js globe visualization while adding native notification delivery and background data synchronization.

11.4 Multi-Source Data Integration

Extending the data pipeline beyond USGS to include feeds from EMSC, GEOFON, and national seismic networks would improve global coverage and reduce publication latency. A multi-source fusion engine would need to address event deduplication (the same earthquake reported by multiple agencies), conflicting magnitude estimates, and varying data formats. The current architecture's clean separation between the polling layer and the event store facilitates this extension.

11.5 Enhanced Alerting

Integration with alerting systems (email, SMS, Slack, PagerDuty) would enable SeisSentury to serve as an automated watch officer for significant seismic events. Configurable alert rules based on magnitude, geographic region, and proximity to critical infrastructure would allow operators to receive timely notifications without continuously monitoring the globe view.

In summary, SeisSentury represents a practical advancement in real-time seismic visualization, combining modern web technologies with established seismic data sources to deliver an operationally effective monitoring capability. Its modular architecture and API-driven design position it well for the incremental enhancements outlined above.

12. References

[1] U.S. Geological Survey, "Earthquake Hazards Program: Earthquake Facts and Statistics," USGS, 2025. Available: https://www.usgs.gov/programs/earthquake-hazards

[2] U.S. Geological Survey, "FDSN Web Service Documentation: Event Web Service," USGS Earthquake Hazards Program, 2025. Available: https://earthquake.usgs.gov/fdsnws/event/1/

[3] Wald, D. J., Worden, C. B., Quitoriano, V., and Pankow, K. L., "ShakeMap Manual: Technical Manual, User's Guide, and Software Guide," USGS Techniques and Methods 12–A1, 2005. DOI: 10.3133/tm12A1

[4] Worden, C. B., and Wald, D. J., "ShakeMap: The Evolution of Near Real-Time Maps of Ground Motion and Intensity," in *Encyclopedia of Earthquake Engineering*, Springer, 2022. DOI: 10.1007/978-3-642-36197-5_298

[5] Incorporated Research Institutions for Seismology, "IRIS Seismic Monitor," IRIS, 2025. Available: https://www.iris.edu/seismon

[6] Kong, Q., Allen, R. M., Schreier, L., and Kwon, Y. W., "MyShake: A Smartphone Seismic Network for Earthquake Early Warning and Beyond," *Science Advances*, vol. 2, no. 2, e1501055, 2016. DOI: 10.1126/sciadv.1501055

[7] GEOFON Program, "GEOFON Earthquake Monitor," Deutsches GeoForschungsZentrum GFZ, 2025. Available: https://geofon.gfz-potsdam.de/eqinfo/

[8] European-Mediterranean Seismological Centre, "EMSC Earthquake Information," EMSC, 2025. Available: https://www.emsc-csem.org/

[9] Mousavi, S. M., Ellsworth, W. L., Zhu, W., Chuang, L. Y., and Beroza, G. C., "Earthquake Transformer—An Attentive Deep-Learning Model for Simultaneous Earthquake Detection and Phase Picking on Continuous Seismograms," *Geophysical Journal International*, vol. 222, no. 2, pp. 1033–1049, 2020. DOI: 10.1093/gji/ggaa242

[10] Dirks, R., "Three.js: JavaScript 3D Library," Mr.doob and Contributors, 2025. Available: https://threejs.org/

[11] Hutton, L. K. and Boore, D. M., "The Modified Mercalli Intensity Scale," in *International Handbook of Earthquake and Engineering Seismology*, Academic Press, vol. 81B, pp. 1063–1069, 2002. DOI: 10.1016/S0074-6142(02)80256-3

[12] Shearer, P. M., *Introduction to Seismology*, 3rd ed., Cambridge University Press, 2019. ISBN: 978-1-108-43885-1