Synthetic Users Monitoring & Observability Policy

Version: 1.0

Effective Date: 15 February 2026

Owner: Security & Compliance Lead

Approved by: CTO

1. Purpose

This policy documents Synthetic Users' monitoring, observability, and alerting infrastructure used to track system health, performance, availability, and security across all production environments.

2. Scope

This policy covers all production systems including the SaaS platform, backend services, task queues, AI/LLM integrations, and supporting infrastructure.

3. Monitoring Stack

3.1 Overview

Layer	Tool	Purpose
Error Tracking & APM	Sentry	Exception capture, performance tracing (100% sampling), profiling

Metrics & Alerting	Prometheus + Grafana	Custom metrics collection, dashboards, threshold-based alerting
Log Aggregation	PaperTrail (via Render log drain)	Centralized log collection and search across all services
Uptime & Status	Better Stack	Public status page (status.syntheticusers.com), uptime monitoring
LLM Observability	Helicone	OpenAI API call monitoring, cost tracking, latency
LLM Tracing	LangSmith	LangChain execution tracing, prompt debugging
Analytics & Feature Flags	PostHog	Product analytics, feature flag evaluation
Alert Delivery	Slack	All operational alerts routed to Slack channels

---

4. Application Performance Monitoring (APM)

4.1 Sentry

- Deployed in production with **100% trace sampling** and **100% profiling**.
- Captures unhandled exceptions across the web application and Celery task workers.
- Provides performance tracing for request latency, database queries, and external API calls.
- `capture_exception()` integrated across critical code paths: subscriptions, workspaces, projects, interviews, knowledge graphs, summaries.

4.2 Health Check Endpoints

Endpoint	Purpose	Checks
`GET /healthz`	Application health	Database connectivity
`GET /checkz`	Task system health	Celery worker connectivity, event publishing

These endpoints are monitored by Better Stack for uptime tracking.

---

5. Metrics & Dashboards

5.1 Prometheus Metrics

Custom Prometheus metrics are exposed at `/metrics` and track:

- **Stuck entity counts** — per entity type (studies, interviews, reports, audience generation, file processing, knowledge graphs, copilot sessions, research plans, suggestions)
- **Entity status distribution** — processing, terminal, and failed states
- **Time elapsed tracking** — identifies entities stuck between 30 minutes and 24 hours

5.2 Grafana Dashboards

Grafana connects to Prometheus and provides:

- Real-time dashboards for system health and workflow status
- Historical trend analysis for capacity planning
- Visual alerting thresholds for operational KPIs

---

6. Log Management

6.1 PaperTrail

- Configured as a **Render platform-level log drain**, capturing logs from all services without application-level integration.
- Provides centralized search, filtering, and retention across web, worker, and infrastructure logs.
- Log levels: INFO in production, DEBUG in development. External library logging (OpenAI, httpx, urllib3) suppressed to WARNING level to reduce noise.

7. Uptime & Status Page

7.1 Better Stack

- **Public status page:** status.syntheticusers.com
- Monitors health check endpoints for availability.
- Provides incident communication to customers during outages.
- Historical uptime metrics available for SLA reporting.

8. LLM Observability

8.1 Helicone

- Monitors all OpenAI API calls in production.
- Tracks request latency, token usage, cost, and error rates.

8.2 LangSmith

- Traces LangChain execution flows for debugging and optimization.
- Captures prompt inputs, model outputs, and chain execution paths.

---

9. Alerting

9.1 Alert Routing

All operational alerts are delivered to **Slack**:

Source	Alert Type	Destination
Grafana + Prometheus	Stuck entities, metric threshold breaches	Slack
Sentry	Unhandled exceptions, error spikes	Slack
Better Stack	Downtime, health check failures	Slack

9.2 In-App Usage Alerts

- Subscription usage alerts at **50%** and **75%** thresholds via Novu (in-app notifications to workspace owners).
- Feature-flag controlled (PostHog: `enable-usage-alerts`).

---

10. Capacity Monitoring

10.1 Task Queue (Celery)

- Redis-backed task queue with connection pool limits.
- Worker concurrency: 10 workers.
- Task timeouts: 15 minutes soft / 16 minutes hard.

- Late acknowledgment strategy with automatic requeue on worker loss.
- Queue prioritization: default queue (priority 5), copilot queue (priority 10).

10.2 Infrastructure

- Render provides platform-level resource monitoring for compute and memory utilization.
- AWS provides CloudWatch metrics for S3, RDS, and EC2 resources.

---

11. Review

This policy is reviewed annually or when changes to the monitoring infrastructure occur.