# ❌ Superadmin Module: Incident Logs & System Alerts

## 🎯 Objective

Create a centralized system for tracking, categorizing, and notifying all infrastructure and functional incidents across the SaaS platform. This includes outages, performance degradation, security flags, third-party failures, and internal module malfunctions.

## 🧩 Feature Scope

### 1. **Real-Time Incident Detection Layer**

* Logs system events, internal errors, third-party failures
* Integrated with:
  + Web servers (e.g., NGINX, Apache)
  + App logs (Node.js, Python)
  + Queue workers (Celery, Redis Queue, etc.)
  + AI modules and webhook dispatchers
* Detects:
  + Downtime
  + 500 errors / API failures
  + Latency spikes
  + Failed jobs or queue overflows
  + Retry loops / throttling
  + Memory/CPU saturation

### 2. **Incident Classifier**

* Severity tagging: LOW, MEDIUM, HIGH, CRITICAL
* Auto-categorization:
  + Platform infrastructure
  + Brand-specific module
  + AI API limits
  + Integration errors (Shopify, WooCommerce, FB/IG, etc.)

### 3. **Incident Logging Engine**

* Schema:
  + timestamp
  + service/module
  + description
  + error log (traceback/snippet)
  + severity
  + linked tenant/brand
  + resolved (true/false)
  + auto-dismiss time (if transient)

### 4. **Admin Dashboard View**

* Global log explorer for superadmins
* Filter by:
  + Time range
  + Brand
  + Severity
  + Module/Service
* UI Components:
  + 🔴 Active incidents panel
  + 🧠 Root cause suggestions (AI-assisted)
  + ✅ Mark resolved + Add notes
  + ⏱️ Auto-dismiss timer panel

### 5. **Notification System**

* Email, Slack, In-app banner
* Configurable by alert level (e.g., notify only for HIGH+)
* Custom escalation matrix:
  + Tier 1: Engineering
  + Tier 2: Brand Account Manager
  + Tier 3: SaaS Founders/Admins

### 6. **Linked Actions + Automation Hooks**

* Integration with custom alert rules (if rules were defined for similar events)
* Trigger auto-muting, log rotation, or module disable for safety
* Webhook to incident response platform (PagerDuty, OpsGenie, etc.)

## 🛠️ Backend Architecture

* Node.js or Python service with queue-based collector
* ELK Stack or Loki/Grafana (for log parsing/visualization)
* DB: PostgreSQL for indexed structured logs, S3 for raw long-term logs

## 🔐 Permissions & Logs Access

* Superadmins: Full logs and alert controls
* Brand Admins: Read-only logs of their incidents only (no global issues)
* Engineering: Read/write with escalated automation triggers

## 🧠 Future Enhancements

* AI-generated root cause insights per pattern
* Anomaly heatmaps on timelines
* Auto-rollbacks if incident detected on new deploys
* Inferred dependencies map (which module causes ripple failures)

This module ensures high resilience, accountability, and faster mitigation across the platform through proactive system-wide observability and incident intelligence.