Multi Model- Observability Tool Requirements

Agent 5: AI Log Viewer for Industry 5.0 [Early Phase]

Value Tagline: Predict, explain, and resolve — the future of observability for Industry 5.0

Value Proposition: An AI-native, multi-modal log viewer that empowers human-AI collaboration to predict, explain, and resolve issues in real time. It reduces downtime, enhances trust and compliance, and future-proofs observability for Industry 5.0.

Market Value: It is estimated that log analytics / management total market market in 2025 is between USD 3 to 4 billion, with a CAGR of 18% growth.

Existing Tools A. Static Search & Filters: Legacy tools rely on regex, keyword search, or manual dashboards — requiring expert operators. B. Reactive, Not Proactive: They show what happened but don't infer why or what's next. C. Limited Contextualization: Logs, traces, and metrics are siloed, requiring cross-tool investigation. D. No Adaptive Learning: Alerts are threshold-based, leading to noise or missed anomalies. E. Scalability Issues: Increasing log volume (IoT, edge, robotics) makes it impractical to manually sift through. F. Not Human-Centric: Industry 5.0 emphasizes human-AI collaboration; legacy tools aren't designed for augmented decision-making.

"Include everything" feature set (organized by modules)

A) Data onboarding (file → project)

- Drag/drop upload, multi-file batch, folder upload
- Large file support: chunked upload, background parsing, gzip
- Encoding detection (UTF-8/UTF-16), line ending handling
- "Sample preview" before import + estimated parse confidence
- Privacy modes: local-only / self-host / cloud; PII redaction toggles

B) Parsing & structuring (make raw logs queryable)

- Automatic format detection:
    - JSON logs
    - common patterns (nginx, apache, syslog, java, python)
    - multiline (stack traces) stitching
- **Parser Studio** (power feature):
    - regex/grok builder, preview, test cases
    - field type mapping: string/int/float/bool/datetime
    - save parsers as reusable "pipelines"
- Enrichment:
    - geoIP (optional), host/service lookup
    - derive fields (e.g., status_class=5xx)
- OTel-style field model (so later can ingest OTel natively)

C) Storage & indexing (fast "instant" analytics)

Two good architecture options:

Option 1: Search-first (text search heavy)

- OpenSearch/Elasticsearch style indexing for full-text + aggregations
  Best when users do lots of “find this string” + fuzzy search.

Option 2: Analytics-first (BI heavy)

- Columnar store (ClickHouse-style) for blazing aggregations over huge volumes
  Best when users want dashboards, group-bys, percentiles, trends.

Hybrid (recommended)

- Hot path: lightweight label/metadata index (Loki-like idea) + compressed raw store
- Analytics path: structured columns in columnar DB for BI

D) Query & exploration UX (what makes it lovable)

- Google-like search bar + filters
- Saved views: “production errors”, “auth failures”, “slow requests”
- Live tail (stream-like) even for file imports (“replay” mode)
- Query language:
  - Simple mode (no syntax)
  - Advanced mode (SQL / LogQL-style concepts)
- “Pin to dashboard” from any query result

E) BI analytics (instant dashboards)

Out-of-the-box dashboards after import:

- Volume over time, error rate, severity breakdown
- Top services/hosts, top messages, top exceptions
- Latency percentiles if duration exists
- “New vs known errors” trend
- Heatmaps by hour/day

Ad-hoc BI:

- Pivot table builder
- Group-by any field
- Drill-down: chart → exact log lines

F) AI analytics (the differentiator)

AI features that feel *observability-native*:

- Auto field extraction: “this looks like request_id / user_id / endpoint”
- Clustering: group similar log messages (template mining)
- Anomaly detection: spikes/drops by message cluster/service (Elastic-like expectation)
- RCA assistant:

    - "What changed around 14:32?"
    - "Show new errors introduced after deployment tag X"
    - "Correlate error spike with host/service fields"
- Summaries:
    - Executive summary ("top 3 incidents")
    - Timeline narrative ("first error at…, spread to…")
- Explain this log: decode stacktrace, likely cause, next checks
- Natural language → query:
    - "show 5xx errors for checkout in the last 30 minutes"
- Guardrails:
    - cite evidence (links to specific log lines / query results)
    - confidence scores, "unknown" allowed
    - redact sensitive tokens before sending to models (if cloud)

G) Alerting & incidents (turn analytics into action)

- Alert rules on:
    - log volume, error rate, specific patterns, anomaly signals
- Notification channels: email, Slack, PagerDuty/webhooks
- Deduping & grouping: by service + error template
- Incident timeline auto-generated from log clusters

H) Governance, security, and enterprise basics

- RBAC + project/workspace separation
- Data retention policies, tiered storage
- Audit logs (who searched what)
- PII detection + masking
- Export: CSV/JSON, shareable links, "download filtered logs"

I) Integrations (future-proof)

- OTel ingestion endpoint (OTLP) so move from "file upload" to "real telemetry" later
- Agents/collectors: Fluent Bit / Vector (later)
- Grafana integration for dashboards (optional)
- Ticketing: Jira / ServiceNow (optional)

Front End Requirements :-

here's a **front-end (UI/UX) design blueprint** for "upload log file → instant BI + AI analytics" observability app. I'll lay it out as **screens, components, interactions, and states** so UI covers everything.

Information architecture (top-level navigation)

Left sidebar

- **Projects / Datasets**
- **Import**
- **Explore (Logs)**
- **Dashboards**
- **AI Insights**
- **Alerts**
- **Settings**

Top bar (global)

- Dataset selector (current imported file set)
- Time range picker (Last 15m / 1h / 24h / Custom)
- Search / Command palette (⌘K)
- User / Workspace

Screen 1: Landing / Projects

Goal: "Pick a dataset quickly or upload a new one."

Layout

- Header: "Your Datasets"
- Cards grid:
    - Dataset name
    - Imported time, size, #lines, time-span covered
    - Quick stats: error rate, top service
    - Actions: Open / Share / Delete
- Primary CTA: **Upload logs**

Empty state

- Big drag-drop zone + "Try sample dataset" button

Screen 2: Import Wizard (Upload + Preview + Parse)

This is where win users. Make it feel *magical*.

Step A: Upload

Components

- Drag & drop zone
- File list (supports multiple)
- Toggle: “Auto-detect multiline stack traces”
- Toggle: “Mask sensitive data (PII)”

Progress UI

- Chunk progress bar per file
- Cancel / pause
- “Parsing starts after first chunk” (so results appear fast)

Step B: Preview & Auto-Parse

Split view

- Left: raw lines (with line numbers)
- Right: parsed fields inspector

Auto-detect panel

- Detected: timestamp format, delimiter, JSON/Plain
- Confidence meter (ex: 82%)
- Suggested fields: timestamp, level, service, message, trace_id, request_id

User controls

- Dropdown: “Log format” (Auto / JSON / Regex / KeyValue)
- Timestamp picker (click a timestamp sample → choose format)
- Multiline stitching rules preview

Step C: Parser Studio (Inline)

If detection fails, user can fix it without leaving.

- Regex builder / grok-like inputs
- Live preview (“10 matched, 2 unmatched”)
- Field mapping table:
    - name, type, example, “index as facet” checkbox
- Save parser as template

Step D: Import Summary

- “You’re ready” card
- Buttons: **Go to Explore**, “Create dashboard”, “Run AI summary”

Screen 3: Explore (Logs Search + Filter + Drill-down)

This is the main daily workspace.

Layout (classic observability layout)

Top row

- Search bar (supports natural language + query)
- Filters chips (service, level, host, env…)
- Buttons: Save view, Export, Share

Middle

- Time histogram (log volume / error volume toggle)
- Brush-to-zoom timeline interaction

Main body (3 columns)

1. **Field sidebar**
    - Search fields
    - Top values per field with counts (click to filter)
    - Pin fields (favorites)
2. **Log table**
    - Virtualized list (fast scrolling)
    - Columns: time, level (colored pill), service, message (truncated), tags
    - Expand row → full log JSON / multiline stack trace
    - Actions: “Add as filter”, “Find similar”, “Copy”, “Open in AI”
3. **Context panel**
    - When a row selected: “Surrounding logs” (±N lines / ±time)
    - “Related fields” correlations (request_id, trace_id)

Must-have interactions

- Click field value → “Include / Exclude”
- Hover on histogram spike → quick insight tooltip (“errors spiked”)
- “Find similar” → groups by template/cluster
- “New since last hour” quick filter
- Bookmark query (Saved Views)

States

- No results: show hints + suggested queries
- Parsing incomplete: show streaming import progress ribbon

Screen 4: Dashboards (BI)

Make dashboards feel instant after import.

Dashboard home

- Prebuilt dashboards:
    - Overview
    - Errors
    - Services
    - Security (optional)
- “Create dashboard” button

Dashboard view layout

- Grid layout with resizable cards:
    - Line chart: logs/sec
    - Bar: top services / hosts
    - Table: top error templates
    - Heatmap: logs by hour/day
    - Pie: levels distribution
- Each widget has:
    - Title
    - Filter scope
    - “View logs” drilldown (click point → opens Explore with filters)

Dashboard builder (no-code)

- Choose metric: count, unique count, percent, p95 if duration field exists
- Group by: any field
- Time bucket: auto / 1m / 5m / 1h
- “Save as widget”

Screen 5: AI Insights (the “wow”)

Two tabs:

Tab 1: “Summary”

- Auto-generated report cards:
    - “Top new error patterns”
    - “Anomaly detected at 14:32”
    - “Most impacted service”
- Each card has:
    - explanation
    - evidence chips (click opens query in Explore)
    - “Create alert from this”

Tab 2: “Assistant”

Chat layout, but grounded in the dataset.

Chat input helpers

- /commands: /summarize, /anomalies, /rootcause, /compare
- Buttons: “Use last selected logs”, “Use current filters”

Response format

- Answer + confidence label
- Evidence section:
    - queries run
    - top log snippets (clickable)
- Suggested follow-ups:
    - “Show me when it started”
    - “Compare to yesterday”

Screen 6: Alerts

Even if don’t implement backend alerts yet, design it now.

Alerts list

- Alert name, status, last triggered, severity, dataset scope
- “Create alert” CTA

Create alert wizard

- Condition builder:
    - “When count of logs where … exceeds …”
    - filters builder (same as Explore)
- Schedule: every 1m/5m
- Notifications: email/webhook placeholders
- Preview: “Would have fired 3 times last 24h”

Screen 7: Settings

- Workspace
- Data retention (even if local)
- PII masking rules
- Parser templates library
- Export settings

UI Components library (design once, reuse everywhere)

Core

- Search bar with query tokens + NL prompt
- Filter chips (include/exclude)
- Facet list with counts
- Virtualized log list row + expand drawer
- Timeline histogram with brush zoom
- “Evidence chip” component (AI → Explore)
- Empty state templates (helpful hints)

Quality-of-life

- Keyboard shortcuts: / focuses search, e expand row, s save view
- Command palette (⌘K): “Go to Explore”, “New Dashboard”, “Create Alert”
- Toasts for saved view/export/copy

Visual style guide (simple & observability-friendly)

- Dark mode first (logs look better), with light mode option
- Level colors: INFO neutral, WARN amber, ERROR red, DEBUG subtle
- Monospace for log message area only; keep rest sans-serif
- Use density toggle: Comfortable / Compact (logs can be huge)

What to implement first (Front-end MVP)

MVP UI that still feels complete:

1. Projects + Import wizard (with preview)
2. Explore (search, histogram, fields sidebar, expand row, saved views)
3. Dashboards (auto generated + drilldown)
4. AI Assistant tab (chat + evidence links)