AI · TEMPLATE

Proving Your AI Features Work: Auditability for LLM Systems

KAANSYSTEMS.COM/LIBRARY/LLM-FEATURE-AUDITABILITY · SEPTEMBER 2, 2026

ABOUT THIS TEMPLATE

Auditors and enterprise buyers now ask for proof your AI review process works. Evidence that answers them without turning your logs into a PHI store.

THE TEMPLATE

The map ties each question you'll be asked to the artifact that answers it. Anywhere you can't fill in the source column for your own stack is the gap to close before the next audit cycle.

QUESTION ASKED	EVIDENCE ARTIFACT	SOURCE	RETENTION
What data did the model see for request X?	Input hashes + source-system names in trace; hash-match against system of record	Trace store	7 years
Who approved output Y before it took effect?	Review record (reviewer from IdP, decision, timestamp) joined on trace id	Review table	7 years
What model/version/prompt was live on date D?	Per-request model snapshot id + template SHA; template content at that SHA	Trace store + prompt repo	7 years / life of repo
Who changed the prompt, and who approved it?	Merged-PR review metadata on the template file (persists with the repo)	Prompt repo	7 years
How do you know quality isn't degrading?	Scheduled eval scores keyed by template SHA + model id	Eval results table in evidence store (CI produces the records)	3 years minimum

QUESTION ASKED	EVIDENCE ARTIFACT	SOURCE	RETENTION
What gates an AI change before release?	Eval threshold config + a failed-promotion example	CI pipeline config; promotion records in evidence store	3 years
What happened during the incident on date D?	Traces filtered to window: volumes, flags, versions, linked overrides	Evidence store query	7 years
Is human review actually happening?	Coverage query: consequential outputs with no linked review = 0 rows	Trace ⋈ review tables	Run live for the assessor
Do you train on customer data?	Vendor agreement no-training clause + sub-processor listing	Contract store	Contract term + 6 years

Two retention traps hide in that source column. GitHub's native audit log holds events for months, not years, so it cannot carry a multi-year retention claim on its own: lean on merged-PR review metadata — which persists with the repository indefinitely — as the primary approval evidence, and stream audit-log events into the evidence store (GitHub audit log streaming) if you want them retained alongside it. Same trap with CI: pipeline artifacts expire in weeks, so CI is the producer of eval records, and the eval-results table in the evidence store is where they survive.

The trace record that makes the top rows answerable:

```
{
  "trace_id": "01J9F3T7Q8Z4",
  "tenant_id": "t_4821",
  "timestamp": "2026-09-02T14:31:07Z",
  "feature": "discharge-summary-draft",
  "model": {
    "provider": "anthropic",
    "id": "us.anthropic.claude-sonnet-4-20250514-v1:0",
    "snapshot": "claude-sonnet-4-20250514",
    "endpoint": "bedrock"
  },
  "prompt_template": {
    "id": "pt-a41f9c2",
    "repo": "acme/prompts",
    "committed_at": "2026-08-19T09:12:44Z"
  },
  "params": { "temperature": 0.2, "max_tokens": 1024 },
  "inputs": [
    {
      "name": "patient_record",
      "source": "records-api",
      "source_version": "v_20260902_1425",
      "hmac_sha256": "9f2b8c…",
      "tokens": 1893
    },
    {
      "name": "clinician_note",
      "source": "notes-api",
      "source_version": "v_20260902_1429",
      "hmac_sha256": "b71e04…",
      "tokens": 412
    }
  ],
  "output": {
    "sha256": "e4c1aa…",
    "tokens": 587,
    "finish_reason": "end_turn"
  },
  "latency_ms": 2140,
  "flags": {
    "safety_filter_triggered": false,
    "low_confidence": false,
    "guardrail_blocked": false
  },
  "decision": {
    "type": "human_review",
    "review_id": "rev-88f2",
    "outcome": "approved_with_edits"
```

```
  }
}
```

Every field is an identifier, a version, a count, a flag, or a hash. The model id is the one the platform actually invoked — here a Bedrock cross-region inference profile, which can route inference outside the calling region, which is why the endpoint field makes no region claim — so it joins directly against invocation logs and CloudTrail, exactly the cross-check an auditor runs; the snapshot field carries the normalized name for comparing across platforms. Nothing in the record is customer content, so it's designed to sit outside the PHI boundary and flow to the evidence store in ordinary observability retention — confirm the hashed-trace determination with your privacy officer, and never hash a small-value-space field unkeyed.

HOW TO USE IT

A hash is only evidence while the source is durable

The keyed hash proves provenance only if the bytes it was computed over can be produced again. If your source systems mutate records in place, the hash of March's input won't match September's record and the proof evaporates. Record the source system's own version id next to the hash (as above), and confirm the source's retention meets your evidence retention. Where a source can't version, snapshot the input into the source system's audit trail at request time. That snapshot belongs to the system that already holds the data, not to your trace.

Don't buy the observability platform before the discipline

The LLM-observability market (LangSmith, Langfuse, Braintrust, and a dozen others) defaults to capturing full prompts and completions, because that's what makes their debugging UX good. Pointed at a PHI-bearing feature with defaults on, they recreate the second losing hand as a SaaS sub-processor you now have to disclose. The tools are fine for eval orchestration and for content-off tracing, and self-hosting inside your boundary changes the calculus. But the compliance artifact is the content-free trace flowing to your own store, and that's 200 lines of middleware. Build that first; evaluate vendors after, with content capture treated as a data-flow decision rather than a default.

The evidence is also your operations substrate

Records built for the auditor answer operational questions all week. Token counts by tenant and feature, multiplied by the vendor's price sheet, is your AI cost model. Latency and guardrail-flag rates by template SHA tell you whether Tuesday's prompt change caused Wednesday's pager noise. Review-decision ratios (edit rates creeping up, rejections clustering on one input source) catch quality drift between eval runs. Teams that treat AI auditability as pure compliance overhead build it grudgingly and thin. Treat it as the feature's flight recorder and it earns its keep monthly, with the audit answer as a byproduct.