

AI Production Readiness Review

Feature reviewed: “Ask Docs”, a RAG support assistant · Company: “Lumen Ledger” (B2B SaaS, 9 people)

This is a sample. The company, the product and every number below are fictional. They were written to show the format, depth and tone of a real review. The setup (a hosted LLM API, a vector index in Postgres, a Python API behind a web app) is typical of small teams that have shipped their first AI feature.

Scope	1 AI feature in production: retrieval-augmented answers over ~4,800 help-centre and contract documents
Stack	FastAPI · OpenAI chat and embedding models · Postgres + pgvector · Redis · background workers · AWS ECS · GitHub Actions
Access used	Read-only: cloud console (viewer), logs, repository (read), LLM provider usage dashboard, 7 days of anonymised request logs
Duration	3 working days · kick-off call (30 min) · walkthrough call (45 min)
Changes made	None. Measurements were taken from logs and a replay of 200 anonymised questions against a staging copy.

1. Executive summary

Ask Docs works and customers use it: about 3,100 questions a week, with good answers most of the time. The demo became a product, but the parts that keep a product healthy were never added. The main risks:

- Nobody can see what an answer cost or why it was wrong.** There is no per-request trace (prompt, retrieved chunks, model, tokens, latency). When a customer complains, the team cannot reproduce the answer.
- Changes ship blind.** Prompt and retrieval changes go straight to production with no evaluation set. The last prompt change lowered answer accuracy on contract questions from 81% to 64% (measured in the replay), and nobody noticed for 11 days.
- Slow and fragile under load.** p95 response time is 9.8 s. There are no timeouts, retries or fallback model, so a provider slowdown becomes an outage.
- Spend is growing faster than usage.** The LLM bill grew 2.4× in 3 months while questions grew 1.5×. About 45% of the spend can be removed without changing answers.

Estimated impact of the 30-day plan: p95 under 4 s, about 45% lower LLM spend (≈ \$1,150/month at current volume), and every answer traceable and covered by an automatic quality check before release.

2. Scorecard

Each area is scored 0–5 against what is reasonable for a small team running one AI feature in production. 0 = missing, 3 = adequate with gaps, 5 = solid and tested.

Area	Score	Summary
Observability & tracing	1	Only HTTP logs; no prompt, retrieval or token-level traces
Quality & evaluation	1	No evaluation set; changes checked by hand on a few questions
Latency & reliability	2	p95 9.8 s; no timeouts, retries or fallback; no streaming
Cost control	2	Oversized context, no caching, premium model for every step
Data, privacy & safety	2	Customer names and emails sent to the provider; no prompt-injection handling
Retrieval pipeline	3	Sound design; re-indexing is manual and not versioned
Delivery & operations	3	CI/CD for code is good; prompts and model versions are not part of it
Overall	2.0	Valuable feature, run without instruments

What is already good

- Clean separation between retrieval, prompt building and generation, so most fixes are local changes.
- Answers cite their source documents, and users can see where an answer came from.
- Container builds, tests and deployments run through GitHub Actions; rollbacks of application code are one click.
- Provider API keys are stored in the cloud secret manager, not in the repository.

3. Findings

Ordered by risk. Effort is an estimate for someone familiar with the stack. “Evidence” describes what was measured, so your team can repeat it.

CRITICAL F1 · Prompt changes are released without a quality check

Evidence	Prompts live in code and ship with every deploy. There is no evaluation set. Replaying 200 real (anonymised) questions shows accuracy on contract questions dropped from 81% to 64% after the change on 2 September.
Risk	Quality can fall sharply with no signal. Customers find regressions before the team does.
Fix	Build a 150–200 question golden set from real traffic, with expected sources and short reference answers. Score faithfulness, answer relevance and retrieval hit rate in CI; block the release when the score drops by more than an agreed margin.
Effort	1.5 days, then a few minutes per release

HIGH F2 · No per-request tracing

Evidence	Logs record status code and total duration only. The prompt, retrieved chunks, model version, token counts and per-step timings are not stored anywhere.
Risk	Bad answers and cost spikes cannot be explained or reproduced. Support escalations take hours.
Fix	Add tracing with an LLM observability tool (self-hosted or managed), linked to the existing request ID. Mask personal data before storing. Dashboard: p50/p95 per step, tokens and cost per request, thumbs-down rate.
Effort	1 day

HIGH F3 · A provider slowdown becomes an outage

Evidence	LLM calls have no timeout and no retry policy. During a provider incident on 23 September, 38% of requests failed or exceeded 60 s. There is no fallback model.
Fix	Set timeouts per step, retry with backoff on 429/5xx only, add a fallback model (another provider or a smaller model) behind a simple gateway, and show a graceful message instead of an error.
Effort	0.5–1 day

HIGH F4 · p95 latency is 9.8 seconds

Evidence	Per-step timing from the replay: query rewrite 1.6 s (premium model), retrieval 0.4 s, reranking 0.3 s, generation 6.9 s with 14 chunks (~11k tokens) of context. Answers are not streamed.
Fix	Use a small model for query rewriting, cut context to the top 5–6 chunks after reranking, and stream tokens to the UI. Expected p95: about 3.5–4 s; first token under 1.5 s.
Effort	1 day

HIGH F5 · Personal data is sent to the LLM provider and documents can carry instructions

Evidence	Questions often include customer names, emails and invoice numbers, and these are sent unmasked. Uploaded contract documents are indexed as-is; a test document containing “ignore previous instructions” changed the assistant’s behaviour.
Fix	Mask personal data before the LLM call and restore it in the answer. Treat retrieved text as data (clear delimiters, an instruction-hierarchy prompt, an output check for policy breaks). Confirm the provider’s data-retention settings and that a DPA is in place.
Effort	1–1.5 days

MEDIUM F6 · About 45% of LLM spend can be removed

Evidence	Monthly LLM spend ≈ \$2,550. Breakdown: oversized context 31%, query rewriting on a premium model 12%, repeated identical questions 9% (no cache).
Fix	Smaller context (see F4), a small model for rewriting and classification, a semantic cache for repeated questions, and provider prompt caching for the fixed system prompt. Track cost per answer on the dashboard from F2.
Effort	1 day · Saving ≈ \$1,150/month

MEDIUM F7 · Re-indexing is manual and not versioned

Evidence	Documents are re-embedded by running a script by hand. 212 help articles changed since the last run. Index and embedding-model versions are not recorded, so there is no way back from a bad re-index.
Fix	Incremental indexing triggered by document changes; store embedding-model version with each vector; build a new index next to the old one and switch only after the evaluation set passes.
Effort	1 day

MEDIUM F8 · Rate limits hit at peak hours

Evidence	2.1% of requests between 14:00 and 16:00 UTC received HTTP 429 from the provider. No queueing or per-tenant limits exist, so one large customer can use up the whole quota.
Fix	Per-tenant rate limits, a short queue with backpressure, and a request for a higher provider tier backed by the usage data from F2.
Effort	0.5 day

LOW F9 · Self-hosting an open model is not worth it yet

Evidence	At ~3,100 questions/week, one dedicated GPU instance running an open-weight model with a serving engine such as vLLM would cost more per month than the optimised API spend after F6, before engineering time.
Fix	Revisit when weekly volume passes roughly 15–20k questions, or if data-residency requirements demand it. Keep the gateway from F3 so the switch stays cheap.
Effort	None now

4. 30-day plan

When	Work	Findings	Effort
Week 1	Tracing and cost dashboard; timeouts, retries and fallback model; mask personal data	F2, F3, F5	~2.5 days
Week 2	Golden evaluation set and quality gate in CI	F1	~1.5 days
Week 3	Latency and cost work: smaller context, small model for rewriting, streaming, caching	F4, F6	~2 days
Week 4	Versioned incremental indexing; per-tenant rate limits; one-page runbook for provider incidents	F7, F8	~1.5 days

Your team can run this plan using the report alone. If you prefer, the same work can be done as a fixed-price follow-up, or covered by a monthly AI Ops care plan (dashboard review, evaluation runs before releases, cost tracking and small fixes).

5. What was checked

Observability & tracing

- Per-request traces: prompt, context, model, tokens, timings
- User feedback capture and review loop
- Alerts on error rate, latency and cost

Quality & evaluation

- Golden set, metrics and release gate
- Retrieval hit rate and citation accuracy
- Regression history after prompt and model changes

Latency & reliability

- Per-step timing, context size, streaming
- Timeouts, retries, fallback, provider incidents
- Rate limits and queueing

Cost control

- Spend by step and by tenant; model choice per step
- Caching (semantic and provider prompt caching)
- Self-hosting break-even

Data, privacy & safety

- Personal data in prompts and logs; provider retention
- Prompt injection through retrieved content
- Secrets and access to provider accounts

Retrieval pipeline & delivery

- Chunking, embedding and index versioning
- Indexing triggers and freshness
- Prompts and model versions in CI/CD

6. How a review works

Step	What happens
Kick-off (30 min)	We agree on the feature in scope and you grant read-only access.
Review (3 days)	Logs, configuration and code are reviewed, and anonymised questions are replayed against staging. No changes are made to production.
Report	A report like this one: scorecard, findings with evidence, measured latency and cost, and a 30-day plan.
Walkthrough (45 min)	We go through the findings together and answer questions from your team.
Price	\$900 fixed for one AI feature.

Prepared by **Taha Bagatur**, AI infrastructure & DevOps engineer. Runs LLM and computer-vision workloads in production for a large retail group: GPU inference servers, RAG services, an edge fleet of 50+ devices, monitoring and drift detection · taha@tahabagatur.com · github.com/mtbagatur
Sample report: the company, the product and all figures are fictional.