SYSTEM CARD · COMPOUND AI SYSTEM

MAGIK

Multimodal Agentic RAG Integrated Knowledge Assistant

A finance-domain retrieval-augmented generation system that ingests text, PDF, DOCX, XLSX, images, audio and video; routes each query through an agentic controller; retrieves with a hybrid lexical and dense pipeline; verifies its own answers against the retrieved evidence before they reach the user; and runs entirely on self-hosted open-weight models — with no third-party language-model API anywhere in the request path. This edition goes deeper than a typical system card into authentication, tenant isolation, observability and AWS infrastructure, and discloses three real operational and security incidents from the current release.

MODALITIES	MODELS COMPOSED	CI MERGE GATES	AWS ACCOUNTS TO DATE
7 pipelines, fully isolated	18 open-weight ~42 GB · 0 fine-tuned	3 enforced retrieval · hallucination · finance	3 · each migration forced by GPU quota

Release **v1.0.2** · 8 September 2026
Compiled 12 September 2026 · MIT License

Author **Vijaya Karthik**

Contents

How to read this card. Every figure is drawn from the repository itself — the model manifest, the evaluation threshold file, the Terraform modules, the monitoring configuration and the test tree — rather than from summary documentation. Open defects, a currently-failing quality gate, and real incidents (including a security one) are included in Sections 06, 12, 14 and 15 rather than omitted.

01 Overview

MAGIK is built around one premise: most retrieval-augmented generation demonstrations fall apart under the conditions real traffic creates — malformed documents, adversarial input, numbers silently rounded in transit, one user's session bleeding into another's context. This system is an attempt to hold up under those conditions, and to show it with a measured evaluation harness rather than a claim.

It specialises in **financial documents** because finance is an unforgiving domain for retrieval-augmented generation: a hallucinated number is worse than a hallucinated sentence. Financial figures in a generated answer are checked against the literal text of the retrieved chunks, and this is gated in continuous integration, not just measured.

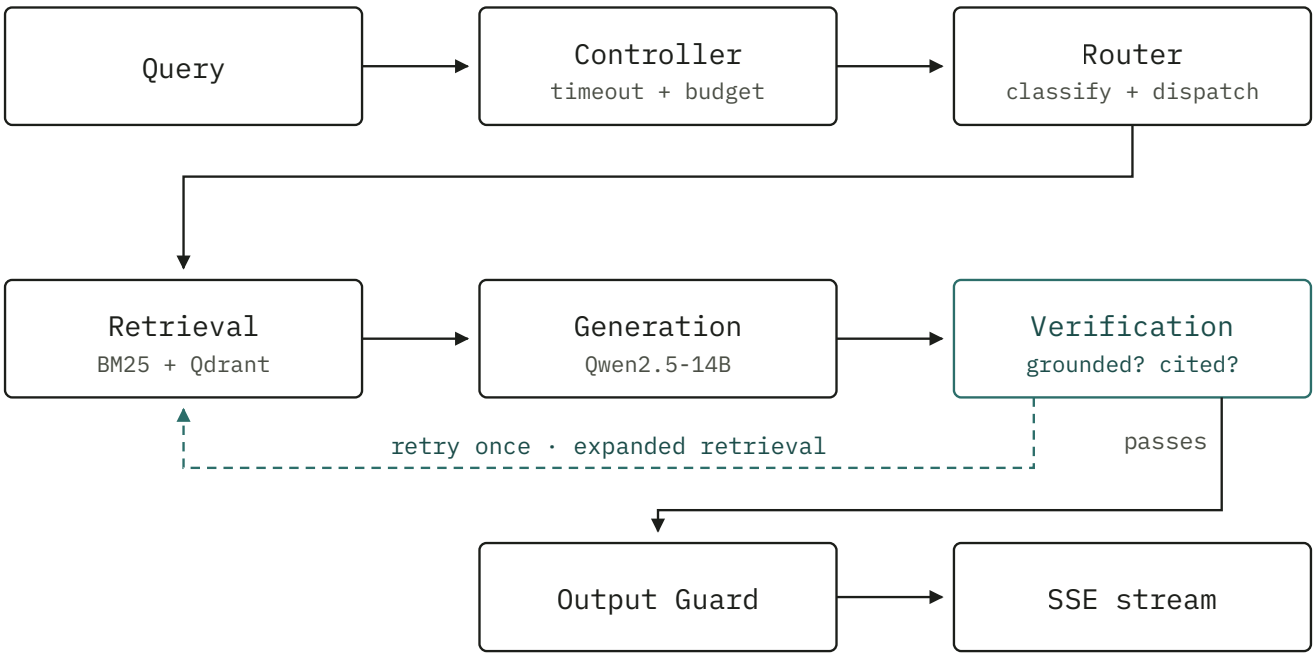
Every model in the stack is open-weight and self-hosted: a quantised GGUF language model served through `llama.cpp`, open embedding and reranking models, and open speech, OCR and vision models. There is no commercial inference API in the request path.

MODALITIES INGESTED	MODELS COMPOSED	TEST FILES	AWS ACCOUNTS TO DATE
7 — text, PDF, DOCX, XLSX, image, audio, video	18 · ~42 GB, all open-weight	143 across 8 categories	3, each forced by GPU quota

02 Architecture

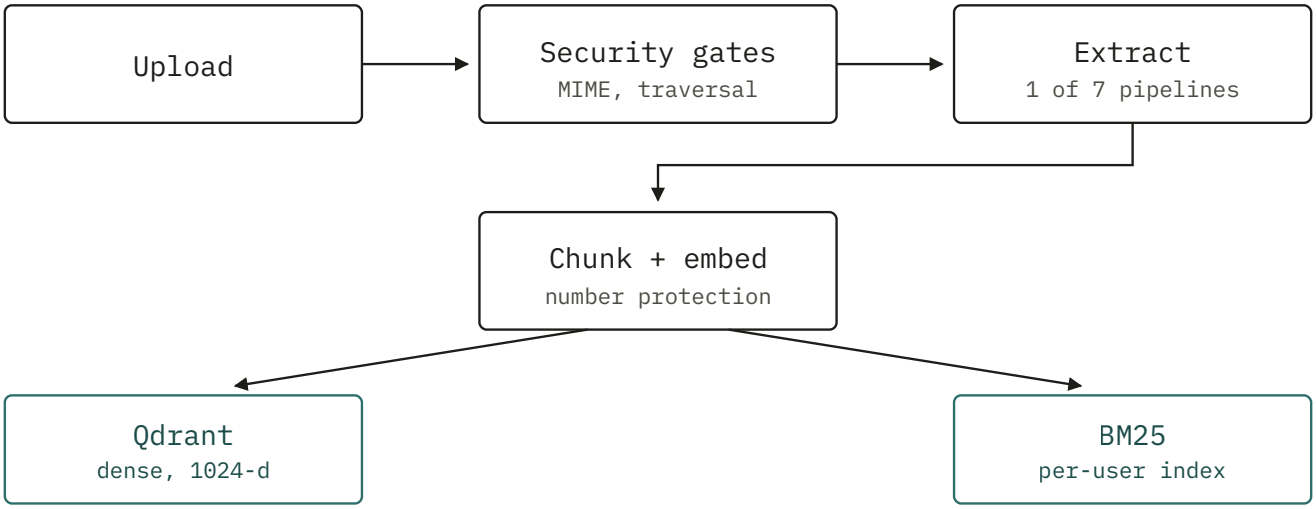
Two independent flows: a query path that answers a question, and an ingestion path that indexes a document. The query path runs under an enforced timeout and token budget; the ingestion path runs security gates before any file content is trusted.

Query flow



The router classifies each query once and dispatches a single tool call — a bounded dispatch, not an open-ended agent loop. A failed groundedness or citation check sends the query back through retrieval once with an expanded strategy, bounded by a hard timeout. Measured retry behaviour is reported in Section 07.

Ingestion flow



Every ingested file is written to both stores. Uploaded originals and the per-user BM25 index land on the instance's local disk — see Section 14 for why that single-copy design already caused one real data-loss incident.

Per-modality isolation

Every one of the seven modalities owns exactly four files — one per processing layer (ingestion, chunking, embedding, lexical indexing) — with no shared per-modality state. This is also what makes the guardrail coverage claim in Section 08 countable: seven modalities across four layers gives 28 text surfaces, each of which must call the single sanitisation entry point.

03 Component models

MAGIK is a compound system rather than a single trained model. Eighteen open-weight checkpoints — roughly 42 GB on disk — are composed through the pipeline above. Each is pinned to an exact upstream commit hash, and the provisioning manifest records a SHA-256 on first download and re-verifies it on every subsequent run.

Provisioned model manifest · all open-weight, none fine-tuned

ROLE	CHECKPOINT	SIZE
Generation (LLM)	Qwen2.5-14B-Instruct · Q4_K_M GGUF	9.00 GB
Vision — charts & images	Qwen/Qwen2-VL-7B-Instruct	16.59 GB
Vision — video frames	Qwen/Qwen2-VL-2B-Instruct	2.20 GB
Evaluation judge	Qwen2.5-7B-Instruct · Q4_K_M GGUF	4.70 GB
Cross-modal embedding	google/siglip-so400m-patch14-384 · 1152-d	1.76 GB
Speech recognition	Systran/faster-whisper-large-v3	1.55 GB
Text embedding	BAAI/bge-large-en-v1.5 · 1024-d	1.35 GB
Reranking	BAAI/bge-reranker-large · cross-encoder	1.34 GB
Image captioning	Salesforce/blip-image-captioning-large	0.90 GB
Speaker diarization	pyannote/speaker-diarization-3.1	0.60 GB
Financial sentiment	yiyanghkust/finbert-tone	0.44 GB
Named-entity recognition	dslim/bert-base-NER	0.43 GB
Toxicity screening	Detoxify (original)	0.42 GB
OCR — printed text	microsoft/trocr-large-printed	0.36 GB
Diarization — segmentation	pyannote/segmentation-3.0	0.20 GB
Diarization — embedding	pyannote/wespeaker-voxceleb-resnet34-LM	0.10 GB
Keyword extraction	all-MiniLM-L6-v2 (KeyBERT)	0.09 GB
OCR — scene text	EasyOCR (en)	0.06 GB

04 Intended use

In scope

- Question-answering over finance documents a user has uploaded.
- Demonstrating production-oriented RAG engineering: hybrid retrieval, answer verification, guardrail coverage, tenant isolation, observability and CI-gated evaluation.
- Single-GPU, low-concurrency deployment.

Out of scope

- Not financial advice.
- Not evaluated on non-English documents.
- Not load-tested at commercial multi-tenant scale.
- The public demonstration runs a deliberately open shared account.

05 Training & provenance

No model in this system has been fine-tuned. All eighteen checkpoints run at their published open weights — no LoRA, no PEFT, no custom training run anywhere in the stack. The engineering investment goes into retrieval quality, answer verification, guardrail coverage, access control and evaluation rigour around off-the-shelf models.

Text splitting uses `langchain-core` and `langchain-text-splitters` only — the agent's tool routing is a hand-built typed dispatch, not a LangChain agent or chain.

06 Evaluation

Eleven suites are runnable from the harness. Three are enforced as hard merge gates: **retrieval**, **hallucination** and **finance numeric fidelity**. Everything else is informational.

Measurement methodology. Three back-to-back runs on identical code once produced faithfulness scores of 0.363, 0.592 and 0.589. The outlier was the first run after a server restart, not judge stochasticity. Process rule since: discard the first run after any restart, average at least three.

Where a confirmation run surfaced a genuinely worse value, the gate is set from the *observed maximum*, not the average.

Retrieval — enforced gate

retrieval suite · n = 56 · CI-gated

METRIC	V5 BASELINE	CI FLOOR	LATEST	STATUS
recall@5	0.5089	0.4835	0.4464	breach
MRR	0.3558	0.3380	0.3069	breach
nDCG@10	0.4024	0.3823	0.3642	breach
recall@10	0.5536	0.5259	0.5625	pass
hit rate	0.6786	0.6447	0.8036	pass
context precision	0.0268	0.0255	0.0321	pass

OPEN, STILL UNATTRIBUTED AS OF THIS RELEASE

3 of 6 gated metrics are breached while 3 improved — coverage improved while ordering degraded. Two candidate causes are under attribution: a metadata-backfill fusion change, or a baseline/staging provenance mismatch. No re-measurement has happened since v1.0.1 because staging itself does not currently exist (Section 12) — attribution is blocked on the same GPU quota increase that blocks staging's rebuild.

Hallucination — enforced gate

hallucination suite · n = 97, all 7 modalities · CI-gated

METRIC	V7	V8	GATE	MEANING
fabrication rate	0.0653	0.0619	max 0.079	Primary safety signal
hallucination rate	0.2715	0.2234	max 0.246	Blended metric
omission rate	0.2302	0.1822	—	Completeness signal, no gate

The fabrication gate is set at 10% above the observed maximum across 4 runs (0.0722). A 4th confirmation run surfaced a real, intermittent hallucination three identical runs had missed — root-caused, not a metric artifact.

Finance numeric fidelity — enforced gate

Financial figures cited in an answer are matched against the literal text of retrieved chunks within a 0.5% tolerance and no unit-scale bridging. Gated at a minimum of 0.95.

Generation and routing — informational

generation suite · n = 42 · **informational**

METRIC	VALUE	METRIC	VALUE
Answer correctness	0.7083	Answer relevancy	0.6528
Context recall	0.8962	Faithfulness	0.5146
Finance fidelity	0.8266	Route accuracy	1.000

Per-modality scorecard — single-run snapshot (2026-08-20)

generation quality · n = 14 / modality

MODALITY	FAITHFULNESS	ANSWER RELEVANCY	ANSWER CORRECTNESS	CONTEXT RECALL
Text	0.4545	0.7045	0.6591	1.0000
PDF	0.5714	0.7679	0.8571	0.2846
DOCX	0.6429	0.8077	0.7857	0.7667
XLSX	0.6154	0.7143	0.7321	0.7917
Image	0.8036	0.8393	0.8929	1.0000
Audio	0.3929	0.6429	0.6786	0.8000
Video	0.3929	0.8214	0.7857	0.8738

hallucination & safety · n = 14 / modality

MODALITY	HALLUC. RATE	FABRIC. RATE	OMISSION RATE	TEMPLATE LEAK	CITATION ACC.
Text	0.0909	0.0909	0.0909	0.0000	1.0000
PDF	0.5000	0.1429	0.3571	0.0000	N/A*
DOCX	0.4286	0.0714	0.3571	0.0000	1.0000
XLSX	0.1429	0.0000	0.1429	0.0000	1.0000
Image	0.0714	0.0714	0.0000	0.0000	1.0000
Audio	0.1429	0.0714	0.1429	0.0000	1.0000
Video	0.2857	0.0000	0.2857	0.0000	1.0000

* PDF's heuristic citation-accuracy metric had zero measurable rows this run — excluded rather than shown as a false 0 or 100.

verification loop · n = 14 / modality

MODALITY	GROUNDING	CITATION V2	RETRY SUCCESS	AVG RETRIES
Text	1.0000	0.8889	0.5000	0.2222
PDF	0.9286	0.7857	0.3333	0.4286
DOCX	1.0000	0.6923	0.0000	0.1538
XLSX	0.9167	1.0000	0.0000	0.0833
Image	0.9286	0.9286	0.0000	0.0714
Audio	0.7692	0.5385	0.2727	0.9231

MODALITY	GROUNDING	CITATION V2	RETRY SUCCESS	AVG RETRIES
Video	1.0000	0.9286	0.0000	0.3571

finance fidelity & latency · n = 14 / modality

MODALITY	FIN. FIDELITY	VER. P50	VER. P95	GEN. P50	GEN. P95	GEN. P99
Text	0.9091	4.01s	11.62s	11.81s	34.07s	44.15s
PDF	0.8433	7.13s	13.95s	9.70s	19.36s	20.52s
DOCX	0.8125	3.91s	9.02s	6.69s	10.91s	11.53s
XLSX	0.9286	4.37s	9.18s	10.20s	14.86s	16.65s
Image	0.9929	3.14s	4.89s	4.33s	6.05s	6.54s
Audio	0.6310	9.21s	15.34s	12.89s	19.29s	22.57s
Video	0.9643	3.97s	6.92s	8.57s	11.03s	11.19s

- **Image is the strongest modality across the board.**
- **Faithfulness is the weakest quality axis system-wide** (0.39 to 0.80).
- **Audio needs real improvement** — weakest citation grounding (0.5385), lowest finance fidelity (0.6310).
- **PDF and DOCX show elevated hallucination rates** (50% and 43%) despite strong correctness.
- **Text has unexplained tail-latency spikes** (p95 34.07s, p99 44.15s) despite being structurally simplest.

07 Answer verification

Before an answer reaches the user, a verification loop checks whether its claims are supported by retrieved context and whether its citations point at real chunks, retrying once with an expanded retrieval strategy on failure.

GROUNDING SUCCESS	CITATION ACCURACY	VERIFICATION LATENCY	MEAN RETRIES / QUERY
0.9384	0.8587	p50 2.44s p95 6.81s	0.2898

V7 TO V8 — WHAT WIRING THE LOOP PROPERLY BOUGHT

Grounding success rose 0.8587 to 0.9384; mean retries per query fell 0.50 to 0.29 while retry success rose 0.071 to 0.186; verification latency roughly halved (p50 4.42s to 2.44s).

Across 140 retried sessions, 88.6% changed nothing, 8.6% raised the score, 1.4% flipped a failing answer to a passing one — a safety net that rarely fires usefully, not a general quality multiplier.

A bug this instrumentation caught. Plain-text documents never triggered the verification loop at all, because ingestion tagged chunks "text" while chunking used the canonical "txt". Fixed by aliasing the two tags.

08 Guardrails

All ingested content is treated as untrusted data, never as instructions. A single entry point is called on every one of the 28 modality-by-layer text surfaces before that text reaches a model.

ATTACK RECALL	FALSE POSITIVES	DETECTION PATTERNS	OWASP LLM TOP 10
64 / 64 (100%)	0.9% · F1 0.994	43, severity-tiered	10 / 10 addressed

Measured against a 109-case red-team corpus. 306 guardrail test cases cover the same surfaces — also what the Auth failure and Guardrail block alerts in Section 11 watch in real time.

09 Authentication & session lifecycle

Authentication is a full implementation, built request-flow by request-flow. Every default below is the code's actual default in `app/core/config.py`.

Password login

`POST /auth/login` verifies the password against an Argon2 hash (time_cost=3, memory_cost=65536 KiB — 64 MiB per verification), with bcrypt as a fallback verifier for legacy hashes.

TOKEN	DEFAULT LIFETIME	CONFIG KEY
Access token	30 minutes	ACCESS_TOKEN_EXPIRE_MINUTES
Refresh token	7 days	REFRESH_TOKEN_EXPIRE_DAYS

Multi-factor authentication (TOTP)

Enrollment generates a standard RFC 6238 TOTP secret plus 8 single-use backup codes, each a 10-character hex string stored only as a bcrypt hash. Verification allows a ±30 second clock-skew window. Login with MFA enabled is two-step: a correct password issues a short-lived MFA token, not the real access token; the client then submits a TOTP or backup code, which only then issues the real JWT pair.

Google OAuth 2.0

Authorization-code flow with CSRF protection via a server-held state parameter — not PKCE. The state store is a bounded in-memory dictionary (capped at 500 entries).

Password reset (OTP)

A 6-digit OTP is stored in Redis with a 10-minute TTL and emailed. 3 wrong guesses trigger a 15-minute lockout. A successful verification issues a one-time reset token (1-hour TTL) consumed on the actual password change.

Logout & revocation

JWTs are stateless, which normally makes logout theater. On logout or password change, the token's jti is written to Redis with a TTL set to the token's own remaining lifetime — the blacklist entry expires at the same moment the token would have anyway, and never grows unbounded.

Rate limiting

Per-user, fixed 60-second window, independent of the OTP-lockout mechanism above.

10 Multi-tenant isolation

Every data layer filters on the user identifier independently — defense-in-depth: a bug in the API's authorization check does not, by itself, leak data, because the storage layer underneath enforces its own filter regardless of what called it.

DATA LAYER	ISOLATION MECHANISM	FAILURE MODE IF MISSING
Qdrant	Typed field condition on user_id per query	Cross-tenant semantic search
BM25	Per-user index file path	Cross-tenant keyword search
Redis	Namespaced keys under user:{id}:*	Session/conversation bleed
MongoDB	Every query filters on user_id	Reading another's transcript

The public demo is a single shared tenant, not a bypass of isolation. What is not isolated is durability — the demo tenant's uploaded files live only on local disk, which is why demo access can delete the corpus (Section 14). That is a data-durability gap, not a tenant-isolation one.

11 Observability & monitoring

A seven-service stack, defined as code in `docker-compose.monitoring.yml`.

SERVICE	ROLE
Prometheus + Pushgateway	Per-modality counters on a dedicated port; Pushgateway for short-lived batch jobs
OTel Collector	Receives distributed trace spans
Tempo	Stores traces; backs the TraceQL alert below
Grafana	Dashboards + unified alerting, behind Caddy
Loki + Promtail	Log aggregation, correlated by trace ID
Arize Phoenix	LLM observability, bound to 127.0.0.1 only
Uptime Kuma	Passive push from wake/idle-stop Lambdas — never polls (would wake the GPU)

Dashboards

Three, provisioned as JSON: `system_health.json`, `rag_quality.json`, `logs.json`.

Alerting — 12 rules across 3 groups

GROUP	ALERT
magik-system	Circuit breaker OPEN
magik-system	Ingestion error rate spike
magik-system	p95 latency breach (online)
magik-system	GPU VRAM critically low
magik-system	Reranker p95 latency breach
magik-system	App error log rate spike (Loki)
magik-system	Span error rate spike (Tempo/TraceQL)
magik-security	Guardrail block rate spike
magik-security	Auth failure spike (login + MFA)
magik-rag-quality	Online hallucination rate drift

GROUP	ALERT
magik-rag-quality	Production traffic drift — WARNING
magik-rag-quality	Production traffic drift — CRITICAL

Disclosed rather than hidden. The TraceQL span-error-rate rule requires Grafana v12.1+ and its own feature toggle. Grafana itself documents TraceQL alerting as experimental. Included anyway, with that tradeoff accepted explicitly and flagged for live verification after the first real deploy.

12 AWS infrastructure & deployment

Three accounts, one recurring cause

ACCOUNT	ERA	WHY IT ENDED
537557168406	Original	Superseded during v1.0.0 bring-up
857194222592	v1.0.0-v1.0.1	GPU vCPU quota of 4 — no room for prod + staging
266901698137	v1.0.2-current	Live account

Every migration shares the same root cause: a g6e.xlarge needs 4 GPU vCPUs, and a fresh account's default quota is exactly 4 — enough for one box, never two. The previous module is kept only as an as-built record, marked DEPRECATED.md, and is no longer applyable.

The AZ capacity fight (2026-09-08 to 2026-09-12)

Production missed us-east-1a, then 1b, and landed in 1c. Staging then missed every availability zone in the region for three straight days before also landing in 1c. Only the Uptime Kuma box stayed on the original AZ.

Network & access

INSTANCE	INBOUND	RATIONALE
Production	SSH (admin CIDR), HTTP (ACME), HTTPS (app + /grafana/)	Public app, break-glass SSH only
Staging	Zero inbound rules	SSM only — no port to scan, no IP to target

IAM is split by trust boundary: a GitHub OIDC provider lets Actions assume a deploy role with no long-lived credentials in the repo; a separate EC2 role is what the running instances use — CI identity and runtime identity are never the same principal.

Staging to gate to promote pipeline

INSTANCES	STAGING ACCESS	PROMOTION GATE	STAGING UPTIME
2 x g6e.xlarge, identical	SSM only	Full Tier-2 suite	Minutes per deploy

The production image is built exactly once, deployed to staging, and the full Tier-2 suite runs against it. Only if that gate passes does the identical image get promoted to production. A failing gate rolls staging back; production is skipped entirely, since nothing was ever deployed there.

CURRENT STATUS — STAGING DOES NOT EXIST RIGHT NOW

Blocked on a second GPU vCPU quota increase. Between 2026-09-08 and 2026-09-12 the pipeline bridged this by keeping staging on the old account's still-intact box while production ran on the new one. That bridge closed when the old account was decommissioned; the Terraform module already supports recreating staging (create_staging = true) — this is a quota wait, not a design gap.

Production wake / idle-stop

ALWAYS-ON COST	SCALE-TO-ZERO COST	COLD START	TRANSPORT
~$1,340 / month	~$12 / month + hourly	60-90s, first request	HTTPS via Caddy

A bug this separation caught. Staging was once found emitting "env": "production" on every log line — a deploy step layered an environment file that never defined ENV on top of an AMI clone that already had it set.

13 Reproducibility

- **Model weights** — pinned to an exact upstream commit hash, SHA-256 verified.
- **Python dependencies** — fully locked.
- **Infrastructure** — codified in Terraform, including the account migration in Section 12.
- **Vector data** — Qdrant collections can be snapshotted and restored.

This mattered twice: strict manifest checking revealed the 7B vision model had been silently absent from provisioning; and `app/bin/restore_demo_kb.py` exists precisely because uploaded files are not yet part of this pinned set — see Section 14.

14 Incident disclosures

Dated, root-caused, and kept here rather than only in CHANGELOG.md.

2026-09-08 — Knowledge-base wipe during AWS account migration

What happened. Uploaded documents and BM25 indexes live only on an instance's root EBS volume, blank on a newly built box. Migrating production replaced the instance and took the demo corpus with it. Managed stores (Qdrant, MongoDB Atlas, Upstash Redis) were unaffected.

Fix. `restore_demo_kb.py` re-uploads through the normal upload route rather than copying files directly, rebuilding the disk copy, vectors and BM25 index together.

Residual risk. A repair tool, not an architecture fix — the same loss recurs on the next instance replacement (Section 15).

2026-09-08 — Chat transcripts lost a turn on every refused answer

What happened. A refused answer was never written to the transcript, but the frontend patched "the last assistant message" regardless — overwriting the previous turn's real answer while the current question never appeared. An empty knowledge base (the state above) makes nearly every answer a refusal, so one incident silently escalated into a second: history corruption.

Fix. The patch is now turn-aware — a mismatched patch is refused rather than blindly applied, and the missing turn is stored instead of discarded. Covered by a dedicated integrity test.

2026-09-12 — PRIVATE SSH KEY COMMITTED TO A PUBLIC REPOSITORY

What happened. Four Terraform plan files were tracked in this public repo. A .tfplan is a zip whose tfstate members are not redacted the way console output is, so each embedded the full PEM of the RSA-4096 break-glass SSH key for the then-live AWS account.

Why two controls both missed it. The .gitignore pattern matched tfplan3.out but not destroy.tfplan — one wrong glob. CI's detect-secrets gate skips binary content, and a .tfplan is a zip — a blind spot in the tool's design.

Impact, assessed rather than assumed. One RSA key leaked, for an account that no longer exists. No SSM parameter values were exposed, only IAM policy ARNs. The current account generates its own independent key pair — fingerprints compared directly and confirmed different.

Fix, and a deliberate non-fix. Both glob spellings are now excluded everywhere. The files are untracked but not scrubbed from git history — force-rewriting a public repo's history was judged to trade a real disruption for no real security gain, since the key opens nothing that still exists.

15 Limitations & known issues

Documented deliberately rather than left implicit.

- **Three retrieval gate metrics are breached and unattributed** — Section 06. Attribution is blocked on staging's rebuild.
- **Staging does not currently exist in the production AWS account** — Section 12. Blocked on GPU quota.
- **Uploaded files and BM25 indexes have no second copy** — repaired by a restore script, not closed (Section 14).
- **Retrieval context precision is low in absolute terms** (0.027 at baseline).
- **A real, intermittent hallucination remains open** on dense audio transcripts.

- **The hybrid web route does not execute a live web search.**
- **The default generation suite covers only text, PDF and DOCX.**
- **Streaming evaluation is manual, not continuous.**
- **A modality-tagging audit is incomplete** beyond the text-tag fix.
- **Finance numeric fidelity is enforced at merge time, not sampled from live traffic.**
- **No formal fairness or bias audit has been conducted.**
- **Single production instance, no horizontal scaling or failover** — the second GPU instance currently does not exist at all.
- **The TraceQL alert rule runs on functionality Grafana calls experimental** (Section 11).

16 Responsible AI considerations

- **No external inference provider** — document and query content never leaves the deployment boundary.
- **PII detection** via Microsoft Presidio across ingestion and memory surfaces.
- **Toxicity screening** on model output via Detoxify.
- **Untrusted-content discipline** — documents, transcripts and web results are always data, never instructions.
- **Auditability** — guardrail violations and auth/MFA failures are monitored and logged.
- **Answer traceability** — citations trace a cited number back to source text.
- **Incident transparency** — real incidents disclosed with root cause and impact (Section 14).

17 References

Repository	github.com/vjkarthik98/MULTIMODAL-AGENTIC-RAG-INTEGRATED-KNOWLEDGE-AI-ASSISTANT
Release	`v1.0.2 · 8 September 2026`
License	MIT
Author	Vijaya Karthik
Engineering log	Full change history in CHANGELOG.md, including root-caused production incidents
Deprecated AWS module	As-built historical record only — deploy/aws/terraform/DEPRECATED.md

Every figure in this card was read from the repository at release v1.0.2. Figures that could not be verified were omitted rather than estimated. Where documentation disagreed with code, the code was treated as authoritative.