VeriCommerce AI

Engineering release record · 18 October 2026
Author: Zahidul Islam

Latest verified release

Version 0.2.0, release 20261018-public-refresh-02, is live from source 26b955. Newer source a82ab55 is published on the release branch; default main remains 1497c06. At the owner's request, release 03 was stopped before promotion. This overview separates the live application from reviewed source-only changes. Author code and documentation use the MIT license.

Website: https://vericommerce.zahidul-islam.com/

Evidence workspace: https://vericommerce.zahidul-islam.com/engine

Repository: https://github.com/Zahidulislam2222/VeriCommerce-AI

Observed engineering evidence

• Live application source: 26b955. Published newer source: a82ab555a6540963c50037acfdf16a8832eb4846, attributed to Zahidul Islam.

• Hosted quality/security passed on a82ab55: 155 PostgreSQL/CPU/infrastructure checks, 17 Node checks and 102 desktop/mobile browser cases, with zero skips or retries. These runner-local checks do not establish release 03 deployment.

• Source, candidate-image and documentation reviews accepted their recorded scopes. A source-only assistant evaluation passed 38/40 generated tasks and 29/29 critical invariant checks; two broad-query relevance misses remain.

• Live native installation matched all 45 locked Python packages with cryptography 50.0.2. Package-only advisory success does not establish a clean container; remaining OS and installer findings are described in section 9.

• Live release 02: 228 matching local/archive/live artifacts, eight HTTPS routes and four healthy constrained services; the safe-stop recheck passed without restarting them.

• Live release 02 snapshots reproduced 27 SQL table counts/JSON payloads including migration metadata, with restricted-role/RLS probes. Earlier encrypted recovery and switchback exercises remain dated evidence; no release 03 restoration or promotion occurred.

• Live release 02 public browser run: 46 passed, zero skips, flakes, failures or retries. Original three approved drafts, volumes, private runtime files and neighboring gateway files were unchanged at the stop; final release 03 public QA was not run.

Current scope and next gates

Live capabilities include persistent account workspaces and four roles, bounded JSON/CSV/PDF/static-image ingestion, current-source retrieval/context inspection, cited quotations, temporal review, durable jobs, explicit approval and provenance export. Shopify integration is read-products-only. Publishing and model training remain future work. New question-history/cancellation assistance is reviewed source, not the live interface.

The 1M+ active-user architecture and 99% rolling-30-day availability objective are future targets. They are supported by a staged capacity model, operational definitions and validation gates; they are not current load or uptime measurements. The following technical overview preserves the project's architecture, public legal/security information, operational limits and ten-phase roadmap.

VeriCommerce AI

Evidence-grounded commerce intelligence

Comprehensive Technical Overview · Zahidul Islam · 18 October 2026

1. Purpose, product and engineering direction

VeriCommerce AI turns product information into a system of traceable evidence. Its purpose is to help people understand what a catalog actually documents, find the passage behind an answer, review content against current sources, and retain the provenance of an approved output. The project is developed and operated by Zahidul Islam.

Commerce information changes across product variants, manuals, policies and catalog updates. A fluent description can still contain the wrong specification. A citation can point to an old revision. A draft can become outdated after someone reviews it. VeriCommerce AI addresses these problems by making source identity, version and approval state part of the application rather than treating them as optional additions to generated text.

The live system ingests structured catalogs and bounded documents/images, stores immutable source revisions, inspects keyword/dense/hybrid/rerank context, returns cited quotations, creates evidence-based draft jobs, checks current-source approval and exports approved content. Accounts, role permissions, temporal relationships and read-only Shopify synchronization extend the same evidence boundary.

The interface combines an editorial public website with a server-backed evidence workspace and a separate browser-storage workspace. Product imagery and starting catalog records are generated test material. They do not represent real customer deployments, orders, endorsements or independently verified product properties.

Temporary cookie-bound workspaces have a configured 12-hour lifetime. Persistent accounts use separate expiring sessions and owner/editor/reviewer/viewer memberships; workspace data survives logout. Shopify authorization and bounded document/image uploads are integrated. Password recovery, administrator MFA, external publishing and training remain absent. Use permitted non-sensitive catalog data only.

The long-term engineering direction is a reusable evidence platform with commerce as its first application. Retrieval, model engineering and guarded publishing should earn their place through measured outcomes. The roadmap does not require every AI framework in every request. It requires reproducible datasets, trustworthy boundaries, comparisons against baselines and release evidence that a reviewer can assess.

Website: https://vericommerce.zahidul-islam.com/

Evidence workspace: https://vericommerce.zahidul-islam.com/engine

Repository: https://github.com/Zahidulislam2222/VeriCommerce-AI

2. User journeys and the evidence lifecycle

A user begins with a generated catalog or permitted non-sensitive records. Each product has an identifier and descriptive fields; each claim identifies a source, passage and page. A source record has metadata and a version. The application validates references before a durable ingestion job applies the supplied changes.

Imports are incremental. Supplying a record updates it; omitting it does not delete it. An identical replay preserves the existing revision. Changed source content creates an immutable new revision and keeps the relationship to the preceding version. This distinction makes retries safe while retaining evidence of what changed.

A source passage stays connected to its product and revision. Search operates on current evidence in the user's workspace. Answers quote retrieved text and identify the source version and page. A source-inspection dialog lets the user inspect the supporting passage without losing the surrounding review context.

Draft creation is asynchronous. The API records a job, the worker claims it, and the user can inspect the resulting persisted draft. Drafts use an evidence-only template. They are not unrestricted language-model completions, and the current API does not accept arbitrary edited draft text.

Approval is an explicit decision about the draft and its evidence. Before recording approval, the application checks that the supporting revisions are still current. A source or product-metadata change invalidates dependent drafts. An already open approval dialog can therefore encounter a conflict; the interface must explain that conflict and require a fresh review rather than quietly treating the old draft as valid.

An approved export contains compatibility metadata and the draft, including product ID/name and citations. It does not contain a full product record, and it does not publish to an external storefront. This boundary matters: an exported artifact is available for the user's next step, while Shopify writes require a future separately authorized workflow.

The same evidence structure supports change-impact analysis. Relationships show which product is documented by a source and which draft derives from evidence. A new manual revision can identify affected content without rewriting history. The current graph is explicit provenance data, not a trained graph model or a claim that community-summary GraphRAG has been evaluated.

The commercial value of this lifecycle is accountable content handling. Source-backed information can still be wrong if the source is wrong, but the system can show its origin, detect supersession and prevent an old approval from silently authorizing new facts.

3. Architecture, isolation and persistence

The implemented application uses React, TypeScript and Vite for the interface; FastAPI and Pydantic for the API; PostgreSQL for durable evidence and work state; and a separate worker for ingestion and drafting. A pinned ONNX encoder supplies optional CPU embedding ranking. The existing server hosts an isolated container stack behind a same-origin HTTPS gateway.

The browser requests the API through the same public origin. The web container serves static assets and forwards the API prefix. PostgreSQL is not published as a public database endpoint. API and worker use restricted runtime credentials; migrations use a separate administrator credential path. Secrets are mounted outside images and release archives.

Temporary browser access uses an opaque HttpOnly, Secure, SameSite cookie; only its hash is stored. Separate account sessions and persistent memberships select durable workspaces. Expected-workspace headers fence stale tabs, and protected writes recheck permission under a tenant lock. Clearing a cookie can lose temporary access without immediately erasing server records; account data survives logout.

Every tenant transaction sets a local context. Nineteen tenant tables enforce forced RLS; seven service-global tables use API authorization. The runtime role owns no public relations and cannot bypass RLS or assume an elevated role. PostgreSQL 17 uses read-committed transactions and a 10-second application statement timeout, with one connection per transaction. Internal DB TLS, connection-pool/load behavior and whole-transaction deadlines remain operating gaps.

Input limits, allowed origins, rate bounds and workspace quotas constrain admission and storage. These quotas are configuration values, not evidence of throughput or simultaneous-user capacity. Raising them requires load and abuse review.

Jobs persist with state, attempt, lease and retry information. A worker must verify that it still owns the claimed attempt under a row lock before producing, completing or failing work. This fences a delayed worker that resumes after another attempt has already taken ownership. Database fencing does not imply exactly-once side effects across an external API; future publishing needs reconciliation and its own idempotency design.

Containers run with nonroot users, read-only roots, dropped capabilities, health checks and resource limits. Owned volumes and temporary filesystems provide only the writable paths needed. Dynamic IP allocation excludes the reserved trusted proxy address, and all enabled Nginx temporary directories must use writable paths.

The current server is one failure domain. Isolation and immutable releases improve operational control, but they do not establish high availability, multi-region failover or protection from a fully compromised host administrator.

Reference: PostgreSQL row security — https://www.postgresql.org/docs/current/ddl-rowsecurity.html

4. Retrieval, model behavior and context

The current retrieval system combines PostgreSQL keyword matching with optional local embedding ranking. Keyword search identifies relevant subject matter; the quantized ONNX model adds semantic ordering through reciprocal rank fusion. Dense similarity alone cannot authorize a factual answer.

This conservative boundary has a trade-off. It reduces the chance of answering from an unrelated semantic neighbor, but purely semantic questions can miss useful evidence when they have no lexical subject match. The system reports its actual retrieval method instead of suggesting that every request used an advanced model.

The current encoder is the BGE-small-en-v1.5 ONNX conversion published by Qdrant. Its repository, full revision, model file, cache and thread/batch settings are configuration. The reviewed artifact is pinned to an immutable revision. Normal API and worker startup use the cache; downloading is an explicit preparation operation. Missing optional cache produces an honest keyword-only status.

A read-only runtime requires every downloader cache to be placed under an owned writable location. A warm cache can conceal a broken first-download path, so empty-cache recovery must be tested separately. Model availability is a readiness fact to observe, not an assumption from a local notebook.

Answers are source quotations with revision and page citations. They abstain when relevant evidence is absent. They do not perform complete question-level entailment checking: a quote may be relevant while failing to answer every part of a complex question. Provenance and truthfulness are related but different responsibilities.

The current release does not train models. It does not claim supervised fine-tuning, DPO, GRPO, distillation or improved recommendation quality. Those experiments remain in the separate model-engineering roadmap.

The current context API compares keyword, dense, hybrid and deterministic reranking against the same frozen questions. Authorized product/SKU/color/category and current-source filters apply before model work; traces record authority, budgets, exclusions and snapshot/version keys. Durable preferences, task state and generated summaries remain separate from product evidence. No general hybrid or graph-retrieval improvement is established.

Training will target extraction, output structure, tool behavior and appropriate abstention rather than memorizing changing price or inventory. It requires permitted data, product-family/source splits, generated-label disclosure, duplicate/leakage checks, base-versus-adapted results and documented regressions. A model that fails to improve can remain unpromoted; an honest negative result is useful engineering evidence.

Model sources: https://huggingface.co/BAAI/bge-small-en-v1.5 and https://huggingface.co/Qdrant/bge-small-en-v1.5-onnx-Q

5. Interface, multimodal work and accessibility

The visual system uses warm ivory, oxblood, olive evidence accents, editorial typography and sculptural product imagery. Cormorant Garamond and DM Sans are self-hosted. The interface aims to make evidence relationships readable while keeping navigation, review and recovery actions concrete.

The public website includes platform, pricing, about, contact, documentation, changelog, security, accessibility, legal hub, privacy, terms, cookies, United States and European Union information. Header and grouped footer links have real destinations. Mobile navigation is a disclosure using native links, expanded state, Escape dismissal and focus restoration.

Dialogs retain keyboard focus and return it to their trigger. Inspecting a source from a review returns to that review. Errors remain visible inside the relevant dialog, including changed-source approval conflicts. Long schema-valid imported text is tested at narrow widths instead of relying only on carefully formatted starting records.

Motion preferences are subscribed while the page is open. Switching to reduced motion removes the film and disables smooth scrolling. The silent hero film can be paused and also stops when offscreen or when the document is hidden. Interaction does not depend on animation.

The three product studies were generated with the built-in image tool. Rendered images are WebP derivatives, with PNG sources available for revision. The five-second film is a local FFmpeg animation of the headphone image; it is not a Seedance-generated film. An image of a product cannot establish a hidden technical specification or certification.

WCAG 2.2 AA is an engineering target, not an independent conformance certificate. Automated axe and keyboard checks are useful evidence, but manual screen-reader, zoom, reflow, high-contrast, touch and cognitive reviews remain necessary. Generated descriptions require human judgment.

Current image-purpose review separates visible observations, verified attributes and the description appropriate to its use. Informative alternatives convey essential meaning; decorative images use empty alternatives; functional images describe the action. Account review records do not establish independent human validation. Scanned or empty documents and images require OCR or reviewer observation rather than invented technical facts.

The service ingests PDFs and static PNG/JPEG/WebP through disposable resource-bounded parser processes and bounded JSON transport. Active/encrypted PDFs, animated images, MIME mismatches and oversized inputs are rejected. Text pages become attributed passages; no OCR/vision model runs in this path. Automated checks do not establish complete WCAG 2.2 AA conformance.

References: https://www.w3.org/TR/WCAG22/ and https://www.w3.org/WAI/tutorials/images/

6. Commerce integration and guarded publishing

The live Shopify connector provides a read-only commercial boundary: only authorized workspace owners connect a permitted store; only read_products is requested. The application imports catalog evidence and supports review/export. It has no checkout, orders/refunds, inventory reservation or product-write transport. External publishing requires a separate authorized workflow.

The implemented OAuth flow binds expiring one-use state to browser/account/workspace/shop, verifies callback HMAC and encrypts tenant/shop-bound tokens. Bounded pagination, refresh and a pinned API version govern reads. HMAC webhooks commit durable receipts and effects before acknowledgment, with duplicate/order fences. Local disconnect clears credentials without claiming remote revocation.

Actual permitted-store synchronization imported 23 parents and 38 variants, supported by 23 sources and 206 passages. Exact SKU/color constraints and a cited answer passed. Two separately approved Admin title changes/restoration produced real signed updates, invalidated a stale draft and reconciled to the original catalog. The connector itself made no product-write calls. Omission remains distinct from deletion; retirement tombstones current evidence and dependent approvals.

Product content work includes FAQs, descriptions, purpose-aware alt review and deterministic structured data. Price, availability and shipping must come from current authoritative inputs. A language model should not invent JSON-LD values or review aggregates. Any structured product data must remain consistent with visible content.

The future publishing workflow binds approval to tenant, actor, exact draft hash, source versions, target fields and expiry. A model proposes; application code validates authority; a person approves consequential changes. Publishing credentials stay outside the model's tool boundary.

A crash after a remote mutation is a difficult case. The system must reconcile the current field value before retrying. An uncertain timeout is not proof that no write occurred. Where a remote mutation lacks atomic compare-and-swap, document the race and use detect-and-reconcile or manual export for unsafe operations.

Reversal must respect field ownership. Store before/after values and reverse only if the current value still matches the operation's output. A merchant's later edit should trigger reconciliation rather than being overwritten by an automated rollback.

Account memberships, connector credentials and real permitted-store acceptance are implemented. Source-backed FAQ/description drafts, purpose-aware image review and deterministic Product/ProductGroup output retain provenance/currentness checks. Password recovery, human benchmark/accessibility validation, graph-quality improvement and scheduled backup operations remain open gates. Merchant-production onboarding is blocked on the documented security, legal and operating decisions.

AI-search visibility is a quality/crawlability question, not a promise that an assistant will recommend a product. Measure accepted content and search usefulness with qualified experiments; do not claim placement or conversion uplift from generated events.

7. Evaluation, release evidence and engineering quality

Verification is organized around behavior, not framework count. Criteria come first, implementation stays scoped, required gates pass, and a fresh reviewer checks the result. The application is exercised through real HTTP/UI paths against PostgreSQL and the CPU model rather than relying solely on mocked responses.

The dated evidence-core release passed 24 local PostgreSQL/HTTP/ONNX integration checks, 28 public browser checks, 23 deployment-safety regressions and five domain/configuration checks. Types, lint, security scans, builds and independent review passed. Those counts describe that release, not every future source revision.

The later public-page release passed eight acceptance criteria, eight domain checks, 32 local browser cases and 44 actual public desktop/mobile cases with no skips, flakes or failures. Fifteen deployment-safety regressions and a scoped visual refresh also passed. Its 84 prepared/archive/live artifact hashes matched, while the existing backend image and service identities remained unchanged.

Browser behavior included citations, unsupported answers, durable approval/export, changed-source invalidation, concurrent approval conflict, expired-cookie recovery, maximum-length imports, dialog focus, navigation, reduced motion and accessibility. A direct unknown route returned a branded HTTP404 with the correct title and zero page errors.

The original ten generated questions remain a regression set. The newer frozen corpus contains 30 generated products, 120 schematic images and 200 questions, including a 64-question test split. Four retrieval methods were measured locally; keyword was selected for that corpus. An independent agent reviewed labels/images under owner delegation; human validation remains pending. These results establish neither merchant accuracy nor a general hybrid improvement.

Evaluation separates retrieval relevance, claim support, contradictions, abstention and task completion. The newer assistant source adds visible guidance, product-scoped follow-ups, at most six volatile question/scope records and Stop/Restart controls, with no old-answer storage or external LLM. Its 40-task independent source evaluation passed 38 relevant tasks and 29 critical checks; two All-products relevance failures remain. These UI changes are not deployed. Human review remains distinct from agent review.

Training comparisons require frozen family/source splits and leakage review. Serving reports require p50/p95 latency, throughput, memory, cold/warm behavior and cost per accepted output. Business reports need actual accepted drafts, manual review effort or appropriately designed search/conversion experiments.

The public documentation was checked for relative links, public language/private-path hygiene, preserved ten-phase structure, SLO arithmetic and license/access consistency. CI configuration and a hosted successful run are distinct evidence. New releases need their own commit-specific checks and observed deployment proof.

8. Deployment, operations, recovery and the future 99% SLO

Local source is the authority. Before editing, compare the active release with the local baseline; reconcile newer remote changes. Build and verify locally, prepare a unique immutable release and checksum manifest, restore the archive locally, then preflight the actual target before promotion. End by proving local/archive/live parity and preserving unrelated services.

A frontend-only release must not package unverified backend work or restart an unchanged API. Runtime rollback preserves database/model volumes and requires schema compatibility. An old image cannot reverse an incompatible migration.

Release 02 pre/post snapshots reproduced 27 public SQL table payloads/counts, including six migration records; isolation probes cover 26 application tables, comprising 19 RLS and seven global tables. The earlier phase3 AES-GCM off-server restore reproduced 249 revisions/15 drafts, and phase5 recovery used 251/16. Current 02 runtime rollback was not exercised. The retained 03 candidate/partial archive is not a verified recovery backup or completed deployment.

Future backup targets are RPO at most 24 hours and RTO at most four hours. Current evidence does not establish an automated cadence or either target. A complete policy needs owned off-server destinations, consistent backups, protected configuration/evidence coverage, verified encryption/key recovery, retention/deletion rules, missed-backup alerts and timed independent restoration.

The future availability target is at least 99% good eligible interactive requests over a rolling 30-day window. Eligible traffic and exclusions must be explicit. Capacity-driven rejection of an admitted valid request counts as failure; missing telemetry is unknown coverage. Static delivery and the working API must be measured separately.

The proposed interactive deadline is five seconds, requiring implementation/cancellation and workload validation before measurement. Proposed non-model latency is p95 within one second; durable job acknowledgment is within two seconds. Long AI job outcomes require separate class deadlines after benchmarks.

The request error budget is 1% of eligible requests. At one million requests it permits 10,000 bad events. A full-outage time equivalent is 432 minutes in 30 days, but traffic-weighted and time-weighted budgets are not interchangeable. Four hours would consume about 56% of that time-equivalent allowance.

There is no current measured 99% uptime or contractual SLA. Instrumentation, alert delivery, burn-rate policy and recovery drills must precede that claim. Incident response scopes impact, contains the owned service, preserves sanitized evidence, validates recovery and records the prevention gate. Suspected privacy incidents require a notification assessment without waiting for technical restoration.

Reference: https://sre.google/sre-book/service-level-objectives/

9. Security, privacy, legal applicability and rights

Controls include account/tenant authorization, forced tenant RLS, restricted credentials, bounded parsing, stale-approval rejection and fenced jobs. Point-in-time live-image scans retain 2 critical/61 high application rows, 4 critical/95 high database rows and one high web row, with vendor/reachability limits and no suppression. The reviewed application candidate has zero critical/44 high rows but was not promoted. These findings and operating gaps remain merchant-production blockers; package-only success is not whole-system assurance.

Report vulnerabilities privately to muhammadzahidulislam2222@gmail.com. Use a safe reproduction and no tokens or another person's records in the first message. No bug-bounty payment, response-time guarantee or independent security certification is offered.

Current data includes account/session metadata, memberships, product/source revisions, private uploaded originals, encrypted Shopify tokens, drafts, jobs, evaluations and audit metadata. Questions reach the project server for retrieval and are excluded as raw text from application audit records. The source-only assistant keeps at most six questions/scopes in volatile page memory, clears them on identity/view/restart changes and stores no past answers or server transcript. Hosting logs may contain request metadata.

The worker expires temporary workspace data; persistent account data survives session expiry/logout. Source deletion clears uploaded original bytes and current eligibility while retaining historical extracted revisions and tombstones. Cookie/local-storage clearing does not erase logs/backups. Retention, restored-data deletion propagation and at-rest encryption coverage need further verification; no fixed complete-erasure promise is offered.

Before merchant personal data is handled, complete a data map, lawful-purpose/role assessment, processor agreements, subprocessors, transfer review, retention schedule, rights/takedown workflow and tested deletion/restoration controls. Future deletion must invalidate descendants and preserve tombstones for recovery. Removing a training row does not erase learned influence; affected adapters may need retraining or disposal.

GDPR applicability depends on establishment, targeting/monitoring and processing—not simply whether an operator is outside the EU or access is free. US obligations depend on federal/sector and applicable state rules. California thresholds/rights require an actual assessment; children's-data processing requires separate review. Privacy pages alone do not establish compliance.

For relevant personal-data incidents, evaluate notification obligations promptly. GDPR supervisory notification generally has a 72-hour awareness-based requirement where applicable, with a risk-based exception and separate affected-person tests. Obtain competent legal support for the actual facts.

Assess the current extractive AI assistant and any future generative chat, recommendations or adapted models against applicable AI transparency, role, risk and data-processing requirements. No EU AI Act exemption or classification is asserted. Keep approved sources and permissions separate from model authority; retain provenance and bounded failure behavior. Do not invent technical, medical or business-performance claims.

The author’s code and documentation use MIT, copyright 2026 Zahidul Islam; preserve its notice and disclaimer. Third-party software, fonts, models and media retain their own terms. Package/license metadata and inventories have scoped evidence, not complete legal clearance. Live application and database images retain OS advisories; a reviewed maintained-base/library candidate was not promoted. Its SBOM also has missing license fields; publisher signing and reproducible-build identity remain unverified.

References:
https://commission.europa.eu/law/law-topic/data-protection/information-business-and-organisations/application-gdpr_en ; https://www.ftc.gov/business-guidance/privacy-security ; https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

10. Future 1M+ users, capacity arithmetic and cost

The one-million-plus active-user objective is a future architecture target, not measured current capacity. Active users, connections, requests/second and CPU jobs must be sized separately. One person can read cached content without an API request; a queued model task can outlive its submitting request.

A stated planning example assumes one million active users making six requests/minute: 100,000 requests/second. If 80% are public content and that portion has a 99% edge hit rate, public origin traffic is 800 requests/second. The 20% private API share remains 20,000 requests/second, for 20,800 total origin requests/second.

If 90% of API requests are reads and 90% of those reads hit a tenant-aware application cache, 1,800 reads and 2,000 writes/second remain. At three database operations per uncached request, that is about 11,400 operations/second before background work. Approval/current-source checks must use authoritative state rather than a stale read cache.

For future model jobs, an assumed 0.001 jobs/user/second yields 1,000 jobs/second. An assumed 0.25 CPU-seconds/job at a 60% utilization ceiling requires at least 417 cores before redundancy. These are planning inputs, not a benchmark; measured service time must replace the assumption. At two-second total job latency, about 2,000 jobs would be in the system.

At 1.2 million connections and an assumed 32 KiB state each, edge state alone is about 36.6 GiB before TLS/OS overhead. At 100,000 requests/second and 10 kB delivered content, sustained traffic is about 2.59 PB over 30 days. Cache hits still consume delivery bandwidth.

Separate public edge delivery, stateless interactive API and asynchronous AI/ingestion workers. Bound connection pools, deadlines, tenant fairness, queue admission and model concurrency. Cache keys include tenant/permissions/source/index/model versions and invalidation.

Scale in measured stages: profile the current envelope, separate bottlenecks, add application replicas, evaluate dedicated data/search tiers, then tenant cells and regional cells. A tenant's transactional approval/evidence state needs a clear write authority. Multi-region requires residency, replication-lag/RPO/conflict policies and actual failover tests.

One small server and free allowances cannot provide this target. Budget compute, database/IOPS, model hardware, egress, storage, backups, telemetry and support. Bounded Azure Free hosting, authenticated AWS validation and GCP retrieval contracts were executed separately; temporary GCP resources were removed. These are outside the main critical path and do not establish large-scale capacity, lasting free eligibility or final billing. New usage needs verified limits/cost authorization.

11. Ten phases with parallel work and deployment

Each phase coordinates frontend, backend, AI/data, infrastructure and QA/security against agreed contracts. Integrated releases implement phases 1–5 features, including permitted-store and bounded three-cloud checks, without proving five sequential phase deployments. Human validation, graph-versus-hybrid quality and automated backup cadence remain open. The table retains the ten-phase acceptance plan; later publishing/training/operations work is not represented as complete.

1. Foundation — UI shell; identity/membership/RBAC; evidence schemas; isolated infra; tenant/role tests

Deployment and evidence: Existing-server foundation and eligible Azure static shell; real allow/deny matrix and eligibility inventory

2. Ingestion — Source/job UX; bounded CSV/PDF/image parsing; versioned corpus; storage/backups; parser/deletion tests

Deployment and evidence: Ingestion/source viewer; attributable records, family/source split and independent restore

3. Shopify/read-only — Connection/sync UX; OAuth/webhook receipts; catalog mapping; AWS validator; replay/order tests

Deployment and evidence: Permitted development-store sync and actual eligible AWS run

4. Retrieval/context — Search/context inspector; filter/permission API; method comparisons; GCP benchmark; injection/isolation tests

Deployment and evidence: Current-version retrieval and actual eligible GCP snapshot benchmark

5. Graph/multimodal — Claim/alt review; temporal relationships/JSON-LD; contradictions/observations; bounded workers; human accessibility review

Deployment and evidence: Evidence impact and deterministic structured-data checks

6. Guarded publishing — Exact diff/approval UX; resumable typed tools; bounded workflow; isolated write worker; stale/replay/unknown-write tests

Deployment and evidence: Approved development-store write, readback and conflict-aware reversal

7. Training — Annotation/results UX; dataset/model registry; LoRA and retrieval training; isolated compute; leakage/regression tests

Deployment and evidence: Actual checkpoint plus base-versus-adapted held-out report

8. Research/ranking — Recommendation/trace UX; simulator; ranking/preferences/distillation; bounded jobs; reward/constraint tests

Deployment and evidence: Executed baseline comparison with gains and regressions

9. Operations — Health/degraded UX; retention/recovery APIs; adversarial/serving benchmarks; backup/canary; failure drills

Deployment and evidence: Measured capacity and timed independent recovery against proposed targets

10. Handover — Final workflows/help; compatibility/export; frozen model/data cards; clean rebuild; independent release review

Deployment and evidence: Reproducible release, operational ownership and actual three-cloud records where eligible

The roadmap requires permitted data and real evidence. Local emulation does not count as a cloud deployment. An unexecuted notebook does not count as training. Automated generated labels do not replace independent review. A model may fail to improve and remain unpromoted.

Cloud trials can expire. Export artifacts and retain local reproducibility rather than upgrading automatically or reopening accounts to evade limits. No cloud trial holds the only copy of important data.

Safety gates remain throughout: source permissions, deletion propagation, token lifecycle, tool authorization, parser/egress constraints, approval expiry and hash binding, queue fencing, model supply-chain integrity, compatible migrations and independent recovery. Advanced research such as voice, graph neural networks, GRPO, distributed training and GPU kernels requires its own baseline and resource justification.

The repository build plan retains the five-lane contracts, dependencies, deployment, acceptance and recovery requirements for all ten phases. The owner stopped the release03 deployment and remaining publication/QA work while requesting this document/PDF refresh. Preserve the candidate and notes; any resumption requires fresh drift, recovery, actual-image and live-flow gates before a completion claim.

12. Reproducibility, lessons and next decisions

A reviewer should be able to understand the current product without private operational access. The repository documentation covers setup, configuration, current API, architecture, security/threats, privacy/legal applicability, accessibility, data/model licenses, evaluation, deployment/CI, incidents/backups, future scale/cost and the ten-phase plan.

The frontend uses a reproducible npm lock. The supplied backend hash lock targets Linux x86_64 and CPython 3.12; other platforms require separately verified locks. Use a dedicated local database, separate runtime/migration credentials, explicit origins, the real worker and a prewarmed pinned model for required CPU checks.

Hosted CI configuration is not an observed hosted result. Record the run URL and commit after it succeeds. Likewise, release publication is not proof of deployment, and uploaded files are not proof of parity. New releases need candidate-specific gate, review, actual-flow and hash evidence.

Several engineering lessons shaped this design. Resource inventory must include networks and ports, not only free RAM. Read-only containers require every cache/temp path to be tested under the real UID. A lock alone does not establish job ownership; attempt fencing does. Responsive design must be tested with maximum valid input. A citation proves origin, not complete question entailment.

These lessons belong in repeatable checks. A failure should produce a small regression gate and a documented recovery path rather than only a one-time repair. Public reporting should explain the problem, outcome and evidence without exposing access details or confidential incidents.

The next commercial decisions concern account recovery/MFA, merchant legal/vendor eligibility, retention/rights and meaningful task quality. Model work still needs permitted frozen data, actual training and held-out results. Operations need owned telemetry/alerts, automated encrypted off-server backup policy and timed restoration. The 1M+ simultaneously active-user and 99% SLO targets remain unmeasured future objectives.

Current limits remain visible: temporary access plus persistent accounts without recovery, read-only storefront integration, no full entailment or broad quality guarantee, no measured large-scale capacity/rolling availability and unfinished merchant-production operating gates. Source-only improvements, reviewed candidates and stopped deployments are distinct from the live release. Accurate boundaries keep the next decisions reviewable.

The project is authored by Zahidul Islam. Product, security, privacy, accessibility and rights questions can be sent to muhammadzahidulislam2222@gmail.com. Contributions follow the repository's rights policy, security reporting and code of conduct. The result should be judged by source, reproducibility and measured behavior, with future ambition stated as future work.