SynqHarbor AI — Comprehensive Technical Overview

Maintainer: Zahidul Islam. Release: 0.2.0. Updated: 2026-10-16.

1. Latest release and verification record

SynqHarbor AI brings clients, developers, operators and bounded AI agents into one human-led collaboration workspace. This edition begins with the published release record, followed by the complete product, architecture, security, legal, operations and growth plan. It retains every substantive paragraph from the twelve-section public technical overview and adds release evidence and legal readiness requirements.

Release 0.2.0 publishes 25 website routes, including the workspace, product and company information, engineering documentation, security and legal pages. The public repository belongs to Zahidul Islam. Its published revision is 6fc0e97ec42b9f09be979e67ecd81d8b36a38f80. The website and repository links below identify this release's public entry points.

The recorded release checks passed 21 automated tests, TypeScript checking, ESLint and the optimized static build; the dependency audit reported zero known vulnerabilities at release time. GitHub Actions run 38812485951 completed its quality and security jobs successfully. These results describe the reviewed release, with time-bounded dependency evidence.

Recorded browser verification covered 100 route/viewport combinations across four widths, with accessible navigation and corrected automated contrast findings. The deployed release returned all 25 routes successfully, with no JavaScript or content-security-policy errors in the tested flows. Task creation and search worked; reload cleared browser-tab state as documented. Deployment verification recorded 175 matching local/live hashes and an archive restoration check.

Authentication, persistent tenant workspaces, collaboration APIs and agent execution remain future phases. The 99% initial availability objective and future 1M+ simultaneous-user envelope are planned qualification targets. Their assumptions, measurement policies and acceptance gates appear later in this document. This edition does not convert targets into achieved operating history.

Live website · Workspace · Public documentation

Public repository and release history · Published quality/security run

2. Product and current release

SynqHarbor AI is an independent, self-hostable collaboration product maintained by Zahidul Islam. It is designed for clients, developers, operators and bounded AI agents working from shared context. The core principle is human ownership of decisions, explicit permissions and evidence attached to meaningful outcomes.

The current release delivers a professional multipage website and interactive workspace screens. Product, features, solutions, agents, integrations, self-hosting, company, security, legal and engineering pages are linked through shared navigation and a full footer directory. The workspace includes overview, rooms, task board, agents, approvals, artifacts, knowledge, activity and settings.

The website uses Next.js, React and TypeScript. Motion supports transitions; Lenis supplies progressive smooth scrolling. Fonts and media are self-hosted. Responsive layouts, focus handling and reduced-motion behavior are part of the design system.

Workspace edits remain in the active browser tab and reset on reload. Account authentication, remote persistence, collaboration APIs, production agent jobs and external actions remain roadmap milestones. The current interface does not connect provider subscriptions or charge for inference.

The public source, documentation and release history belong to the maintainer's GitHub project, Zahidulislam2222/synqharbor-ai. This overview explains current capability, operating objectives and future architecture separately. A future target becomes an available capability only after implementation, acceptance testing and deployment evidence.

3. People, workflows and available interactions

The initial audience is software and automation teams working with clients. Clients need a clear request, understandable progress and a reviewable result. Developers need bounded tasks, implementation evidence and authority over release. Operators need visibility into blocked work and recovery.

The current workspace supports local task creation, search, project filtering, status changes, conversations, artifact inspection, task-report downloads and appearance preferences. Approval interactions distinguish client content acceptance from developer release decisions. A revision clears outdated authorization in the interface model.

Three records remain conceptually distinct. A message contributes to a conversation. A task defines work, ownership and acceptance criteria. A run records one execution attempt. Future backend implementation must preserve these distinctions so an informal message cannot silently become execution authority.

The target delivery workflow begins with a request and explicit criteria. A bounded Builder produces an artifact. Independent review examines the exact result and retained tests. The client approves content, while an authorized developer separately approves release.

Business accountability and content operations are later workflows built on the same foundation. Source-linked commitments, follow-up drafts, campaign research and publishing all need scoped information and explicit external-action permissions.

Current browser state is not an authoritative shared business record. Until persistent services are delivered, retain important information in your existing systems. No local approval action publishes software or sends a communication.

4. Architecture and trust boundaries

The delivered architecture is static: Next.js generates HTML and browser assets; a hardened static server distributes them; the browser runs interface interactions. Caddy supplies automatic HTTPS on the existing deployment, and Cloudflare provides network delivery and security services.

Maintained content and navigation live in validated data files. Public origin, routes, media and motion settings have a central typed configuration owner. Public-prefixed build variables must never contain credentials. No provider key is required to run the website.

The planned application adds a control plane for identity, membership, tasks, runs, artifacts, reviews, approvals and connector grants. PostgreSQL is proposed as the first authoritative durable store. Private object storage holds larger artifacts; notifications and caches remain separate from task authority.

A durable dispatcher assigns bounded work to qualified adapters and isolated outbound runners. The web server must never execute arbitrary agent-generated shell commands. Runtime, model provider, permissions and execution location are independent settings.

Start the backend as a modular monolith with one authoritative dispatcher. Split services only when measured bottlenecks, security isolation or operational ownership justify the boundary. Competing schedulers must not claim the same task.

The future flow is authenticated request, authorization, persistent work, bounded execution, retained artifact, review, approval and authorized action. Each boundary needs a clear owner, observable state and recovery behavior. Public diagrams omit private infrastructure addresses and access material.

5. Backend and API development roadmap

A persistent collaboration backend is outside the current website release. Future API documentation will be generated from implemented schemas and verified against real HTTP behavior. Planned routes must not be presented as deployed endpoints.

Initial data domains include organizations, workspaces, memberships, rooms, messages, projects, tasks, runs, artifacts, reviews, approvals, grants, usage and action receipts. Records need stable identifiers, timestamps, provenance and explicit tenant ownership.

Authentication must cover secure server-side sessions, expiry, revocation, account recovery and protection against CSRF. Every object access requires backend tenant and role authorization. A visible UI role is not sufficient authority.

Task changes need optimistic revision checks. Artifact versions and content hashes must be immutable. Approval records bind an exact version, destination, environment and arguments. Changed content invalidates previous authorization. Client acceptance and developer release require distinct permissions and, where policy requires, distinct actors.

The API contract must document validation, pagination, body limits, stable errors, rate limits and compatibility. External writes need idempotency and an action-intent ledger. After an uncertain timeout, inspect the destination before retrying rather than duplicating an accepted action.

Acceptance tests cover guessed identifiers, cross-tenant access, revoked grants, concurrent updates, stale approvals, duplicate events and interrupted work. Database migration, backup and restoration procedures must be established before persistent customer workloads are accepted. These requirements remain separate from the published frontend milestone.

6. Agents, artifacts and human authority

Builder and Reviewer are the initial role model. An agent role is not a model, provider or permission grant. Qualified runtimes should remain replaceable without changing the task and approval foundation.

The website currently presents agents through profiles and local interactions. It does not execute models, connect subscriptions, run shell commands or publish to external systems. Real execution requires an explicitly authorized route and independently enforced tools.

Each future assignment should carry the objective, relevant sources, acceptance criteria, allowed actions, artifact references and remaining budget. Start with deterministic routing to explicitly assigned agents. Enforce step, time, concurrency and spending bounds.

Independent review examines exact artifacts and retained test evidence. A second model can contribute findings but does not prove correctness. Outcome qualification combines protected automated checks, structured review and human judgment.

Coding execution requires disposable project copies, restricted filesystem access, controlled networking and resource limits. A worktree separates changes but does not isolate the operating system. Production credentials, unrestricted home directories and the Docker socket must remain inaccessible to jobs.

The evidence trail should distinguish prepared work, completed review, granted approval, submitted action, destination acceptance and confirmed outcome. These states must survive interruption. Chat text cannot override backend authority, and retrieved documents cannot expand an agent's permissions. External communications and release actions remain human-controlled capabilities.

7. Installation, configuration and delivery

Use the documented Node.js 24 runtime and the pinned package lock. Install with npm ci. Configure NEXT_PUBLIC_SITE_URL to the intended HTTPS origin, then run tests, type checking, lint and the production build. The output directory is out/.

Serve the export with correct MIME types, directory indexes, real 404 responses, compression and media range support. Static output does not provide request-dependent server actions, authentication or API routes. Self-hosted fonts avoid runtime font-CDN requests.

Release locally first. Compare the current live release with the local manifest before replacing it. Reconcile newer live changes into local source. Build and review the change, upload a versioned release, validate its server configuration, check health and atomically activate it.

Preserve the previous release and an off-host archive. After activation, check HTTPS, routes, interactive behavior and exact file hashes. Scope shared-host changes to the project's own site, port and application identity; do not modify unrelated services.

Configuration ownership is explicit: deployment/environment values in typed settings, maintained copy in data files, secrets in approved private storage and recovery records. Public examples contain safe placeholders.

CSP policies must be generated from the exact UTF-8 scripts in the release and tested in a real browser. A successful build or HTTP 200 cannot establish that hydration and interactions work under production headers. Future backend releases additionally need migration and state-aware rollback planning.

8. Security controls and verification

The current static release uses HTTPS, self-hosted assets, a generated script policy, dependency checking, secret scanning and reviewed deployment. The serving container is nonroot and read-only, with dropped capabilities, resource limits and loopback-only publishing.

Private credentials, project memory, recovery records and unshipped work are excluded from Git publication. A public repository must be scanned across staging and history before pushing. Configuration examples must never contain live access values.

Future application controls should map to OWASP ASVS 5.0.0 with status, evidence and ownership. Priority controls include object-level authorization, tenant isolation, safe database queries, output encoding, secure sessions, CSRF protection, validated uploads and bounded resource use.

Agent systems also require defenses against prompt injection, sensitive-data disclosure, excessive agency and unsafe output handling. Treat retrieved instructions as untrusted. Enforce permissions outside the model, and never use prompts as a substitute for backend authorization.

Pin dependencies, review licenses, triage advisories and require focused changes. Gitleaks and Semgrep complement behavioral tests and independent review; a passing tool is evidence within a defined scope, not a certification.

The project enables GitHub private vulnerability reporting. Reports should include affected release, minimal reproduction and impact. Do not publish credentials or sensitive exploit information, access other people's data, scan unrelated infrastructure or run denial-of-service tests. Disclosure and verification are coordinated with the maintainer.

9. Privacy, legal responsibility and accessibility

The current website has no account registration, contact intake form or advertising analytics. Workspace interactions remain local to the browser tab. Hosting and security services can process technical request metadata, including network and browser information.

The current deployment uses an existing Contabo server and Cloudflare delivery/security services. Third-party links enter their own services. Server logs are capacity-rotated; a fixed age-based retention period has not been verified. Privacy concerns should use the contact page to arrange private communication, not publish personal data in issues.

The operator is Zahidul Islam. A corporate entity, governing jurisdiction, signed DPA or universal compliance claim is not invented. Website terms, acceptable use, privacy and cookies policies describe current operation; commercial contracts and customer processing schedules require separate agreement.

Future self-host operators must document purposes, roles, categories, retention, exports, deletion, subprocessors and inference destinations. Self-hosting does not automatically keep connected model context local. Assess international transfers and applicable rights before accepting customer data.

Where GDPR applies, distinguish processor notification to its controller from conditional controller authority-notification duties. Source software uses Apache 2.0; dependencies and typefaces retain their licenses.

Accessibility targets WCAG 2.2 AA. Evaluate keyboard navigation, visible focus, contrast, reflow, touch targets, announcements, forms and reduced motion. Automated checks support human workflow evaluation. A target or axe result is not an accessibility certification.

10. Legal readiness, processing lifecycle and customer agreements

Before an authenticated customer deployment, identify the contracting operator, customer/controller and any processor or subprocessors. Document each data category, purpose, lawful basis where required, destination, retention period and access role. Determine applicable obligations from the actual service and affected people; neither the maintainer's location nor a hosting region establishes universal compliance.

Where GDPR applies, establish procedures for identity-verified access, correction, deletion, restriction, portability and objection where applicable. Respond through a private, authenticated channel. A data export must exclude other tenants' records. Deletion must address primary records, search indexes, caches and objects, while documenting backup expiry and lawful preservation requirements. These are implementation requirements for future account services.

A processing agreement should describe authorized instructions, confidentiality, security measures, subprocessor changes, audit cooperation, incident reporting and return/deletion at termination. Review any international transfers and required safeguards before transmitting customer context to a model or connector. Keep provider destinations visible to an authorized customer administrator.

An incident procedure must separate containment, evidence preservation and risk assessment from notification decisions. Under applicable GDPR duties, processor-to-controller notification and controller-to-authority obligations differ. Evaluate the relevant notification threshold, awareness time and required content rather than treating every technical fault as a universal 72-hour notification obligation.

Commercial terms need separately reviewed service scope, ownership/licensing of customer content and generated artifacts, acceptable use, payment rules if introduced, support, suspension, termination and dispute arrangements. Any future service-level agreement must define eligible events, measurement window, exclusions, remedies and responsibilities. The present 99% objective is not such an agreement.

Plan consent and cookie disclosures around actual enabled technologies and applicable rules. Maintain dependency and media license notices. Introduce payments, sensitive-data workflows or regulated use cases only after their specific legal and security requirements are addressed. No regulatory certification, independent penetration-test report or signed customer processing agreement is claimed by this release.

EDPB guidance: individual rights · International transfers · Personal-data breaches

11. Availability, monitoring and support

The proposed initial operating objective is 99% availability over rolling 30 days. A future multi-zone stage targets 99.9% after redundancy, capacity and incident response are qualified. These are objectives, not measured uptime history or a contractual SLA.

For a 30-day period, time-based 99% availability permits 7 hours 12 minutes of unavailability; 99.9% permits 43 minutes 12 seconds. Request-based availability must be calculated from actual eligible events rather than inferred from these time equivalents.

Define success, eligible requests, maintenance treatment and telemetry gaps before reporting. Count timeouts, server errors and valid-workload throttling as failures as appropriate. Rejecting legitimate traffic must not manufacture a better availability result.

Separate website delivery, authenticated API success/latency and asynchronous completion. Planned indicators include latency percentiles, queue age, completion success, reconnect behavior and artifact durability. Per-tenant views must reveal customers hidden by aggregate success.

Monitor public reachability, TLS expiry, resource saturation, disk capacity and release health. Future workers need heartbeats and stale-run detection. Telemetry must avoid sensitive content. The current status page reports a point-in-time browser connection check rather than historical uptime.

Use an error-budget policy to prioritize reliability when objectives are threatened. Incident procedures define severity, owner, escalation, communication and follow-up. Community support has no contracted response-time commitment. Continuous staffed support requires separately agreed people and operating capacity.

12. Recovery, portability and operational ownership

Recovery scope includes source, locked dependencies, release artifacts, configuration and protected access material. A backup stored only on the same host does not protect against losing that host.

The current static website can be rebuilt from the public repository and public build configuration. Versioned release archives are retained off-server, extracted into a separate restore directory and compared with a SHA-256 manifest. The previous deployed release remains available for rollback.

Future persistent operation requires encrypted off-host database/object backups, monitored failures, retention and timed restore drills. Keep recovery keys separate from encrypted backups and control access without making recovery dependent on an undisclosed personal account.

Proposed startup objectives are RPO no greater than 24 hours and RTO no greater than 8 hours. A later qualified stage targets RPO no greater than 15 minutes and RTO no greater than 1 hour. These remain targets until actual drills establish feasibility.

RPO describes the acceptable data-loss window. RTO describes the target restoration time for required functionality. Define incident start/end and application integrity checks. Successful backup creation is not proof of a usable restore.

PostgreSQL point-in-time recovery requires appropriate base backups and retained WAL. Drills should cover a lost host, broken release, failed migration, compromised credential and unavailable provider. Record elapsed time, recovered data point and unresolved dependencies. Portability requires exportable configuration, artifacts, account inventory and a clear operator handover.

13. Future 1M+ simultaneous-user architecture

The growth architecture targets more than one million simultaneously connected users. Connections, active requests, retained data and executing jobs are different capacity dimensions. The current website does not establish authenticated million-user collaboration capacity.

The proposed envelope assumes 1,000,000 connections, 10% active users and 0.2 API requests per active user per second: 20,000 requests/second sustained and 40,000 burst. At an assumed 20KB per connection, gateway memory approaches 20GB before transport and application overhead.

If 1% submit one agent job per hour, arrivals are approximately 2.78 jobs/second. At a three-minute mean duration, approximately 500 jobs execute concurrently. Model throughput, sandbox resources and tool latency require independent measurements.

The future architecture separates static delivery, stateless APIs, connection gateways, durable dispatch and isolated workers. Regional cells, tenant routing, bounded database pools, private object storage, authorization-aware caches and controlled fan-out reduce shared failure domains. Partition data only after query evidence justifies it.

Qualify 1K, 10K, 100K and 1M connections progressively. Include realistic writes, tenant skew, reconnect storms, spikes, soak, dependency failures and recovery. Test generators need capacity checks; dropped iterations invalidate conclusions.

Proposed interactive targets include p95 at most 500ms, p99 at most 1500ms and error ratio at most 1% under the defined envelope, with no lost accepted writes or stale authorization. These are qualification targets. Funding, staffing and published test evidence are prerequisites for this operating stage.

14. Ten-phase roadmap, cloud costs and references

The full build plan is authoritative. Its ten phases are: foundation/deployment; identity/tenancy; collaboration and durable controlled execution; artifacts/evidence/approvals; local AI and isolated runners; complete client delivery; permission-aware knowledge; business workflows/scheduling; operations/reliability/portability/capacity; production pilot and handover.

Each phase coordinates frontend, backend, agent/integration, infrastructure and quality tracks. Each ships a scoped increment with rollback and measured exit gates. Website publication does not close backend or cloud engineering gates.

Startup targets zero new service charges using an existing host within measured capacity. Existing infrastructure still has ownership costs. AWS, GCP and Azure initially serve bounded public surfaces without competing task authorities.

CloudFront Free flat-rate is distinct from pay-as-you-go allowances. Published startup limits include 1M monthly requests, 100GB transfer and 5GB storage, with no SLA. Firebase Spark is quota-limited and can stop affected services; linking billing upgrades to Blaze. Azure Static Web Apps Free has bounded storage/domains and no SLA. Recheck eligibility before deployment.

One million 1MB initial page loads already transfer about 1TB. Free startup plans cannot supply the complete target envelope. Paid capacity, inference, egress and staffing need current estimates and explicit approval; alerts are not universal spending caps.

Primary references: sre.google/workbook/implementing-slos/; owasp.org/projects/asvs; w3.org/TR/WCAG22/; postgresql.org/docs/current/continuous-archiving.html; aws.amazon.com/cloudfront/pricing/; firebase.google.com/docs/projects/billing/firebase-pricing-plans; learn.microsoft.com/en-us/azure/static-web-apps/plans. Detailed guides preserve direct links and verification boundaries.

Google SRE — implementing service-level objectives

OWASP — Application Security Verification Standard

W3C — Web Content Accessibility Guidelines 2.2

PostgreSQL — archiving and point-in-time recovery

AWS — CloudFront pricing and plan boundaries

Google — Firebase billing plan boundaries

Microsoft — Azure Static Web Apps plans