PraxisForge AI

A Hermes-powered agent operations and engineering platform

Comprehensive technical overview | Zahidul Islam | 16 October 2026

PraxisForge AI turns business requests into bounded agent work, explicit human decisions and inspectable outcomes. It combines a professional web interface, authenticated operations, permission-scoped knowledge, durable task ownership and three specialized Hermes roles. The initial release is self-hosted and serves a configured operator; broader customer admission and external application delivery have separate roadmap gates.

Release evidence was verified on 15 October 2026. A live agency workflow completed after a worker interruption, produced two source-supported commitments, required exact approval and recorded an internal receipt. An encrypted off-host production backup restored into a fresh isolated database with matching schema and business records. These results describe the tested scope; future capacity and availability objectives are specified separately.

Reading map: product and workflow; architecture; agent reasoning and knowledge; durable execution and approval; identity and security; privacy and legal applicability; integrations and governed improvement; engineering and deployment; reliability objectives; million-user capacity planning; evaluation and recovery; phased roadmap and references.

Live product: https://praxisforge.zahidul-islam.com/ | Operations: https://praxisforge.zahidul-islam.com/operations/ | Source and detailed guides: https://github.com/Zahidulislam2222/PraxisForge-AI

The source is available for inspection under the repository's copyright notice. Frameworks, dependencies, fonts and models retain their own licenses. This report contains public project information and excludes access credentials, private infrastructure details and confidential operating records.

01 Product and business workflow

The first workflow supports agencies and operational teams that need consistent follow-through on meeting commitments. A member supplies meeting text and a request, selects permitted sources and identifies the intended client and recipient. The platform records the request durably before agent processing begins. Work is asynchronous; the operator can inspect stage, attempt, outcome and evidence rather than keeping a conversation open.

The coordinator frames the bounded task, the researcher examines the selected knowledge, and the operator prepares a draft and commitments. The result is validated and presented for a human decision. Approval records an internal draft and commitments together with a receipt and episodic summary. It does not send an email or claim a third-party CRM update. Future connectors must verify their own account authorization and provider outcomes.

Current capabilities include owner authentication, workspace membership checks, versioned text sources, durable tasks, retries and cancellation, source-linked structured outputs, exact approval envelopes, internal receipts, downloadable artifacts and persisted workflow memory. The operations interface provides Work, Knowledge, Memory and Connections views, with an approval dialog inside Work. The public website contains product, pricing, contact, guidance, security, status and legal pages.

The workspace route uses example records; authenticated Operations executes business workflows. Open registration, subscription checkout and independent-customer operating guarantees are not enabled. Displayed dashboard counts describe the latest 100 tasks, rather than authoritative all-time queue aggregates.

Successful work means the requested internal action committed with supporting evidence. Failed, blocked, rejected and cancelled work remain explicit states. A generated draft alone is not completion. This distinction is the reusable foundation for future executive assistance, marketing, commerce and technical-operations workflow packages.

Product expansion requires evidence: each package needs typed inputs, permitted accounts and tools, outcome rubrics, exception handling, approval rules and a deployment/rollback plan. Specialists are added when permission, knowledge or evaluation needs differ; adding agents by itself is not a product metric.

02 Architecture and data ownership

The presentation layer uses Next.js, React and TypeScript with a static export. Motion and Lenis support restrained interaction, scrolling and reduced-motion behavior. FastAPI and typed Python schemas provide the business control plane. PostgreSQL owns authentication, memberships, sources, task state, attempts, leases, events, approvals, commitments, receipts and episodes. Private local inference supplies the model endpoint.

The main path is browser to TLS/static edge, then authenticated API, transactional business records and a leased worker. The worker invokes fresh bounded Hermes role processes against the local model. Model outputs return through schema and evidence validation before any human decision or business mutation. Browsers have no direct database or inference access.

There is one business-workflow owner: PostgreSQL. Hermes is an execution activity rather than a competing scheduler. This avoids two systems claiming the same task or disagreeing about whether an action finished. If a separate workflow engine is introduced for cross-service processes, ownership and activity boundaries must be explicit; adding Temporal or n8n does not justify duplicate scheduling.

The deployed services include database, model, API, worker and core web. The public website can be released independently through a separate resource-limited web container. API and worker run with restricted privileges; database and model stay on a private network. Public ingress binds behind the existing TLS edge.

State has distinct lifetimes. Working context is transient and bounded. Business state is transactional. Episodic memory records completed work. Procedural knowledge belongs to reviewed skills; semantic knowledge belongs to versioned sources. None of these stores substitutes for an approval or provider receipt.

The initial topology has a single-host failure domain. Workspace filtering and resource-level authorization do not establish complete isolation between hostile independent customers. Future customer cells need enforced storage, process, credential, retrieval, browser, cache and network boundaries, plus explicit admission and recovery policies. Architecture guides describe the migration criteria rather than presenting a multi-region system as already deployed.

03 Hermes reasoning, context and knowledge

Nous Research's Hermes Agent is integrated through its AIAgent surface at a pinned revision. The application uses three fresh role subprocesses: coordinator, researcher and operator. Tools, shell/browser execution, unrestricted host access, autonomous skill changes and external-provider fallback are disabled in this path. Each role receives only the inference settings and selected task context it needs, without database or owner credentials.

The current knowledge path accepts bounded text. Each source has workspace ownership, source metadata, a version and content hash. Retrieval is restricted to active selected documents in the permitted workspace, ranked with PostgreSQL full-text functions. The context budget limits total source text and allocates space across the selected documents. This is an explicit baseline, not a claim of OCR, embeddings or graph retrieval.

Role output is untrusted. Typed contracts validate the draft, subject, summary, commitments and cited source identifiers. Sources outside the permitted set are rejected. Insufficient evidence produces a blocked or failed outcome instead of granting new access. Document instructions cannot authorize tools or turn retrieved text into a system permission.

Completed episodes and commitments are stored as application records. Corrections, retention and deletion must address the underlying business state and source snapshots; conversational memory alone cannot prove that an invoice, email or record change occurred. Passwords and API secrets must never enter knowledge or agent memory.

Local inference is mandatory for the current role path. The actual successful production run used one model call per role and about 14.5 minutes of model work in total. It is one measured run on CPU hardware, not a latency promise or reliability estimate. A local-only workspace must not silently fall back to a paid external endpoint.

Planned knowledge improvements include incremental document ingestion, native extraction before OCR, duplicate detection, deletion propagation, permission metadata, hybrid retrieval and reranking. Graph-based retrieval and compression strategies must outperform simpler baselines on held-out tasks. Prompt, skill, retrieval and model changes require versioned evaluations and controlled promotion.

04 Durable execution and exact approvals

Task submission is recorded before processing and uses workspace-scoped idempotency. Reusing a key with identical input returns the existing task; conflicting input is rejected. Persistent workers claim rows with transactional locking, a lease and an attempt identity. Runtime, retry count, input size, context and queue admission are bounded by configuration.

Lease fencing prevents a stale worker from settling work after ownership changes, cancellation or expiry. Interrupted read-only planning can recover into a new bounded attempt. Cancellation and rejection remain business transitions, not suggestions hidden in a prompt. Recovery does not require a second user submission or manual database edits.

The live verification deliberately interrupted the owned worker during the initial coordinator attempt. The same task recovered after natural lease expiry, completed all three roles and reached an approval decision. This shows the tested recovery path; it does not establish survival of every host, storage or network failure.

Approval binds the actor, workspace, task, operation, account, recipient, content, commitments, source identities/hashes and expiry. The operator sees the exact proposed artifact. Changing approved content or using a stale source invalidates the decision. The API locks the relevant rows and validates policy at execution time, outside the model.

For the current internal operation, draft, commitments, receipt, episode and final task state commit in one transaction. Altered approval and replay were rejected in the actual UI flow. A receipt identifies the completed internal transaction; it is not proof of mailbox delivery.

Future external execution follows propose, validate policy, review, approve, execute, verify and reconcile. Provider responses must distinguish accepted, completed, failed and unknown outcomes. A timeout after possible submission requires reconciliation before retrying. Stable provider identifiers, deduplication contracts and failure injection around every submission boundary are prerequisites for consequential connectors. Batch workflows must account for every input, including blocked and failed items.

05 Identity, isolation and security

Security is enforced by application and execution boundaries. Opaque session and CSRF tokens are hashed server-side. Production cookies use secure, HTTP-only and strict same-site settings as applicable to their purpose. Origin checks, CSRF validation, authenticated dependencies, workspace membership and task ownership protect business operations. Owner password rotation revokes existing sessions.

Authentication has per-identity limiting and an edge budget that also catches rotating-email floods. The final live probe used twelve distinct reserved test identities and received six expected unauthorized responses and six rate-limited responses. This verifies the defined control, not protection against every distributed attack.

API and worker run nonroot with read-only filesystems, dropped capabilities, bounded temporary storage and resource limits. Agents have no container-management socket or host administrator tools. Local inference and database access use private networking. The tested worker egress probes were blocked at selected destinations; a limited probe set is not universal network certification.

Threats include prompt injection, malicious source data, cross-workspace access, approval tampering, replay, stale workers, credential leakage, unsafe dependencies, webhook spoofing and resource exhaustion. Controls need negative tests at the API, retrieval, worker and provider boundaries. Source content never grants permissions, and a model-generated identifier is checked against permitted evidence.

Private secrets, recovery files, instance topology, backups and incident records are excluded from public source. The repository uses secret scanning, Python static analysis, Semgrep, tests, type checks and builds. Dependency pins and upstream hashes support provenance. Scanners are detection layers; they do not prove that the application is invulnerable.

Further customer readiness includes enterprise identity, strong tenant cells, per-account scoped secret stores, rotation/revocation exercises, outbound-network policy tests, sandbox hardening, admission limits and an incident response process. A compromised host administrator remains outside the protection offered by ordinary container restrictions. Security policy and reporting instructions are maintained in the repository.

06 Privacy and US/EU legal applicability

The platform processes account information, session/security records, source text, requests, drafts, commitments, receipts and workflow history. Selected context is sent to the configured private local model. Hosting and edge services also handle request metadata. Encrypted backups contain full database records, including authentication tables; encryption does not remove retention obligations.

Operators must define purposes, user population, controller/processor roles, permitted source material, retention periods, access rules, export/deletion procedures and incident ownership. Removing an active document from retrieval does not erase prior task snapshots, receipts, logs or backup copies. Complete self-service erasure and export are not current features. Restored data must respect deletion and retention decisions.

US requirements depend on operator activity, people served, data and jurisdiction. FTC privacy/security guidance addresses honoring representations and appropriate safeguards. State privacy and breach-notification laws require an applicability assessment; California privacy-policy obligations and CCPA thresholds are distinct considerations. The release does not provide health, children's, employment, credit or other specially regulated operating arrangements.

Where GDPR applies, assess lawful basis, transparency, minimization, storage limitation, rights handling, appropriate security, processor agreements and international transfers. Determine whether a DPIA or other review is needed for the intended processing. Breach reporting follows the applicable legal trigger and authority; a generic incident timer is not a complete legal procedure.

EU AI Act duties depend on role, intended use and the applicable provisions. AI interaction/content transparency and human-review expectations must be assessed against current official guidance. The product is not a compliance certificate, and human approval alone does not establish legal conformity. Generated material still needs factual, copyright, confidentiality and contractual review.

Public privacy, terms, cookies and accessibility notices should match actual behavior. No physical address is invented. The public contact channel is maintained at https://praxisforge.zahidul-islam.com/contact/. Operators should keep provider agreements, data maps, rights-request verification and incident records current before admitting new customers or enabling external applications.

Primary references: FTC privacy/security guidance; official GDPR text; European Commission AI Act guidance; relevant California official guidance. NIST AI RMF is voluntary risk-management guidance, separate from legal obligations and certification. Detailed source links appear in the repository's privacy/legal and security guides.

07 Integrations and governed improvement

The current active account is internal draft storage. Gmail, Drive, Telegram and external CRM delivery are not configured operational connectors. Connection status must identify what is active, unavailable or awaiting authorization. A browser session or framework integration surface does not establish permission to use every application feature.

The first connector package should integrate a small set deeply: Gmail, Google Drive, Telegram and one CRM. Each requires separate account identity, minimum scopes, OAuth lifecycle, refresh/revocation handling, synchronization state, verified webhooks, bounded retries, timeouts, rate limits and provider receipts. Public customer onboarding may require provider verification or a separate security assessment; test-account access does not establish that readiness.

MCP should reuse Hermes's supported integration surface while preserving authenticated user, tenant, account, operation, resource and budget. Typed contracts classify read/write behavior and required scopes. Protected-service tokens must have the intended audience; credentials must not be blindly forwarded to unrelated upstream services. Agent-to-agent exchange also needs durable task ownership and authorization boundaries.

Browser and coding workers are future adapters for workflows without an appropriate API. They need isolated customer sessions, restricted destinations, bounded downloads, disposable code workspaces, dependency controls, timeouts and evidence capture. Generated code is a deliverable only after it runs and meets its task contract.

English/Bangla and voice intake can expand task access, with transcript review and retained authentication/approval controls. Audio, images, document batches and OCR are adapters rather than reasons to grant unrestricted host access. Every batch item needs an outcome and a recoverable processing identity.

Skills need ownership, versions, permissions, dependencies, test fixtures, rollback and promotion gates. Candidate skills move through isolated tests, evaluation, human review and limited rollout before production. Memory updates, procedural skill changes and model-weight training are distinct operations.

The engineering lab remains separate from customer execution. Authorized de-identified datasets, prompt/retrieval baselines, classical ML, parameter-efficient adaptation, model routing and simulated agent-learning experiments require reproducible reports and held-out tests. Improvements must report regressions and quality/cost trade-offs rather than promote themselves automatically.

08 Build, configuration and deployment

Frontend, backend, configuration, deployment definitions, automated tests and public engineering guides are published together so reviewers can trace one release. Maintained UI content and prompts are data files. Typed settings own application, bootstrap, deployment and inference-child variables. Safe public examples are distinct from private instance configuration; secrets never belong in browser output or committed source.

Hermes is pinned by a full revision and archive checksum. The Python base image and service images use verified immutable identities. Public build preparation checks the official archive, limits download/expanded size, rejects unsafe paths and creates a fresh allowlisted Docker context. A metadata-only upstream dependency patch is documented; execution-code changes are not concealed.

CI runs frontend tests, types, lint and static export, plus backend regressions against a disposable real PostgreSQL database. Security jobs run Gitleaks, Bandit and Semgrep. Workflow actions are pinned and permissions minimized. Standard public GitHub runners are used; paid runners, model calls and automatic production deployment are outside this pipeline.

Release discipline begins with live drift detection and a retained snapshot. Changes are made locally, tested and reviewed, then assembled into a new immutable release with matching CSP, static assets and configuration. Services start privately before the owned proxy switches. Final verification checks file hashes, actual image/configuration identity, health, TLS, DNS and protected business state.

The latest website release matched all 159 release files. Both 73-file core releases and retained website/frontend artifacts were preserved. Core API/worker use the configuration correction while database, model and retained core web keep their verified identities. The public website can change without recreating business services.

Rollback is scoped and guarded. Retained artifacts are not permission to overwrite newer data or restore an old database over active work. After legitimate authentication or business changes, review a fresh compatible baseline instead of weakening a preservation guard. Deployment records and actual access material stay private; public guides use placeholder examples and explain required operator decisions.

09 Observability and the 99% objective

The future availability objective is 99% over a defined 30-day window. For a time-based calculation this permits 7.2 hours of unavailability. This is a planning SLO, not an achieved uptime measurement or contractual SLA. Define which authenticated API and web operations count, the observation point, interval, eligible requests, timeout threshold and error classification before reporting a percentage.

Track web/API availability separately from durable task completion, approval correctness and model latency. An accepted task is not a successful workflow. Useful indicators include queue age, lease recovery, failed/blocked outcomes, approval turnaround, verified receipts, retrieval failures, model/tool errors, cancellation latency and backup freshness. Workload-level measures must retain the tested scope and uncertainty.

The current release records task stages, attempts, timestamps, events, pinned engine identity and final outcomes. The dashboard is limited to its recent 100-task history. Broader authoritative queue aggregates, distributed tracing and notification delivery are roadmap work. A static product-status page is not a live monitoring service.

Future tracing should connect request identity, permitted context, model calls, tool contracts, approvals, artifacts and receipts. Logs need redaction and bounded retention. Diagnostic exports must remove tokens, personal content and private account details. Cost reporting should measure cost per successfully verified workflow, alongside quality and latency.

The error-budget policy should specify when reliability work takes precedence over new releases. Alert on user-facing burn rate and actionable symptoms, with an owner, escalation path and tested runbook. Investigate incidents using evidence, communicate impact and uncertainty, restore service, then record causes and corrective gates. Scheduled maintenance and third-party failures must be treated consistently in the published definition.

Reliability improvements include redundant stateless services, durable queue ownership, overload admission, bounded retries, capacity reserve, tested backups and recovery exercises. Replica count alone does not remove shared database, model or host failure domains. Availability reporting needs measured observations over the whole window; one successful workflow or container restart cannot establish the objective.

10 Future scale: 1M+ concurrent users

One million or more concurrent users is a future architecture target. Capacity planning must distinguish simultaneously connected sessions, active requests, task submissions and concurrent inference. A million registered accounts is not the same target. The current small-server deployment has no measured capacity near this scale.

The shared repository planning scenario assumes one million connected users, five percent making one API request per ten seconds: 5,000 requests per second before retries and bursts. If 0.01 percent of connected users submits one agent task per minute, arrival is about 1.67 jobs per second. At the observed roughly 866 seconds of serial model work per workflow, that hypothesis would require about 1,444 continuously busy equivalent workflow slots before headroom, or about 2,407 at 60 percent target utilization. These are planning calculations, not benchmark results; they show why user traffic and AI capacity require different controls.

The future edge serves cacheable static content independently from authenticated APIs. Stateless API replicas scale behind an appropriate load-balancing layer; sessions and authorization remain consistent. Connection pooling, cache permissions, partitioned business data, read replicas and indexed queries need measured limits. Evidence objects, ingestion pipelines and indexes must retain tenant authorization and deletion propagation.

Task admission, per-tenant budgets, queue limits, fair scheduling and backpressure protect inference and storage. Separate interactive requests from long-running agent jobs. Scale workers from queue age, measured service time and resource saturation; scale model serving from token throughput, context size, batch efficiency and memory capacity. Autoscaling needs adequate node capacity and cannot create free compute.

Progressive load gates should define workload mixes at increasing concurrency, then measure p50/p95/p99 latency, errors, queue growth, verified task success, database saturation and cost. Include bursts, noisy tenants, provider failures, worker loss and data-isolation tests. A browser/socket test must not be reported as a million simultaneous agent executions.

Scaling requires additional infrastructure, inference capacity and operating ownership when measurements justify it. The roadmap does not authorize paid resources. AWS, GCP and Azure milestones remain conditional on verified account-wide free eligibility; exhausted credits are not an allowance. Start with the simplest measured design and publish reproducible capacity reports before raising supported limits.

11 Evaluation, recovery and evidence

Verification covers complete business outcomes and failure boundaries. The released backend passed 23 real PostgreSQL acceptance regressions; the frontend passed 26 tests, type checks, lint and production build. Actual live public checks covered 20 routes, 30 responsive cases, seven accessibility scans and sixteen interactions with no JavaScript page errors. Authenticated operations passed 21 viewport checks and ten accessibility scans, including full engine revision visibility on a 320-pixel screen.

The live workflow recovered naturally after interruption, completed the three Hermes roles, rejected altered approval and replay, committed an internal receipt, persisted memory and supported receipt download, refresh and logout. Human comparison with the supplied evidence passed seven semantic checks. Editorial review remains appropriate; the result is not a statistical reliability estimate or proof that arbitrary tasks will succeed.

Source and dependency scanning reported no findings in the scoped final public checks. Sixteen actual configured secret/private values were compared in memory against all 152 exported public files, including common encodings, with no matches. Security evidence is bounded by tool coverage and test inputs; it is not a certification.

An explicitly authorized production PostgreSQL export was encrypted off-host, authenticated and restored into a fresh isolated database on the same server. Schema and eleven business tables with 26 rows matched the exported snapshot. Authentication tables were included in the dump; volatile session, login-attempt and heartbeat tables were excluded only from equality fingerprints. Tampering was rejected and the source was unchanged by the helper.

This proves logical database restoration, not full host replacement. Scheduled retention, alert delivery, WAL/PITR and measured host-failure RPO/RTO remain future work. Recovery procedures must protect keys separately, retain the last valid backup, validate clean restores and reconcile external actions before retries. Restoring data cannot undo a message already accepted by a future provider.

The evaluation roadmap adds held-out agency tasks, repeated stochastic runs, adversarial documents, stale sources, permission failures, broken integrations and calibrated human/automated rubrics. Prompt, skill, retrieval and model releases should be blocked when defined regressions occur. Reports must include failures, uncertainty, latency and cost, with reproducible input/version information and privacy-safe evidence.

12 Ten-phase roadmap and handover

Each phase advances frontend, backend/agent runtime, infrastructure and verification in parallel, with a deployed increment and exit evidence. The completed initial release supplies part of this foundation; it does not mark every phase complete. Detailed lane ownership, rollback and conditional cloud milestones are maintained in BUILD-PLAN.md and the roadmap guide.

1. Establish reproducible pinned runtime, configuration, security gates and operating baselines. 2. Deliver identity/workspaces, usable operations and early encrypted recovery. 3. Prove durable three-role execution, interruption recovery and task accounting. 4. Add permission-aware document ingestion, hybrid retrieval, memory correction/deletion and knowledge-quality evaluation.

5. Extend exact approvals into governed external execution with receipts, unknown-outcome reconciliation and duplicate prevention. 6. Integrate the initial Gmail/Drive/Telegram/CRM package with separate account identities and revocation tests. 7. Add production observability, budgets, backpressure, diagnostics and operational alerts.

8. Harden customer isolation, adversarial evaluation, full recovery and controlled admission. 9. Promote context/graph/skills/multilingual improvements only after measured baseline comparisons. 10. Run the isolated model/research lab, inference benchmarks, reviewed adaptation datasets and reproducible simulation experiments, then complete operational handover.

All cloud work must verify actual account eligibility and supporting-service costs first. An emulator is development evidence, not a provider deployment. Additional workflow packages reuse the same authorization, state, evaluation and recovery contracts. Production learning and customer credentials remain separated.

Handover includes source and release identity, configuration inventory, capability matrix, threat model, evaluation report, backup/restore procedure, incident ownership, operating limits and upgrade/rollback instructions. Sensitive access details belong only in the private recovery source. The public source is available at https://github.com/Zahidulislam2222/PraxisForge-AI; the operations entry point is the live product URL.

Selected primary references: Nous Hermes https://github.com/NousResearch/hermes-agent ; Kubernetes autoscaling https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/ ; SRE objectives https://sre.google/workbook/error-budget-policy/ ; FTC https://www.ftc.gov/business-guidance/privacy-security ; GDPR https://eur-lex.europa.eu/eli/reg/2016/679/oj ; EU AI Act https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai ; NIST AI RMF https://www.nist.gov/itl/ai-risk-management-framework . Repository guides retain the detailed source ledger and current implementation boundaries.