The Ideal Workflow for 100 Professions — and which parts of Knitweb each one touches

> A vision artefact from the Knitweb project · Published at knitweb.github.io · Version 2.0 · 2026

Introduction

This guide makes one claim, a hundred times over: in 2026 the ideal workflow for nearly every profession has converged on the same *shape* — a human captain directing AI agents that read and write a shared, content-addressed memory and produce verifiable, provenance-cited work. What differs is the **domain**, the **technical literacy** of the worker, and therefore **which parts of the system they ever touch**.

That last point is the new lens of this second edition. A software engineer builds on the `pulse` SDK and gateway directly; a farmer or a chef never sees an engine, a ledger, or a line of code — they meet Knitweb only through a friendly app and the shared memory underneath it. So each of the hundred entries below records not just the loop, but **which repositories the role uses (and how), which it deliberately avoids (and why), its technical level, and the entry point that fits that level** — and from that, a concrete list of **features the role is effectively requesting**. Those 490 requests, deduplicated, become the product backlog.

The professions are the variables. The loop is the constant. What changes most is how close to the substrate each role ever needs to stand.

How to read this

Each role lists: **Uses / Avoids** (which Knitweb repos it touches), **Level** (Low / Medium / High technical literacy), **Entry** (the right surface), and **Wants** (its feature requests). Read your own role, then one far from it — the loop repeats; the surface and the depth do not.

Software & Data

Software engineer

Builds real apps on knitweb primitives via the App/SDK layer, treating the content-addressed Web as a verifiable shared backend.

Intake a feature as a code task, plan against the App API (identity, economy, persistent Web, validation, provenance), then implement with agents pairing in virtualpc while every woven record/edge is content-addressed into the shared Web. Verify with property/interop tests over the same web_state_cid, then ship a signed bundle the gateway can serve over HTTP/JSON.

Uses: pulse (Builds on knitweb.gateway.App + knitweb.sdk: actor() identity from external ids, persisted shared Web, faucet/transfers, serve() over HTTP/JSON; runs nodes via the CLI) · **lens** (Calls the interpret lobe (retrieve/distill) for reasoning over the woven graph and wires its read-only /interpret gateway hook into the app) · **knitweb** (Treats the content-addressed Web/fabric as the canonical shared memory the app reads and writes records/edges into) · **virtualpc** (Runs paired coding agents to draft, test, and review implementation branches before the human opens PRs) · **gither** (Navigates the multi-repo pulse/lens/molgang layout and sequences cross-repo build/migration steps)

Avoids: bt (No off-grid Bluetooth-mesh need; the app reaches the fabric over normal HTTP/p2p, not a mesh transport) · vbank (Governance voting/treasury is an org concern, not part of building the app's feature code) · molgang (Only relevant as a domain template if building chemistry apps; a general engineer doesn't touch its game internals)

Level: High · **Entry:** Python SDK + knitweb gateway App layer (HTTP/JSON), with CLI for node ops

Wants: - **pulse** — Stable, versioned SDK/gateway App contract (semver, OpenAPI for serve()) so client code doesn't break when internals move — *High-literacy builders need a frozen API surface and machine-readable schema to integrate against confidently* - **pulse** — Production key custody hooks replacing clear-text dev wallets, with a from_seed warning path for non-custodial identities — *Engineers shipping real apps must not rely on spendable plain-seed identities the docs warn against* - **lens** — Deterministic /interpret gateway hook callable from any runtime with pinned web_state_cid and reproducible output order — *Determinism + provenance depth is exactly what a high-literacy builder requires to test and trust reasoning calls* - **gither** — Cross-repo build/dependency planner that resolves pulse<->lens<->molgang version compatibility — *Multi-repo navigation is only realistic with a tool that sequences builds and flags incompatible CID/contract drift* - **knitweb** — Content-addressed record/edge query API returning minimal CIDs with lazy full-record fetch — *Engineers need efficient, deterministic reads against the shared memory without pulling whole subgraphs*

Data scientist

Pulls verifiable, provenance-anchored data from the shared Web and runs reasoning/distillation over it with reproducible lineage.

Intake a question, plan a query/subscription scope against the fabric, then retrieve a deterministic CandidateSet and distill relations over it while agents propose feature selections. Verify by re-running against the same web_state_cid for byte-identical results and checking OriginTrail provenance ancestry, then ship a model/dataset whose inputs are content-addressed and citable.

Uses: lens (Primary surface: interpret.retrieve for deterministic structured retrieval and interpret.distill for bounded recursive reasoning/feature selection over candidates) · **knitweb** (Reads the content-addressed Web as the dataset substrate; every input record/edge is a CID with provenance ancestry) · **pulse** (Uses the SDK/App read paths and provenance walker + OriginTrail anchoring to pin and cite dataset lineage; pays PLS for compute jobs) · **monitor** (Explores the knowledge graph visually to scope subscriptions and sanity-check relations before querying)

Avoids: pulse (Touches read/SDK surfaces but NOT the L0-L2 engine (crypto, CBOR, p2p sync) - too low-level for data work) · bt (No off-grid need; datasets are pulled over normal network paths) · vbank (Treasury/voting governance is irrelevant to data modeling) · virtualpc (Single-analyst notebook flow rarely needs a multi-agent coordination runtime)

Level: High · **Entry:** Python SDK + notebooks over the interpret lobe (retrieve/distill); REST for pulling provenance-anchored datasets

Wants: - **lens** — retrieve()/distill() with strict determinism keyed on (query, subscription, web_state_cid) and logged loop/sub-call counts — *Reproducibility is the data scientist's core requirement; identical inputs must yield byte-identical CandidateSets* - **pulse** — Provenance ancestry export (full lineage graph as CIDs + OriginTrail receipts) for any dataset slice — *High-literacy users need deep provenance to cite, audit, and reproduce training inputs* - **lens** — Pluggable memory/feature-store backend behind distill so intermediate selections persist and are addressable — *Reusable, addressable intermediate artifacts let analysts iterate without recomputing* - **monitor** — Subscription-scoped knowledge-graph explorer with CID copy-out into notebooks — *Scoping queries visually then pulling exact CIDs bridges exploration and reproducible code* - **pulse** — Python SDK dataframe/iterator adapter over Web.traverse/neighbors with lazy full-record fetch — *Analysts need ergonomic, memory-efficient access to graph data in familiar tooling*

DevOps / platform engineer

Runs, scales, and observes knitweb nodes/hosts and the p2p fabric as production infrastructure.

Intake an ops change as a deploy task, plan node/host topology and peer/relay config, then roll it out via CLI/IaC while agents draft manifests and the persisted node state + content-addressed feeds give a verifiable rollback point. Verify node convergence (same web_state_cid across peers) and PoUW scheduler health, then ship with monitoring wired in.

Uses: pulse (Operates the node/host lifecycle via CLI (run node, peer/relay, payment, host status --json), tunes the pouw/scheduler compute guardrail, manages persisted state and serve() endpoints) · **monitor** (Runs dashboards for node health, peer convergence, PLS earned/spent/escrowed/slashed, and PoUW dispute rates) · **gither** (Plans multi-

repo builds/releases and coordinates migrations across pulse/lens deployments) · **virtualpc** (Coordinates multi-agent runners and bounded GPU jobs that must respect the compute guardrail) · **bt** (Optionally provisions off-grid Bluetooth-mesh transport for nodes in disconnected/edge environments)

Avoids: lens (Reasoning/orchestration is an app concern; platform engineers operate the substrate, not the interpret lobe) · molgang (A domain knitweb's game logic is out of scope for infra operation) · vbank (Governance treasury/voting is not an infrastructure responsibility)

Level: High · **Entry:** CLI + REST (gateway serve()) + infra-as-code; runs and operates knitweb nodes/hosts

Wants: - **pulse** — Long-running host service with peer discovery, relay, provider ops, health/readiness endpoints, and structured JSON logs — *Operators need a real daemon with observability hooks, not a dev-mode foreground process* - **monitor** — Operator dashboard for node convergence (web_state_cid drift), PoUW sampling/dispute rates, and PLS accounting — *Platform engineers must see fabric health and economic flows at a glance to run it safely* - **pulse** — Provider accounting reports (PLS earned/spent/escrowed/slashed) exportable as JSON/metrics — *High-literacy operators need machine-readable accounting for billing, alerting, and capacity planning* - **pulse** — Compute guardrail with enforced per-job time/resource bounds and quota policy in pouw/scheduler — *Bounded GPU work must be policy-enforced infra-side, not left to per-experiment discipline* - **gither** — Release/migration planner that gates dependent repo deploys on upstream CID-stability sign-off — *Coordinated multi-repo rollouts need a hard gate so a downstream deploy can't land before an upstream contract stabilizes*

QA / test engineer

Proves determinism and contract-conformance of the engine, knitwebs, and reasoning lobe across peers and versions.

Intake a behavior/contract to validate, plan property + interop + golden-CID cases, then run them while agents generate edge fixtures and replay traces against a pinned web_state_cid. Verify byte-for-byte agreement across clients and check the owner-direction/canonical guards, then ship green proofs recorded in the experiments ledger.

Uses: pulse (Runs the property/interop/knitweb proof suites, golden-CID and canonical-encoding tests, replay-protection and PoUW dispute scenarios via the test SDK/CLI) · **lens** (Tests interpret retrieve/distill determinism: same (query, subscription, web_state_cid) must yield identical CandidateSet/Selection) · **knitweb** (Validates content-addressed record/edge integrity and provenance ancestry as test oracles) · **molgang** (Uses a domain knitweb as an end-to-end fixture/template to exercise the full stack) · **monitor** (Reads graph/dashboard state to confirm expected post-conditions after a test run)

Avoids: bt (Mesh transport is out of the functional/contract test scope for most suites) · vbank (Governance flows are not part of engine/reasoning conformance testing) · virtualpc (Multi-agent orchestration is a tool QA may benefit from but rarely tests directly) · gither (Build navigation is a dev/ops tool, not a test surface)

Level: Medium · **Entry:** CLI test harness + Python test SDK; property/interop test suites over fixtures

Wants: - **pulse** — First-class deterministic test harness with golden-CID fixtures, cross-client interop replay, and a fixture generator — *Medium-literacy QA needs ready-made harnesses and templates rather than hand-rolling canonical-byte comparisons* - **lens** — Determinism conformance suite for the interpret lobe with seedable, pinned-web_state_cid replay — *Reasoning outputs must be provably reproducible; QA needs a turnkey way to assert that* - **pulse** — Plain-language assertion helpers (assert_same_cid, assert_provenance_intact) wrapping canonical/CBOR checks — *Guardrail-style helpers let QA write correct determinism tests without mastering the low-level encoding* - **knitweb** — Fixture catalog of canonical sample webs/records with known CIDs as shared test oracles — *Reusable golden fixtures make conformance tests portable across repos and contributors* - **monitor** — Test-mode snapshot diff view comparing expected vs actual graph state — *A visual diff lowers the literacy bar for diagnosing where a contract test diverged*

Security analyst

Audits the engine's cryptographic invariants, PoUW integrity, and provenance chains for tampering or equivocation.

Intake an audit scope or incident, plan checks against signatures (secp256k1+SHA-256), canonical bytes, replay-protection, PoUW sampling, and equivocation, then trace provenance ancestry over the content-addressed Web while

agents flag anomalies. Verify findings by independent re-execution and CID re-derivation, then ship an attested report with linked evidence CIDs.

Uses: pulse (Audits L0-L4: crypto, canonical CBOR/CID float-rejection, network-id replay protection, PoUW sampled re-execution + commit-before-sample challenge, equivocation detection in fabric/equivocation.py) · **knitweb** (Walks content-addressed provenance/ancestry as the tamper-evident audit trail and verifies record/edge integrity) · **lens** (Reviews the read-only /interpret gateway hook and export boundaries to ensure reasoning can't leak or mutate out-of-scope CIDs) · **vbank** (Reviews governance voting/treasury flows for authorization and quorum-integrity risks) · **monitor** (Uses the knowledge-graph explorer to visualize suspicious edges, dispute clusters, and slashing events)

Avoids: bt (Off-grid mesh transport is outside the value-path crypto/provenance audit scope unless a mesh-specific incident arises) · molgang (A domain knitweb's game content is not a core security surface beyond how it uses the engine) · gither (Build navigation tooling isn't a security-relevant runtime surface) · virtualpc (Agent coordination runtime is operational, not part of cryptographic/provenance auditing)

Level: High · **Entry:** CLI + Python SDK for forensic queries; REST/provenance walker for audit; reads engine source

Wants: - **pulse** — Provenance-depth audit API: full ancestry walk, signature re-verification, and CID re-derivation for any artifact — *High-literacy auditors need deterministic, deep provenance and independent re-derivation to prove or disprove tampering* - **pulse** — Equivocation/double-sign detection report with PoUW sampling-mismatch and slashing evidence linked by CID — *Security analysts require machine-verifiable evidence of integrity violations, not just dashboards* - **lens** — Enforced read-only export boundary on /interpret with per-call scope attestation — *The reasoning lobe must be provably unable to mutate or exfiltrate out-of-subscription CIDs* - **knitweb** — Tamper-evident attestation receipts (attest.py) covering record provenance with verifiable signatures — *An auditable, signed evidence trail is the substrate of any credible security finding* - **pulse** — Canonical-encoding fuzz/conformance hooks asserting float-rejection and byte-identical hashing across clients — *Auditors need to confirm the hash-critical surface cannot be coerced into divergence by malformed input*

Healthcare

Physician

Diagnose and treat patients, with agents drafting differentials and orders that the physician verifies and signs.

Intake pulls the patient's longitudinal record from the shared content-addressed memory (labs, imaging, prior notes as CID-pinned facts); a Lens reasoning agent proposes a ranked differential, workup plan, and draft orders with citations to the source facts. The physician edits and signs; the signed decision and its provenance chain are written back to knitweb-memory as a new immutable record so the next clinician sees exactly what was decided and why.

Uses: lens (Runs the diagnostic-reasoning agent: builds differentials, drug-interaction checks, and order drafts, each linked to the source facts it reasoned over.) · **knitweb** (The shared longitudinal patient record - every lab, note, and decision is a content-addressed, provenance-stamped fact the physician reads and appends signed conclusions to.) · **monitor** (Knowledge-graph explorer to trace why an agent suggested something - walk from a recommendation back to the labs/guidelines that produced it before signing.)

Avoids: pulse (Never sees the engine; CIDs, PLS accounting, and ledger mechanics are below the clinical surface and irrelevant to bedside work.) · bt (No off-grid Bluetooth-mesh need in a connected hospital; transport is invisible to the clinician.) · gither (Multi-repo build planning is a developer tool with no clinical use.) · virtualpc (Multi-agent runtime orchestration is plumbing the Lens app hides entirely.) · vbank (Treasury/voting governance is unrelated to direct patient care.)

Level: Medium · **Entry:** clinical web app on top of a Lens surface (with REST API access for EHR/order-entry integration)

Wants: - **lens** — Signed, citation-bound differential drafts - every suggested diagnosis/order must link to the exact source facts (lab CID, guideline CID) and require an explicit physician signature before it becomes part of the record. — *Medium-literacy clinicians need auditable, defensible reasoning they can verify rather than a black-box suggestion; the sign-off enforces human accountability.* - **knitweb** — Immutable, append-only clinical record with full provenance

chain and tamper-evident history. — *Medico-legal defensibility requires that no prior note can be silently altered and that any clinician can reconstruct who decided what, when, and on which evidence.* - **monitor** — One-click 'explain this recommendation' graph view tracing a suggestion to its evidence. — *A physician with medium technical literacy needs to audit agent reasoning visually and fast, mid-consult, without reading raw data structures.* - **lens** — Real-time drug-interaction and contraindication guardrails that block unsafe draft orders before sign-off. — *Safety guardrails catch errors at the moment of ordering, matching a clinician who wants assistance but not a tool that can auto-commit harm.* - **knitweb** — REST API for bidirectional EHR/order-entry sync. — *Physicians work inside existing hospital systems; the shared memory must integrate, not become yet another login.*

Nurse

Deliver and document care at the bedside, with agents pre-filling tasks and flagging changes the nurse confirms.

At shift intake the app pulls the patient's active care plan and open tasks from the shared memory and presents them as a plain-language checklist; the nurse does the care (meds, vitals, observations) and confirms each step with a tap, an agent timestamping and writing it back. Verification is a built-in barcode/photo confirmation, and abnormal vitals trigger an agent alert the nurse acknowledges, with every action shipped to knitweb-memory for the next shift's handover.

Uses: knitweb (Reads the active care plan and writes each completed task/vital observation back as a confirmed, timestamped record feeding the shift handover.) · **lens** (Agent that turns the care plan into a simple task checklist, flags out-of-range vitals in plain language, and drafts the handover summary.)

Avoids: pulse (Engine internals (CIDs, ledger) are entirely hidden; a bedside nurse never touches the substrate.) · monitor (Graph-explorer dashboards are too technical and desk-bound for fast bedside use.) · bt (No off-grid transport need on a connected ward.) · gither (Developer multi-repo tooling with zero clinical relevance.) · virtualpc (Agent-runtime orchestration is invisible plumbing.) · vbank (Governance and treasury are outside the nursing role.)

Level: Low · **Entry:** consumer mobile app / tablet bedside app (template-driven, plain-language)

Wants: - **lens** — Plain-language, template-driven task checklist generated from the care plan, with large tap-to-confirm buttons. — *Low technical literacy plus a fast, hands-busy bedside context demands guardrailed templates and minimal typing, not free-form data entry.* - **lens** — Automatic abnormal-vitals alerting with a one-tap acknowledge-and-escalate. — *A guardrail that surfaces danger and routes escalation suits a role that must act fast without interpreting raw numbers.* - **knitweb** — Auto-generated shift-handover summary assembled from the shift's confirmed records. — *Removes error-prone manual handover and guarantees the next nurse inherits a complete, provenance-stamped picture.* - **knitweb** — Barcode/photo med-administration confirmation written as a verified record. — *Built-in verification gives a low-literacy user a foolproof right-patient/right-drug check that is also an audit trail.* - **lens** — Offline-tolerant mobile capture that queues and syncs when reconnected. — *Bedside connectivity is unreliable; the nurse needs the app to never block care or lose a confirmation.*

Pharmacist

Verify, dispense, and reconcile medications, with agents cross-checking interactions and dosing against the shared record.

Intake pulls the patient's full active medication list and allergies from the shared memory; a Lens agent runs interaction, dose, renal-adjustment, and duplicate-therapy checks and presents flagged issues in a structured worklist. The pharmacist reviews each flag, approves or intervenes (contacting the prescriber), and ships the verified dispensing decision plus any intervention back to knitweb-memory as a provenance-linked record.

Uses: lens (Runs the interaction/dosing/duplicate-therapy checking agent and produces a structured, ranked worklist of flags with rationale.) · **knitweb** (The authoritative shared medication record and allergy list the pharmacist reconciles against and writes verified dispensing decisions into.) · **monitor** (Knowledge-graph view to trace a flag back to the specific orders/labs that triggered it before accepting or overriding.)

Avoids: pulse (Ledger and CID mechanics are below the clinical surface; the pharmacist never touches the engine.) · bt (No off-grid mesh need in a pharmacy.) · gither (Developer build tooling, no pharmacy use.) · virtualpc (Agent-runtime orchestration is hidden plumbing.) · vbank (Governance/treasury is outside the dispensing role.)

Level: Medium · **Entry:** spreadsheet-like / structured web app over a Lens surface (with REST API for dispensing-system integration)

Wants: - **lens** — Ranked interaction/dose/duplicate-therapy worklist with explicit rationale and a required accept-or-intervene action per flag. — *A medium-literacy verifier needs structured, auditable flags they can adjudicate, not raw alerts, and the forced action ensures nothing is silently passed.* - **knitweb** — Single reconciled medication source-of-truth with full provenance across prescribing, dispensing, and administration. — *Medication reconciliation depends on one tamper-evident list every party agrees on, eliminating conflicting copies.* - **monitor** — Trace-a-flag graph view linking an alert to its causal orders and labs. — *The pharmacist must justify overrides; visual provenance lets them audit the agent's reasoning before accepting or rejecting.* - **lens** — Override-with-documented-reason capture that records the clinical justification on the record. — *Defensible practice requires that every override is a documented, attributable decision, not a dismissed popup.* - **knitweb** — REST API to sync verified decisions with the dispensing/inventory system. — *Pharmacists work in existing dispensing software; the shared memory must integrate to be adopted.*

Physiotherapist

Assess movement and run rehab programs, with agents proposing exercise plans and tracking progress the therapist adjusts.

Intake pulls the referral, diagnosis, and prior session notes from the shared memory; a Lens agent proposes a templated rehab program (exercises, sets, progression) the therapist tailors with simple drag/edit. After the session the therapist logs progress and pain/range measures via guided forms, and the agent updates the plan and ships the session record plus next-visit goals to knitweb-memory for continuity across the care team.

Uses: knitweb (Reads referral and prior-session history and writes each session's progress, measures, and updated goals as a continuity record.) · **lens** (Agent that drafts and progresses templated rehab programs and summarizes progress trends in plain language.)

Avoids: pulse (Engine internals are fully hidden; a therapist never touches the substrate.) · monitor (Graph-explorer tooling is too technical for a template-driven clinic workflow.) · bt (No off-grid transport need in clinic.) · gither (Developer tooling, irrelevant.) · virtualpc (Agent orchestration is invisible plumbing.) · vbank (Governance/treasury outside the role.)

Level: Low · **Entry:** no-code web app / tablet app (template library, plain-language)

Wants: - **lens** — Drag-and-edit rehab-program templates with built-in safe-progression guardrails (no unsafe jumps in load). — *Low technical literacy plus clinical safety needs no-code templates and progression guardrails rather than free-form programming.* - **knitweb** — Visual progress timeline (range-of-motion, pain, function) assembled from session records. — *A plain-language trend view lets a non-technical therapist and patient see improvement without reading data tables.* - **lens** — Auto-drafted, plain-language next-visit goals and home-exercise sheet for the patient. — *Reduces admin and gives a low-literacy user patient-ready output generated from the record automatically.* - **knitweb** — Shared continuity record readable by the wider care team (physician, nurse). — *Rehab progress must feed back into the patient's single shared memory so care is coordinated, not siloed.* - **lens** — Guided structured-form capture for measures (sliders, pickers) instead of free text. — *Templates and guardrailed inputs match low literacy and keep the data clean enough for agents to track trends.*

Veterinarian

Diagnose and treat animals across species, with agents adapting dosing and differentials per species and syncing field visits.

Intake pulls the animal's record (species, weight, history, herd context) from the shared memory; a Lens agent proposes species-aware differentials and weight/species-adjusted dosing the vet verifies. In the field with no connectivity, records are captured locally and relayed over the off-grid mesh, then reconciled into knitweb-memory on reconnection so clinic and field records stay one provenance-stamped whole.

Uses: lens (Runs species-aware diagnostic and dosing agents (per-species drug formularies, weight-based calculations) with verifiable rationale.) · **knitweb** (The shared per-animal/per-herd record read at intake and appended

with verified diagnoses, treatments, and field observations.) · **monitor** (Knowledge-graph view for herd-level patterns and tracing a recommendation to its evidence.) · **bt** (Off-grid Bluetooth-mesh transport so rural/large-animal field visits capture and relay records without connectivity, syncing later.)

Avoids: pulse (Engine/ledger internals stay below the clinical surface even though the field mode relies on them indirectly; the vet never touches CIDs or PLS.) · gither (Developer multi-repo tooling, no clinical use.) · virtualpc (Agent-runtime orchestration is hidden plumbing.) · vbank (Governance/treasury is outside clinical practice.)

Level: Medium · **Entry:** clinical web app over a Lens surface, with offline/bt-backed field mode for rural/large-animal work

Wants: - **lens** — Species-aware dosing and differential engine with per-species formularies and weight-based calculators, plus citation-backed rationale. — *Cross-species range is the defining vet hazard; a medium-literacy clinician needs verifiable, species-correct math, not a human-medicine model.* - **bt** — Robust off-grid field-capture-and-relay with automatic conflict-safe reconciliation on reconnect. — *Rural and large-animal work genuinely needs off-grid transport - the one healthcare role with a real bt use case - and records must merge cleanly when back online.* - **knitweb** — Herd/flock-level aggregate records linking individual animals to a group context. — *Veterinary decisions are often population-level; the shared memory must model herds, not just single patients.* - **monitor** — Herd-pattern outbreak/trend graph explorer. — *A vet with medium literacy needs to spot population-level disease patterns visually, which a graph view over the shared memory enables.* - **lens** — Provenance-stamped treatment records suitable for food-safety/withdrawal-period compliance. — *Production-animal treatment carries regulatory withdrawal-period obligations; auditable provenance turns the record into a compliance artifact.*

Legal & Finance

Lawyer

Drafts, reviews and reasons over contracts and case files against a shared, citable memory of clauses, statutes and prior matters.

Intake: lawyer uploads a matter (contract, brief, correspondence) into a friendly Lens app, which weaves it into the shared content-addressed memory as immutable, citable records. Plan/Do: a Lens reasoning agent retrieves relevant clauses, statutes and precedents from that memory and distills a draft or risk-flag list, every assertion carrying a provenance link back to a source CID. Verify/Ship: the lawyer reviews each cited passage (no un-sourced claims surface), accepts or edits, and ships the final document while the reasoning trail is pinned in memory for later audit or dispute.

Uses: lens (Primary surface: retrieve-and-distill reasoning over contracts/precedents, with provenance-gated output so every drafted statement links to a source clause.) · **knitweb** (The shared content-addressed memory where matters, clauses and reasoning trails live as immutable, citable records the lawyer can re-reference.)

Avoids: pulse (Never sees the engine; CIDs, ledger and P2P are too low-level and irrelevant to drafting.) · bt (No off-grid Bluetooth-mesh need in a law office; always online.) · gither (Multi-repo build navigation is developer infrastructure with no legal use.) · virtualpc (Multi-agent coordination runtime is plumbing hidden beneath the Lens app.) · molgang (A chemistry domain template; unrelated to legal work.)

Level: Low · **Entry:** no-code web app (a friendly Lens surface) with a document/chat panel

Wants: - **lens** — Mandatory citation-or-silence mode: the agent must attach a source CID to every clause/risk it asserts, and refuses (rather than fabricates) when no provenance exists. — *A low-literacy lawyer can't audit a vector store; they need a hard guardrail against hallucinated case law they would be liable for.* - **lens** — Plain-language 'why this clause' explainer that renders the reasoning chain as readable bullet points with click-through to the original text. — *Matches Low technical literacy; the lawyer needs to trust output without reading CIDs or graph queries.* - **knitweb** — Matter-scoped private memory with simple share toggles (this client only / firm-wide). — *Confidentiality is non-negotiable; the surface must express scoping without exposing the underlying subscription/scope mechanics.* - **monitor** — Read-only timeline view of a matter showing every document and AI-drafted version as immutable, timestamped, source-linked entries. — *Gives a non-technical user a defensible chain-of-custody for litigation without*

touching graph internals. - **lens** — One-click redline export to Word/PDF that embeds footnoted provenance links. — *Lawyers live in documents, not apps; the deliverable must leave the system in their native format with citations intact.*

Accountant

Reconciles transactions, classifies entries and produces filings against a shared, immutable record of source documents and prior treatments.

Intake: receipts, bank feeds and invoices are imported into a spreadsheet-like Lens app and woven into shared memory as immutable source records. Plan/Do: a Lens agent proposes classifications and reconciliations by retrieving the client's prior treatments and applicable rules from memory, leaving a provenance trail from each booked entry to its source document. Verify/Ship: the accountant reviews exceptions, approves the deterministic ledger, and ships the period close; the immutable trail makes a later audit a replay rather than a reconstruction.

Uses: lens (Suggests classifications/reconciliations and answers 'how did we book this last year' by distilling over prior entries with source links.) · **knitweb** (Holds source documents, the chart of accounts and every booked entry as immutable, content-addressed records for replayable closes.) · **monitor** (Dashboard view of balances, exceptions and the document-to-entry graph for the close.)

Avoids: pulse (The PLS/CID ledger engine is the substrate; the accountant works on a financial ledger surface, never the protocol ledger internals.) · bt (No off-grid mesh requirement in accounting.) · gither (Developer multi-repo tooling with no accounting relevance.) · virtualpc (Agent-coordination runtime hidden under the app.) · vbank (Governance/voting is not part of bookkeeping; treasury voting is an org-governance concern, not the close.)

Level: Low · **Entry:** spreadsheet-like UI / no-code web app with ledger and document import

Wants: - **lens** — Rule-and-template classification packs (e.g. local VAT/GAAP) with plain-language prompts the accountant fills in, not free-form queries. — *Low literacy means guardrailed templates beat open-ended AI; reduces misclassification risk.* - **knitweb** — Immutable, replayable period snapshots: a sealed CID for each close that can be re-opened read-only. — *Accountants need a defensible, tamper-evident close; content addressing gives that for free if surfaced simply.* - **lens** — Every booked entry shows a one-tap 'source document' link and a confidence flag on AI-suggested classifications. — *Builds trust and keeps a non-technical user in control of what the agent proposed.* - **monitor** — Exception dashboard: unreconciled items, duplicate detection and missing-document alerts surfaced as a simple worklist. — *Mobile/visual triage suits Low literacy far better than running graph queries.* - **knitweb** — Spreadsheet-style import/export (CSV/XLSX) that round-trips into memory without losing provenance. — *Accountants live in spreadsheets; the bridge must be frictionless and preserve the audit trail.*

Financial analyst

Builds models and theses by querying a shared, provenance-rich knowledge graph of filings, metrics and prior analyses with reproducible agent assistance.

Intake: the analyst pulls filings, market data and internal memos into a notebook via the SDK, weaving them into shared memory with provenance. Plan/Do: a Lens agent retrieves comparables and prior models from the graph and distills a draft thesis or metric set, each figure traceable to a source CID; the analyst iterates in code. Verify/Ship: the analyst re-runs the query against a pinned memory-state CID to confirm reproducibility, then ships the model and memo back into memory for the desk to reuse.

Uses: lens (Programmatic retrieve+distill over the knowledge graph to assemble comparables, summarize filings, and surface contradictions, with provenance on every figure.) · **knitweb** (The shared memory of filings, metrics and prior analyses, addressed by CID so a model can be pinned to an exact graph state.) · **monitor** (Knowledge-graph explorer to visually trace how an entity links to filings, peers and prior theses before coding against it.)

Avoids: pulse (Uses the SDK's high-level memory/Lens APIs, not the raw CID/encoding/P2P internals of the engine.) · bt (No off-grid transport need on a connected desk.) · gither (Build-planning across repos is not part of financial analysis.) · virtualpc (Multi-agent runtime is invoked indirectly through Lens, not operated directly.) · vbank (Governance voting is outside the analyst's modeling remit.)

Level: Medium · **Entry:** Python SDK / notebook plus a spreadsheet-like UI for ad-hoc pulls

Wants: - **lens** — Reproducible-query API: pass a query plus a memory-state CID and get byte-identical results on re-run. — *Medium-literacy analysts demand determinism so a thesis can be defended and rebuilt months later.* - **lens** — Provenance-depth control: ask for the full ancestry chain (filing -> footnote -> derived metric) on any number the agent reports. — *Analysts must trace a figure to primary source to trust it in a model.* - **knitweb** — Python SDK for scoped graph queries and weaving derived analyses back as first-class CID nodes. — *Matches notebook-driven workflow; lets derived metrics become reusable, citable graph nodes.* - **monitor** — Entity-comparables graph view with export to the SDK session. — *Visual exploration before coding speeds comparable selection without leaving the analyst's toolchain.* - **lens** — Contradiction/stale-data flags when two sources in memory disagree on a metric. — *Surfacing conflicts is higher-value than a confident single answer for someone building a defensible model.*

Auditor

Independently verifies that reported figures and reasoning trace deterministically to attested source records, replaying rather than trusting.

Intake: the auditor receives a sealed period snapshot CID and the client's report. Plan/Do: via the SDK/CLI they replay each assertion against shared memory, walking provenance ancestry and checking attestations and signatures back to original source documents, and re-run any Lens-distilled conclusion deterministically to confirm it reproduces. Verify/Ship: discrepancies (unattested entries, broken provenance, non-reproducible figures) are flagged; the auditor ships a signed audit opinion whose own evidence chain is itself woven into memory as immutable, independently re-verifiable records.

Uses: pulse (Directly exercises provenance, attestation and signature-verification primitives and content-addressing to replay and prove the integrity of records.) · **knitweb** (The immutable shared memory under audit; every claimed figure must resolve to an attested CID with intact ancestry.) · **lens** (Re-runs any agent-distilled conclusion deterministically against a pinned state to confirm it reproduces and is provenance-gated.) · **monitor** (Knowledge-graph explorer to inspect ancestry chains and spot orphaned or unanchored nodes.)

Avoids: bt (Off-grid mesh transport is irrelevant to verifying records in a connected engagement.) · gither (Build-planning tooling is not an audit surface.) · virtualpc (Agent-coordination runtime is out of scope; the auditor verifies outputs, not how agents were scheduled.) · molgang (A chemistry domain template unrelated to financial audit.)

Level: High · **Entry:** CLI + Python SDK against the provenance/attestation APIs

Wants: - **pulse** — Provenance-ancestry verification API: given any record CID, return the full attested chain to primary sources and fail loudly on any gap or unsigned hop. — *High literacy; the auditor's entire job is replaying integrity, not trusting a summary.* - **pulse** — Deterministic replay/digest endpoint that proves a sealed snapshot CID reconstructs bit-identically. — *Determinism is the audit invariant; an auditor must show the close cannot have been altered.* - **lens** — Provenance-gate enforcement report: an attestation that every relation in a distilled conclusion was source-backed, with any dropped/fabricated candidates logged. — *Lets the auditor sign off on AI-assisted conclusions because un-sourced claims are provably excluded.* - **knitweb** — Tamper-evidence diff: detect and pinpoint any record whose content no longer matches its CID, or whose anchor is missing. — *Content addressing makes tampering detectable; the auditor needs it surfaced as concrete findings.* - **pulse** — Signed, content-addressed audit-opinion record type so the auditor's own evidence chain is independently re-verifiable. — *An audit trail that is itself immutable and attested closes the loop and supports peer review/regulator inspection.*

Insurance underwriter

Prices and decides risk by combining application data with a shared memory of loss history, exposure facts and prior decisions, with traceable rule application.

Intake: an application and supporting documents arrive through the underwriting app (or via REST from the policy system) and are woven into shared memory. Plan/Do: a Lens agent retrieves comparable risks, loss history and applicable guidelines, distills a recommended rating/decision, and shows which rules and data points drove it, each linked to a source CID. Verify/Ship: the underwriter reviews the rationale, overrides where judgment differs (the override itself recorded), and ships the bound decision; the full basis is pinned in memory for regulatory and reinsurance scrutiny.

Uses: lens (Retrieves comparables/loss history and produces an explained rating recommendation with each factor traced to source data and a guideline.) · **knitweb** (Shared memory of exposure facts, prior decisions and guidelines, addressed by CID so a decision's full basis is immutable and re-citable.) · **monitor** (Portfolio/exposure dashboard and risk-relationship graph to sanity-check accumulation before binding.)

Avoids: pulse (Consumes high-level Lens/memory and REST surfaces; the CID/ledger/P2P engine stays hidden.) · bt (No off-grid mesh need in underwriting operations.) · gither (Developer build navigation has no underwriting use.) · virtualpc (Agent-coordination runtime is invoked through Lens, not operated directly.) · vbank (Treasury/governance voting is unrelated to risk pricing.)

Level: Medium · **Entry:** no-code/low-code web app with a REST API for system integration

Wants: - **lens** — Explainable decision card: every rating shows the driving factors, the guideline applied, and a source link for each, in plain language. — *Medium literacy plus regulatory duty to explain declines/pricing means transparency must be built in, not reconstructed.* - **knitweb** — Immutable decision-basis record capturing inputs, agent recommendation and the underwriter's override at the moment of binding. — *Regulators and reinsurers demand a frozen, defensible basis for each bound risk.* - **lens** — Guideline/guardrail packs that constrain the agent to approved underwriting rules and appetite, refusing out-of-appetite binds. — *Guardrails protect a non-expert-coder underwriter from the agent recommending non-compliant terms.* - **monitor** — Accumulation/exposure graph showing correlated risks already on the book before a new bind. — *Visual accumulation control prevents silent concentration, suited to a Medium-literacy user.* - **knitweb** — REST API to ingest applications from and push decisions back to the core policy-admin system. — *Underwriters work inside an existing policy system; integration, not a standalone app, makes the workflow real.* - **lens** — Reproducible-rationale pin: re-running the same application against the same memory-state CID yields the same recommendation. — *Determinism supports dispute handling and demonstrates non-arbitrary, consistent pricing.*

Science & Research

Chemist

Propose, react, and validate molecular and reaction knowledge as content-addressed records that peers can re-verify.

Intake a target molecule or reaction in the chemistry-knitweb app; a Lens agent plans candidate routes by retrieving prior reactions from the shared content-addressed memory and ranking them by provenance depth. The chemist runs/records the reaction, peers vote (PLS pulses) to validate the bond/Fiber, and on a confirm-quorum the verified record is woven into the Web and anchored so the next query reuses it.

Uses: molgang (The chemistry-knitweb template is the friendly surface: propose bonds/molecules, see peer validation, build a verified collection of reactions) · **lens** (Reasoning/orchestration over the web: retrieves prior reactions, plans synthesis candidates, ranks by provenance — the chemist never writes the traversal) · **knitweb** (The shared content-addressed memory/fabric where every validated reaction/molecule lives as a re-fetchable Fiber) · **monitor** (Knowledge-graph explorer to browse the reaction graph and confirm a record was woven and validated)

Avoids: pulse (Never touches the ledger/CID engine internals; the app and SDK hide it) · bt (No off-grid Bluetooth-mesh need in a connected lab) · gither (Multi-repo build planning is irrelevant to a bench chemist) · virtualpc (Does not author multi-agent runtimes; consumes a single Lens agent)

Level: Medium · **Entry:** Domain web app (chemistry-knitweb / MOLGANG-style surface) plus a Python SDK for batch work

Wants: - **molgang** — Reaction/structure import templates (SMILES, MOL, common reagent presets) with guardrails that flag chemically-impossible bonds before proposal — *Medium literacy: needs structured templates and sanity-guardrails, not raw record construction* - **lens** — Plain-language synthesis-route query ('how do I make X from Y') that returns ranked routes with their provenance roots — *Lets a chemist orchestrate retrieval without writing graph traversals* - **knitweb** — Per-molecule provenance card: every validated record links to its raw-material origins and processing steps — *Reaction trust requires seeing the full derivation chain, not just a result* - **monitor** — Reaction-graph filter view (by element, functional group, validation status) — *Browsing the woven chemistry graph needs domain*

filters at medium literacy - **molgang** — Peer-validation status indicator (proposed / under-quorum / confirmed) on each record — *Chemist needs to know which knowledge is peer-verified before relying on it*

Biologist

Capture assays, samples, and observations as linked, content-addressed records so findings are traceable and reusable across the lab.

Intake sample/assay metadata via a spreadsheet-like form; a Lens agent plans which prior records and protocols to pull from the shared memory and suggests linked entities. The biologist records results, the records are woven into the Web with derived-from links, and a provenance walk verifies the lineage before sharing/anchoring the dataset.

Uses: knitweb (Shared content-addressed memory for samples, assays and observations, each a re-fetchable record with typed links) · **lens** (Retrieves matching protocols/prior results and proposes entity links over the web on the biologist's behalf) · **molgang** (Used only as the domain-knitweb template pattern — a biology knitweb is instantiated the same way the chemistry one is) · **monitor** (Knowledge-graph explorer to inspect sample lineage and dataset relationships)

Avoids: pulse (Engine is too low-level; the form app and SDK abstract CIDs and ledger away) · bt (No off-grid mesh requirement for in-lab capture) · gither (Not building or navigating multiple code repos) · vbank (No voting/treasury governance role in routine assay capture)

Level: Medium · **Entry:** No-code / spreadsheet-like web app over a biology domain-knitweb, with optional Python SDK

Wants: - **knitweb** — Schema-templated record types for common bio entities (sample, assay, organism, protocol) with required-field guardrails — *Medium literacy: needs ready-made schemas so records link correctly without manual modeling* - **lens** — Protocol-suggestion query that pulls the closest prior protocol and pre-fills the form — *Reduces manual entry and reuses validated lab knowledge* - **knitweb** — Sample-lineage provenance view (parent sample -> derivatives -> results) — *Biological traceability depends on full derived-from chains* - **monitor** — Mobile-friendly read-only dataset/lineage viewer for field or bench checks — *Low-to-medium literacy benefits from a phone-readable view away from a workstation* - **lens** — Duplicate/near-match detection on intake ('a similar sample exists') — *Prevents redundant records and keeps the shared memory clean without expert dedup logic*

Academic researcher

Build cited, reproducible claims whose every input is a content-addressed, peer-validatable record.

Intake a research question; a Lens agent plans a literature/data retrieval over the shared content-addressed memory and assembles a candidate evidence set with provenance. The researcher analyses and drafts claims, links each claim to its source records, and ships a dataset/paper whose citations resolve to anchored Fibers others can re-verify.

Uses: knitweb (Shared content-addressed memory holding datasets, claims and citations as immutable, re-fetchable records) · **lens** (Orchestrates evidence retrieval and ranks candidates by provenance depth for a defensible claim chain) · **monitor** (Knowledge-graph explorer to visualize citation/claim networks and check what derives from what) · **docs** (Reads the protocol/record-format documentation to cite and structure records correctly)

Avoids: pulse (Does not need the P2P/ledger engine; works at the memory + Lens layer) · bt (No off-grid transport need for desk-based research) · gither (Not coordinating multi-repo builds) · virtualpc (Uses prebuilt Lens agents rather than authoring a coordination runtime)

Level: Medium · **Entry:** Web app / notebook-style surface over the shared memory, plus a Python SDK for analysis

Wants: - **knitweb** — Stable citation handles: a CID-backed permalink for any record that never breaks — *Reproducibility needs immutable, content-addressed references over fragile URLs* - **lens** — Evidence-set builder that returns sources with confidence and full provenance roots — *Claims must be backed by a traceable, ranked evidence chain* - **monitor** — Citation/claim graph view with 'what cites this' and 'what this derives from' both directions — *Researchers reason about evidence networks, needing bidirectional graph navigation* - **knitweb** — Reproducibility bundle export: a claim plus the closure of all records it depends on — *Lets reviewers re-run/re-verify the full input set deterministically* - **docs** — Plain-language guide to record/claim formatting and provenance linking — *Medium literacy: needs documentation, not engine internals, to participate correctly*

Clinical-trial coordinator

Run trial logistics on guardrailed forms while every consent, visit, and event becomes an auditable, tamper-evident record.

Intake participant/visit data through a guided mobile/no-code form; a Lens agent plans the next protocol step, surfaces the right consent template, and flags missing items. The coordinator confirms in plain language, the event is woven into the shared content-addressed memory as a signed, immutable record, and an audit/provenance view verifies the full trail before reporting.

Uses: lens (The entire experience: a plain-language assistant that plans the next step, validates inputs, and explains records — the coordinator never sees the graph directly) · **knitweb** (Shared content-addressed memory stores consents, visits and adverse events as tamper-evident, auditable records (behind the app)) · **monitor** (Read-only audit/timeline dashboard to confirm the trail is complete before reporting)

Avoids: pulse (Never sees the engine; far too low-level for a low-literacy operational role) · bt (No off-grid mesh need for clinic-based workflows) · gither (Not a developer; no multi-repo navigation) · virtualpc (Consumes a single friendly agent, never authors a multi-agent runtime) · molgang (Chemistry-game template is irrelevant to trial logistics)

Level: Low · **Entry:** Consumer-grade / no-code web + mobile app over a friendly Lens surface

Wants: - **lens** — Step-by-step guided protocol walkthrough in plain language with hard stops on missing/invalid data — *Low literacy: needs strong guardrails and a wizard, not freeform record entry* - **lens** — Consent/eligibility templates with auto-fill and validity checks — *Templates remove the need to understand record schemas* - **knitweb** — Immutable, signed audit-trail records with a human-readable 'who/what/when' card per event — *Regulatory audit needs tamper-evidence shown without technical jargon* - **monitor** — Mobile read-only participant timeline and completeness checklist — *Low-literacy coordinators work on phones and need at-a-glance completeness, not a graph UI* - **lens** — Plain-language anomaly/missing-data alerts ('Visit 3 consent not recorded') — *Surfaces problems without the user querying the data themselves*

R&D engineer

Wire experiments, models, and verifiable compute into reproducible, content-addressed pipelines with deterministic provenance.

Intake a research/engineering task as code; the engineer drives Lens programmatically to retrieve inputs from the shared content-addressed memory, dispatches verifiable compute (proof-of-useful-work) via the SDK, and pins outputs as CIDs. Results are woven into the Web with explicit derived-from edges, re-verified by sampled re-execution, and the deterministic provenance closure is exported for downstream reuse.

Uses: knitweb (Shared content-addressed memory and verifiable-compute substrate accessed for deterministic inputs/outputs and re-execution) · **lens** (Programmatic reasoning/orchestration: scripts retrieval, planning, and provenance-ranked candidate selection via SDK) · **pulse** (The one science role that legitimately touches the engine: SDK/CLI for canonical CIDs, the ledger, PLS-metered compute, and provenance/anchor APIs) · **gither** (Multi-repo navigator/build planner to coordinate pulse + lens + a domain-knitweb across builds) · **monitor** (Dashboards to verify pipeline runs, node health, and the woven knowledge graph) · **virtualpc** (Multi-agent coordination runtime when an experiment needs several agents orchestrated)

Avoids: bt (Off-grid Bluetooth-mesh transport is irrelevant to a connected R&D pipeline (skip unless field-deployed)) · vbank (Governance voting/treasury is outside an engineering experiment loop)

Level: High · **Entry:** Python SDK and REST API, with the CLI for node/pipeline operations

Wants: - **pulse** — Stable, versioned Python SDK + REST API over canonical CIDs, ledger, and PLS-metered compute — *High literacy: needs deterministic programmatic access to the engine, not an app* - **pulse** — Byte-for-byte determinism + reproducibility guarantees and sampled re-execution hooks exposed in the SDK — *Reproducible pipelines demand verifiable, re-runnable compute with deterministic encoding* - **knitweb** — Deep provenance-query API returning the full DAG closure with edge types and origins — *Engineering reuse requires full-depth, machine-readable provenance, not a summary card* - **lens** — Scriptable retrieval/orchestration API with pluggable backends and ranking control — *Engineers compose custom pipelines and need programmatic control over the reasoning layer* -

gither — Cross-repo build-plan + dependency graph for pulse/lens/domain-knitweb checkouts — *Coordinating multiple repos editably needs a deterministic build planner* - **monitor** — Run/pipeline observability with per-step CID, provenance, and re-execution verdict — *High-literacy debugging needs deterministic, drill-down telemetry tied to provenance*

Engineering & Trades

Mechanical engineer

Designs and validates physical parts/assemblies, needing deterministic, provenance-backed material and load data she can cite in a stamped report.

Intake a design spec and pull verified material/component records from the shared content-addressed memory; a Lens agent plans the analysis and runs deterministic checks (mass, tolerance, load) over knitweb records. The engineer verifies by re-deriving the CID-stable result and re-running the proof, then ships a signed design Fiber whose every input is provenance-walkable back to source.

Uses: knitweb (Stores design records, material datasheets and component specs as content-addressed, citable memory shared across the team.) · **lens** (Orchestrates reasoning over the fabric: pulls candidate materials, runs trade-off analysis, drafts the verification narrative.) · **pulse** (Via the Python SDK only — reads canonical CIDs and provenance walks to guarantee a cited input is byte-identical to source; does not operate the engine.) · **monitor** (Knowledge-graph explorer to trace which design depends on which material/component record before sign-off.)

Avoids: bt (Bench/office engineering work is online; no off-grid Bluetooth mesh need.) · vbank (Governance/treasury voting is irrelevant to part design.) · virtualpc (Does not author multi-agent runtime coordination; consumes Lens results, not the coordination substrate.) · molgang (Chemistry-game domain template, not mechanical parts.)

Level: High · **Entry:** Python SDK + REST API (Lens for design queries)

Wants: - **pulse** — Deterministic, versioned material-property records with a stable CID per (material, revision) so an analysis result is reproducible byte-for-byte. — *High-literacy roles demand determinism and a fixed citation that survives report audits.* - **pulse** — Provenance-depth query: walk a design Fiber back to every source datasheet/test report with one SDK call. — *Stamped engineering needs full traceability of every cited input.* - **lens** — Read-only /interpret gateway exposing analysis (mass/tolerance/load) without write access to the ledger. — *She wants to run reasoning over the fabric without risking accidental state changes.* - **knitweb** — Unit-typed numeric fields (integer base units, declared dimension) on shared records. — *Float-free integer discipline prevents silent unit errors in load/mass math she relies on.* - **monitor** — Dependency diff view: highlight which downstream designs are affected when a material record is revised. — *Impact analysis before changing a shared spec is core to safe engineering.*

Civil engineer

Coordinates structural/site designs against codes, surveys and supplier records that must be auditable for permits and liability.

Intake permit and site data, then a Lens agent assembles the relevant code clauses, geotech and material records from the shared memory and plans the compliance check. The engineer verifies against anchored, timestamped records and re-runs the structural proof, then ships a signed submission whose provenance (which code version, which survey) is anchored for the permit file.

Uses: knitweb (Holds code clauses, geotech reports, material certs and as-built records as shared, content-addressed memory across firms and inspectors.) · **lens** (Plans compliance checks, cross-references code versions, and drafts the justification narrative over the fabric.) · **pulse** (Through REST/SDK for anchored provenance receipts (OriginTrail anchoring) proving which record version was used at submission time.) · **monitor** (Graph explorer to inspect dependency chains between site records, codes and supplier certs before sealing.)

Avoids: bt (Field connectivity is handled by phones; no need to operate a Bluetooth mesh transport directly.) · vbank (Treasury/voting governance is outside structural sign-off.) · gither (Multi-repo build planning is a developer concern, not civil design.) · molgang (Chemistry domain template, unrelated to civil works.)

Level: High · **Entry:** REST API + web app (Lens-backed), monitor graph explorer

Wants: - **pulse** — Timestamped anchoring receipt that freezes the exact code/spec version cited at the moment of submission. — *Permit and liability files need a tamper-evident 'this is what the rules were when I signed' proof.* - **knitweb** — Versioned regulatory-code records with supersession links between revisions. — *He must always cite the in-force clause and show it was current at submission.* - **lens** — Provenance query-contract that surfaces missing or unverified inputs in a compliance run. — *High-literacy users want gaps flagged explicitly rather than silently assumed.* - **monitor** — Permit-pack export: a single bundle of every cited record + its anchor for the inspector. — *Submissions must be reviewable by third parties without giving them engine access.* - **pulse** — Integer-grams/units mass and quantity fields on supply records so material takeoffs reconcile exactly. — *Conservation/quantity checks must be exact, not floating-point approximate.*

Electrician

Installs and certifies wiring on site, needing plain-language guidance, scannable part records and a tamper-proof job certificate.

On site, scans a panel/part and the mobile app's Lens agent pulls the right wiring rule and part record from shared memory and walks her through the steps in plain language. She verifies by photographing the result and answering guardrailed checks; the app ships a signed completion certificate to the shared memory for the building owner and inspector.

Uses: knitweb (Shared memory of part/cable specs and job certificates she reads via scan and writes a signed completion record into — never sees it as a database.) · **lens** (The friendly mobile assistant that plans the job, gives plain-language steps and runs the safety checks behind the app.) · **bt** (Off-grid Bluetooth mesh so the app still works and syncs certificates in basements/new-builds with no signal.)

Avoids: pulse (Never sees the engine, CIDs or ledger; far too low-level for an on-site app user.) · gither (Multi-repo build planner is a developer tool with no trade use.) · virtualpc (No multi-agent runtime authoring; consumes one guided assistant.) · vbank (Governance/treasury is irrelevant to wiring jobs.) · molgang (Chemistry game template, unrelated to electrical work.)

Level: Low · **Entry:** Consumer mobile app (Lens-guided, on-site)

Wants: - **lens** — Plain-language, step-by-step on-site wizard with photo-based 'is this correct?' guardrails. — *Low technical literacy needs guidance and confirmation, not raw data or queries.* - **knitweb** — Scan-to-record (barcode/QR/photo) lookup of cable and part specs. — *A scan, not a search, is the right interaction for a phone-on-a-ladder.* - **lens** — One-tap signed completion certificate generated from the answered checklist. — *She needs a trustworthy job record without understanding signatures or CIDs.* - **bt** — Offline-first sync that queues certificates and pushes them when signal returns. — *New-builds and basements have no connectivity, so the mesh must carry the proof.* - **knitweb** — Template job-records per common install type with sensible defaults pre-filled. — *Templates over blank forms keep low-literacy data entry fast and correct.*

Architect

Composes building designs from components and specs across many disciplines, needing a shared, traceable model the whole project team trusts.

Intake a brief into a visual web app; a Lens agent proposes a design composition by pulling vetted components and specs from the shared content-addressed memory and flags conflicts. The architect verifies via the graph explorer and human review, then ships a signed design package whose every component is provenance-linked for downstream engineers and contractors.

Uses: knitweb (Single shared model/memory where components, finishes and specs live content-addressed and are reused across the project team.) · **lens** (No-code assistant that composes options, checks consistency and drafts the design narrative over the fabric.) · **monitor** (Knowledge-graph explorer to see and present how parts of the design depend on shared components and specs.)

Avoids: pulse (Works through the app/Lens surface; never touches CIDs, ledger or the engine.) · bt (Studio-based work; no off-grid mesh need.) · gither (Developer multi-repo tooling, irrelevant to design composition.) · virtualpc (Does

not author agent-runtime coordination; consumes Lens results.) · vbank (Treasury/voting governance outside the design role.)

Level: Medium · **Entry:** No-code/low-code web app (Lens-backed) + monitor graph view

Wants: - **lens** — Visual, drag-and-compose design canvas backed by vetted shared-memory components. — *Medium literacy wants a no-code composition surface, not an API.* - **lens** — Automatic consistency/clash flags when two chosen specs conflict. — *He needs the assistant to catch cross-discipline conflicts he can't all hold in his head.* - **knitweb** — Reusable, versioned component library shared across the project with 'who else uses this' visibility. — *A shared model only works if components are reusable and their reuse is visible.* - **monitor** — Presentable dependency view exportable for client and contractor reviews. — *He must communicate the model to non-technical stakeholders.* - **knitweb** — Signed design-package snapshot that downstream engineers can provenance-walk. — *Handoff requires a traceable, frozen package without exposing engine internals.*

Logistics / supply-chain planner

Plans flows of goods across suppliers and sites, needing balanced, auditable transformation and allocation records that reconcile exactly.

Intake demand/inventory into a spreadsheet-like UI; a Lens agent plans routing and allocation, emitting mass-conserving supply-chain transformation events and capacity-feasible allocation events into the shared memory. The planner verifies that mass balances and no resource is oversubscribed (events that violate are refused pre-signature), then ships signed, content-addressed flow records auditable by partners.

Uses: knitweb (Shared memory of SKUs, supplier records and signed transformation/allocation events that partners read and reconcile against.) · **lens** (Plans routing/allocation and drafts what-if scenarios over the fabric in the spreadsheet UI.) · **pulse** (Via REST/SDK: emits supply-chain ProcessEvents (mass conservation) and operational AllocationEvents (capacity feasibility), which the ledger refuses if unbalanced.) · **monitor** (Dashboards tracking flow status and the provenance of each shipment/allocation record.)

Avoids: bt (Planning is desk-based and online; no off-grid mesh need.) · gither (Developer multi-repo navigator, irrelevant to logistics planning.) · virtualpc (Consumes Lens planning, does not author the multi-agent runtime.) · vbank (Treasury/voting governance is outside supply planning.) · molgang (Chemistry-game template, not goods logistics.)

Level: Medium · **Entry:** Spreadsheet-like UI + REST API (Lens-backed)

Wants: - **pulse** — Mass-conservation guard on supply-chain transformation events that rejects unbalanced inputs/outputs before signing. — *Auditable flows require records that are physically impossible to fudge; she relies on the guard, not manual checks.* - **pulse** — Capacity-feasibility guard on allocation events so no resource pool is oversubscribed. — *Plans must be provably feasible against declared capacity.* - **lens** — Spreadsheet-like what-if planner with one-click commit of the chosen scenario to shared memory. — *Medium literacy wants a familiar grid UI with safe commit, not raw API scripting.* - **monitor** — Shipment provenance dashboard linking each unit back to its transformation/allocation events. — *Partners and audits need end-to-end traceability of goods.* - **knitweb** — Integer-grams/unit-mass typed SKU records shared with suppliers. — *Exact integer quantities make takeoffs and reconciliations match byte-for-byte.* - **pulse** — Reference field linking an allocation event to its priced ResourceItem/settlement. — *She needs the feasible plan tied to the actual PLS obligation for cost reconciliation.*

Creative & Media

Graphic designer

Turns briefs into on-brand visual assets with an AI assistant, while every asset and brand rule lives as reusable content-addressed memory.

Intake: drop the brief + brand kit into the app; the Lens agent reads the brand's pinned memory (logo, palette, type, do/don't) from the shared fabric and proposes 3 directions. Plan->Do: designer picks one, the agent generates variations and the designer nudges them visually; do not touch CIDs or any engine. Verify->Ship: a guardrail Lens

checks brand-compliance and license-cleanliness of every source asset before the design is published back to the fabric as a new content-addressed Fiber so reuse and provenance are automatic.

Uses: lens (Brand-aware design assistant: reads the brand memory, generates and refines variations, runs the pre-publish brand/compliance check) · **knitweb** (Shared brand memory + asset library: every logo, template, and finished design is a content-addressed entry so the right asset is always reused, never re-fetched) · **monitor** (Light read-only 'where did this asset come from / where is it used' view, opened only when a license or version question comes up)

Avoids: pulse (Never sees the engine; CIDs/ledger/CBOR are too low-level and would break the visual-first workflow) · bt (No off-grid mesh need; works online in a studio app) · gither (Multi-repo build planning is irrelevant to a non-coder) · virtualpc (Single-assistant flow; no need to orchestrate a fleet of agents) · vbank (No governance/treasury voting in a solo design task)

Level: Low · **Entry:** consumer/desktop design app with a Lens side-panel (Canva/Figma-style), plus a friendly mobile companion

Wants: - **lens** — One-tap 'stay on brand' guardrail that hard-blocks off-palette/off-type output with a plain-language reason — *Low-literacy designers need protective defaults, not config; the agent should refuse to drift rather than expect the user to police it* - **knitweb** — Brand-kit template: a prebuilt memory bundle (logo/palette/type/voice) you fill in once via a form, no IDs visible — *Templates + form-filling match Low literacy; hides content-addressing while still giving reuse* - **lens** — Plain-language license/usage-rights tag on every source asset ('OK for client work', 'not for resale') — *A non-technical user can't read provenance graphs; the agent must surface rights in human words before they ship* - **monitor** — 'Used in' badge: tap an asset to see, in friendly cards, which projects reused it — *Gives reuse/version safety without exposing the raw knowledge-graph explorer* - **lens** — Mobile approval card: review and approve generated variations from a phone — *Mobile-first approval fits a Low-literacy designer working between meetings*

Video editor

Assembles, captions, and versions video while agents auto-log every clip, edit, and rights-status into shared memory.

Intake: ingest footage; the Lens agent transcribes, tags scenes, and registers each clip as a content-addressed entry in the project memory. Plan->Do: editor lays the cut on the timeline while agents handle rough-cut suggestions, captions, and B-roll matching from the shared library; the editor stays in control of the timeline. Verify->Ship: a Lens pass checks music/footage licensing and caption accuracy, then publishes the master plus an edit-decision record as a provenance-linked Fiber so any version is reconstructable.

Uses: lens (Transcription, scene-tagging, caption generation, B-roll/music matching, and a pre-export rights check) · **knitweb** (Project + clip memory: footage, captions, music cues, and edit-decision lists are content-addressed so versions and reuse are deterministic) · **monitor** (Knowledge-graph view to trace which master came from which clips/licenses when a takedown or re-version question arises)

Avoids: pulse (The editor uses CIDs only implicitly through the app; the ledger/p2p engine is never opened directly) · bt (Renders/exports happen on workstation or cloud, not over a Bluetooth mesh) · gither (No multi-repo navigation in a media workflow) · virtualpc (A single assistant suffices; no agent-fleet coordination runtime needed) · vbank (No treasury/voting in editing work)

Level: Medium · **Entry:** timeline-based editing app (Premiere/DaVinci-style) with an embedded Lens panel and a project-memory sidebar

Wants: - **knitweb** — Content-addressed clip dedupe + edit-decision-list (EDL) snapshots so any cut version is reconstructable by reference — *Medium literacy can handle a versions panel; deterministic reuse saves storage and makes 'restore that cut' reliable* - **lens** — Auto music/stock-footage license check with a clear pass/fail per asset before export — *Editors understand rights but want it automated; a per-asset gate prevents shipping infringing masters* - **lens** — Caption/transcript accuracy review loop with confidence flags on uncertain lines — *Medium users want assistance plus a verify step, not blind auto-captioning* - **monitor** — Provenance trace 'this master ← these clips ← these licenses' as a navigable graph — *Useful for takedowns/re-versioning; a Medium user can read a focused graph view* - **knitweb** — Shared team B-roll/music library keyed by content hash with usage tags — *Enables cross-project reuse without duplicating large media files*

Photographer

Captures, organizes, and delivers shoots while every original and edit is fingerprinted into shared memory for proof-of-origin and licensing.

Intake: shoot and import; the app fingerprints each original as a content-addressed Fiber the moment it lands, establishing proof-of-capture. Plan->Do: the Lens agent culls, auto-tags, and proposes edits; the photographer accepts/rejects with taps. Verify->Ship: a guardrail confirms model/property releases and license terms, then publishes the delivery gallery to the shared fabric so clients get verifiable, rights-clean originals.

Uses: knitweb (Each original and edit is stored as a content-addressed entry, giving tamper-evident proof-of-origin and a reusable, deduplicated library) · **lens** (Auto-cull, tagging, edit suggestions, and a plain-language release/license check before delivery) · **monitor** (Simple 'origin certificate' view a client can open to confirm an image's source and rights)

Avoids: pulse (Photographer never touches the ledger/CID engine; fingerprinting must be invisible) · bt (No off-grid mesh delivery; uploads when back online) · gither (Irrelevant developer tooling) · virtualpc (No multi-agent orchestration in a solo-shoot flow) · vbank (No governance/treasury role)

Level: Low · **Entry:** consumer mobile app (capture + cull + deliver) with a Lens assistant; optional desktop for batch delivery

Wants: - **knitweb** — Capture-time fingerprinting: every shot is auto-registered as a content-addressed original with no user action — *Low literacy = invisible guardrails; proof-of-origin must 'just happen' to be useful in disputes* - **lens** — Release/consent checklist in plain language ('do you have a signed model release?') gating delivery — *Protects a non-technical shooter from delivering rights-unclean images; templates over config* - **monitor** — Shareable 'origin certificate' card a client can view without an account — *Turns provenance into a simple trust artifact for Low-literacy users on both sides* - **lens** — One-tap mobile cull/tag with undo, no folders to manage — *Mobile-first, gesture-based flow suits Low literacy* - **knitweb** — Automatic dedupe of near-identical frames in the library — *Saves storage and confusion without the user understanding content addressing*

Musician / producer

Writes and produces tracks while agents register every stem, sample, and contributor split into shared memory for clean rights and reuse.

Intake: start a project; each imported sample/stem is registered as a content-addressed entry with its source and license. Plan->Do: the producer arranges in the DAW while a Lens agent suggests progressions, matches royalty-clear samples, and tracks who contributed what. Verify->Ship: a Lens pass reconciles sample clearances and contributor splits, then publishes the master + a split-sheet as a provenance-linked Fiber so payouts and reuse are unambiguous.

Uses: lens (Generative/assistive ideas, royalty-clear sample matching, and a clearance + split-sheet reconciliation pass) · **knitweb** (Stem/sample/contributor memory: content-addressed so a sample's license and a collaborator's split travel with the audio) · **monitor** (Graph view to trace a master back to stems, samples, and contributors when splits or clearances are questioned) · **vbank** (Optional: when a track is co-owned, route agreed contributor splits to a shared treasury/payout)

Avoids: pulse (Producer relies on CIDs implicitly via the app; the raw ledger/p2p engine stays hidden) · bt (No off-grid Bluetooth mesh need in studio production) · gither (No multi-repo build navigation) · virtualpc (Single assistant is enough; no agent-fleet runtime)

Level: Medium · **Entry:** DAW-adjacent app or plugin (Ableton/Logic-style) with a Lens panel; project memory sidebar for stems and splits

Wants: - **knitweb** — Content-addressed stem + sample registry carrying license and contributor metadata with the audio — *Medium users can manage a stems panel; binding rights/splits to the hash prevents lost clearances* - **lens** — Royalty-clear sample matcher that only surfaces samples whose license permits the intended use — *Automates the riskiest step while letting the producer verify; fits Medium literacy* - **vbank** — Split-sheet → treasury payout so agreed percentages auto-route to collaborators — *Co-owned tracks need transparent, governed payouts; this is the one governance touch a producer wants* - **monitor** — Master→stems→samples→contributors provenance trace —

Resolves split and clearance disputes; a Medium user can read a focused graph - **lens** — Pre-release clearance gate that blocks export on any uncleared sample — *Deterministic verify step prevents shipping infringing masters*

Copywriter

Produces on-voice, fact-checked copy while every brief, source, and approved line is stored as reusable, provenance-linked memory.

Intake: paste the brief; the Lens agent loads brand-voice memory and cited source facts from the shared fabric. Plan->Do: it drafts variants grounded in those cited sources while the writer edits for voice; claims stay linked to their origins. Verify->Ship: a Lens pass checks claims against provenance and runs a brand-voice/compliance gate, then publishes approved copy back as a content-addressed entry so future briefs reuse proven, fact-checked lines.

Uses: lens (Source-grounded drafting, claim-to-source checking, and a brand-voice/compliance gate before publish) · **knitweb** (Brand-voice guide, approved-claims library, and finished copy stored as content-addressed entries for reuse and provenance) · **monitor** (Read-only 'which source backs this claim' view when legal/compliance asks)

Avoids: pulse (Writer never opens the engine; content addressing is implicit through the app) · bt (No off-grid mesh need) · gither (No multi-repo developer navigation) · virtualpc (Single writing assistant suffices; no agent-fleet runtime) · vbank (No treasury/voting in copy production)

Level: Medium · **Entry:** no-code web app / writing workspace with a Lens assistant and a brand-voice memory panel; optional REST API for power users

Wants: - **lens** — Claim-grounding: every factual sentence links to a cited source, with unsupported claims flagged before publish — *Medium-literacy writers want provenance depth as an automatic check, not a manual citation chore* - **knitweb** — Brand-voice memory + approved-claims library reused across briefs by reference — *Content-addressed reuse keeps voice consistent and lets proven copy be recombined safely* - **lens** — Brand-voice/compliance gate with a plain-language diff of what was off-voice — *Gives a verify step the writer can act on without reading model internals* - **monitor** — 'Source behind this claim' lookup for legal/compliance review — *Turns provenance into an auditable trust artifact for a Medium-literacy reviewer* - **lens** — Optional REST API to push briefs and pull approved copy for power users integrating a CMS — *Higher end of Medium literacy wants programmatic access; keeps the friendly app default while enabling automation*

Education & Knowledge

School teacher

Runs a class by pulling trustworthy, grade-appropriate material from shared memory and turning it into lessons and quizzes, with an agent doing the heavy lifting.

Intake: teacher types a topic and grade level into the app; a Lens agent retrieves provenance-cited facts from the shared content-addressed memory (the Web). Plan/do: the agent drafts a lesson + quiz and the teacher tweaks it on the phone; verify: every claim shows a plain-language source badge from the memory so the teacher can trust it before sharing. Ship: approved lesson is woven back into the class knitweb as a signed record other teachers can reuse.

Uses: knitweb (Reads and contributes to the shared content-addressed memory (lessons, facts, vetted sources) as the durable class fabric; never sees CIDs directly, just "saved/shared" affordances.) · **lens** (The reasoning layer behind the app: retrieves grade-appropriate, provenance-cited material and drafts lessons/quizzes from the converged graph.) · **molgang** (Uses the molgang chemistry knitweb (and similar domain templates) as a ready-made classroom game where curriculum content already lives.) · **monitor** (A simplified 'class map' view to see which topics the class has covered, as a read-only knowledge-graph explorer.)

Avoids: pulse (Never sees the engine; CLI/ledger/CID internals are too low-level for a classroom phone app.) · bt (No off-grid Bluetooth-mesh need in a connected school.) · gither (Multi-repo build planning is irrelevant to a non-developer teacher.) · virtualpc (Multi-agent coordination runtime is below the app surface and not exposed.) · vbank (Governance voting/treasury is not part of classroom delivery.)

Level: Low · **Entry:** consumer mobile app (a friendly Lens-backed app, e.g. a molgang-style classroom knitweb)

Wants: - **lens** — Grade-band guardrails: a 'reading level / age' control that filters retrieval and rewrites answers to the target level, with hallucination-blocking (provenance-gated, drop un-sourced claims). — *A low-literacy teacher needs the agent to be safe-by-default and age-appropriate without configuring anything.* - **knitweb** — Plain-language 'source badge' on every fact (where it came from + how trusted) instead of raw CIDs/provenance graphs. — *Teachers must judge trust at a glance; CIDs and attestation chains are meaningless to them.* - **molgang** — Reusable lesson/quiz templates with a one-tap 'make a class set' so a topic becomes a playable activity. — *Templates and mobile one-tap flows match low technical literacy and save prep time.* - **monitor** — A simplified, color-coded 'coverage map' of curriculum topics for the class, read-only on mobile. — *A friendly visual is the right altitude; the full knowledge-graph explorer is too dense for this role.*

University lecturer

Builds rigorous, citable course material and research-grounded lectures by querying the shared memory and demanding traceable provenance for every claim.

Intake: lecturer poses a research/teaching query in the web app; a Lens agent retrieves candidate relations from the shared memory and runs a bounded distill loop. Plan/do: the agent assembles an argument/lecture with inline citations; verify: the lecturer inspects the provenance trail (originator + source) before accepting. Ship: vetted material is woven back into the course knitweb and optionally exported via REST into the LMS/slides.

Uses: knitweb (Primary shared memory for course content and cited facts; contributes signed, provenance-bearing course relations others can build on.) · **lens** (Runs retrieve + provenance-gated distill to produce cited lecture material and answers grounded in the converged graph.) · **monitor** (Knowledge-graph explorer to inspect how concepts and sources connect and to audit a claim's provenance chain.) · **vbank** (Light use for departmental decisions/curriculum proposals via polls when a course knitweb is shared-governed.)

Avoids: pulse (Does not operate the P2P node/ledger directly; consumes the engine only through the app/API.) · bt (No off-grid transport need on campus.) · gither (Build planning across repos is a developer concern, not a lecturer's.) · virtualpc (Multi-agent runtime stays under the hood; not a surface the lecturer drives.)

Level: Medium · **Entry:** no-code web app + occasional REST API for pulling cited material into slides/LMS

Wants: - **lens** — Provenance-depth control: ability to request 'distill with full citation chain' and see originator + asset_cid for each relation in the answer. — *A medium-literacy academic demands traceable, defensible citations, not just an answer.* - **knitweb** — Citable stable references: a human-readable handle for a content-addressed record that resolves to its CID + originator, suitable for a bibliography. — *Lecturers need to cite shared-memory facts in publications/syllabi with a stable, verifiable pointer.* - **pulse** — Read-only REST query endpoint (retrieve + provenance) so material can be pulled into an LMS/slide tool without running a node. — *Medium literacy means comfort with an API but not with operating the substrate.* - **monitor** — Side-by-side source comparison and 'why was this claim accepted' provenance drill-down. — *Auditing conflicting sources is core to scholarly rigor and benefits from a real graph explorer.*

Librarian / information specialist

Curates, catalogues, and certifies the shared memory's provenance so others can trust what they read.

Intake: ingests collections/sources via batch import into the shared memory; a Lens agent proposes metadata and links to existing records. Plan/do: librarian reviews, de-duplicates, and attests sources (signs what they vouch for); verify: provenance and attribution are checked so the memory isn't polluted. Ship: curated, attested records are woven into the Web and exposed to patrons through the explorer.

Uses: knitweb (The core working surface: catalogues, links, de-duplicates, and attests records in the shared content-addressed memory (validate-at-read provenance).) · **monitor** (Knowledge-graph explorer to map collections, find orphaned/duplicate records, and audit provenance coverage.) · **lens** (Agent-assisted metadata enrichment and link suggestion across the converged graph.) · **vbank** (Occasional governance participation on collection/retention policy via polls when a knitweb is community-governed.)

Avoids: pulse (Engine internals (CBOR/CID/ledger) are abstracted away behind cataloguing tools.) · bt (No off-grid mesh need for library cataloguing.) · gither (Developer multi-repo navigation is out of scope.) · virtualpc (Agent-

coordination runtime is not a librarian-facing surface.)

Level: Medium · **Entry:** spreadsheet-like UI + no-code web app for cataloguing, with batch import

Wants: - **knitweb** — Batch attestation + de-duplication workflow: import a collection, auto-detect duplicate/near-duplicate records by content address, and sign-off in bulk. — *Librarians work at collection scale and need trustworthy, attributable curation without per-record CID handling.* - **monitor** — Provenance-coverage dashboard: highlight records lacking a verified originator or trusted source. — *Information specialists are accountable for source quality; a gap dashboard makes curation actionable.* - **lens** — Controlled-vocabulary / authority-file aware metadata suggestion that maps to existing graph nodes instead of inventing new ones. — *Cataloguing demands consistency with authority files, not free-text drift.* - **knitweb** — Patron-facing read-only 'verified source' view with plain-language trust labels. — *Bridges the curated memory to non-technical patrons who only need the trust signal.*

Instructional designer

Designs reusable, provenance-backed learning experiences as domain-knitweb templates that agents and learners run.

Intake: defines a learning objective and audience in the builder; a Lens agent pulls grounded content and learning objects from shared memory. Plan/do: designer assembles a module/path from templates and adds assessments; verify: checks alignment to objectives and that each content node is sourced. Ship: publishes the module as a reusable knitweb template woven into shared memory for instructors to adopt.

Uses: molgang (Uses the molgang chemistry game as the reference domain-knitweb template, cloning its 'the game is the protocol' pattern for new subjects.) · **knitweb** (Stores reusable learning objects and module structures as shared, content-addressed records others can fork and adopt.) · **lens** (Agent-assisted content sourcing, sequencing, and assessment-item drafting grounded in the graph.) · **monitor** (Visualizes the learning-path graph (prerequisites, objective coverage) for design review.)

Avoids: pulse (Builds on top of the engine via the app; never touches node/ledger/CID internals.) · bt (No off-grid transport relevance to course design.) · gither (Multi-repo build planning is a developer task.) · virtualpc (Agent-orchestration runtime is below the design surface.) · vbank (Treasury/voting not part of the design loop unless explicitly governing a shared template.)

Level: Medium · **Entry:** no-code web app (template/flow builder) over a Lens surface

Wants: - **molgang** — A 'new domain knitweb from template' scaffold that clones the molgang structure (nodes, knits, quorum-vote, PLS) for an arbitrary subject with no code. — *A medium-literacy designer needs a no-code clone of a proven template, not to build a knitweb from primitives.* - **lens** — Objective-aligned generation: feed a learning objective and get sequenced, sourced learning objects + assessment items back. — *Instructional design is objective-driven; the agent should plan against objectives, not just retrieve facts.* - **knitweb** — Forkable/versioned learning-module records with adoption tracking (who reused this template). — *Reuse and iteration are the core value of content-addressed shared memory for course design.* - **monitor** — Learning-path graph view showing prerequisite gaps and objective coverage. — *A graph explorer tuned to learning paths makes design quality visible at the right altitude.*

Translator

Produces verified, traceable translations as linked records in the shared memory, keeping terminology consistent across languages.

Intake: opens a source record (or batch) in the app; a Lens agent drafts a translation grounded in glossary/termbase nodes from shared memory. Plan/do: translator post-edits in a side-by-side view; verify: terminology consistency and the link back to the signed source are checked. Ship: the translation is woven into the Web as a signed record linked to its source CID, so provenance and the original are always traceable.

Uses: knitweb (Stores translations as signed records explicitly linked to their source record's content address, with a shared multilingual termbase as graph nodes.) · **lens** (Agent-assisted draft translation and terminology lookup grounded in the converged termbase/graph.) · **monitor** (Explorer to see which source records lack translations and to audit term consistency across languages.) · **pulse** (Power users only: Python SDK for batch translation jobs that read/write records programmatically.)

Avoids: bt (No off-grid mesh need for translation work.) · gither (Developer multi-repo navigation is irrelevant.) · virtualpc (Multi-agent runtime is below the translator's surface.) · vbank (Governance/treasury not part of the translation loop.)

Level: Medium · **Entry:** no-code web app with side-by-side editor; Python SDK for power users doing batch jobs

Wants: - **knitweb** — Source-linked translation records: a translation is woven as a child of its source CID so original↔translation provenance is permanent and traceable. — *Translators are accountable to the source; an immutable link back is essential for trust and revision.* - **lens** — Termbase-locked drafting: the agent must reuse existing glossary nodes for terms-of-art and flag novel terms rather than improvise. — *Terminology consistency is the translator's core quality bar; the agent must respect the shared termbase.* - **knitweb** — Multilingual graph: a concept node holds all its language variants, surfacing 'missing translation' gaps. — *Content-addressed shared memory should make a concept's translations one linked cluster, not scattered files.* - **pulse** — Python SDK batch translate/retrieve helpers (read records, write linked translation records) for high-volume jobs. — *This medium-literacy role's power users want programmatic batch access without operating a full node.*

Business & Management

Product manager

Turns customer signals and a backlog into a verifiable, agent-executed product plan whose decisions are provenance-tracked.

Intake: drop research, tickets, and metrics into the shared content-addressed memory so every artifact has a stable CID. Plan: a Lens agent drafts a prioritized backlog and rationale, the PM edits in the no-code app, and verify by asking Lens for the provenance chain behind each priority before shipping the plan as a signed, immutable spec into memory that engineering and agents read from.

Uses: knitweb (Shared memory of immutable, content-addressed specs, research, and decision records — the single source of truth the PM and agents both read/write.) · **lens** (Reasoning/orchestration surface that drafts backlogs, summarizes feedback, and answers 'why is this prioritized' with traceable provenance.) · **monitor** (Dashboard + knowledge-graph explorer to see how specs, experiments, and outcomes link, and to track delivery against the plan.) · **vbank** (Lightweight governance for prioritization calls — stakeholder votes on roadmap bets with an auditable tally.)

Avoids: pulse (Never touches the engine; ledger/CID internals are abstracted away behind Lens and the app.) · bt (No off-grid mesh or DEX need in a product workflow.) · gither (Repo/build planning is an engineering concern, not the PM surface.) · virtualpc (Agent-runtime operations are handled for them; they consume agent output, not orchestrate runtimes.)

Level: Medium · **Entry:** No-code web app (Lens-driven workspace) with a REST API for power use

Wants: - **lens** — A 'decision provenance' panel that, for any roadmap item, shows the exact source artifacts (CIDs) and reasoning steps that led to its priority. — *Medium-literacy PMs need trustworthy, auditable rationale to defend roadmap calls to leadership without reading the ledger.* - **knitweb** — Versioned, immutable spec records with human-readable diffs between content-addressed versions. — *PMs reason in product terms; they need 'what changed and when' without learning CBOR/CID encoding.* - **monitor** — A roadmap-to-outcome graph view linking each shipped spec to its experiment and metric nodes. — *A visual knowledge-graph closes the verify loop so PMs can prove a feature moved a metric.* - **vbank** — Embeddable 'roadmap poll' widget with weighted stakeholder voting and a one-click auditable result. — *Prioritization needs defensible, tamper-evident stakeholder input, not a Slack thread.* - **lens** — REST API for read access to the backlog and provenance graph. — *Higher-literacy PMs want to wire roadmap data into their own dashboards and BI tools.*

Project manager

Coordinates tasks, owners, and deadlines while agents do the legwork and every status update is a verifiable record.

Intake: pick a project template in the app and assign tasks to people and agents, all stored in shared memory. Plan/do: Lens agents update status, flag blockers, and draft standups; the PM reviews in a familiar board/table view, verifies

completion against the immutable task record, and ships the status report and milestone sign-off back into memory for stakeholders.

Uses: knitweb (Shared memory holding the immutable task/milestone records and status history — an audit trail of who committed to what and when.) · **lens** (Friendly agent surface that summarizes progress, drafts standups, and surfaces at-risk tasks in plain language.) · **monitor** (Read-only dashboard showing timeline, task status, and dependencies for stakeholder visibility.)

Avoids: pulse (Far too low-level; never sees CIDs, ledgers, or P2P sync.) · vbank (Governance voting/treasury is outside task coordination scope.) · bt (No off-grid transport need.) · gither (Code-forge/build planning is for engineers, not project tracking.) · virtualpc (Doesn't configure agent runtimes; just assigns and reviews work.)

Level: Low · **Entry:** Consumer/spreadsheet-like web app with templates and guardrails

Wants: - **lens** — Plain-language 'what's blocked and who do I nudge' daily digest with suggested actions. — *Low-literacy PMs need agents to translate state into next steps, not jargon.* - **knitweb** — Guardrailed task templates (with required fields and owners) that prevent malformed or untracked tasks. — *Guardrails keep non-technical users from creating records that can't be tracked or audited.* - **monitor** — Mobile-friendly Gantt/timeline view with simple red/amber/green health. — *PMs check status on the move and need at-a-glance health, not a graph query language.* - **lens** — One-tap auto-drafted status report a human approves before it's published. — *Keeps a human in the loop while removing the manual reporting burden for a low-literacy user.* - **knitweb** — Immutable, timestamped sign-off records for milestone approvals. — *Gives PMs a tamper-evident paper trail of approvals without any crypto knowledge.*

Marketer

Plans and runs campaigns with agent-generated content, where every claim is traceable to a verified source.

Intake: define audience and goal in the app; pull approved product facts from shared memory. Plan/do: Lens agents draft copy, variants, and a schedule; the marketer edits and approves; verify that every factual claim links back to a source CID (no hallucinated specs) before shipping the campaign and storing performance results back into memory for the next iteration.

Uses: knitweb (Shared memory of approved brand facts, product specs, and prior campaign results that agents must cite from.) · **lens** (Generative + orchestration surface that drafts copy and content and grounds claims in cited memory.) · **monitor** (Simple dashboard for campaign performance and content-to-result links.)

Avoids: pulse (Engine internals are invisible to a marketing user.) · vbank (No governance/voting role in campaign execution.) · bt (No off-grid or trading need.) · gither (Not a code or build context.) · virtualpc (Doesn't run agent infrastructure; consumes a managed Lens app.)

Level: Low · **Entry:** Consumer mobile/web app (campaign studio) on a Lens surface

Wants: - **lens** — Claim-grounding mode that refuses or flags any generated marketing claim not backed by a memory source. — *Low-literacy marketers need automatic guardrails against false/unapproved claims for compliance safety.* - **knitweb** — An 'approved facts' library with locked, citable brand/product records. — *Gives non-technical users a safe, reusable source-of-truth so agents stay on-brand and accurate.* - **lens** — Mobile content studio with brand-voice templates and one-tap variant generation. — *Mobile + templates match a low-literacy, on-the-go workflow.* - **monitor** — Plain-language campaign performance cards (reach, conversion) with no query setup. — *Marketers need outcomes in marketing terms, not graph queries.* - **knitweb** — Immutable approval/publish records so legal can audit exactly what was published and when. — *Provides a tamper-evident trail for regulated claims without exposing the marketer to crypto plumbing.*

HR manager

Runs hiring, policy, and people processes with agent help while keeping a private, auditable record of sensitive decisions.

Intake: capture roles, policies, and cases in the app, stored in access-scoped shared memory. Plan/do: Lens agents screen, draft job posts, summarize policy questions, and schedule; the HR manager reviews with a human-in-the-loop

gate, verifies fairness/consistency against policy records, and ships decisions with an immutable, access-controlled audit record.

Uses: knitweb (Shared, access-scoped memory of policies, role definitions, and tamper-evident decision records for sensitive people data.) · **lens** (Friendly agent surface for screening, drafting, policy Q&A, and scheduling, with plain-language explanations.) · **vbank** (Auditable voting for hiring-committee or promotion decisions when multiple approvers must sign off.)

Avoids: pulse (Engine-level access would be a privacy and complexity liability; never touched.) · monitor (Open knowledge-graph exploration conflicts with sensitive-data confidentiality.) · bt (No off-grid or financial-trading need.) · gither (Not a code context.) · virtualpc (Doesn't operate agent runtimes; uses a managed, governed app.)

Level: Low · **Entry:** No-code web app with strong privacy guardrails on a Lens surface

Wants: - **knitweb** — Fine-grained, role-based access control with redaction on sensitive (PII) records. — *Low-literacy HR users need privacy enforced by the platform, not by manual discipline.* - **lens** — Bias/consistency check that flags when a hiring recommendation deviates from stated policy. — *Guardrails help a non-technical user defend fairness and meet compliance.* - **lens** — Plain-language policy assistant that always cites the exact policy clause. — *HR needs trustworthy, source-cited answers, not opaque AI text.* - **vbank** — Private, auditable committee-vote workflow with sealed results until quorum. — *Promotion/hiring panels need tamper-evident, confidential sign-off.* - **knitweb** — Immutable, access-logged decision records with retention controls. — *Gives HR a compliant audit trail and 'who viewed what' log without crypto knowledge.*

Management consultant

Diagnoses a client problem and builds an evidence-backed recommendation where every conclusion traces to a verifiable source.

Intake: load client data, interviews, and benchmarks into shared memory with stable CIDs. Plan/do: Lens agents run analyses, build option models, and draft the deck; the consultant stress-tests assumptions and verifies each recommendation's provenance chain before shipping a signed, immutable findings package the client can independently audit.

Uses: knitweb (Shared memory of client artifacts, analyses, and findings as immutable, content-addressed records the client can later verify.) · **lens** (Reasoning/orchestration surface for analysis, scenario modeling, and deck drafting with traceable provenance.) · **monitor** (Knowledge-graph explorer to map how findings, evidence, and recommendations interconnect for the client narrative.) · **vbank** (Auditable stakeholder/steering-committee voting to converge on a recommended option.)

Avoids: pulse (Engine internals are below the consultant's altitude; abstracted by Lens.) · bt (No off-grid mesh or DEX relevance to advisory work.) · gither (Not a software-build engagement surface.) · virtualpc (Consumes agent output via Lens; doesn't run the multi-agent runtime.)

Level: Medium · **Entry:** No-code web app plus REST API for data pulls and exports

Wants: - **lens** — Deterministic, reproducible analysis runs where the same inputs yield the same outputs and the trace is exportable. — *Medium/high-literacy consultants need defensible, repeatable analysis to withstand client scrutiny.* - **knitweb** — A signed, client-verifiable findings bundle (evidence + reasoning + conclusion CIDs). — *Lets a client independently audit the basis of every recommendation — a differentiator for trust.* - **lens** — REST API to pull analysis outputs and provenance into the consultant's own models/decks. — *Consultants want programmatic access to integrate with existing tooling at their literacy level.* - **monitor** — Evidence-to-recommendation graph that highlights any unsupported conclusion. — *Surfaces gaps in the argument chain so the consultant can close them before delivery.* - **vbank** — Steering-committee decision poll with weighted votes and an immutable record of what was agreed. — *Engagements need a clean, auditable record of client buy-in on the chosen option.*

Public & Social

Social worker

A frontline caseworker who coordinates support for vulnerable people and needs a trustworthy shared record without ever seeing the plumbing.

Intake a client situation by talking to the Lens assistant in plain language on mobile; the Lens drafts a care plan and pulls relevant prior context from the shared content-addressed memory (only entries in the worker's authorized scope). The worker reviews/edits, an agent fills the routine paperwork, the worker verifies the plain-language summary and taps to ship it into the shared memory as a signed, provenance-stamped case note.

Uses: knitweb (the shared content-addressed memory holds case notes/care plans as provenance-stamped records the worker reads and contributes to, scoped to their consent/authorization) · **lens** (the chat/reasoning surface that drafts care plans, retrieves prior context, and translates jargon to plain language)

Avoids: pulse (never sees the engine; CIDs/ledger/PLS internals are far too low-level for a caseworker) · bt (no off-grid Bluetooth-mesh need in an office/field-with-signal setting) · gither (multi-repo build navigation is irrelevant to a non-developer) · virtualpc (no need to orchestrate a multi-agent runtime by hand) · molgang (a chemistry domain-knitweb is unrelated to social casework) · vbank (no voting/treasury governance in individual casework)

Level: Low · **Entry:** consumer mobile app (case-companion) with a friendly Lens chat surface

Wants: - **lens** — Plain-language mode with reading-level control and a 'are you sure?' confirmation before anything is shared — *Low literacy means guardrails and human-readable summaries must come before any commit to shared memory* - **knitweb** — Consent-scoped visibility: per-record sharing controls so a note is only readable by named parties, enforced at the memory layer — *Sensitive client data demands the privacy boundary be a built-in template, not something the worker must configure* - **lens** — Templated intake/care-plan forms with safeguarding red-flag prompts the agent surfaces automatically — *Low-literacy roles want fill-in templates and prompts, not free-form structuring* - **monitor** — A simple mobile 'who saw this and when' provenance ribbon on each case note — *Caseworkers need accountability they can read at a glance without learning a graph explorer* - **knitweb** — Right-to-be-forgotten / redaction flow that tombstones a record while preserving an audit trail — *Vulnerable-client data needs deletion semantics that are safe by default for a non-technical user*

Urban planner

A planner who weaves zoning, infrastructure, and consultation data into evidence-backed proposals and wants every claim traceable to its source.

Intake a planning question and assemble datasets (parcels, transit, consultation responses) in a map/spreadsheet web app; an agent (via Lens) drafts options and traverses the shared content-addressed memory to attach provenance to each figure. The planner verifies sources and impact estimates in the knowledge-graph explorer, then ships a versioned proposal record whose underlying evidence is content-addressed and re-checkable.

Uses: knitweb (shared memory stores parcels, datasets, consultation inputs and the proposal itself as content-addressed, versioned records with provenance) · **lens** (reasoning/orchestration to draft scenario options and bind each claim to its source records) · **monitor** (knowledge-graph + map dashboards to inspect relationships, provenance lineage, and scenario diffs)

Avoids: pulse (the CID/ledger engine is below the planner's altitude; touched only through friendly surfaces) · bt (planning is done with connectivity; no off-grid mesh transport needed) · gither (build/repo planning is a developer concern) · virtualpc (no hand-rolled multi-agent runtime; the Lens app manages agents) · molgang (chemistry domain template is unrelated to urban planning) · vbank (the planner proposes; formal voting/treasury sits with elected governance, not their tool)

Level: Medium · **Entry:** no-code/low-code web app with map + spreadsheet-like views over the shared memory

Wants: - **monitor** — Geospatial knowledge-graph explorer that overlays content-addressed records on a basemap and shows their relation/provenance edges — *Medium literacy planners can use a visual explorer but need spatial*

context the substrate doesn't render - **knitweb** — Versioned scenario snapshots with diff between two proposal CIDs — *Planners iterate options and need to compare versions; content-addressing makes immutable diffs natural* - **lens** — Source-binding mode that refuses to state a figure without an attached source CID — *Evidence-backed planning needs provenance depth, appropriate for a medium-literacy user who trusts but verifies* - **knitweb** — Bulk dataset import with schema templates for common GIS/census formats that auto-content-address each row — *A no/low-code planner needs templated ingestion, not scripting, to populate shared memory* - **monitor** — Consultation-response heatmap aggregating citizen inputs by area while preserving per-response provenance — *Planners summarize public input but must keep each contribution auditable*

Policy analyst

An analyst who runs reproducible, provenance-deep analyses on shared evidence and demands byte-for-byte determinism.

Intake a policy question, then script a reproducible retrieval against the shared content-addressed memory via the SDK; the Lens interpretation lobe (deterministic structured retrieve -> recursive distill -> provenance-gate) returns a Selection of relations with source CIDs. The analyst verifies determinism by re-running against a pinned web-state CID, then ships a signed, fully-traceable analysis whose every conclusion links back to immutable evidence.

Uses: knitweb (queries the shared memory directly for content-addressed evidence, ancestry, and provenance; pins web-state CIDs for reproducibility) · **lens** (calls the interpretation lobe (retrieve/distill/provenance-gate) programmatically to turn evidence into a sourced Selection) · **monitor** (cross-checks results and audits provenance lineage visually before publication) · **pulse** (uses canonical CID/CBOR and provenance APIs read-only via the SDK to confirm determinism and verify signatures — the one high-literacy role that legitimately touches the substrate's read surface)

Avoids: bt (no off-grid transport need; analysis runs on connected infrastructure) · gither (multi-repo build planning isn't part of an analyst's workflow) · virtualpc (doesn't need to operate the raw multi-agent runtime; the Lens lobe suffices) · molgang (chemistry domain template is out of scope for policy) · vbank (analyzes policy options but doesn't run the voting/treasury machinery itself)

Level: High · **Entry:** Python SDK + REST API over the shared memory and the Lens interpretation lobe

Wants: - **lens** — Deterministic retrieve+distill with a stable output contract: same (query, subscription, web_state_cid) yields byte-identical Selection — *High literacy means reproducibility is the headline requirement, not friendliness* - **knitweb** — Provenance/ancestry query API returning full lineage and missing-node visibility for any record CID — *Analysts need provenance depth and to know exactly what evidence is absent, not just present* - **pulse** — Read-only SDK to pin and resolve a web-state CID and verify record signatures without running a node — *A high-literacy role wants determinism guarantees from the substrate without operating the engine* - **lens** — Provenance-gate that attests the distiller output and rejects any unsourced relation — *Sourced-only conclusions are non-negotiable for credible policy analysis* - **monitor** — Side-by-side diff of two analysis runs highlighting which source CIDs changed — *Analysts must explain why an updated dataset changed a conclusion, with full traceability*

Journalist

A reporter who investigates with shared evidence and needs every fact tied to a tamper-evident, dated source before publishing.

Intake a story lead and ask the Lens assistant to gather and cross-reference records from the shared content-addressed memory, surfacing provenance for each claim. The journalist verifies sourcing and contradictions in the explorer, an agent drafts the piece with inline citations, and they ship the article alongside a content-addressed evidence bundle that readers/editors can independently re-check.

Uses: knitweb (shared memory provides content-addressed, timestamped source documents and their provenance, plus a publishable immutable evidence bundle) · **lens** (chat surface to retrieve, cross-reference, fact-check, and draft with inline source CIDs) · **monitor** (knowledge-graph explorer to spot contradictions and trace how a claim connects to its sources)

Avoids: pulse (the engine is too low-level; the journalist consumes provenance through friendly surfaces) · bt (no routine off-grid need, though field journalism is the closest case (still not their default tool)) · gither (developer build navigation is irrelevant) · virtualpc (the Lens app manages agents; no raw runtime needed) · molgang (chemistry domain template is unrelated to reporting) · vbank (no voting/treasury role in journalism)

Level: Medium · **Entry:** no-code research web app with a Lens chat/verification surface

Wants: - **knitweb** — Tamper-evident, timestamped source pinning that produces a shareable evidence-bundle CID readers can independently verify — *Medium literacy reporter needs verifiability packaged as a click, not as cryptography* - **lens** — Cross-reference/contradiction finder that flags when two sourced records disagree — *Fact-checking is the core loop; the agent should surface conflicts plainly* - **lens** — Inline citation mode that attaches a source CID to every generated sentence and refuses unsourced claims — *Publishing demands every fact be attributable, balanced with a friendly drafting UX* - **knitweb** — Source-confidentiality controls (sealed/anonymized provenance) that still prove a record existed at a date — *Journalists must protect sources while retaining tamper-evident dating — a guardrail the memory layer should own* - **monitor** — Timeline view reconstructing the order events entered shared memory — *Investigations hinge on chronology; a visual timeline suits a medium-literacy user*

NGO / programme coordinator

A coordinator who runs field programmes, tracks beneficiaries and grant outcomes, and reports transparently to funders — sometimes from low-connectivity areas.

Intake programme and beneficiary data through a spreadsheet-like app (or mobile in the field); an agent via Lens checks it against the grant plan and writes outcomes into the shared content-addressed memory, syncing later if offline. The coordinator verifies a plain-language progress summary and ships a funder-facing report whose figures are provenance-stamped and re-checkable, with any pooled-fund decisions handled through the governance layer.

Uses: knitweb (shared memory stores beneficiary records, activities, and outcomes as provenance-stamped entries that roll up into reports) · **lens** (friendly assistant that validates entries against the grant plan, drafts reports, and translates to plain language) · **monitor** (simple dashboards showing programme progress and outcome provenance for funders) · **vbank** (governance surface for transparent voting on pooled-fund allocation and an auditable treasury when a programme manages shared money) · **bt** (off-grid Bluetooth-mesh transport so field data captured without connectivity still syncs to shared memory later — the one low-literacy role with a real off-grid need)

Avoids: pulse (the CID/ledger engine is invisible; touched only via the app) · gither (developer multi-repo tooling is irrelevant to a coordinator) · virtualpc (no need to operate a raw multi-agent runtime) · molgang (the chemistry game template is the wrong domain (though it shows the domain-knitweb pattern an NGO template would mirror))

Level: Low · **Entry:** spreadsheet-like web UI + mobile app on a friendly Lens/domain-knitweb surface

Wants: - **knitweb** — Offline-first capture that queues provenance-stamped records and reconciles on reconnect without duplicates — *Low-connectivity fieldwork needs an offline guardrail that just works for a non-technical user* - **vbank** — Simple, auditable pooled-fund voting + treasury ledger with plain-language ballots — *Transparent fund decisions need a low-literacy governance template, not raw voting primitives* - **lens** — One-tap funder report generator that maps outcomes to the grant logframe with sources attached — *Coordinators want a templated, plain-language report rather than building one from scratch* - **bt** — Bluetooth-mesh relay so multiple field workers' devices sync to one another and forward to shared memory when any reaches signal — *Off-grid programmes need peer relay, surfaced as 'nearby devices syncing', not protocol detail* - **monitor** — Beneficiary-privacy-safe progress dashboard that aggregates outcomes without exposing individuals — *Funder transparency must coexist with low-literacy-safe privacy defaults*

Service & Commerce

Chef / kitchen lead

Run service from a tablet: recipes, prep par-levels, allergens and supplier orders all live in one shared memory the kitchen trusts.

Intake: tickets, counts and supplier deliveries are scanned or tapped into the mobile app, which writes them to the shared content-addressed Web memory. Plan/do: a Lens agent reads today's covers + on-hand stock and proposes prep lists and reorders, the chef approves with one tap; verify: allergen and cost flags are checked against attested supplier records before anything is sent. Ship: approved orders and the day's recipe versions are woven back into shared memory so every shift and the next location sees the same source of truth.

Uses: knitweb (Shared memory for recipes, allergen data, prep pars and supplier batches — the kitchen's single source of truth, read/written through the app.) · **lens** (Reasoning agent that turns covers + stock into prep lists and reorder suggestions and answers plain-language questions ('what can I sell out tonight?').) · **molgang** (Used only indirectly as a domain-knitweb template — ingredient/allergen chemistry and substitution knowledge surfaced read-only inside the app.)

Avoids: pulse (Never sees the engine; CIDs, ledger and P2P sync are below the app surface and irrelevant to running a kitchen.) · bt (No off-grid mesh need; a kitchen has wifi and a tablet on the pass.) · gither (Multi-repo build navigation is a developer tool with no kitchen use.) · virtualpc (Multi-agent runtime internals are hidden; the chef only sees one assistant in the app.) · vbank (No voting/treasury governance role in daily service.)

Level: Low · **Entry:** consumer mobile app (tablet on the pass) backed by a Lens

Wants: - **lens** — Plain-language, guard-railed prep assistant ('86 the salmon, what changes?') that only proposes — never auto-orders — and shows a one-line reason for each suggestion. — *Low literacy needs guardrails and confirm-before-act, not raw queries.* - **knitweb** — Photo/voice intake of deliveries and counts that auto-writes attested stock records to shared memory. — *A chef can't type structured data mid-service; capture must be one tap or one photo.* - **molgang** — Template allergen + substitution knowledge pack a kitchen can clone and tailor without coding. — *Low-literacy roles want a ready-made domain template, not to model chemistry themselves.* - **monitor** — A simple mobile food-cost / waste tile (big numbers, color flags) instead of a graph explorer. — *Dashboards must be glanceable on a greasy tablet, not analyst tooling.* - **knitweb** — Shared recipe versioning so an updated recipe propagates to every shift/location with allergen diffs highlighted. — *Consistency across locations is the core value of shared memory for a chain kitchen.*

Real-estate agent

List, match and close properties from a phone, with every disclosure and offer provably recorded in shared memory.

Intake: listings, photos, viewings and buyer criteria are entered in the app and woven into the shared Web memory. Plan/do: a Lens agent matches buyers to listings, drafts disclosures and answers client questions from attested property facts; verify: every disclosure and price claim is checked against its provenance (cadastre, prior sale, inspection) before it's shown to a client. Ship: signed offers and disclosures are committed to shared memory, giving an auditable trail without the agent ever touching the engine.

Uses: knitweb (Shared memory holding listings, disclosures, viewing history and offers as attested, content-addressed records.) · **lens** (Buyer-to-listing matching, draft disclosures, and a plain-language Q&A assistant for clients.) · **monitor** (A friendly pipeline/funnel view of leads, viewings and offers in flight.)

Avoids: pulse (The ledger/CID engine is below the app; agents only need provable records, not protocol internals.) · bt (No off-grid mesh transport need for a connected agent.) · gither (Developer multi-repo tooling, no relevance to selling homes.) · virtualpc (Agent-coordination runtime is hidden behind the single in-app assistant.) · vbank (No governance/treasury voting in a sales workflow.)

Level: Low · **Entry:** consumer mobile app + no-code web app over a Lens

Wants: - **lens** — One-tap, template-driven disclosure draft that always cites its source facts in plain language. — *Low literacy + legal exposure means templates and visible provenance, not free-form generation.* - **knitweb** — Provenance-backed property record so each claim (sqm, last sale, zoning) links to its attested origin. — *Agents need defensible disclosures; shared memory provenance makes claims auditable.* - **lens** — Buyer-criteria matching that explains 'why this home' in one sentence per match. — *Friendly, explainable matching beats an opaque score for a non-technical agent and their clients.* - **monitor** — Mobile deal-pipeline tiles (leads -> viewings -> offers -> closed) with reminders. — *A glanceable funnel, not a graph explorer, fits a phone-first agent.* - **knitweb** — Shared client-consent + GDPR flags on

every contact record. — *Guardrails for privacy are essential and must be enforced at the memory layer, not left to the agent.*

Sales representative

Work a quota: shared account memory plus a Lens that drafts, prioritizes and forecasts, with light automation the rep can wire up themselves.

Intake: emails, calls, meeting notes and CRM updates flow into the shared Web memory via connectors and a spreadsheet-like UI. Plan/do: a Lens agent scores and sequences accounts, drafts follow-ups and surfaces next-best-action grounded in attested account history; verify: forecast and claims are checked against provenance (who said what, when) so pipeline numbers are defensible. Ship: committed deals, notes and forecasts are woven back to shared memory and optionally pushed to downstream systems via the REST API.

Uses: knitweb (Shared account/opportunity memory — emails, notes, commitments as content-addressed, attributed records.) · **lens** (Lead scoring, next-best-action, follow-up drafting and forecast reasoning over that memory.) · **monitor** (Pipeline and forecast dashboards plus light knowledge-graph view of account relationships.)

Avoids: pulse (Engine/ledger internals are below the rep's surface; the REST API and Lens are the contract they use.) · bt (No off-grid mesh need for a connected sales rep.) · gither (Multi-repo build planning is a developer concern.) · virtualpc (Multi-agent runtime is abstracted; the rep configures automations, not the runtime.) · vbank (No treasury/governance voting in a quota-carrying sales role.)

Level: Medium · **Entry:** no-code/low-code web app + spreadsheet-like UI over a Lens, with optional REST API

Wants: - **lens** — REST endpoint for next-best-action + draft reply, callable from the rep's existing CRM/email. — *Medium literacy can wire APIs; meeting them where they already work beats a new UI.* - **knitweb** — Provenance on every account fact so forecast claims trace to the email/call that backs them. — *Defensible pipeline requires attributable memory, valued by a numbers-accountable rep.* - **monitor** — Forecast vs. actual dashboard with drill-down to the underlying records. — *A medium-literacy rep wants explainable, drillable analytics, not just tiles.* - **lens** — Spreadsheet-like 'rules + AI' builder to auto-draft or auto-route on simple conditions. — *Low-code automation lets the rep self-serve without engineering.* - **knitweb** — Connector/sync so the shared memory two-ways with their CRM and inbox. — *Adoption depends on not double-entering data; sync is the make-or-break feature.*

Customer-support lead

Run a support org on shared, attested knowledge: a Lens drafts and triages, humans approve, every resolution provably enriches the knowledge base.

Intake: tickets, chats and KB articles land in shared Web memory through connectors. Plan/do: a Lens agent triages, suggests resolutions grounded in attested KB articles, and auto-drafts replies for agent approval; verify: answers are provenance-gated so the assistant can't cite a fabricated or unverified article, and confidence/sources are shown to the human. Ship: approved resolutions are woven back as new attested KB knowledge, so the shared memory compounds with every solved ticket.

Uses: knitweb (Shared, attested knowledge base + ticket history as content-addressed records that improve with each resolution.) · **lens** (Triage, retrieval-grounded answer drafting, deflection, and macro suggestion over the KB.) · **monitor** (Queue health, CSAT, deflection and KB-coverage dashboards with knowledge-graph view of gaps.)

Avoids: pulse (Ledger/CID engine is below the console; the lead consumes provenance, not protocol internals.) · bt (No off-grid mesh transport need.) · gither (Developer multi-repo tooling, not a support concern.) · virtualpc (Multi-agent runtime is hidden behind the workflow builder and Lens.) · vbank (No governance/treasury voting in support operations.)

Level: Medium · **Entry:** web console + no-code workflow builder over a Lens, plus REST API for integrations

Wants: - **lens** — Provenance-gated answer drafting that refuses to cite unattested articles and shows sources + confidence to the agent. — *Medium literacy + customer trust demands grounded, no-hallucination answers with visible provenance.* - **knitweb** — Auto-capture of approved resolutions as new attested KB nodes (with dedupe). — *The flywheel — every solved ticket enriches shared memory — is the core value.* - **monitor** — KB-coverage / knowledge-

gap explorer showing which question clusters lack attested answers. — *A lead needs to see and close knowledge gaps; the graph explorer makes that concrete.* - **lens** — No-code triage/routing workflow builder with human-in-the-loop approval gates. — *Self-serve automation with guardrails fits a medium-literacy ops lead.* - **knitweb** — REST API + connectors to Zendesk/Intercom so tickets sync into shared memory. — *Integration into the existing helpdesk is required for real adoption.*

Recruiter

Source, match and track candidates against shared, attributed role/candidate memory, with a Lens that drafts outreach and flags bias.

Intake: roles, candidate profiles, notes and interview feedback enter the shared Web memory via the app and resume parsing. Plan/do: a Lens agent matches candidates to roles, drafts outreach and interview kits, and ranks with explainable reasons grounded in attested skills/experience; verify: matches are provenance-checked (claims trace to the resume/feedback that backs them) and screened by fairness guardrails before a human reviews. Ship: stage moves, feedback and offers are committed to shared memory, giving an auditable, bias-checked hiring trail.

Uses: knitweb (Shared memory of roles, candidates, feedback and stages as attributed, content-addressed records.) · **lens** (Candidate-role matching, outreach/interview-kit drafting, and explainable ranking over that memory.) · **monitor** (Funnel, time-to-fill and diversity dashboards with a relationship view of pipelines.)

Avoids: pulse (Engine/ledger internals are below the recruiter's surface; only provenance and the app matter.) · bt (No off-grid mesh transport need.) · gither (Developer multi-repo tooling, irrelevant to hiring.) · virtualpc (Multi-agent runtime is abstracted behind a single in-app assistant.) · vbank (No governance/treasury voting in a recruiting workflow.)

Level: Medium · **Entry:** no-code web app + spreadsheet-like UI over a Lens, with optional REST API

Wants: - **lens** — Explainable match scoring with a one-line reason per candidate and a fairness/bias guardrail on ranking. — *Hiring is legally sensitive; medium-literacy recruiters need explainability and bias guardrails, not a black box.* - **knitweb** — Provenance on every candidate claim so a 'matched skill' links to the resume/feedback line that backs it. — *Defensible, auditable hiring decisions require attributable shared memory.* - **lens** — Template-driven, tone-adjustable outreach + interview-kit drafting from the role record. — *Templates and friendly drafting fit the literacy level and ensure consistency.* - **monitor** — Diversity + time-to-fill funnel dashboard with drill-down to stage records. — *A medium-literacy recruiter wants explainable, drillable hiring analytics.* - **knitweb** — REST/connector sync with the ATS so candidate data isn't double-entered. — *Integration with the existing applicant-tracking system is required for adoption.*

Skilled Trades

Plumber

A field tradesperson who diagnoses, quotes, and logs plumbing jobs with verifiable parts/warranty provenance.

Intake: snap a photo of the leak/fixture and speak the symptom into the mobile app; a Lens agent reads the shared content-addressed memory for that fixture model, code rules, and prior jobs at this address. Plan/do: the agent drafts a fix + parts list + quote the plumber edits with tap-to-confirm guardrails, then logs work-done photos. Verify/ship: the agent gates the warranty/compliance record against parts provenance in knitweb-memory and ships a signed job record the customer and next plumber can both trust.

Uses: knitweb (Shared memory holds the address/fixture history, parts batch provenance, and signed job records reused across visits and across plumbers.) · **lens** (Voice/photo agent that diagnoses, drafts quotes, pulls local code rules, and writes the job record - the only 'brain' surface the plumber sees.) · **monitor** (Simple read-only view of a property's plumbing history / open warranties when a dispute or follow-up arises.)

Avoids: pulse (Never sees the engine; CIDs/ledger internals are too low-level for a phone-first tradesperson.) · bt (Jobs are at serviced buildings with cell/wifi; no off-grid mesh need.) · gither (Multi-repo build planning is irrelevant to field service.) · virtualpc (Runs a single guided agent, not a multi-agent runtime.) · vbank (No governance/treasury voting in a one-person job-logging flow.)

Level: Low · **Entry:** consumer mobile app

Wants: - **lens** — Photo+voice symptom intake that returns a plain-language fix, parts list, and editable quote with one-tap confirm — *Low literacy: hands are dirty and on-site; typing and jargon are blockers, so the entry must be camera/mic with guardrailed taps.* - **knitweb** — Per-fixture/per-part provenance record (batch, install date, installer) keyed to a property — *Warranty and recall claims need trustworthy shared history without the plumber understanding CIDs.* - **lens** — Built-in local plumbing-code/permit checklist templates by region — *Low-literacy users need pre-baked compliance guardrails rather than authoring rules themselves.* - **monitor** — Customer-shareable, mobile-friendly job history card — *Plain-language, link-shareable proof of work builds trust without exposing engine data.* - **lens** — Offline-tolerant draft mode that syncs the signed record when back in coverage — *Basements/crawlspaces drop signal; a low-literacy user shouldn't lose a job log.*

Carpenter

A builder who plans cut lists, tracks materials, and ships verifiable build records for custom and site carpentry.

Intake: photograph the space or import a sketch and describe the build; a Lens agent reads the shared memory for material specs, prior builds, and supplier batch data. Plan/do: the agent generates a cut list, material order, and step plan the carpenter adjusts, logging progress photos as the build proceeds. Verify/ship: the agent checks measurements/material provenance against knitweb-memory and ships a signed build record (materials, dimensions, finish) for the client and any future renovation.

Uses: knitweb (Shared memory stores reusable cut-list/material templates, lumber/hardware batch provenance, and signed as-built records.) · **lens** (Mobile agent that turns a sketch/photo into a cut list, material order, and step-by-step plan.) · **monitor** (Visual project timeline / as-built gallery for the client.)

Avoids: pulse (Engine-level CIDs and ledger mechanics are below what a phone-first carpenter touches.) · bt (On-site builds typically have connectivity; no mesh requirement.) · gither (Code-repo navigation has no carpentry analog.) · virtualpc (One guided agent suffices; no multi-agent orchestration.) · vbank (No treasury or voting in a solo build workflow.)

Level: Low · **Entry:** consumer mobile app

Wants: - **lens** — Sketch/photo-to-cut-list generator with waste-minimizing board layout — *Low literacy: the value is an automatic plan from a picture, not a CAD tool the carpenter must learn.* - **knitweb** — Reusable build templates (cabinet, deck, framing) seeded and remixable — *Templates are the right guardrail so a low-literacy user starts from a known-good plan.* - **lens** — Imperial/metric unit-safe measurement capture with mistake warnings — *Plain-language safety nets prevent costly mis-cuts for non-technical users.* - **knitweb** — Material/finish provenance attached to the as-built record — *Future renovators and clients need trustworthy material history without engine knowledge.* - **monitor** — Client-facing progress timeline with before/after photos — *A friendly visual surface communicates value without exposing the substrate.*

Welder

A fabricator who runs weld procedures and ships inspectable, provenance-backed joint records for code-bound structural and pipe work.

Intake: log the joint, base metal, filler, and WPS in a spreadsheet-like UI; a Lens agent pulls matching procedure specs and prior qualified records from shared memory. Plan/do: the agent confirms the procedure parameters and required NDT, the welder records actuals (amps, passes, heat) per joint. Verify/ship: the agent gates each weld record against the WPS/PQR and inspector sign-off in knitweb-memory, shipping a signed, traceable joint log auditors can replay.

Uses: knitweb (Shared memory holds WPS/PQR specs, material/filler heat-number provenance, and signed per-joint weld + inspection records.) · **lens** (Agent that matches joints to procedures, flags out-of-parameter welds, and assembles the inspection package.) · **monitor** (Weld-map / knowledge-graph view linking joints, procedures, materials, and NDT results for audit.)

Avoids: pulse (The welder consumes verifiable records but doesn't operate CIDs/ledger internals directly.) · bt (Shop and most jobsites have connectivity; no off-grid mesh need.) · gither (Multi-repo build planning is unrelated to

fabrication.) · virtualpc (Single procedural agent fits; no multi-agent runtime.) · vbank (No governance/treasury role in weld QA.)

Level: Medium · **Entry:** spreadsheet-like UI

Wants: - **knitweb** — Heat-number / filler-lot provenance traceability per joint — *Medium literacy: code work demands material traceability the user can query and trust by lot.* - **lens** — WPS-aware parameter checker that flags welds outside qualified ranges — *Mid-literate welders want a rules engine that validates against the procedure, not just a logbook.* - **monitor** — Interactive weld map overlaying joints, NDT status, and provenance — *A graph explorer matches a procedural user who reasons about joint relationships.* - **lens** — Inspector sign-off workflow producing a replayable inspection package — *Verification must yield an auditor-grade, tamper-evident bundle.* - **knitweb** — Importable WPS/PQR templates by code (AWS/ASME) reused across shops — *Shared, versioned procedure templates raise consistency without bespoke setup.*

HVAC technician

A service tech who commissions, services, and verifies HVAC systems with provenance-backed refrigerant, efficiency, and warranty records.

Intake: pull up the unit in a no-code web app and log readings (pressures, superheat, refrigerant charge); a Lens agent reads the equipment's history and manufacturer specs from shared memory. Plan/do: the agent diagnoses, recommends parts/charge adjustments, and tracks refrigerant added/recovered against regulatory limits. Verify/ship: the agent gates the service record against spec and compliance (e.g. refrigerant logs) in knitweb-memory and ships a signed commissioning/service report.

Uses: knitweb (Shared memory stores per-unit service history, refrigerant chain-of-custody, and signed commissioning/service reports.) · **lens** (Diagnostic agent that reads readings, matches manufacturer specs, and drafts the compliant service record.) · **monitor** (Fleet/site dashboard of unit health, efficiency trends, and open warranties.)

Avoids: pulse (Trusts verified records but never touches ledger/CID internals.) · bt (Serviced buildings have connectivity; no mesh transport need.) · gither (No multi-repo navigation relevance.) · virtualpc (A single guided agent is sufficient.) · vbank (No treasury/governance in service operations.)

Level: Medium · **Entry:** no-code web app

Wants: - **lens** — Reading-driven diagnostic agent (superheat/subcooling) with plain recommendations — *Medium literacy: techs enter numbers and want actionable diagnosis, via a no-code form not an API.* - **knitweb** — Refrigerant chain-of-custody ledger (added/recovered) per unit — *Regulatory traceability needs trustworthy shared records the tech can rely on.* - **monitor** — Per-unit efficiency-trend dashboard with warranty flags — *A friendly dashboard matches a mid-literate user tracking fleets over time.* - **lens** — Manufacturer spec/template library auto-matched to the unit model — *Pre-loaded specs reduce setup and keep diagnoses grounded in trustworthy data.* - **knitweb** — Signed commissioning report shareable with building owner and next tech — *Portable, trustworthy records prevent re-diagnosis and disputes.*

Automotive mechanic

A technician who diagnoses vehicles and ships provenance-backed repair histories tied to VIN, parts, and recall data.

Intake: enter the VIN and OBD codes in a no-code web app; a Lens agent reads the vehicle's repair history, OEM bulletins, and recall data from shared memory. Plan/do: the agent proposes a diagnosis and parts list with genuine/aftermarket provenance, and the mechanic logs labor and replaced parts. Verify/ship: the agent gates the repair record against parts provenance and warranty rules in knitweb-memory, shipping a signed service history that follows the VIN.

Uses: knitweb (Shared memory holds VIN-keyed repair history, OEM bulletins/recalls, and part serial/batch provenance reused across shops.) · **lens** (Diagnostic agent mapping OBD codes + symptoms to fixes and a provenance-checked parts list.) · **monitor** (Per-vehicle service-history and recall-status view for the customer and shop.)

Avoids: pulse (Consumes verifiable records; never operates CIDs or the ledger directly.) · bt (Shops have connectivity; no off-grid mesh need.) · gither (Multi-repo build planning has no automotive analog.) · virtualpc (One guided diagnostic agent suffices; no multi-agent runtime.) · vbank (No governance/treasury function in shop work.)

Level: Medium · **Entry:** no-code web app

Wants: - **lens** — VIN + OBD-code diagnostic agent returning ranked fixes and a parts list — *Medium literacy: structured code entry with an actionable diagnosis fits a no-code form, not raw APIs.* - **knitweb** — Part serial/batch provenance to flag counterfeit or non-OEM parts — *Trustworthy parts traceability protects warranty and safety without engine knowledge.* - **knitweb** — VIN-keyed portable service history reusable across independent shops — *Shared content-addressed history is the core value: the record follows the car, not the shop.* - **lens** — Recall/TSB auto-match against the entered VIN — *Pulling trustworthy OEM data into diagnosis is a mid-literacy guardrail against missed safety items.* - **monitor** — Customer-facing service-history and recall-status card — *A friendly, shareable surface conveys trust without exposing the substrate.*

Agriculture & Environment

Farmer

Run the farm day-to-day and log what actually happened in the field so the shared memory stays trustworthy.

Intake: snap a photo / dictate an observation (pest, irrigation, planting) into the mobile app; an on-device Lens agent plans the next task from a crop template and the farm's history in the shared content-addressed memory. Do the task, then verify by confirming the agent's plain-language summary ('sprayed Field 3, 40L'); on tap it is woven into the knitweb fabric as a content-addressed record, syncing to the engine when signal returns.

Uses: knitweb (The farm's living record (fields, crops, treatments, harvests) lives in the shared content-addressed memory; the farmer reads/adds entries through the app without ever seeing CIDs.) · **lens** (An assistant agent over the memory turns photos/voice into logged events, surfaces 'what to do next', and translates jargon into plain language.) · **molgang** (Used only as the friendly template pattern (faucet/propose/confirm flow) that the farm app's onboarding and guided tasks are modeled on; not used for chemistry itself.)

Avoids: pulse (Never sees the engine; CIDs, CBOR, ledger and PLS settlement are entirely behind the app — too low-level.) · bt (No DEX/basket-trust need; the farmer is logging activity, not trading orders.) · gither (Not a developer; no repo, build, or release surface.) · virtualpc (No multi-agent server to host or operate; a single in-app assistant is enough.) · vbank (Individual operation, not running votes or a treasury.)

Level: Low · **Entry:** consumer mobile app (field-facing, voice/photo-first, offline-tolerant)

Wants: - **lens** — Offline-first voice + photo capture that an agent turns into structured field events, with a one-tap plain-language confirm before anything is written. — *Low literacy and poor connectivity: the farmer must add trustworthy records without typing, jargon, or worrying about sync.* - **knitweb** — Crop/field templates (planting, spray, harvest) that pre-fill the memory schema so logging is fill-in-the-blank, not free-form. — *Guardrails and templates keep low-literacy entries consistent enough to be useful downstream.* - **knitweb** — Per-farm private memory scope with simple opt-in sharing to an agronomist or buyer. — *Farmers need to control who sees field data without understanding keys or permissioning.* - **lens** — Plain-language daily 'next best task' card driven by the farm's own history plus weather. — *Turns the shared memory into actionable advice instead of a database the farmer must query.*

Agronomist

Advise many farms by reading their field memory, designing trials, and writing back agronomic recommendations.

Intake: pull a portfolio of farms' field records from the shared content-addressed memory into a spreadsheet-like view; plan trials or prescriptions with a Lens agent that compares fields and flags anomalies. Do the analysis, verify recommendations against provenance (which observations support each claim), then ship a signed recommendation back into the knitweb fabric so the farmer's app surfaces it.

Uses: knitweb (Reads cross-farm field history and writes back recommendations/trial designs as content-addressed records in the shared memory.) · **lens** (Orchestrates comparison and trial analysis over many farms, drafts prescriptions, and exposes the provenance chain behind each suggestion.) · **monitor** (Uses the knowledge-graph explorer to see how fields, treatments, and outcomes relate across a region.) · **vbank** (Occasionally participates in a cooperative's poll (e.g. approving a shared trial protocol) through a friendly voting surface.)

Avoids: pulse (Touches the engine only via the app/SDK abstractions; does not run nodes or handle canonical encoding.) · bt (No off-grid mesh or trading need in advisory work.) · gither (Not building software; consumes finished surfaces.) · virtualpc (No need to host a multi-agent runtime; a hosted Lens app suffices.)

Level: Medium · **Entry:** no-code/low-code web app with a spreadsheet-like UI over the shared memory

Wants: - **monitor** — Region-level knowledge-graph view that overlays treatments against yield/health outcomes across many farms. — *Medium literacy wants visual cross-farm comparison without writing graph queries.* - **lens** — Trial-design assistant that proposes control/treatment splits and tracks them as linked records with provenance. — *Agronomists need rigor (which fields, which observations) but at no-code level, not API depth.* - **knitweb** — Signed recommendation records that link back to the exact field observations they cite. — *Advice must be auditable so farmers and buyers can trust why a prescription was made.* - **lens** — Spreadsheet-like export/import that round-trips with the shared memory without breaking content-addressing. — *Agronomists live in spreadsheets; they need their familiar tool to read/write the fabric safely.* - **monitor** — Anomaly flags (outlier yields, unexpected pest spread) pushed as a review queue. — *Lets an advisor cover many farms by triaging exceptions instead of reading every record.*

Forester

Survey, mark, and manage forest stands in the field, often with no connectivity, building a verifiable stand history.

Intake: in the field (often offline) record stand observations, GPS, and marked trees via the mobile app; a local Lens agent plans the survey route and the next silvicultural action from the stand's prior record. Do the survey, verify the agent's summary on the device, then ship the records over Bluetooth mesh to a base node that weaves them into the shared content-addressed memory when a connected peer is reached.

Uses: knitweb (Each stand's history (inventory, thinning, fire/pest events) is the shared content-addressed memory the forester adds to via the app.) · **lens** (On-device assistant plans survey routes, identifies species/condition from photos, and summarizes actions in plain language.) · **bt** (Off-grid Bluetooth mesh carries field records back from no-signal forest to a connected base node — the one substrate piece this role genuinely needs.)

Avoids: pulse (Never operates the engine directly; sync, CIDs and settlement are hidden behind the app and the base node.) · gither (Not a developer; no code-forge surface.) · virtualpc (No multi-agent server to run; a single field assistant is enough.) · vbank (Field management is not governance/voting.) · monitor (Rarely needs a desktop graph explorer; a planning manager would, but the field forester works in the app.)

Level: Low · **Entry:** consumer mobile app with offline + off-grid mesh support

Wants: - **bt** — Background field-record relay over Bluetooth mesh that auto-flushes to a base node when any peer comes in range, with a clear 'X records pending sync' indicator. — *Foresters work where there is no signal; off-grid capture must be invisible and reliable, surfaced in plain language.* - **lens** — Photo-based species/health identification and a guided stand-survey checklist that works fully offline. — *Low literacy + no connectivity: the agent must guide and capture without cloud calls or jargon.* - **knitweb** — Geo-anchored stand records (GPS + stand ID) as a template so survey entries are consistent across crews. — *Templates and guardrails keep multi-crew field data comparable over years.* - **lens** — Plain-language conflict resolution when two crews logged the same stand differently after mesh sync. — *Low-literacy users need merge conflicts explained and resolved by tapping, not by understanding content-addressing.*

Fisheries manager

Track catch, enforce quotas, and steward stocks across many vessels with an auditable shared record.

Intake: catch reports and vessel logs flow in from boats (app or REST) into the shared content-addressed memory; a Lens agent reconciles them against quota and stock models and flags overruns. Do the allocation/closure decision,

verify it against the provenance of each catch record, then ship quota decisions and stock assessments back into the knitweb fabric where vessels' apps and regulators can read them.

Uses: knitweb (The shared, append-only record of catches, quotas, and stock assessments that all parties read from and trust.) · **lens** (Reconciles catch reports against quota, runs stock-status summaries, and explains flags with the underlying records.) · **monitor** (Dashboard view of fleet activity, quota burn-down, and stock-health knowledge graph.) · **vbank** (Runs governance polls for cooperative quota allocation and treasury of shared management funds.)

Avoids: pulse (Uses the REST/app abstractions; does not run engine nodes or touch canonical encoding.) · bt (Reporting is connectivity-dependent at dock/office; no off-grid mesh requirement for the manager.) · gither (Not building software.) · virtualpc (No multi-agent runtime to self-host.)

Level: Medium · **Entry:** no-code web app + REST API for catch/quota integrations

Wants: - **knitweb** — Tamper-evident catch records with provenance so a logged catch cannot be silently altered after submission. — *Quota enforcement needs an auditable record regulators and vessels both trust.* - **lens** — Quota reconciliation agent that flags overruns and explains each flag from the cited catch records. — *Medium literacy wants automated enforcement with a human-readable why, not raw query writing.* - **vbank** — Cooperative quota-allocation polls with deterministic one-vessel/one-person tallying and signed, auditable results. — *Fair shared-stock governance requires transparent, independently verifiable voting.* - **monitor** — Fleet quota burn-down and stock-health dashboard updated live from the fabric. — *Managers need an at-a-glance operational view across many vessels.* - **knitweb** — Simple REST ingestion for vessel catch reports that maps to the memory schema. — *Boats use varied logging gear; an API lets reports flow in without manual re-entry.*

Environmental scientist

Build reproducible, provenance-deep environmental analyses on shared field/sensor data and publish citable results.

Intake: pull sensor streams and field observations from the shared content-addressed memory via the SDK; plan an analysis pipeline that a Lens orchestration agent runs, keeping every input pinned by CID. Do the computation, verify by replaying the provenance walker so results are byte-for-byte reproducible, then ship the dataset and findings back to the fabric as anchored, citable records.

Uses: knitweb (Treats the fabric as a content-addressed data lake — pulls inputs and publishes datasets/findings as immutable, anchored records.) · **pulse** (Uses the Python SDK/interpret layer directly for deterministic CIDs, the provenance walker, and OriginTrail anchoring — the rare role that does touch engine APIs.) · **lens** (Orchestrates multi-step analysis agents over the data with full provenance capture.) · **monitor** (Knowledge-graph explorer to inspect data lineage and relationships before/after analysis.) · **vbank** (Occasionally for governance of a shared research dataset or open-data commons decisions.)

Avoids: bt (Lab/cluster compute is connected; no off-grid mesh transport need (their field collaborators use bt, they don't).) · gither (Uses standard dev tooling; not adopting the knowledge-forge for analysis work itself.) · virtualpc (Typically runs analysis on their own compute, not a self-hosted multi-agent OS.) · molgang (A chemistry learning game; not their domain or workflow.)

Level: High · **Entry:** Python SDK + REST API over the shared memory and provenance walker

Wants: - **pulse** — Stable, documented Python SDK + REST endpoints for pinning inputs by CID and replaying the provenance walker. — *High literacy needs determinism and reproducibility as a first-class API, not behind a no-code wall.* - **pulse** — Deterministic dataset versioning so a published analysis cites exact input CIDs and is byte-for-byte re-runnable. — *Scientific reproducibility demands content-addressed inputs and stable hashing guarantees.* - **knitweb** — OriginTrail-anchored, citable dataset records with a stable external identifier (UAL). — *Findings must be independently verifiable and citable in publications, not just internal.* - **lens** — Provenance-depth query API that returns the full lineage subgraph for any result. — *Auditors and peer reviewers need to trace every claim back to source observations programmatically.* - **monitor** — Lineage-diff view between two analysis runs over the knowledge graph. — *High-literacy users debug and defend results by comparing exactly what changed between runs.*

Extended Health

Dentist

A treating dentist who weaves patient findings, imaging, and treatment plans into shared memory and lets a Lens cross-check against verified clinical guidance.

Intake: at chairside the dentist dictates/forms the exam (charting, radiograph findings) into the no-code app, which weaves each finding as a signed, content-addressed record into the shared memory with the dentist as attributed originator. Plan: a Lens reasons over that patient subgraph plus provenance-traced clinical guidelines to draft a treatment plan and flag contraindications; the dentist edits and approves. Ship: the approved plan is re-anchored as a new immutable CID with a provenance edge back to the source findings, so referrals/insurers/recall agents read the same verified version.

Uses: lens (Reasons over the patient subgraph and guideline provenance to draft plans, surface contraindications, and answer 'why' with cited source records) · **knitweb** (The shared content-addressed memory holding signed charting, imaging refs, and treatment-plan records the practice and referrals all read) · **monitor** (Knowledge-graph explorer to visually trace a patient's findings-to-plan history and provenance before approving)

Avoids: pulse (Never sees the engine; CID/ledger/P2P internals are abstracted by the app, too low-level for a clinician) · bt (No off-grid Bluetooth-mesh need in a wired clinic) · gither (Multi-repo build navigation is a developer concern, irrelevant to chairside work) · virtualpc (Multi-agent runtime orchestration is plumbing the friendly app hides) · molgang (Chemistry knowledge-game template, unrelated to dentistry) · vbank (No treasury/voting governance role in single-practice clinical work)

Level: Medium · **Entry:** no-code web app (clinical Lens surface) on desktop + tablet chairside

Wants: - **lens** — Cited-contraindication checker: every flagged risk must link to the originator+asset_cid of the guideline it came from, shown in plain language — *Medium-literacy clinician needs to trust and defend a recommendation without reading the engine; provenance must surface as a readable citation, not a raw CID* - **knitweb** — Dental charting template (tooth/surface schema) that weaves structured findings with one form, no JSON — *A Medium-literacy dentist wants a guardrailed clinical template, not to hand-author content-addressed records* - **monitor** — Per-patient timeline view that diffs successive treatment-plan CIDs and highlights what changed and who signed it — *Clinician needs an auditable, visual 'what changed since last visit' rather than a developer graph query* - **lens** — Approve-before-ship gate so no agent-drafted plan enters shared memory until the dentist signs it — *Clinical liability requires a human-in-the-loop guardrail appropriate to a non-engineer surface* - **knitweb** — Scoped subscription so a referral specialist sees only the relevant patient subgraph, not the whole practice memory — *Patient-privacy boundary must be enforced by the memory layer, expressed as a simple share toggle for a Medium-literacy user*

Clinical psychologist

A therapist who keeps confidential session notes and outcome tracking in a private, plain-language app while a Lens summarizes progress and surfaces evidence-based suggestions.

Intake: after a session the psychologist types notes and a few outcome-scale numbers into the mobile app, which weaves them as private, signed records into their own slice of shared memory. Plan/Do: a Lens summarizes trends across sessions and suggests evidence-based next steps in plain language, citing the source guidance. Verify/Ship: the therapist reviews and confirms; only confirmed notes persist, and nothing leaves the private scope unless they explicitly share with a supervisor.

Uses: lens (Summarizes session-over-session progress and suggests plain-language, evidence-cited next steps the therapist approves) · **knitweb** (Private content-addressed store for confidential, signed session notes and outcome scales, scoped to the therapist)

Avoids: pulse (Engine internals (CIDs, P2P, ledger) are fully hidden; a Low-literacy clinician never touches them) · monitor (Graph-explorer dashboards are too technical; the app gives a simple progress view instead) · bt (No off-grid mesh need in a consulting room) · gither (Developer multi-repo tooling, irrelevant) · virtualpc (Agent-runtime plumbing hidden by the app) · molgang (Chemistry game template, unrelated) · vbank (No governance/treasury role)

Level: Low · **Entry:** consumer mobile app + spreadsheet-like UI for session notes

Wants: - **knitweb** — Private-by-default encrypted scope where notes never sync off-device until an explicit share, with a one-tap 'who can see this' control — *Low-literacy role handling highly sensitive data needs confidentiality as an unmissable default guardrail, not a config flag* - **lens** — Plain-language progress summaries with a 'where did this come from' link to the cited evidence, no jargon or CIDs — *Low technical literacy demands readable output and gentle provenance, not raw graph or hashes* - **lens** — Safety-flag prompt that surfaces risk language (e.g. self-harm) for clinician review, never auto-acting — *Guardrail suited to a non-technical, high-stakes context: assist and flag, human always decides* - **knitweb** — Validated outcome-scale templates (PHQ-9, GAD-7) so scores weave as structured records from a simple form — *Low-literacy user needs templates and dropdowns, not to model data by hand* - **lens** — One-tap supervision export bundle that shares only a chosen client's de-identified summary — *Sharing must be a single guarded action with built-in de-identification for a non-technical user*

Dietitian

A dietitian who builds client nutrition plans from templates and lets a Lens check them against verified nutritional and clinical guidance.

Intake: the dietitian captures client goals, restrictions, and biometrics through guided forms that weave a signed client record into shared memory. Plan/Do: a Lens drafts a meal plan from verified nutrition data, flags allergen/condition conflicts, and explains choices in plain language; the dietitian adjusts and approves. Ship: the approved plan is anchored as a shareable record the client app reads, with provenance back to the guidance used.

Uses: lens (Drafts and checks nutrition plans against verified food/clinical data and flags conflicts with the client's restrictions) · **knitweb** (Holds signed client profiles and approved plans as content-addressed records shared between dietitian and client apps)

Avoids: pulse (Engine layer fully abstracted; a Low-literacy practitioner never sees CIDs or the ledger) · monitor (Graph dashboards too technical; the app shows a simple plan/adherence view) · bt (No off-grid mesh use case) · gither (Developer tooling, irrelevant) · virtualpc (Agent-runtime plumbing hidden) · molgang (Chemistry game template, unrelated to dietetics) · vbank (No governance/treasury role)

Level: Low · **Entry:** consumer mobile app with template-driven meal/plan builder

Wants: - **lens** — Allergen and condition conflict guardrail that hard-blocks a plan containing a flagged item until resolved, in plain language — *Low-literacy role needs a strong safety guardrail rather than a subtle warning they could miss* - **knitweb** — Curated, provenance-backed food/nutrient reference set the plan builder draws from, so values trace to a verified originator — *Trustworthy plans require verified source data, but the user should never have to source or verify it themselves* - **lens** — Template meal-plan generator (cultural/budget/restriction presets) producing an editable plan from a few taps — *Low technical literacy wants templates and presets, not free-form construction* - **knitweb** — Client-shareable plan link that the client's consumer app reads read-only, with adherence check-ins flowing back — *A simple share-and-track loop must be built into the friendly surface, hiding all addressing* - **lens** — Plain-language 'why this food' explanations citing the guidance behind each recommendation — *Provenance must be readable reassurance for a non-technical practitioner and client*

Paramedic

A field paramedic who records assessments and interventions hands-busy, syncing to shared memory when connectivity returns, with a Lens giving protocol-driven decision support.

Intake: en route and on-scene the paramedic captures vitals, history, and interventions by voice/quick-tap into an offline-first app that weaves signed records locally even with no signal. Plan/Do: a Lens runs protocol-based decision support (dosing, triage, contraindications) against verified guidance available on-device. Ship: when a link (cellular or Bluetooth mesh between units) returns, records sync to shared memory and hand off to the receiving ED as a verified, time-stamped chain.

Uses: lens (On-device protocol decision support: triage category, drug dosing, and contraindication checks against verified field guidance) · **knitweb** (Local-first signed capture of vitals/interventions that syncs into shared memory and

hands off to the ED record) · **bt** (Off-grid Bluetooth-mesh transport to relay records between crews and vehicles when cellular is down)

Avoids: pulse (The sync engine works under the app; a paramedic never touches CID/P2P/ledger internals) · monitor (Graph-explorer dashboards are an ops/back-office tool, not for the field) · gither (Developer multi-repo tooling, irrelevant in the field) · virtualpc (Agent-runtime orchestration is hidden plumbing) · molgang (Chemistry game template, unrelated) · vbank (No governance/treasury role on an ambulance)

Level: Medium · **Entry:** ruggedized consumer mobile app (offline-first), voice-driven

Wants: - **knitweb** — Robust offline-first weave-and-queue: every record is signed and content-addressed locally, then merges without conflict on reconnect — *Medium-literacy field user must trust that nothing is lost off-grid and that ordering/integrity hold once synced* - **bt** — Crew-to-crew and vehicle-to-ED Bluetooth-mesh relay of patient records when cellular is unavailable — *This is the one role with a genuine off-grid need; mesh transport is mission-critical, not optional* - **lens** — Voice/quick-tap protocol decision support (triage, weight-based dosing) that works fully on-device with cited protocol source — *Hands-busy, intermittently-connected context needs local, fast, defensible support — provenance for medico-legal defensibility* - **knitweb** — Tamper-evident, time-stamped handoff chain so the ED can verify the field record was not altered — *Medium-literacy user relies on the memory layer to provide a defensible chain of custody automatically* - **lens** — Guarded auto-fill of repetitive fields from voice with confirm-before-commit — *Speed with a confirm guardrail fits a Medium-literacy, high-pressure workflow*

Optometrist

An optometrist who imports exam-device data and prescriptions into shared memory and uses a Lens to track ocular changes and flag referral-worthy findings over time.

Intake: exam results (refraction, IOP, fundus images, OCT) import into the no-code app and weave as signed records into the patient's subgraph in shared memory. Plan: a Lens compares against prior visits, flags progression (e.g. glaucoma risk) and referral thresholds with cited guidance; the optometrist reviews. Ship: the confirmed prescription/referral is anchored as a new CID with provenance back to the device readings, shareable to dispensing or an ophthalmologist.

Uses: lens (Compares longitudinal exam data, flags progression/referral thresholds, and explains findings with cited clinical guidance) · **knitweb** (Stores signed refraction/imaging/prescription records as a content-addressed patient subgraph shared with dispensing and referrals) · **monitor** (Knowledge-graph explorer to visualize trend lines across visits and inspect provenance before confirming a referral)

Avoids: pulse (Engine internals abstracted by the clinical app; too low-level for a clinician) · bt (No off-grid mesh need in a fixed clinic) · gither (Developer multi-repo tooling, irrelevant) · virtualpc (Agent-runtime plumbing hidden by the app) · molgang (Chemistry game template, unrelated) · vbank (No governance/treasury role in clinical optometry)

Level: Medium · **Entry:** no-code web app (clinical Lens surface) with device-import

Wants: - **knitweb** — Device-import adapters (refractor/OCT/fundus) that weave structured signed records without manual entry — *Medium-literacy clinician needs verified instrument data captured by template/import, not hand-keyed* - **lens** — Longitudinal progression detector with cited referral thresholds (e.g. IOP/cup-disc trend) surfaced in plain language — *Trustable, defensible flagging requires provenance to the guideline, readable without engine knowledge* - **monitor** — Visit-over-visit trend visualization with provenance hover showing which device reading and originator each datapoint came from — *Medium-literacy user benefits from a visual, auditable trend rather than raw graph queries* - **lens** — Confirm-before-ship gate on prescriptions and referrals so no agent output enters shared memory unsigned — *Clinical accountability needs a human sign-off guardrail at a non-engineer surface* - **knitweb** — Scoped referral share that exposes only the relevant ocular subgraph to the receiving ophthalmologist — *Privacy boundary enforced by the memory layer, expressed as a simple share control for a Medium-literacy user*

Extended Finance

Investment banker

Structures deals and valuations on verifiable, provenance-tracked financials rather than emailed spreadsheets of unknown lineage.

Intake: pull a target's signed finance-knitweb entries and comparables from the shared content-addressed memory by CID. Plan/do: a Lens orchestration runs valuation/scenario models against those Fibers, citing every input CID; agents fan out to fetch filings and re-run sensitivity. Verify/ship: confirm each model input resolves to a canonical CID and the deal memo's provenance walks clean, then publish the signed memo + model bundle back to the Web for the syndicate.

Uses: lens (Orchestrates multi-step valuation/scenario agents and produces a cited, reproducible deal memo over the shared memory) · **knitweb** (Reads and writes deal artifacts (models, comps, memos) as content-addressed records so every input is referenceable by CID) · **monitor** (Knowledge-graph explorer for diligence: traces a target's financial provenance and counterparty relations) · **pulse** (Via SDK only — resolves canonical CIDs and verifies signatures on finance-entry records; does not operate the node internals)

Avoids: bt (Off-grid Bluetooth mesh transport is irrelevant to a desk with connectivity) · molgang (Chemistry domain-knitweb template, unrelated to deal structuring) · gither (Multi-repo build planner is engineering tooling, not a banking surface) · virtualpc (Lens already abstracts agent coordination; no need to touch the raw multi-agent runtime)

Level: High · **Entry:** Python SDK + REST API (Lens-orchestrated), with monitor dashboards for diligence review

Wants: - **lens** — Deterministic, reproducible valuation pipelines that emit a full input-CID manifest with every model run — *High-literacy users demand determinism and provenance depth so a model output can be independently re-derived and defended in committee* - **pulse** — Provenance-walk API that returns the complete CID lineage and signature chain for any finance-entry record — *Diligence requires proving where each number came from, not trusting a counterparty's spreadsheet* - **knitweb** — Comparables/precedent-transaction record schema addressable by CID with versioned revisions — *Lets bankers cite an exact frozen snapshot of a comp set rather than a mutable file* - **monitor** — Counterparty relationship graph with drill-down from any deal node to its underlying signed records — *Diligence and conflict checks need to traverse ownership/relation edges, not just read totals* - **vbank** — Syndicate sign-off / approval ledger tying memo CIDs to who voted to proceed — *Deal governance needs an auditable record of committee approvals bound to the exact artifact approved*

Actuary

Builds reserving and pricing models on integer-exact, reproducible data whose every assumption and input is content-addressed and re-runnable.

Intake: pull policy/claims cohorts and assumption sets from the shared memory by CID. Plan/do: run reserving/pricing models in the SDK; Lens-orchestrated agents fan out to stress-test scenarios and re-execute against the same Fibers for reproducibility. Verify/ship: confirm results round-trip byte-for-byte (no float drift), attach the assumption-set CID to the result, and publish the signed valuation back to the Web for sign-off.

Uses: lens (Orchestrates scenario/stress-test runs and assembles a cited reserving report with reproducible steps) · **knitweb** (Stores assumption sets, mortality/lapse tables, and valuation outputs as content-addressed, versioned records) · **pulse** (SDK use of integer/canonical-encoding and CID primitives so monetary results are exact and reproducible) · **monitor** (Reviews portfolio aggregates and traces a reserve number back to its contributing records)

Avoids: bt (No off-grid need for an actuarial desk) · molgang (Unrelated chemistry domain template) · vbank (Treasury/voting governance is outside the actuarial modelling loop) · gither (Repo build-planning tooling is not part of the modelling surface)

Level: High · **Entry:** Python SDK + notebook, with monitor for portfolio/graph review

Wants: - **pulse** — First-class fixed-point / integer money type guaranteed float-free through canonical encoding for actuarial math — *Reserving cannot tolerate float drift; high-literacy users need byte-exact reproducibility across clients*

and re-runs - **knitweb** — Immutable, versioned assumption-set records (mortality, lapse, discount) addressable by CID — *Regulators require that a reported reserve be reproducible from the exact assumptions used at valuation date* - **lens** — Reproducible scenario-grid orchestration that re-executes each stress against the frozen input CIDs — *Stress testing demands determinism so two runs of the same scenario give identical numbers* - **monitor** — Reserve-attribution drill-down: from an aggregate figure down to contributing cohorts and assumption CIDs — *Actuaries must explain movement in reserves by tracing to underlying signed inputs* - **pulse** — Provenance receipts proving which data snapshot and code version produced a valuation — *Audit and peer review need a verifiable record of exactly what was run*

Tax advisor

Computes positions and files returns from client financials whose provenance is verifiable, with rules applied transparently and citably.

Intake: import a client's signed finance-knitweb ledger entries into a spreadsheet-like Lens app. Plan/do: rule-templated agents classify items and compute the position, each step citing the source record CID and the rule applied. Verify/ship: review the explained computation, confirm inputs resolve to canonical CIDs, then publish the signed return/advice bundle to the shared memory for the client and the firm.

Uses: lens (Runs rule-based tax-treatment agents and presents an explainable, cited computation in a friendly UI) · **knitweb** (Reads client ledger entries and writes the signed return/advice as content-addressed records) · **monitor** (Optional review view to trace a tax line back to the underlying transactions)

Avoids: pulse (Never sees the engine; CID resolution and signing are abstracted behind the Lens app) · bt (No off-grid mesh need in a tax practice) · molgang (Chemistry domain template, irrelevant) · virtualpc (Raw multi-agent runtime is too low-level; Lens handles orchestration) · gither (Engineering multi-repo tooling, not a tax surface)

Level: Medium · **Entry:** Spreadsheet-like / no-code web app over a Lens surface, with REST API for firm integrations

Wants: - **lens** — Plain-language explanation panel showing which rule was applied to each line and which source record it used — *Medium-literacy advisors need to defend a position to clients and authorities without reading code or internals* - **lens** — Versioned rule/template packs per jurisdiction and tax year that are pinned to a return — *A return must record exactly which rule version produced it; advisors pick templates, not write logic* - **knitweb** — One-click signed export bundle of return + cited inputs as a content-addressed package — *Filing and client handoff need a tamper-evident package without the advisor handling CIDs manually* - **monitor** — Click-through from a tax line to its supporting transactions — *Mid-literacy review depends on visual traceability rather than querying the graph by hand* - **lens** — Guardrails that block filing when an input record fails signature/provenance verification — *Protects a non-expert from filing on data of unverified lineage*

Bookkeeper

Records day-to-day transactions into a self-balancing ledger that refuses errors before they are saved.

Intake: enter or import receipts/invoices in a friendly grid app. Plan/do: a templated assistant categorizes each entry and the finance-knitweb enforces debits-equal-credits, refusing any unbalanced entry before it is signed and woven into the shared memory. Verify/ship: review the auto-flagged exceptions in plain language, approve, and the balanced entries are stored as content-addressed records for the accountant.

Uses: knitweb (Each approved entry is stored as a signed, content-addressed finance record in the shared ledger) · **lens** (Friendly assistant suggests categories and explains exceptions; never exposes CIDs or crypto)

Avoids: pulse (Never touches the engine; balancing, signing, and CIDs all happen invisibly under the app) · bt (No off-grid use case) · molgang (Unrelated chemistry template) · monitor (Graph explorer is too technical; the bookkeeper works in a simple grid) · vbank (Governance/treasury is out of scope for data entry) · gither (Developer tooling, irrelevant) · virtualpc (Multi-agent runtime is far too low-level)

Level: Low · **Entry:** No-code web app / spreadsheet-like UI (mobile-friendly) over the shared memory

Wants: - **knitweb** — Inline plain-language explanation when the double-entry invariant refuses an unbalanced entry — *Low-literacy users need to understand why an entry was rejected without seeing a stack trace or accounting jargon* -

lens — Receipt-photo / mobile capture that auto-suggests account and amount — *Low-literacy, on-the-go users need a phone-first capture flow, not a desktop ledger* - **lens** — Prebuilt chart-of-accounts and category templates for common small-business types — *Guardrails and templates keep a non-expert from miscategorizing; they choose, not configure* - **lens** — Undo / draft-before-commit so entries can be corrected before they are signed — *Once woven into the shared memory records are immutable; a beginner needs a safe edit window first* - **knitweb** — Simple monthly close summary auto-generated from the signed entries for handoff to the accountant — *The bookkeeper needs a clean, trustworthy package to hand up without touching provenance internals*

Mortgage broker

Matches borrowers to lenders using verifiable income and obligation records, with affordability checks applied transparently.

Intake: collect a borrower's income/obligation records, ideally pulled as signed entries from the shared memory rather than re-keyed PDFs. Plan/do: a guided Lens app runs affordability and eligibility checks against lender rule-templates, each result citing the source record. Verify/ship: review the plain-language eligibility result, then publish a signed, content-addressed application package the lender can independently verify.

Uses: lens (Guided eligibility/affordability flow with templated lender criteria, presented in plain language) · **knitweb** (Reads verified borrower income records and writes the signed application package as a content-addressed bundle)

Avoids: pulse (Never sees the engine; verification and CIDs are hidden behind the app) · bt (No off-grid mesh need) · molgang (Chemistry template, irrelevant) · monitor (Graph explorer is too technical for this surface) · vbank (Governance/treasury is unrelated to brokering) · gither (Developer tooling, not a broker app) · virtualpc (Raw multi-agent runtime is too low-level)

Level: Low · **Entry:** Consumer/web app over a Lens surface (mobile-friendly), with REST API for lender integrations

Wants: - **lens** — Guided step-by-step application wizard with plain-language affordability results — *Low-literacy brokers and borrowers need a friendly mobile flow, not a query interface* - **knitweb** — Verified-income record import so a borrower's payslip/statement entries are pulled with proven provenance — *Replaces re-keying PDFs with tamper-evident records the lender can trust without the broker handling crypto* - **lens** — Versioned per-lender eligibility rule packs the broker selects, not codes — *Guardrails and templates let a non-expert apply the correct current criteria safely* - **lens** — Block-and-explain when a required income record fails verification — *Protects a low-literacy user from submitting an application on unverified data* - **knitweb** — One-click signed application package the lender can independently verify by CID — *Clean, tamper-evident handoff to the lender without the broker managing provenance manually*

Government & Defense

Judge

Weighs evidence and writes rulings, demanding that every fact be traceable to a signed, tamper-evident source before it enters the record.

Intake: clerks and parties submit evidence/filings that land as signed, content-addressed records on the shared knitweb memory. Plan/Do: the judge asks a Lens-driven assistant to assemble and cross-check the evidence chain, with every claim resolving to a CID and its full ancestry. Verify/Ship: the judge confirms each cited fact's provenance and attestation, then issues the ruling as its own signed record anchored back into the fabric so the chain of reasoning is auditable on appeal.

Uses: knitweb (The case file lives as content-addressed, immutable records (filings, exhibits, prior rulings); citing a fact means citing a stable CID, not a mutable document.) · **lens** (Drives evidence retrieval and reasoning over the case web; the judge queries in plain language and gets back claims that each carry source CIDs and ancestry, with no fabricated relations allowed past the provenance gate.) · **monitor** (Knowledge-graph explorer to visually walk an exhibit's provenance DAG (derived-from / consumes edges) and spot gaps or unsupported links before relying on

them.) · **vbank** (Lightweight read-only use: confirms which clerk/role identities are authorized to file, and reviews the immutable audit trail of who submitted or sealed what.)

Avoids: pulse (Never sees the engine; CIDs, ledger and PLS-wei settlement are substrate the Lens surface abstracts away — too low-level for a judge.) · bt (Courtroom and chambers are connected; no off-grid Bluetooth-mesh transport need.) · gither (Multi-repo build planning is a developer concern, irrelevant to adjudication.) · virtualpc (Multi-agent orchestration runtime is below the abstraction a judge works at; the Lens app already mediates any agent use.) · molgang (Chemistry domain-knitweb template; unrelated to judicial work.)

Level: Medium · **Entry:** no-code web app (Lens-backed case-record explorer + provenance viewer in monitor)

Wants: - **lens** — A 'cite-or-refuse' answer mode: every sentence the assistant produces must attach the exact source CID(s) and refuse (rather than paraphrase) when a claim has no attested provenance. — *A Medium-literacy judge needs hard guardrails against hallucinated authority; the value is trust, not fluency, so unsupported claims must be impossible, not merely flagged.* - **monitor** — Plain-language provenance timeline that renders an exhibit's ancestry as a readable 'who/what/when' narrative with one-tap drill-down to the underlying signed record. — *Judges reason in narrative and chronology, not graph theory; the no-code surface must translate the DAG into something defensible in a written opinion.* - **knitweb** — Tamper-evident 'evidence seal' receipt: a human-readable certificate that a record's bytes and signature are unchanged since admission, exportable as a PDF for the file. — *Rulings must survive appeal; a Medium-literacy user needs a portable, plain artifact proving integrity without reading CBOR or hashes.* - **vbank** — Role-scoped authorization checks surfaced in the app (e.g. 'this filing was sealed by an authorized clerk') with an immutable submission audit log. — *The integrity of the record depends on knowing only authorized parties wrote to it; this must be visible without the judge touching governance internals.* - **lens** — Deterministic, reproducible evidence bundles: the same query over the same case-state CID always returns the identical citation set, with the bundle itself addressable. — *Appellate review requires that a higher court can re-run the judge's exact evidence assembly and get byte-identical results — determinism is a due-process guarantee.*

Police officer

Captures field evidence and checks people/items on the spot, producing chain-of-custody records that can't be quietly altered later.

Intake: the officer photographs/scans/logs an item or stop in the field, and the app signs it to the shared memory the moment it's captured (queued if offline). Plan/Do: a Lens assistant answers simple lookups ('is this item flagged?', 'prior related records?') in plain language, pulling only from records the officer's unit is authorized to see. Verify/Ship: the app confirms the capture is sealed and time-stamped, and once back in coverage it syncs the signed record so the custody chain is intact end to end.

Uses: knitweb (Each captured photo, scan, or note becomes a signed, content-addressed custody record; the officer never edits history, only appends, so tampering is detectable.) · **lens** (Plain-language field lookups and 'what should I do next' prompts, scoped to the officer's authorized data, with results that trace back to source records.) · **bt** (Off-grid Bluetooth-mesh transport so captures still record and sync peer-to-peer in dead zones (basements, rural, disaster scenes) until a relay is reached.)

Avoids: pulse (Never sees the engine; signing, CIDs and sync happen invisibly under the mobile app — far too low-level for a Low-literacy field user.) · vbank (Governance/voting/treasury is an administrative back-office concern, not a patrol task.) · monitor (Graph dashboards are for analysts and supervisors; the officer needs a single simple capture-and-lookup screen.) · gither (Developer multi-repo tooling; no relevance to fieldwork.) · virtualpc (Agent-orchestration runtime is hidden behind the app; the officer never coordinates agents directly.) · molgang (Chemistry-game domain template; irrelevant.)

Level: Low · **Entry:** consumer mobile app (field app with offline capture)

Wants: - **knitweb** — One-tap signed capture: photo/scan/note is hashed, signed, and time-stamped automatically on shutter with no fields to configure. — *A Low-literacy user in a stressful moment can't manage keys or metadata; chain-of-custody integrity must be a zero-thought default.* - **bt** — Transparent offline mesh queue with a clear 'captured & sealed, will sync' indicator and automatic peer relay when another officer/device is in range. — *Dead-zone capture is the core field reality; the guardrail is reassurance that evidence is already safe before it ever reaches a tower.* - **lens** — Big-button, voice-friendly plain-language lookups ('Is this plate flagged?') that return a simple yes/no/uncertain with

the source record, never a guess. — *Low literacy demands minimal text input and a refusal-over-guess design so an officer never acts on a fabricated answer.* - **lens** — Authorization-scoped results baked in: the app only ever surfaces records the officer's role and jurisdiction permit, enforced server-side. — *Field users can't be trusted to manage data-access rules manually; privacy and legality must be enforced as guardrails, not policy reminders.* - **knitweb** — Auto-redaction templates for sensitive fields (faces, minors, ID numbers) applied at capture, keeping the original sealed but sharing a redacted derivative. — *Plain-language, template-driven redaction lets a non-technical officer stay compliant without understanding the underlying derived-from provenance link.*

Diplomat

Negotiates and reports across borders on a shared, verifiable record, while controlling exactly what each counterpart can see.

Intake: cables, agreements, and counterpart positions land as signed records on the shared memory, scoped to the right clearance and bilateral channel. Plan/Do: a Lens assistant drafts briefings and tracks commitments, citing the signed source for every claimed position so the diplomat negotiates from verifiable, not remembered, history. Verify/Ship: agreed text is sealed as a jointly-signed record and selectively disclosed to the counterpart's node, giving both sides a tamper-evident common reference without exposing internal deliberations.

Uses: knitweb (Treaty drafts, position papers and joint statements live as signed, content-addressed records; a 'final agreed text' is a fixed CID both parties can independently verify.) · **lens** (Drafts briefings, summarizes counterpart history, and tracks who-committed-to-what, with each assertion tied to a source record for accountability.) · **vbank** (Governance for multi-party / coalition decisions: recording votes, mandates, and authorizations among delegations in an auditable, tamper-evident way.) · **monitor** (Dashboard view of commitment status and relationship history across counterparts and dossiers.)

Avoids: pulse (Never touches the engine; selective disclosure and signing are handled by the Lens/app surface, not raw ledger and CID code.) · bt (Embassies and missions have connectivity; no off-grid mesh requirement.) · gither (Developer multi-repo navigation is irrelevant to diplomatic work.) · virtualpc (Agent-runtime orchestration sits below the diplomat's surface; the app mediates any agent use.) · molgang (Chemistry domain template; unrelated.)

Level: Medium · **Entry:** no-code web app (briefing + shared-record workspace with selective disclosure)

Wants: - **knitweb** — Selective-disclosure sharing: share a verifiable derivative or specific record with a named counterpart node while keeping internal antecedents private, with proof the shared item is unaltered. — *Diplomacy is about controlled transparency; a Medium-literacy user needs to share a verifiable 'agreed text' without leaking the deliberation chain behind it.* - **vbank** — Multi-party authorization and mandate records (who is empowered to commit to what, with quorum/sign-off captured immutably). — *Cross-delegation agreements need provable authority; this must be a governance primitive, not a trust-me email, yet usable without touching governance internals.* - **lens** — Citation-backed briefing drafts where every stated counterpart 'position' links to the exact signed record and date it came from. — *Negotiating from verifiable history (not staff memory) is the differentiator; the guardrail is that the assistant can't assert a position no record supports.* - **knitweb** — Joint co-signed sealing: two delegations' signatures bound to one agreed-text CID, producing a single artifact both can independently verify forever. — *A neutral, tamper-evident common reference removes 'whose copy is authoritative' disputes — high value, delivered without exposing engine mechanics.* - **monitor** — Commitment-tracking dashboard that flags drift between a counterpart's signed commitments and their later signed statements. — *A Medium-literacy negotiator benefits from an at-a-glance accountability view rather than manually diffing records.*

Tax inspector

Audits financial and supply-chain records by reconstructing full, signed provenance chains and re-running deterministic checks at scale.

Intake: filer submissions, invoices, and supply-chain events arrive as signed records, often via the finance/supply-chain domain knitwebs. Plan/Do: the inspector scripts queries through the SDK to walk each transaction's full ancestry to its raw origins and runs deterministic re-checks (totals, VAT chains, related-party links) against the web-state. Verify/Ship: mismatches are re-executed to confirm they're real, then findings are sealed as signed, reproducible audit records that a tribunal or the filer can independently replay.

Uses: knitweb (All evidence is content-addressed and signed; the inspector references exact CIDs and integer (wei-style) balances, so figures agree byte-for-byte and can't be silently restated.) · **lens** (Programmatic retrieve+distill over the audit web with subscription-scoped access, returning candidate sets whose every relation is provenance-gated.) · **monitor** (Knowledge-graph explorer to visualize related-party structures and trace fund/goods flow across the finance and supply-chain knitwebs.) · **vbank** (Reviews treasury/disbursement governance records when auditing public funds, including the immutable approval trail.)

Avoids: bt (Office-based, fully connected work; no off-grid mesh need.) · gither (Build/dev-repo planning is outside an auditor's remit even at High literacy.) · virtualpc (Doesn't run a multi-agent orchestration cluster; the SDK and Lens cover the inspector's automation needs.) · molgang (Chemistry-game template; not relevant to financial audit (though the supply-chain knitweb is).)

Level: High · **Entry:** Python SDK + REST API (with a spreadsheet-like UI for casework)

Wants: - **lens** — Deterministic, reproducible audit bundles: a documented SDK call that, for a fixed (query, subscription, web_state_cid), always returns the identical citation set and an addressable bundle CID. — *A High-literacy auditor needs results a tribunal can replay exactly; reproducibility and a stable bundle digest are the evidentiary backbone.* - **knitweb** — Full-depth provenance/ancestry API exposing every raw-origin leaf and processing edge (derived-from / consumes / settles) for any record, as a typed DAG. — *Audit means following the money/goods all the way to roots; the inspector requires depth and typed edges, not a hop-capped traversal.* - **lens** — Provenance-gated query results that hard-drop any relation lacking an attested originator + asset_cid, with a machine-readable rejection log. — *High-literacy users want to trust the substrate refuses unattested data automatically and to audit what was excluded and why.* - **knitweb** — Anchor/attestation export tying audited records to external provenance (e.g. OriginTrail DKG) with verifiable origin and timestamp. — *Cross-border audits need a neutral external anchor proving a record existed and originated when claimed — depth of provenance is the differentiator.* - **monitor** — Related-party / circular-flow detection over the finance+supply-chain graph, exportable to the spreadsheet-like casework UI. — *Scaling audits requires automated structural anomaly surfacing; a High-literacy user wants it as both an API result and a reviewable graph.* - **vbank** — Immutable disbursement-approval trail query (who authorized which public payment, under what quorum). — *Auditing public funds demands a provable governance chain alongside the financial records, queryable programmatically.*

Military logistics officer

Coordinates supply, transport, and resource allocation across distributed units, keeping a verifiable common operating picture even when links are intermittent.

Intake: requisitions, stock levels, and movement events land as signed records on the shared memory, replicated peer-to-peer across units. Plan/Do: the officer scripts allocation and routing against the supply-chain knitweb, with agents (coordinated via the runtime) proposing plans that resolve against current, content-addressed inventory state. Verify/Ship: plans are re-checked for feasibility against signed stock records, then issued as signed tasking records that sync over whatever transport is available, including off-grid mesh, so forward units act on a verified picture.

Uses: knitweb (Inventory, requisitions, and movements are signed, content-addressed records with integer balances; every unit converges on the same byte-for-byte state, giving a tamper-evident common operating picture.) · **lens** (Reasons over the logistics web to generate and explain allocation/routing plans, each tied to the source stock and demand records it relied on.) · **virtualpc** (Multi-agent coordination runtime to run planner/validator agents that propose and stress-test logistics plans across distributed units.) · **bt** (Off-grid Bluetooth-mesh transport keeps forward and disconnected units in sync over long delays when satellite/radio links are degraded.) · **monitor** (Ops dashboard giving the field team a no-code view of stock, in-transit movements, and plan status.)

Avoids: pulse (Even at High literacy, operates through the SDK/Lens/runtime; doesn't hand-edit ledger, CBOR, or CID internals of the engine itself.) · vbank (Treasury/voting governance is not a logistics function; tasking authority comes from command structure, not on-fabric treasury votes.) · gither (Developer multi-repo build planning is unrelated to operational logistics.) · molgang (Chemistry-game domain template; not the supply-chain domain this role uses.)

Level: High · **Entry:** REST API + CLI (with a no-code ops dashboard for the field team)

Wants: - **knitweb** — Conflict-quarantine + deterministic merge for delayed/partitioned syncs so two units that updated the same stock offline reconcile to one agreed integer state on reconnection. — *Intermittent links are the operational*

norm; a High-literacy officer needs provable, deterministic reconciliation rather than last-writer-wins guesswork. - **bt** — Store-and-forward mesh relay with long-delay tolerance and signed-feed replication across hops, prioritizing critical tasking records. — *Forward units operate disconnected; verifiable eventual sync over off-grid transport is the core differentiator for the field picture.* - **virtualpc** — Sandboxed planner/validator agent orchestration whose proposals must resolve against signed inventory state, with each plan emitting its source-record citations. — *High-literacy automation at scale needs agents, but every machine-proposed allocation must be auditable and grounded in verified stock, not hallucinated availability.* - **lens** — Deterministic feasibility re-check: re-run a routing/allocation plan against a fixed web-state CID and get byte-identical pass/fail, with a clear infeasibility reason. — *Command must be able to independently replay and trust a plan's feasibility check; determinism turns 'the system said so' into something verifiable.* - **monitor** — Real-time no-code logistics dashboard (stock, in-transit, ETA, shortfalls) that degrades gracefully to last-synced state with a clear staleness indicator on intermittent links. — *The field team is mixed-literacy; they need a guardrailed read-only picture that never silently shows stale data as current.* - **knitweb** — Signed tasking records with role-scoped issuance and an immutable order-of-events log per unit. — *Accountability and after-action review require that every order is attributable, ordered, and tamper-evident across the distributed force.*

Extended Education

Special-education teacher

Builds and adapts individualized learning plans for students with diverse needs, with agents handling the paperwork and pacing.

Intake: the teacher dictates or taps a student goal into the mobile app and a Lens agent drafts an IEP-style plan from templates. Do/verify: the teacher reviews each step in plain language, approves or edits, and the agent records adaptations to the shared content-addressed memory so the next session resumes from the exact same state. Ship: a parent/aide summary is generated; nothing is published without an explicit teacher tap.

Uses: knitweb (Per-student plans, accommodations, and progress notes are written to the shared content-addressed memory so any aide, therapist, or substitute resumes from the same canonical record.) · **lens** (Drafts IEP goals, suggests next activities, and translates jargon into plain language; runs read-only over the memory so it never silently edits a child's record.) · **bt** (Off-grid Bluetooth mesh lets the app sync plans device-to-device in low-connectivity classrooms, special-needs field trips, or rural sites without wifi.)

Avoids: pulse (Never sees the engine; ledger/PLS/CID internals are far too low-level for a classroom user.) · virtualpc (Multi-agent coordination runtime is invisible plumbing; the teacher only meets one assistant in the app.) · gither (Multi-repo build planner is a developer tool with no classroom use.) · vbank (No voting/treasury governance role in a single classroom.) · molgang (Chemistry-game domain template is unrelated to special-ed planning.)

Level: Low · **Entry:** consumer mobile app (with offline mode)

Wants: - **lens** — Plain-language IEP template library with one-tap accommodation suggestions and a 'why' explanation for each — *Low-literacy user needs guardrailed templates and explanations, not a blank reasoning prompt.* - **knitweb** — Per-student access guardrails (consent scopes) so a record is only readable by named aides/therapists — *Child privacy is non-negotiable; low-literacy users need this enforced by default, not configured.* - **lens** — Voice/dictation intake and read-aloud review of the generated plan — *Reduces friction for a busy teacher and improves accessibility for the educator themselves.* - **bt** — Conflict-free offline sync with a simple 'merged / needs your review' badge — *Field-trip and low-connectivity reality; merges must be surfaced in plain terms, never as a git-style conflict.* - **monitor** — Single-student progress timeline with milestone celebration view — *Visual, mobile-friendly progress tracking the teacher can show parents without reading a graph database.*

Private tutor

Runs 1:1 sessions, lets an agent generate practice and track each learner's mastery across sessions.

Intake: tutor picks a subject/learner and the Lens agent pulls that learner's prior mastery from the shared memory and proposes today's plan. Do/verify: tutor teaches, marks what the student got, and the agent updates mastery and

generates the next problem set; the tutor approves before sending homework. Ship: a short progress note and next-session plan are committed to memory and shared with the parent on tap.

Uses: knitweb (Each learner's mastery map and session notes persist in shared memory so progress carries across sessions and tutors.) · **lens** (Generates leveled practice problems, hints, and a recap; reads the learner's history to avoid repeating mastered material.) · **monitor** (Simple mastery dashboard the tutor shows the parent to justify the next package of sessions.)

Avoids: pulse (Engine internals irrelevant; the tutor pays/gets paid through an app surface, not the raw ledger.) · bt (Tutoring is online or in-home with normal connectivity; no off-grid mesh need.) · gither (Developer multi-repo tool, no relevance.) · virtualpc (Agent-coordination runtime is hidden plumbing.) · vbank (No governance/treasury role for an individual tutor.)

Level: Low · **Entry:** consumer mobile app

Wants: - **lens** — Curriculum-aligned problem generator with difficulty auto-tuned to the learner's mastery, plus an 'answer key with worked steps' — *Low-literacy tutor needs trustworthy, ready-to-use practice without prompt engineering.* - **knitweb** — Portable learner profile the parent owns and can move between tutors — *Builds trust and continuity; the family controls the record, not a single tutor.* - **monitor** — Parent-facing one-page progress report exported as a friendly PDF — *Tutors need a simple artifact to demonstrate value and renew sessions.* - **lens** — Guardrail that flags when generated content is above/below the learner's level before it's sent — *Prevents over- or under-challenging a child; a safety rail for non-technical users.*

School principal

Oversees whole-school outcomes, staffing, and policy, using agents to roll up data and surface where to intervene.

Intake: the principal sets a question (e.g. 'which grades are falling behind in reading?') in the dashboard and a Lens agent aggregates from teachers' shared-memory records. Plan/do: agent proposes interventions and resource shifts; the principal reviews provenance (which classrooms/records fed the conclusion) before acting. Ship: decisions and policy changes are recorded to shared memory and, where they need staff buy-in, sent to a lightweight vote.

Uses: knitweb (Reads the school-wide content-addressed memory of student progress and program notes as the single source of truth for roll-ups.) · **lens** (Aggregates and explains trends with provenance (which records support each claim) and drafts intervention plans.) · **monitor** (School-wide dashboards and a knowledge-graph explorer to see how programs, cohorts, and outcomes connect.) · **vbank** (Runs lightweight staff votes on policy changes and tracks small discretionary budget allocations.)

Avoids: pulse (Never touches the engine; consumes outcomes through dashboards, not CIDs or the ledger directly.) · bt (School has connectivity; no off-grid mesh need.) · gither (Developer build-planning tool, not an administrator surface.) · virtualpc (Agent-coordination runtime stays hidden behind the dashboard.) · molgang (A single subject-domain template is not a whole-school admin tool.)

Level: Medium · **Entry:** no-code web dashboard with spreadsheet-like configuration

Wants: - **monitor** — Configurable cohort/intervention dashboard with drill-down from school -> grade -> classroom (no code) — *Medium-literacy admin needs self-serve views without querying a graph by hand.* - **lens** — Provenance panel on every aggregate metric showing the underlying records and any teacher overrides — *A principal must defend data to boards/parents; medium literacy wants traceability surfaced, not raw CIDs.* - **vbank** — Quorum-gated staff vote widget with plain-language proposals and an audit trail — *Policy changes need legitimate, recorded buy-in without a governance learning curve.* - **knitweb** — Role-scoped read access so the principal sees aggregates without exposing individual confidential SEN records inappropriately — *Balances oversight against student privacy obligations.* - **monitor** — Early-warning alerts (e.g. attendance/reading decline) pushed to the dashboard — *Turns passive data into actionable intervention timing.*

Curriculum developer

Designs reusable learning units and standards mappings that agents can instantiate and tutors/teachers can adopt.

Intake: developer drafts a unit and standards map in the authoring UI; a Lens agent checks coverage and suggests gaps. Plan/do: the agent generates aligned activities and assessments, the developer curates them, and the validated unit is committed to shared memory with a stable content address so every downstream classroom references the exact same version. Ship: the unit is published as a reusable template (a domain-knitweb), versioned and forkable.

Uses: knitweb (Publishes versioned, content-addressed curriculum units to shared memory so adoptions are pinned to an exact, citable version.) · **lens** (Generates aligned activities/assessments, checks standards coverage, and flags reading-level mismatches.) · **molgang** (Uses the domain-knitweb template pattern as the model for packaging a subject curriculum as a reusable, forkable knitweb.) · **monitor** (Reviews how units connect to standards and how adopted units perform across classrooms via the graph explorer.)

Avoids: pulse (Relies on stable content addressing but works at the curriculum layer, not the ledger/CID engine itself.) · bt (Authoring happens online; no off-grid transport need.) · gither (Multi-repo build planning is a developer-infra concern, not curriculum authoring.) · virtualpc (Agent orchestration runtime is hidden behind the authoring tools.) · vbank (No treasury/voting role in unit authoring.)

Level: Medium · **Entry:** spreadsheet-like authoring UI plus optional REST API

Wants: - **knitweb** — Curriculum versioning with stable content addresses, fork/merge, and a 'pin to version' adoption model — *Medium-literacy author needs reproducible, citable units so classrooms aren't broken by silent edits.* - **lens** — Standards-coverage checker that maps generated items to a chosen framework and reports gaps — *Alignment is the core deliverable; the agent should verify it, not just generate.* - **molgang** — Reusable domain-knitweb scaffold (template repo) for packaging a subject curriculum with activities, rubrics, and metadata — *Lets authors ship forkable units following a proven domain-template pattern.* - **knitweb** — REST API to publish/query units programmatically — *Medium-to-high authors and LMS integrations need automated pipelines beyond the UI.* - **monitor** — Adoption + outcome analytics per unit version — *Closes the loop so authors iterate on what actually works in classrooms.*

Career counselor

Guides learners toward pathways and credentials, with an agent matching interests and verified achievements to options.

Intake: counselor and student answer a short interest/skills questionnaire in the app and a Lens agent reads the student's verified achievements from shared memory. Plan/verify: the agent proposes pathways (courses, credentials, jobs) with the evidence behind each match, and the counselor reviews and personalizes before sharing. Ship: an action plan is saved to the student's portable record and revisited at the next meeting.

Uses: knitweb (Reads the student's portable, verifiable achievement record from shared memory and saves the agreed pathway plan back to it.) · **lens** (Matches interests and verified skills to pathways and explains each recommendation in plain language.) · **monitor** (Simple pathway-explorer view showing how credentials connect to careers.)

Avoids: pulse (Never sees the engine; credentials are consumed as friendly verified badges, not raw ledger entries.) · bt (Counseling sessions have connectivity; no off-grid need.) · gither (Developer tooling, irrelevant to counseling.) · virtualpc (Coordination runtime stays hidden behind the app.) · vbank (No governance/treasury function in counseling.)

Level: Low · **Entry:** no-code web app / consumer mobile app

Wants: - **lens** — Interest-to-pathway matcher with a plain-language 'why this fits you' rationale and links to real programs — *Low-literacy user needs explained, actionable suggestions rather than an opaque ranking.* - **knitweb** — Student-owned verifiable credential wallet that imports achievements once and reuses them across counselors — *Portability and trust; the learner controls their record across schools and advisors.* - **monitor** — Visual pathway map (credential -> course -> career) with skill-gap highlights — *Mobile-friendly visualization makes options concrete for the student.* - **lens** — Guardrail that grounds every recommendation in verified achievements and labels anything speculative — *Prevents over-promising careers; keeps a non-technical counselor honest and safe.*

Extended Creative

Novelist

Writes long-form fiction with an AI co-writer that remembers every character, place, and plot thread across the whole manuscript.

Intake: the novelist drafts a scene or asks for one in plain language, and Lens pulls the relevant character/world facts from the shared content-addressed memory (the 'story bible' lives as immutable knitweb fibers). Plan->do: agents propose continuations, continuity checks, and rewrites against that memory; the writer accepts, edits, or rejects. Verify->ship: a continuity pass flags contradictions before the writer 'publishes' a chapter revision, which saves a new content-addressed version (old drafts never overwritten) that they can hand to an editor or keep private.

Uses: knitweb (The story bible (characters, timelines, world facts, voice notes) lives as the shared content-addressed memory; every chapter revision is an immutable versioned fiber the writer can branch and compare.) · **lens** (Reasoning/co-writer surface: drafts prose, runs continuity checks, answers 'when did X learn Y?' over the bible without the writer seeing any internals.) · **docs** (Plain-language help and templates for setting up a story bible and version history.)

Avoids: pulse (Never sees the engine; CIDs/ledger/PLS are below the friendly app and irrelevant to writing prose.) · bt (No off-grid mesh need; a desk writer is online and works in a normal app.) · gither (Multi-repo build navigation is a developer tool with no creative-writing surface.) · virtualpc (Multi-agent runtime is plumbing; the writer wants one co-writer, not an orchestration console.) · vbank (No voting/treasury governance in a solo manuscript.) · molgang (Chemistry domain template is unrelated to fiction.)

Level: Low · **Entry:** consumer desktop/web writing app (Lens-backed, like a friendly word processor)

Wants: - **lens** — Plain-language continuity guardrail: a one-click 'check this chapter against my bible' that returns contradictions in human terms ('Chapter 3 says Mara is left-handed; here she writes with her right') with no jargon. — *Low-literacy role needs guardrails and plain language, not query syntax.* - **knitweb** — Visual version timeline of immutable revisions with side-by-side diff and one-tap restore, hiding CIDs behind friendly labels ('Draft from Tuesday'). — *Writer needs safe, reversible versioning without understanding content addressing.* - **lens** — Story-bible auto-population: highlight a name/place in the draft and 'add to bible', so the memory grows from writing itself. — *Removes setup friction for a non-technical user.* - **docs** — Genre starter templates (mystery, romance, fantasy) that pre-shape the bible schema. — *Templates let a low-literacy user start in minutes.*

Animator

Builds animated sequences where AI agents handle in-betweening, asset consistency, and render orchestration against a shared asset library.

Intake: the animator defines a shot and references characters/props from the shared content-addressed asset library (model sheets, rigs, palettes stored as fibers). Plan->do: Lens plans a shot breakdown and dispatches render/in-between jobs to spider workers, pulling exact asset versions from memory so every frame uses the same canonical rig. Verify->ship: a review pass checks frame-to-frame consistency and the animator approves a take, which is published as an immutable shot version that downstream comp/edit can reference.

Uses: knitweb (Shared asset library: model sheets, rigs, palettes, approved shots as content-addressed immutable versions so every render references the exact same asset.) · **lens** (Shot-breakdown planning, in-between/consistency reasoning, and natural-language direction over the asset memory.) · **monitor** (Render/job dashboard and asset-graph explorer to see which shots use which rig version and job status.) · **docs** (Guides for wiring render hooks and structuring an asset library.)

Avoids: pulse (The animator pays for render compute through the app surface; raw ledger/CID/PLS internals stay hidden below it.) · bt (Studio work is online; no off-grid Bluetooth mesh need.) · gither (Source-repo build planning isn't part of an animation pipeline.) · vbank (No governance voting in a single production.) · molgang (Chemistry template is unrelated.)

Level: Medium · **Entry:** no-code/low-code web app with a node/timeline canvas and optional REST hooks

Wants: - **monitor** — Asset-version graph view: 'which shots depend on rig v4' with one-click impact when a rig is updated. — *Medium-literacy role benefits from a visual dependency/provenance explorer over the asset memory.* - **lens** — Consistency-lock on dispatch: render jobs pin the exact content-addressed asset version so re-renders are deterministic. — *Reproducible renders require determinism, which a medium-literacy user can request without touching CIDs directly.* - **knitweb** — Approved-take publishing with immutable shot versions and an 'approved' tag downstream tools can resolve. — *Pipeline handoff needs stable, referenceable versions.* - **monitor** — Render-job queue dashboard (status, cost in friendly units, retries) over spider workers. — *Medium users want visibility and cost control, not a CLI.* - **lens** — Optional REST/webhook so an existing DCC tool (Blender/Maya) can trigger shot plans. — *Medium-literacy power users want light API integration with current tools.*

Interior designer

Designs rooms by photographing a space and placing AI-suggested furniture/materials pulled from a shared, provenance-tracked product catalog.

Intake: the designer scans a room and sets a brief (budget, style) in plain language; Lens reads the shared catalog memory for in-stock, on-budget products. Plan->do: agents propose layouts and material palettes, the designer drags items in/out, and selections snap to real catalog SKUs with sourcing/provenance carried from memory. Verify->ship: a budget/availability check confirms everything is orderable before the designer 'ships' a client-ready board, saved as a versioned proposal the client can review.

Uses: knitweb (Shared product/material catalog and saved room proposals as content-addressed versions; product provenance (origin, price, stock) lives in shared memory.) · **lens** (Plain-language room planning, style/palette suggestions, and budget reasoning over the catalog.) · **docs** (Simple how-to and style templates.)

Avoids: pulse (Never touches the engine; settlement/CID internals are invisible behind the app.) · bt (On-site but online via phone; no mesh transport need.) · gither (Developer multi-repo tool, irrelevant.) · virtualpc (No need to orchestrate agent fleets; one design assistant suffices.) · monitor (Knowledge-graph dashboards are too technical for this consumer surface.) · molgang (Chemistry template unrelated.)

Level: Low · **Entry:** consumer tablet/mobile app (camera + drag-and-drop room canvas)

Wants: - **lens** — Budget guardrail: a live 'you are 200 over budget' bar that blocks or warns before adding an item. — *Low-literacy role needs a hard, plain guardrail rather than a query.* - **knitweb** — Provenance-backed catalog: every placed item carries origin/stock/price from shared memory so the board is automatically orderable. — *Trustworthy sourcing without the designer understanding provenance plumbing.* - **lens** — Photo-to-layout: scan a room and get drag-and-drop suggestions in style presets. — *Mobile, template-driven entry suits a low-literacy user.* - **knitweb** — Versioned client proposals with a friendly 'Option A / Option B' compare. — *Safe, reversible proposal history without exposing CIDs.*

Fashion designer

Designs garments and generates production-ready tech packs, sourcing materials from a provenance-tracked supply catalog.

Intake: the designer sketches or describes a piece and pulls fabrics/trims from the shared supply catalog (content-addressed, provenance-bearing). Plan->do: Lens proposes silhouettes, builds graded tech packs, and reconciles each material against real supplier records in memory; the designer iterates. Verify->ship: a manufacturability/cost check validates the tech pack before publishing an immutable version a factory or molgang-style supply knitweb can consume.

Uses: knitweb (Shared supply catalog (fabrics, trims, suppliers) and immutable versioned tech packs; material provenance/origin tracked in shared memory.) · **lens** (Silhouette/grading suggestions, tech-pack generation, and cost/sourcing reasoning over the catalog.) · **monitor** (Supply-graph explorer to trace a fabric's origin/availability and see which designs depend on it.) · **molgang** (As a domain-knitweb template, the supply-chain knitweb pattern is reused for textile sourcing/provenance (not chemistry content).) · **docs** (Tech-pack structure and CSV import guides.)

Avoids: pulse (Sourcing/settlement happens through the app; ledger/CID internals stay hidden.) · bt (Studio/office is online; no off-grid mesh need.) · gither (Developer build tool, irrelevant to garment design.) · virtualpc (No agent-fleet

orchestration console needed.) · vbank (No governance voting in a design workflow.)

Level: Medium · **Entry:** no-code web app with a tech-pack canvas plus optional spreadsheet/CSV import

Wants: - **monitor** — Material provenance trace: tap a fabric to see origin, supplier, certifications, and which tech packs use it. — *Medium-literacy designers want provenance depth and a supply-graph view.* - **knitweb** — Immutable tech-pack versioning with a factory-resolvable 'released' tag. — *Production handoff needs stable, referenceable versions.* - **lens** — CSV/spreadsheet import to map an existing bill-of-materials into the catalog. — *Medium users bridge current tooling without a full API.* - **molgang** — Reuse the supply-chain knitweb template for textiles (supplier nodes, lot provenance) as a starter pack. — *Leverages an existing domain template for a medium-literacy user instead of building from scratch.* - **lens** — Cost/manufacturability check before release with plain warnings. — *Guardrail validation suited to a medium-literacy producer.*

Game designer

Builds game systems, content, and live-balance loops where agents reason over a deterministic, versioned design/knowledge graph.

Intake: the designer specs systems and content as data in the shared content-addressed graph (entities, rules, economy curves as fibers). Plan->do: agents via the Lens API generate/validate content, run balance simulations, and propose changes against deterministic, provenance-tracked memory; the designer scripts pipelines through the SDK. Verify->ship: deterministic re-runs and provenance checks confirm a balance change is reproducible before publishing an immutable content version that the live game and teammates' no-code editor resolve.

Uses: knitweb (Design/knowledge graph: entities, rules, economy curves, and content as content-addressed immutable versions with full provenance and branchable history.) · **lens** (Reasoning/orchestration over the graph for content generation, balance simulation, and validation; accessed via API for scripted pipelines.) · **molgang** (Studied/forked as the canonical domain-knitweb template (a knowledge game) when scaffolding a new game-domain knitweb.) · **monitor** (Knowledge-graph explorer and balance/telemetry dashboards over the design graph.) · **virtualpc** (Multi-agent coordination runtime to fan out content-generation and simulation agents in parallel.) · **docs** (SDK/API reference and determinism/provenance guarantees.)

Avoids: pulse (Even a high-literacy designer works through the SDK/Lens API, not raw ledger/CID/canonical-encoding internals; the engine is the substrate, not a design surface.) · bt (No off-grid mesh requirement for a studio pipeline.) · gither (Multi-repo build planning is engine-team tooling, not game-content design.) · vbank (No on-chain governance voting/treasury in the design loop.)

Level: High · **Entry:** Python SDK + CLI, with a Lens API and a no-code editor for designers on the team

Wants: - **lens** — Stable, versioned reasoning API with deterministic outputs and provenance on every generated asset. — *High-literacy role needs APIs, determinism, and provenance depth for reproducible pipelines.* - **knitweb** — Branch/merge on the content graph with content-addressed diffs for A/B balance experiments. — *Power users want git-like branching over deterministic memory.* - **virtualpc** — Parallel agent-fleet orchestration with per-run isolation for large content/sim batches. — *High-literacy designers script multi-agent runs and need a real coordination runtime.* - **molgang** — Documented 'fork this domain-knitweb template' scaffold to bootstrap a new game domain. — *Lets an expert reuse the canonical L5 knitweb pattern instead of reinventing it.* - **monitor** — Provenance-deep balance dashboard linking every metric back to the exact content version that produced it. — *High-literacy role demands traceability from telemetry to canonical CID.* - **docs** — Determinism + provenance contract reference for the SDK. — *Expert users need exact guarantees to build reproducible tooling.*

Extended Science

Physicist

Run reproducible simulations and theory-vs-data comparisons whose every derived relation is content-addressed and independently re-executable.

Intake a hypothesis as a query against the shared content-addressed memory (Web) and pull a deterministic CandidateSet of prior results via lens.retrieve scoped by subscription; plan and run the simulation/derivation as a bounded distill loop, with agents proposing parameter sweeps and a PoUW job re-executing the deterministic pieces. Verify by provenance-gate (fabricated or unattested relations are dropped) and a deterministic digest independent of insertion order, then ship the verified answer as a distill_bundle woven back into the Web with distilled-from ancestry edges.

Uses: lens (Primary surface: deterministic retrieve + recursive distill controller + distill_bundle SDK to derive and re-verify results over the fabric) · **knitweb** (Shared content-addressed memory: queries, weaves verified relations, and reads provenance ancestry as the canonical record of prior results) · **pulse** (Touched as the engine for canonical CIDs, attestation, and PoUW sampled re-execution that makes a simulation result reproducible byte-for-byte; uses CLI for node/provenance walks) · **monitor** (Knowledge-graph explorer to inspect the derivation DAG and provenance anchors of a result set)

Avoids: bt (Off-grid Bluetooth DEX/transport is irrelevant to lab compute on networked clusters) · vbank (Governance voting/treasury is not part of an individual derivation workflow) · molgang (Chemistry-game domain template; physics has its own domain, not the molgang knitweb) · gither (Repo-ownership forge is for software shipping, not running physics derivations)

Level: High · **Entry:** Python SDK (lens distill SDK) + CLI for node/provenance, notebooks on top

Wants: - **lens** — Numeric-precision-aware distill: declare float tolerances and units so deterministic digests survive floating-point derivations without breaking reproducibility — *High-literacy physics is float-heavy; the engine's no-float canonical rule must be bridged by an explicit tolerance/units contract at the reasoning layer* - **pulse** — Deterministic seed + environment manifest pinned into the PoUW job manifest (RNG seed, library versions, hardware class) — *Reproducibility of a simulation requires the re-executor to match the exact stochastic and numeric environment, not just the query* - **knitweb** — Provenance depth query returning the full citation ancestry of a derived relation as a traversable subgraph — *Physicists must audit which upstream measurements/constants a result depends on; provenance depth is the trust currency at this literacy level* - **lens** — Re-run-on-new-evidence subscription hook that re-executes a distill_bundle when a subscribed upstream relation changes CID — *Theory-vs-data must auto-invalidate when an input dataset is revised, keeping conclusions live rather than stale*

Astronomer

Cross-match catalogs and time-domain alerts into a shared, provenance-anchored sky memory that other observatories can trust and re-derive.

Intake an object/region query against the shared content-addressed memory and retrieve a deterministic candidate set of prior detections scoped to subscribed surveys; plan the cross-match/photometry pipeline, letting agents fan out per-catalog while a PoUW class re-executes the deterministic reduction. Verify each emitted association passes the provenance-gate (only attested, OriginTrail-anchored source observations survive) and ship the confirmed relations woven into the Web with an anchored UAL so peer observatories cite the same CID.

Uses: knitweb (Shared sky memory: weaves cross-match relations (object -observed_in-> survey) as signed, content-addressed records other observatories read) · **lens** (Deterministic retrieve + distill to assemble candidate associations and reduce time-series, with bounded recursive reasoning over the fabric) · **pulse** (Engine for CIDs, OriginTrail anchoring of source frames, and PoUW re-execution of the reduction; CLI for bulk survey ingest) · **monitor** (Force-directed knowledge-graph view of the cross-match web with anchor markers to spot mis-associations visually)

Avoids: bt (No off-grid mesh need; observatory pipelines run on networked compute) · vbank (Treasury/voting governance is outside catalog reduction) · gither (Code-ownership forge unrelated to running survey science) · virtualpc (Self-hosted agent OS is heavier than needed; lens already provides the reasoning orchestration)

Level: High · **Entry:** Python SDK + notebooks; CLI for large survey ingest and provenance anchoring

Wants: - **pulse** — Spatial index over the fabric Web keyed on sky coordinates (HEALPix/RA-Dec) for deterministic cone-search retrieval — *Astronomy queries are inherently positional; the interpretation lobe's SpatialIndex must speak sky geometry to make cross-match retrieval real* - **knitweb** — High-volume frame anchoring with batched OriginTrail UALs so terabyte survey artifacts are provenance-linked without per-frame overhead — *Surveys ingest millions of*

frames; provenance must scale or the anchoring becomes the bottleneck - **lens** — Time-domain re-distill trigger that fires when a transient alert (new CID) lands in a subscribed scope — *Time-domain astronomy is latency-sensitive; the reasoning layer must react to new evidence automatically to be useful for alerts* - **monitor** — Provenance-anchored sky overlay that renders the cross-match graph against catalog positions with mismatch highlighting — *Visual verification of associations catches systematic mis-matches faster than tabular audit at this data scale*

Geologist

Build a shared, citable record of samples, sites, and assays where field observations are attested and peers validate interpretations.

Intake field/lab data through a spreadsheet-like form that writes attested records into the shared content-addressed memory; plan an interpretation (e.g. stratigraphic correlation) with agent assistance that retrieves prior site relations from the Web. Peers validate the proposed correlation via a quorum vote (molgang-style propose-then-confirm) before it is woven as a confirmed relation, and ship it anchored so the assay-to-conclusion chain is auditable.

Uses: knitweb (Shared memory of samples/sites/assays as signed, attributable records with provenance from lab to interpretation) · **molgang** (Used as a domain-knitweb template: the propose -> peers vote -> confirm-quorum -> woven-Fiber loop is reused for validating geological interpretations) · **lens** (Friendly reasoning helper that retrieves related sites and suggests correlations in plain language without exposing the substrate) · **monitor** (Knowledge-graph explorer to see how a conclusion links back to specific samples and which peers validated it)

Avoids: pulse (Never sees the engine internals (CIDs, CBOR, ledger); too low-level for a medium-literacy field scientist) · bt (No off-grid mesh transport in the workflow (though field-offline is a wishlist, not a touched repo)) · gither (Code forge is irrelevant to geological data curation) · virtualpc (Multi-agent OS is over-scoped; the app + lens cover the agent needs)

Level: Medium · **Entry:** Spreadsheet-like UI + no-code web app over a domain knitweb; light REST API for field-data import

Wants: - **molgang** — Reusable domain-knitweb starter kit (faucet + propose + quorum + leaderboard) re-skinnable from chemistry to a generic field-science domain via config — *Medium-literacy geologists need a templated propose/validate loop, not to build consensus plumbing from scratch* - **lens** — Plain-language correlation suggestions with each suggestion carrying a one-click provenance trail back to source samples — *Guardrails and explainability matter more than API depth at medium literacy; the geologist must trust a suggestion without reading bytecode* - **knitweb** — Spreadsheet/CSV import that auto-attests each row to the field collector with site/date metadata — *Field data arrives as tables; attribution must be automatic so contributions are trustworthy without manual signing steps* - **monitor** — Sample-to-conclusion lineage view in a non-technical layout showing who validated each step — *A medium-literacy user verifies trust through a readable lineage picture, not a raw provenance graph traversal*

Statistician

Produce reproducible analyses whose data, code path, and result are one content-addressed, independently re-runnable bundle.

Intake a dataset and analysis spec, retrieve prior analyses and the exact source data CIDs from the shared content-addressed memory deterministically; plan and execute the model as a bounded distill controller while a PoUW job re-runs the deterministic estimation on sampled peers. Verify the result digest is identical regardless of relation insertion order and that every input is attested, then ship a distill_bundle that anyone can re-execute to byte-identical numbers.

Uses: lens (Core surface: retrieve + recursive distill + distill_bundle to express an analysis as a deterministic, re-runnable pipeline) · **knitweb** (Shared memory of datasets and prior results as content-addressed inputs/outputs with full provenance ancestry) · **pulse** (Engine for canonical CIDs, attestation, and PoUW sampled re-execution that turns a reproducibility claim into a verified one; CLI for audit) · **vbank** (Occasionally, for deterministic tallying primitives when the analysis is over ballots/polls (auditable one-person-one-vote results))

Avoids: bt (Off-grid DEX mesh is unrelated to statistical compute) · molgang (Chemistry-game template is the wrong domain for general statistics) · gither (Repo forge is for shipping software, not running analyses) · virtualpc (Full agent OS exceeds needs; lens orchestration suffices)

Level: High · **Entry:** Python SDK + REST API; notebooks for analysis, CLI for re-execution audits

Wants: - **lens** — Float-tolerance + RNG-seed contract on distill so stochastic estimators (MCMC, bootstrap) yield stable deterministic digests within declared bounds — *Reproducible statistics needs determinism that accommodates seeded randomness and finite precision, exactly the gap between the no-float engine and real analysis* - **pulse** — Re-execution audit report listing which deterministic pieces were sampled, by whom, and the match/mismatch digest — *A statistician's trust in a result is the audit trail of its independent re-runs; high literacy wants the raw verification record, not a summary* - **knitweb** — Dataset-version provenance with diff-able lineage between two source CIDs — *Analyses must pin and compare exact data versions; provenance depth and diffing are first-class needs at this literacy* - **lens** — REST/SDK endpoint to submit an analysis spec and receive a verified distill_bundle CID plus reproduction command — *High-literacy users want programmatic, scriptable access and a one-command path for reviewers to reproduce results*

Bioinformatician

Run provenance-tracked genomics/omics pipelines where every intermediate is a cached content-addressed node peers can re-verify.

Intake sequences/variants and retrieve prior annotations and reference CIDs deterministically from the shared content-addressed memory scoped by subscription; plan the pipeline as a distill DAG where each stage weaves its output as a content-addressed node (identical input slice = cache hit) and agents fan out across samples. A PoUW job re-executes deterministic stages and the provenance-gate drops any annotation lacking attested lineage, then the verified call set ships woven into the Web with distilled-from ancestry for downstream reuse.

Uses: lens (Pipeline-as-distill-DAG: deterministic retrieve, content-addressed intermediates with cache-hits, and provenance-gated outputs) · **knitweb** (Shared memory of references/annotations/call-sets as signed content-addressed nodes with reusable distilled-from ancestry) · **pulse** (Engine for CIDs, attestation, OriginTrail anchoring of reference databases, and PoUW re-execution; CLI for batch jobs) · **monitor** (Knowledge-graph explorer to trace a variant call back through pipeline stages to raw reads and reference version)

Avoids: bt (Off-grid Bluetooth transport irrelevant to cluster/cloud genomics) · vbank (Governance voting is outside a bioinformatics pipeline) · molgang (Chemistry-game domain template, not the genomics domain) · gither (Code-ownership forge is for software releases, not running pipelines)

Level: High · **Entry:** Python SDK + CLI; REST API for pipeline orchestration, notebooks for exploration

Wants: - **lens** — Content-addressed intermediate caching keyed on (tool version, params, input CID) so re-running a pipeline skips unchanged stages — *Genomics pipelines are expensive and re-run often; deterministic cache-hits on intermediates are the difference between usable and not at this scale* - **knitweb** — Reference-database version anchoring so an annotation node records the exact reference build it was called against — *Bioinformatics conclusions are meaningless without the pinned reference version; provenance must capture it as a first-class edge* - **pulse** — PoUW job class for large deterministic alignment/variant stages with bounded sampled re-execution — *Trust in a call set comes from peers re-running the deterministic stages; the engine must register genomics-scale jobs within the compute guardrail* - **lens** — Provenance-gate with a fabricated-annotation rejection test that drops any call lacking attested upstream reads — *Hallucinated or unsourced annotations are a safety risk; high-literacy users demand the gate proves no fabricated relations entered the result*

Operations

Warehouse manager

Keeps stock accurate and goods moving by turning every receive/pick/ship into a signed, content-addressed inventory event.

Intake: scans and counts flow in via the mobile/scanner companion into the warehouse app. Plan/do: a Lens agent reconciles counts, proposes putaway and replenishment, and the operator confirms; each confirmed movement is written as a signed supply-chain process event into the shared content-addressed memory so the on-hand state is a

verifiable CID. Verify/ship: agents reconcile against expected mass/quantity balances, flag discrepancies, and the corrected ledger is woven into the Web for downstream procurement and quality.

Uses: lens (Drives reconciliation, replenishment suggestions, and cycle-count planning over the warehouse's inventory graph; the operator approves agent proposals.) · **knitweb** (On-hand stock, lot/SKU records, and movement history live as content-addressed records in the shared fabric so every count is provable and shared.) · **monitor** (Dashboard view of stock levels, transfer activity, and the inventory knowledge-graph for the warehouse.)

Avoids: pulse (Never sees the engine; integer ledger/CID/CBOR internals are too low-level for a stock operator.) · bt (A warehouse has Wi-Fi/LAN; no off-grid Bluetooth-mesh transport need.) · gither (Multi-repo build navigation is irrelevant to running a warehouse.) · virtualpc (Multi-agent runtime is plumbing under Lens; not touched directly.)

Level: Medium · **Entry:** Spreadsheet-like UI plus a Lens-backed web app, with a barcode/scanner mobile companion

Wants: - **knitweb** — supplychain knitweb 'inventory movement' record type with quantity+lot conservation gates (receive/pick/transfer/ship) so an event is refused if on-hand would go impossibly negative — *Medium-literacy operator needs guardrails that block physically impossible stock states without understanding CBOR or balances.* - **lens** — Cycle-count and replenishment agent that proposes actions as a confirm/reject checklist, not free-form output — *Keeps the human in control with simple approvals matching a floor manager's literacy.* - **monitor** — Mobile-friendly stock dashboard with discrepancy alerts (expected vs scanned) and one-tap drill-down to the offending lot — *Discrepancy triage happens on the floor on a phone, not at a desk.* - **knitweb** — Lot/serial provenance lookup that returns the upstream chain of a SKU from one scan — *Recalls and audits require tracing a unit back through the woven fabric without engine knowledge.* - **lens** — Offline-queue mode that captures scans locally and re-weaves them when connectivity returns — *Warehouse dead zones must not block intake; Medium users want it to 'just sync' later.*

Procurement officer

Sources, negotiates, and approves purchases against verifiable supplier and stock signals in the shared memory.

Intake: demand and reorder triggers arrive from the warehouse/inventory records in shared memory. Plan/do: a Lens sourcing agent drafts RFQs, compares supplier quotes and lead-time history, and routes a recommended PO for human approval; the approved PO and supplier commitment are written as signed records. Verify/ship: receipts are matched three-way (PO vs goods-received event vs invoice) by agents against the content-addressed ledger, and exceptions are escalated to the officer.

Uses: lens (Generates RFQs, ranks suppliers on price/lead-time/quality history, and performs three-way match; officer approves spend.) · **knitweb** (Purchase orders, supplier scorecards, and goods-received events are stored as shared content-addressed records linking to warehouse inventory.) · **vbank** (Routes spend approvals and budget caps through governance/treasury so larger POs need a vote or sign-off.) · **monitor** (Spend dashboards, open-PO aging, and supplier-performance graph views.)

Avoids: pulse (Treasury and CIDs are surfaced through vbank/app; the officer never touches the ledger engine.) · bt (Office-based role with normal connectivity; no mesh transport need.) · gither (Build/repo tooling is unrelated to sourcing.) · virtualpc (Agent-coordination runtime is hidden behind Lens.)

Level: Medium · **Entry:** No-code web app with approval workflows; optional REST API for ERP integration

Wants: - **lens** — Three-way-match agent (PO / goods-received event / invoice) that surfaces only mismatches for review — *Medium user wants exception-based work, not to reconcile every line manually.* - **vbank** — Budget-cap and approval-threshold policy where POs over a limit auto-route to a treasury vote — *Spend governance must be enforced by guardrails, not officer discipline.* - **knitweb** — Supplier scorecard records with signed delivery/quality history so reputation is portable and tamper-evident — *Sourcing decisions need trustworthy provenance the officer can cite to management.* - **lens** — Plain-language RFQ drafting and quote-comparison templates — *Lowers the bar so a non-technical buyer can launch a sourcing round quickly.* - **monitor** — Open-PO aging and lead-time-variance dashboard — *Spots slipping suppliers before stockouts without querying the ledger.*

Quality manager

Defines, enforces, and audits quality gates so only conforming lots are released, with full provenance.

Intake: inspection results and process events stream in from warehouse/supplier records in shared memory. Plan/do: the manager codes acceptance rules (spec limits, sampling plans) that Lens agents evaluate per lot, writing signed pass/fail/quarantine attestations into the content-addressed fabric. Verify/ship: agents re-check a sample of attestations (proof-of-useful-work style) and the manager signs the release; the audit trail is permanently woven and queryable for recalls/CAPA.

Uses: lens (Runs deterministic quality-gate evaluations and root-cause/CAPA reasoning over the lot-provenance graph.) · **knitweb** (Inspection records, non-conformance reports, and release attestations are signed, content-addressed, and immutably linked to lots.) · **monitor** (Knowledge-graph explorer to trace defect propagation across lots, suppliers, and processes.) · **docs** (References contract/provenance docs and the supply-chain knitweb record schemas when authoring gates.)

Avoids: bt (In-facility lab/QA work; no off-grid transport.) · gither (Not building multi-repo software; build planning is out of scope.) · virtualpc (Uses Lens-level orchestration, not the raw multi-agent runtime.) · vbank (Quality release is technical/regulatory, not treasury/voting.)

Level: High · **Entry:** Python SDK and REST API over Lens, plus a web app for non-technical reviewers

Wants: - **lens** — Deterministic, versioned quality-gate rule engine (spec limits + sampling plans) with provenance-depth query API — *High-literacy QA needs reproducible, auditable verdicts and deep traceback, not opaque suggestions.* - **knitweb** — Immutable non-conformance and CAPA record types linked to the originating lot/process CID — *Regulated audits require tamper-evident chains the manager can prove byte-for-byte.* - **lens** — Sampled re-verification hook so a fraction of pass/fail attestations are independently re-evaluated — *Matches proof-of-useful-work discipline and catches gate drift or tampering.* - **monitor** — Defect-propagation graph query that returns every downstream lot sharing a suspect input — *Recall scoping must be exhaustive and provable, surfaced visually from the woven graph.* - **docs** — Stable, versioned schema reference for supply-chain/operational record types — *Authoring gates against the fabric requires a precise, pinned contract.*

Facilities manager

Keeps buildings, assets, and maintenance running by logging issues and dispatching work as simple shared records.

Intake: staff report issues via the mobile app (photo + location), creating a work-order record in shared memory. Plan/do: a Lens agent triages priority, suggests a vendor/technician and a slot, and the manager taps approve; the assignment and completion are written as signed records. Verify/ship: photo/sign-off on completion closes the work order in the fabric, and recurring preventive-maintenance tasks are auto-scheduled by an agent.

Uses: lens (Triages and prioritizes work orders, schedules preventive maintenance, and suggests vendors; manager confirms with taps.) · **knitweb** (Work orders, asset records, and maintenance history live as shared records so any approved tech sees the same status.) · **monitor** (Simple dashboard of open tickets, overdue PM, and asset status.)

Avoids: pulse (Low-literacy role; engine, CIDs, and ledger are completely hidden behind the app.) · bt (Buildings have connectivity; no mesh need.) · gither (Software build tooling is irrelevant.) · virtualpc (Never sees the agent runtime.) · vbank (Routine facilities spend handled via the app; no direct treasury/voting role.)

Level: Low · **Entry:** Consumer mobile app plus a no-code web app for work orders

Wants: - **lens** — Photo-to-work-order intake: snap a picture, agent fills category/priority/location automatically — *Low-literacy users need near-zero typing; the agent does the structuring.* - **knitweb** — Plain-language work-order record templates with required fields enforced as guardrails — *Prevents incomplete tickets without expecting the user to know any schema.* - **lens** — Auto-scheduling agent for recurring preventive maintenance with simple reminders — *Removes the cognitive load of tracking PM calendars.* - **monitor** — Mobile traffic-light dashboard (green/amber/red) for open and overdue tasks — *Status must be glanceable for a non-technical manager on the go.* - **lens** — One-tap vendor dispatch with completion photo sign-off — *Closing the loop should be as easy as taking a picture.*

Event planner

Coordinates venues, vendors, schedules, and logistics into one shared, agent-assisted plan.

Intake: event brief, date, headcount, and budget entered in the app. Plan/do: a Lens agent generates a task checklist, vendor shortlist, and timeline, and the planner approves bookings; confirmations and contracts are stored as shared records so the whole team sees one source of truth. Verify/ship: agents track RSVPs, deliverables, and deadlines, nudging on slippage, and the run-of-show is shared with all participants.

Uses: lens (Builds the checklist, timeline, and vendor shortlist; tracks RSVPs and deadlines; planner approves.) · **knitweb** (Bookings, contracts, vendor confirmations, and the run-of-show live as shared records the whole team references.) · **monitor** (Simple countdown/checklist dashboard of what's done, pending, and at-risk.)

Avoids: pulse (Low-literacy creative role; never touches engine internals.) · bt (Venues have connectivity; no off-grid mesh need.) · gither (No software build relevance.) · virtualpc (Agent runtime stays hidden under the app.) · vbank (Event budgets handled in-app; no governance/treasury role.)

Level: Low · **Entry:** Consumer mobile app plus a no-code web app with calendar/checklist views

Wants: - **lens** — Template-driven event planner: pick event type and the agent generates a full checklist and timeline — *Low-literacy users want a ready-made plan, not a blank canvas.* - **knitweb** — Shared run-of-show and vendor-confirmation records with simple read-only sharing links — *Everyone on the team needs one trustworthy source of truth without accounts on the substrate.* - **lens** — Deadline and RSVP nudge agent with plain-language reminders — *Keeps a non-technical planner ahead of slippage automatically.* - **monitor** — Countdown dashboard highlighting at-risk tasks before the event date — *Glanceable risk view suits a busy, non-technical coordinator.* - **knitweb** — Mobile-first vendor booking record with photo/contract attachment — *Bookings happen on the phone; attachments must be effortless.*

Everyday & Emerging

Entrepreneur / small-business owner

Runs the business while agents handle the busywork, with every decision and document traceable in shared memory.

Intake: owner describes a need in plain language ('chase unpaid invoices', 'price this quote') in the mobile app, which an agent turns into a plan it shows back for one-tap approval. Do: the agent acts and writes every artifact (invoice, quote, contract draft) as a content-addressed record in the shared knitweb memory so it is reusable and auditable. Verify+ship: lens re-checks the result against the business's own past records before the owner approves with a tap; nothing touches money or customers without that confirmation.

Uses: knitweb (Shared memory of record for the business: customers, quotes, invoices, suppliers stored as content-addressed records the owner never has to file or search by hand.) · **lens** (Plain-language agent that turns 'do X for my business' into a checked plan and verifies output against the business's own history before asking for approval.) · **knitweb.github.io** (Sign-up, account, and the friendly mobile/web app shell the owner actually opens.)

Avoids: pulse (Never sees the engine; CLI, CIDs, and the ledger are too low-level for someone who just wants invoices chased.) · bt (No off-grid Bluetooth-mesh need; runs on a normal phone with internet.) · gither (Multi-repo build navigation is a developer tool, irrelevant to a shopkeeper.) · virtualpc (Multi-agent runtime internals are abstracted away behind the one-tap app.) · vbank (No DAO governance or treasury voting in a single-owner small business.)

Level: Low · **Entry:** consumer mobile app + no-code web app

Wants: - **lens** — One-tap approval gate with a plain-language 'here is what I am about to do and why' summary before any agent action that spends money or contacts a customer — *Low-literacy owner needs a hard guardrail and a human-readable preview, not raw agent logs.* - **knitweb** — Templated business memory packs (invoicing, quoting, customer list) that pre-structure records so the owner never designs a schema — *Low-literacy roles need ready-made templates, not a blank content-addressed graph.* - **knitweb.github.io** — Mobile-first onboarding that creates the identity/wallet behind the scenes with no key handling shown — *Per the identity and mobile-fit docs, key custody must be invisible for a phone-only non-technical owner.* - **monitor** — A single plain-language 'business health' card (cash in/out, who owes what) instead of a graph explorer — *Low literacy means a simplified dashboard, not the raw knowledge-graph view.* - **lens** — Undo/rollback that reverts an agent action by pointing at the prior content-addressed record — *Owner needs a safety net because they cannot reason about state themselves.*

Freelancer / solopreneur

A one-person service business that uses agents to deliver client work and keeps a verifiable, portable record of everything delivered.

Intake: freelancer logs a client brief in a spreadsheet-like board; an agent drafts a plan and timeline they edit directly. Do: the agent executes deliverables and proposals, writing each version as a content-addressed record in shared memory so client work is provenance-stamped and reusable across gigs. Verify+ship: lens diffs the deliverable against the brief and prior accepted work, the freelancer reviews, then ships to the client with a shareable proof-of-delivery record.

Uses: knitweb (Portable, provenance-stamped record of every deliverable, proposal, and client interaction that the freelancer owns and can carry between platforms.) · **lens** (Drafts deliverables, checks them against the brief and past accepted work, and surfaces a diff for review.) · **monitor** (Lightweight dashboard of active gigs, deliverable status, and what has been shipped vs. pending.) · **knitweb.github.io** (Account, billing, and the no-code app surface.)

Avoids: pulse (Spends PLS through the app but never touches the engine, ledger internals, or canonical CIDs directly.) · bt (No off-grid mesh requirement for a connected freelancer.) · gither (A build/repo-navigation tool with no place in service delivery.) · virtualpc (Multi-agent coordination is handled for them; a solo operator does not orchestrate fleets.) · vbank (No collective treasury or voting; it is a one-person business.)

Level: Medium · **Entry:** no-code web app + spreadsheet-like UI (light REST/automation hooks)

Wants: - **knitweb** — Exportable, signed proof-of-delivery record per gig that a client can verify without joining knitweb — *Medium-literacy freelancer wants portable provenance they can hand to clients on any platform.* - **lens** — Brief-to-deliverable diff that highlights what changed vs. the agreed scope — *Solo operator needs scope-creep and accuracy checks they can audit, a step above pure guardrails.* - **knitweb.github.io** — Light automation/REST hooks (e.g. webhook on 'deliverable approved') for connecting existing tools like calendars and invoicing — *Medium literacy can wire simple integrations but will not write SDK code.* - **monitor** — Per-client timeline view showing every version and decision tied to its content-addressed record — *Needs traceability of who-asked-for-what to defend billing and revisions.* - **knitweb** — Reusable deliverable templates seeded from the freelancer's own best past work — *Lets a solopreneur compound prior gigs without rebuilding from scratch each time.*

Content creator

Produces and publishes content with agents, while keeping a verifiable record of originals and sources to prove authorship.

Intake: creator drops an idea or rough draft into the mobile app; an agent expands it into a content plan and drafts. Do: the agent generates and schedules content, writing each draft and its source material as content-addressed records in shared memory so originals and provenance are preserved. Verify+ship: lens checks the draft for the creator's voice and flags borrowed/sourced material, the creator approves on the phone, and it publishes with an authorship-stamped record.

Uses: knitweb (Tamper-evident store of originals, drafts, and sources so the creator can prove what they made and when.) · **lens** (Drafts content in the creator's voice and flags sourced material and possible accuracy issues before publishing.) · **knitweb.github.io** (The mobile app and account surface where ideas go in and approvals happen.)

Avoids: pulse (Never sees CIDs, ledger, or the P2P engine; the app abstracts all of it.) · bt (No off-grid publishing need.) · gither (Developer build tool, irrelevant to content work.) · virtualpc (Agent orchestration is hidden behind a single creative assistant.) · vbank (No governance/treasury role for an individual creator.) · molgang (A chemistry domain-knitweb template, unrelated to general content creation.)

Level: Low · **Entry:** consumer mobile app

Wants: - **knitweb** — Automatic authorship/provenance stamp on every original, exportable as a shareable proof — *Low-literacy creator needs one-tap proof-of-authorship without understanding content addressing.* - **lens** — Plain-language 'source flag' that warns when a draft leans on borrowed material before publish — *Guardrail framed in everyday words, not a citation graph, suits low literacy.* - **knitweb.github.io** — Mobile schedule-and-publish flow with preview, fully phone-native — *Mobile-first is essential; this role lives on a phone.* - **lens** — Voice/style consistency

check against the creator's past published work — *Keeps output on-brand using the creator's own memory, no config needed.* - **knitweb** — Templated content packs (caption, thread, short-video script) so no schema setup is required — *Low literacy needs ready-made templates.*

Community manager

Keeps a community healthy and informed, with agents triaging conversations and a shared memory of decisions and FAQs.

Intake: incoming questions, reports, and threads flow into a moderation board where an agent triages and proposes responses or actions. Do: with the manager's approval, agents answer FAQs, route issues, and log every decision and canonical answer into shared memory so the community has one consistent source of truth. Verify+ship: lens checks proposed answers against the community's logged canon and policy before posting; the manager spot-checks the dashboard for sentiment and escalations.

Uses: knitweb (Canonical, content-addressed store of FAQs, rulings, and decisions so answers stay consistent over time and across moderators.) · **lens** (Triages incoming conversations, drafts policy-consistent responses, and flags items needing a human.) · **monitor** (Community-health dashboard: sentiment, open issues, escalations, and a knowledge-graph view of recurring topics.) · **vbank** (Lightweight: runs community polls/decisions when the group needs to vote on a rule or budget.) · **knitweb.github.io** (The no-code app surface and account.)

Avoids: pulse (Never touches the engine, ledger, or CIDs; works entirely on the friendly surface.) · bt (No off-grid mesh need for an online community.) · gither (Developer multi-repo tool, irrelevant to moderation.) · virtualpc (Agent fleet orchestration is abstracted; the manager approves, it doesn't configure runtimes.) · molgang (A specific chemistry domain template, not general community work.)

Level: Medium · **Entry:** no-code web app + dashboard (monitor)

Wants: - **lens** — Policy-grounded answer drafting that cites the community's own logged ruling for each suggested reply — *Medium literacy wants auditable consistency, not just a guardrail.* - **monitor** — Sentiment + escalation dashboard with a topic knowledge-graph of recurring issues — *This role can read a richer dashboard and needs to spot trends, not just a single card.* - **knitweb** — Canonical-answer registry so an FAQ has one content-addressed source of truth that updates everywhere when revised — *Consistency across many threads and moderators requires a single addressed record.* - **vbank** — Simple no-code community poll with transparent, tamper-evident tally — *Needs light governance for rule/budget decisions without standing up a full DAO.* - **lens** — Human-in-the-loop escalation queue with one-click approve/edit/reject on agent-proposed actions — *Keeps a human gate on moderation actions at a manager-appropriate level of control.*

Personal / virtual assistant

Manages a principal's schedule, communications, and tasks through agents, keeping a shared, permissioned memory of context and decisions.

Intake: requests arrive by message, email, or calendar; an agent interprets them against the principal's preferences stored in shared memory and proposes a plan. Do: the agent books, drafts replies, and tracks tasks, writing each action and its context as a content-addressed record so the principal's history is consistent and handoff-safe. Verify+ship: lens checks proposed actions against standing preferences and permissions, the assistant (or principal) approves sensitive items, and it ships with an audit trail.

Uses: knitweb (Permissioned shared memory of the principal's preferences, contacts, and past decisions so context survives across tasks and assistants.) · **lens** (Interprets requests against stored preferences, drafts replies and bookings, and verifies actions against permission rules before acting.) · **monitor** (A simple agenda/task dashboard of what is scheduled, pending approval, and done.) · **knitweb.github.io** (The mobile app, account, and connection of calendar/email tools.)

Avoids: pulse (Never sees the ledger, CIDs, or P2P engine; operates on the app surface only.) · bt (No off-grid mesh requirement.) · gither (A developer build/navigation tool with no assistant use.) · virtualpc (Multi-agent runtime details are hidden behind the assistant experience.) · vbank (No collective treasury/voting in a personal-assistant context.) · molgang (Chemistry domain template, irrelevant to scheduling and comms.)

Level: Medium · **Entry:** consumer mobile app + light automation hooks (calendar, email, REST)

Wants: - **lens** — Standing-preference engine that checks every proposed action against the principal's rules and asks for approval only on sensitive items — *Medium literacy needs configurable trust levels, not just blanket approve/deny.* - **knitweb** — Permissioned, scoped memory so the principal controls exactly what an assistant agent can read or write — *Handling someone else's data demands fine-grained, content-addressed access control.* - **knitweb.github.io** — Calendar/email/REST connectors with a clear consent screen per integration — *Medium literacy can connect tools but needs transparent, revocable permissions.* - **monitor** — Pending-approval queue distinct from auto-handled items — *Keeps a human gate on high-stakes actions while letting routine ones flow.* - **knitweb** — Handoff-safe context export so a new assistant inherits the full provenance-stamped history — *Continuity across assistants requires portable, addressed context records.*