SANITIZED TECHNICAL PORTFOLIO

Private Systems Case Studies

Two production systems that demonstrate customer-style discovery, end-to-end delivery, workflow design, applied AI judgment, secure deployment, and operational ownership.

“The repositories and live data remain private. The architecture, decisions, and verification evidence are presented here without exposing credentials, personal records, infrastructure addresses, or production access.”

Alexandria Job Search Command Center

Multi-source ingestion, structured LLM evaluation, deterministic application state, dashboard operations, and Telegram delivery.

Daily Momentum Command Center

FastAPI task platform with browser and Telegram interfaces, durable shared state, reminders, and private deployment.

Bryan Oyloe
boyloe@gmail.com · linkedin.com/in/bryan-oyloe · github.com/boyloe

CASE STUDY 01

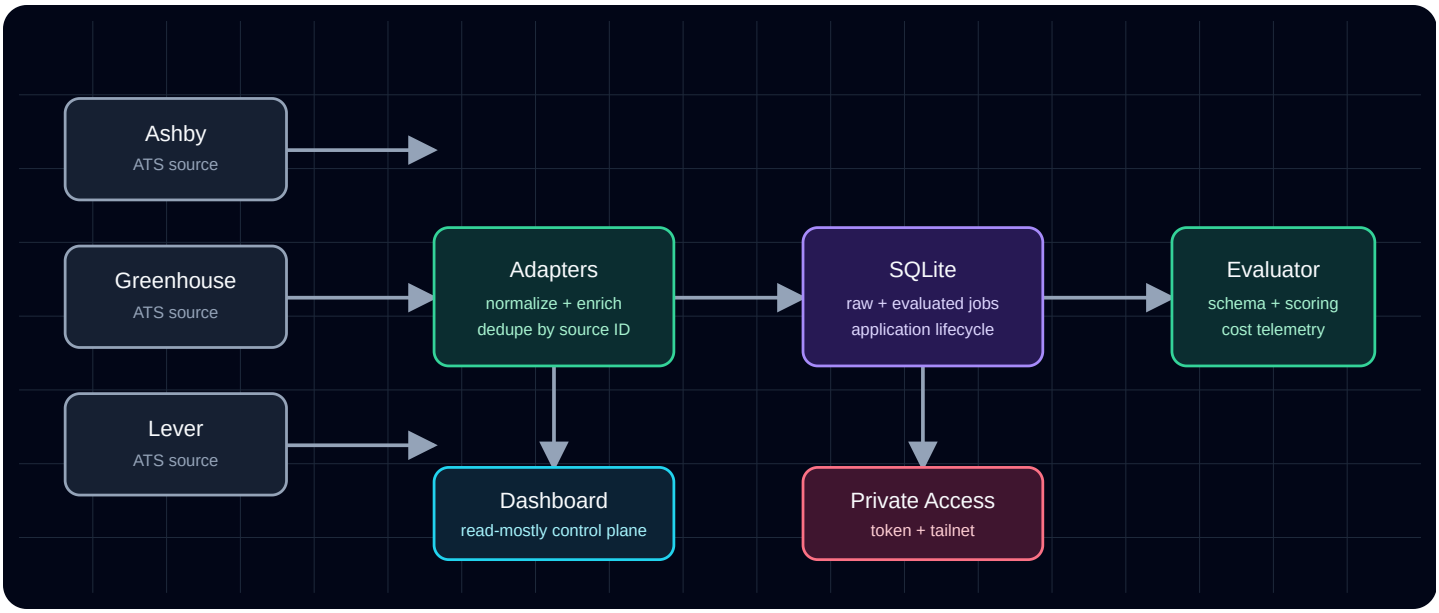
Alexandria Job Search Command Center

PROBLEM

Job leads arrived from different applicant-tracking systems with inconsistent fields and duplicated lifecycle work. Generic keyword matching obscured seniority, location, compensation, and application-plausibility risks. Model calls also needed explicit cost controls.

OWNERSHIP

Designed and built the ingestion adapters, normalized data model, evaluation contracts, application-state workflow, operational dashboard, scheduled pipeline, Telegram digest, private deployment, and test coverage.



3
ATS ADAPTERS

89
TESTS PASSING

1
DURABLE SOURCE OF TRUTH

FDE relevance: The system starts with an ambiguous business workflow, integrates inconsistent external systems, combines deterministic and probabilistic components, and remains owned through deployment and day-to-day operation.

Designing for trust, cost, and operability

Source-specific adapters; source-agnostic evaluation

Each ATS preserves its own retrieval and normalization logic. The evaluator consumes one normalized contract, preventing source quirks from leaking into scoring.

Deterministic state around probabilistic output

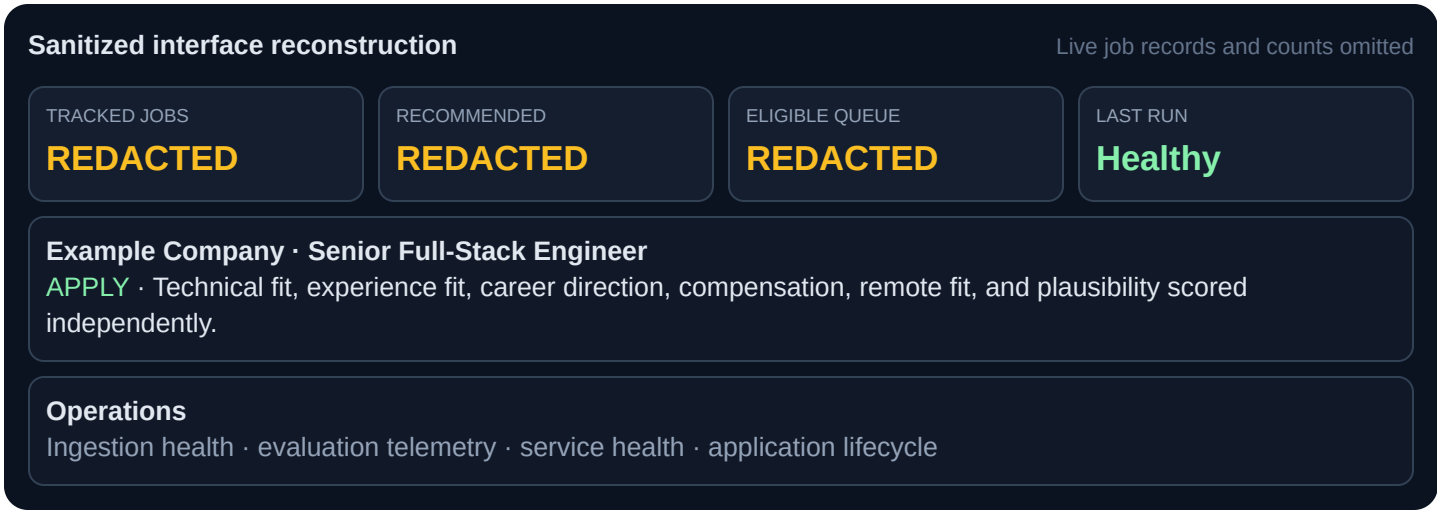
Application status, deduplication, eligibility, and workflow transitions remain deterministic. LLM output is schema-validated and stored with model, latency, token, prompt-version, and cost telemetry.

Cost as a product constraint

Already-evaluated, applied, closed, incomplete, or implausible records are filtered before model use. Paid evaluation cannot be triggered from the dashboard.

Read-mostly operational surface

The dashboard exposes summaries, recommendations, run health, and lifecycle updates while keeping ingestion, scheduler changes, and paid model calls outside the UI.



TRADEOFFS

- SQLite favors operational simplicity over horizontal scale.
- A private network and shared token fit a single-user threat model; this is not multi-tenant auth.
- Explicit workflow boundaries add code but reduce accidental cost and state corruption.

VERIFICATION EVIDENCE

- Verified 89 tests plus 2 subtests pass.
- Three independently implemented ATS integrations.
- Schema-constrained evaluation with per-run telemetry.
- Private service deployment with health and admin monitoring.

CASE STUDY 02

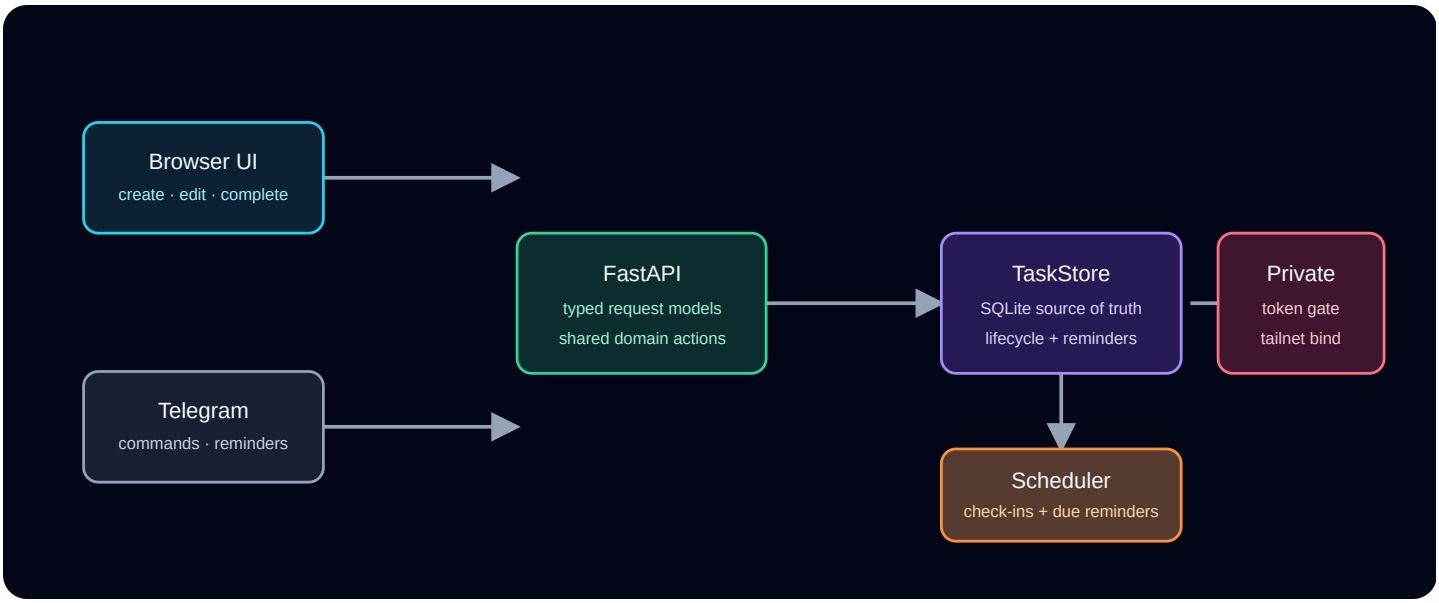
Daily Momentum Command Center

PROBLEM

Tasks entered in chat, reminders, and a browser dashboard easily drift into separate sources of truth. The system needed one durable task model with consistent behavior across interfaces and reliable reminder semantics.

OWNERSHIP

Designed and built the API, SQLite task store, typed domain models, Telegram command layer, reminder workflow, dashboard actions, private deployment, and automated tests.



2
USER INTERFACES

38
TESTS PASSING

1
SHARED TASK STATE

FDE relevance: Multiple user touchpoints were reconciled into one operational workflow. The implementation emphasizes adoption, predictable state transitions, direct user feedback, and production handoff rather than a standalone prototype.

One workflow across browser and chat

Shared source of truth

The browser UI, API, Telegram commands, check-ins, and reminder sender all use the same SQLite-backed TaskStore instead of maintaining interface-specific state.

Explicit lifecycle semantics

Create, update, complete, snooze, and permanent delete have distinct behavior. Reminder delivery records the send and advances the next reminder rather than silently duplicating notifications.

Typed API boundary

Pydantic models constrain task areas, priorities, statuses, time fields, completion criteria, materials, location context, and estimates before state reaches storage.

Private production footprint

The service runs under a supervised user service, binds to a private network interface, uses a token gate, persists locally, and exposes a health endpoint for operational verification.

Sanitized interface reconstruction

Sample tasks; no production records

MAIN MOVE	OPEN	DUE	SERVICE
1	REDACTED	REDACTED	Healthy

Review architecture notes
career · high priority · computer

Prepare interview walkthrough
career · medium priority · reminder enabled

TRADEOFFS

- SQLite is appropriate for a single-user service but not a multi-writer SaaS deployment.
- A lightweight command parser favors predictability over unrestricted natural language.
- Private-network deployment reduces exposure but intentionally limits public demos.

VERIFICATION EVIDENCE

- Verified 38 automated tests pass.
- CRUD, snooze, reminders, Telegram parsing, dashboard, API, and scheduler behavior are covered.
- Health checks and direct database readbacks support release verification.
- The same state is exercised through both browser and messaging workflows.

VERIFICATION GUIDE

How an employer can evaluate this work

Private does not mean unverifiable. It means verification happens through controlled evidence instead of public production access.

Available during an interview

- Guided screen-share of sanitized workflows
- Architecture and data-flow walkthrough
- Representative code excerpts with secrets and personal data removed
- Live or recorded automated-test execution
- Discussion of failure modes, tradeoffs, and next steps

Intentionally not shared

- Production URLs or private-network addresses
- Tokens, environment files, or deployment credentials
- Personal job records, task data, or Telegram identifiers
- Unrestricted repository or database access
- Access to the live production environment

SUGGESTED 15-MINUTE WALKTHROUGH

1. **Frame the user problem**
   What failed with the manual workflow and what “success” meant.
2. **Trace one request end to end**
   Input → validation → persistence → workflow decision → interface output.
3. **Explain one hard tradeoff**
   Deterministic vs. probabilistic logic, cost controls, SQLite, or private access.
4. **Show reliability evidence**
   Tests, health checks, telemetry, failure isolation, and release verification.
5. **Close with what changes at scale**
   Authentication, tenancy, queueing, managed persistence, and observability.

> **Positioning statement:** “These are private, actively operated systems. I can demonstrate them through a sanitized screen share and explain the code, architecture, test strategy, deployment model, and tradeoffs without exposing personal data or production credentials.”