From quoting by gut to quoting from source — cited end-of-life answers in seconds

An internal AI workflow that answers Verizon partner fleet questions — still sold? official replacement? how do alternatives compare? — with source-of-truth citations, never a guess.

1,024
Devices supported — current + end-of-life

656
Validated replacement mappings, with authority levels

Seconds
Per fleet question — down from a dozen manual lookups

0
Specs invented — cited or dropped

THE PROBLEM

A Verizon Client Partner fields several fleet conversations a week — each a list of router SKUs across Cradlepoint, Sierra Wireless, Cisco, Peplink and more. Every SKU raises the same three questions: still sold, official replacement, how the alternatives compare. The answers live in a twelve-tab device workbook — but one fleet question meant a dozen lookups across four sheets and two CSVs, so under time pressure the analysis got skipped and quotes went out on gut. The cost of gut isn't hypothetical: quote a discontinued router and the best case is a re-quote; the worst case is a deal that walks with the credibility.

WHAT WE BUILT

- A compiled, queryable source-of-truth catalog — built from the canonical device workbook, which stays canonical.
- An agent that selects tools — catalog lookup → find replacements → spec compare → cited docs — to answer each question.
- Browser access for partners with no VPN or IT lift, plus the same tools over MCP for agent clients.
- The citation contract: every factual claim carries a tool-result citation, or it is dropped.

The safeguard. Persona prompts encode hard rules: no fabricated pricing, lead times, or Verizon promo policy. Feature gaps are disclosed, never filled. The device workbook remains the single source of truth.

THE RESULT

In internal active use since the March–April 2026 build sprint. A fleet question that used to take a dozen manual lookups now returns a cited answer in seconds — still-sold status, the official replacement, and how the alternatives compare — with zero invented specs. A user-facing accuracy measurement window is in progress.

Have a source of truth buried in a workbook your team quotes from by hand?

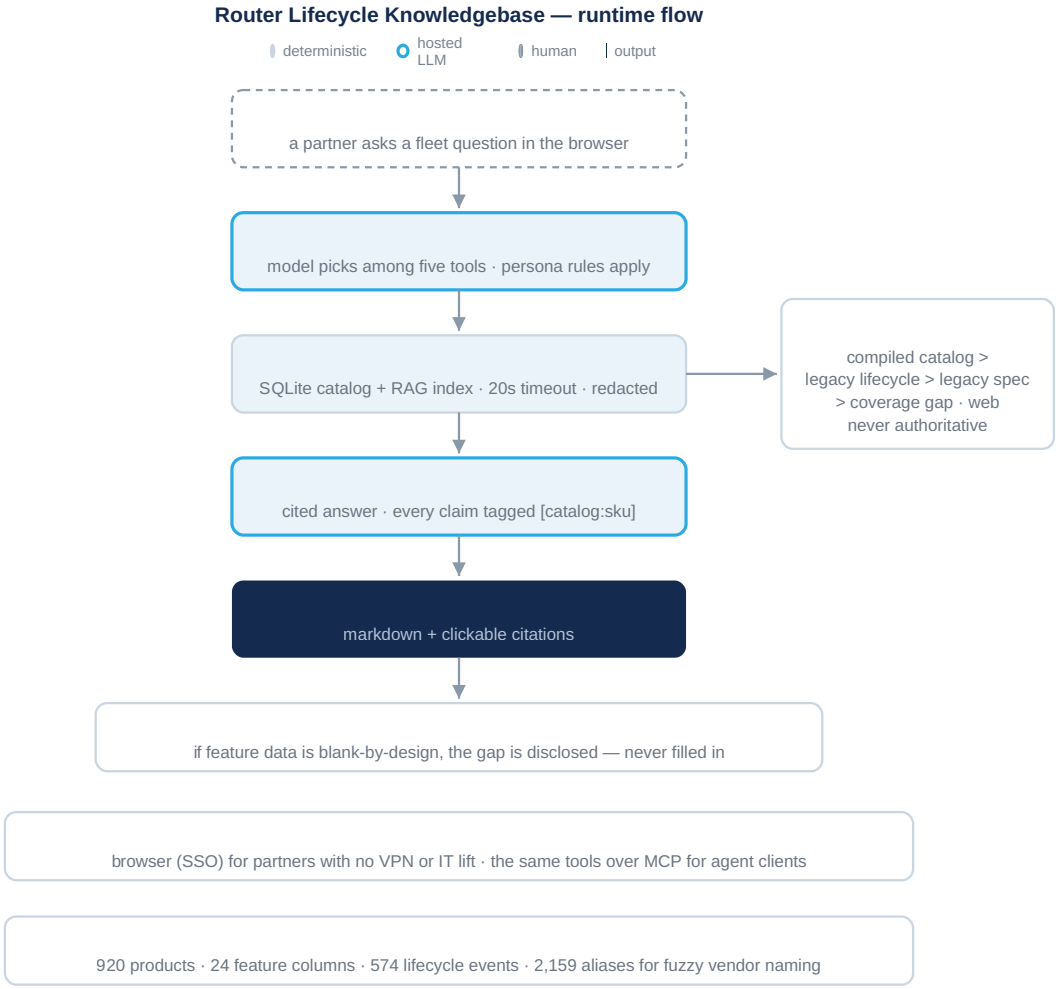
Start with an AI Workflow Teardown →

Pete Dunn · PD Insights · pete@pdinsights.ai · pdinsights.ai

Full technical detail on the following pages

HOW IT WORKS — RUNTIME FLOW

A partner's fleet question is answered by an agent that selects among five tools; tool calls run inside the application container against a local SQLite catalog and a local RAG corpus — the model never sees the workbook directly, only the structured fields tools return. Tool outputs are redacted of pricing, internal IDs, and absolute paths before they reach the model. Reps describe routers, not customers — no customer identifiers enter the system.



The agent does not “know” what an IBR350 is — the catalog does. Every factual claim carries a tool-result citation, or it is dropped.

Two design choices stand out. **Source-of-truth precedence:** tools consult the compiled catalog first; legacy files fill gaps only, and web results are never authoritative. **The fabrication ban:** no invented pricing, lead times, or Verizon promo policy — encoded in the persona prompt as a tool-output contract.

AI DESIGN — THE SCHEMA IS THE PRODUCT

The defensible technical idea is not the model — it is the catalog schema plus source-of-truth precedence. The catalog's 920 products carry 24 normalized feature columns, 656 replacement mappings with authority levels, 574 lifecycle events, and 2,159 aliases for fuzzy matching across vendor naming.

The prompt tells the model: use the recommended replacement verbatim for end-of-life devices; if feature data is unavailable, disclose the gap and never invent specs. The defensible idea is the set of rules governing when the model is — and is not — allowed to fill gaps, encoded as a tool-output contract.

RELIABILITY AND GUARDRAILS

MECHANISM	WHAT IT DOES
Per-surface authentication	Distinct auth gates on each access surface — SSO for the web app, token auth for MCP clients
Build-time data-integrity guards	The container build fails if catalog or RAG artifacts arrive as unresolved pointers instead of real data
Build-provenance diagnostics	Artifact integrity is verifiable on the live runner for incident response, without exposing data
Source-of-truth precedence + coverage flag	Compiled catalog first; legacy CSVs fill gaps only. The agent sees which products have populated specs vs. blank-by-design (411 of 920 lacked feature norms in v25)
Workbook-rebuild manifest	Records the source workbook checksum and per-table row counts; rebuilds reproduce the artifact bit-for-bit
~1,000 backend tests + ~12 real-DB end-to-end regressions	Tests exercise the production code path against the actual seed catalog — the class of test that would have caught the original tool-nesting bug pre-deploy

INTEGRATION STACK

LAYER	TOOL	ROLE
AI engine	Hosted OpenAI model (gpt-5-mini at time of writing)	Tool selection, answer synthesis, citation insertion
Backend	FastAPI + Uvicorn	Tool dispatch, persona routing, audit logging
Frontend	React + Vite + TypeScript	Routers tab, knowledgebase tab, citation rendering
Auth	SSO (web) + token auth (MCP)	Partner access + agent-client access
Catalog	SQLite + custom 28-table relational schema	Compiled from the device master workbook via a build script
Knowledgebase	Pre-chunked JSONL + on-disk index	Manufacturer datasheets, official EOL pages, replacement guides
Deploy	Docker (multi-stage) + Git LFS · Hugging Face Spaces	Single-image deploy, catalog + RAG baked in (~22 MB); auto-rebuild on push

WHAT I BUILT — VS. WHAT THE TOOLS DID

- **Workflow discovery** — mapped the Client Partner's fleet-replacement workflow; identified the “remember to look it up” bottleneck.
- **Catalog schema** — 28-table relational model: products, aliases, lifecycle events, replacement mappings with authority levels, port maps, antenna flow, fitment rules.
- **Source-of-truth precedence** — compiled catalog exact > legacy lifecycle > legacy spec fallback > coverage gap; web never authoritative.
- **Agent runtime + tools + prompts** — persona-scoped dispatcher (20s timeouts, audit log, redaction); five tools; prompts encoding citation rules, gap disclosure, and fabrication bans.
- **Deploy infrastructure** — single-commit deploy flow, build-time data guards, build-provenance diagnostics.
- **Test coverage** — end-to-end real-DB regressions exercising the seed catalog — the tests that would have caught the original tool-nesting bug pre-deploy.

WHAT'S MEASURED VS. WHAT'S PENDING

METRIC	OBSERVATION
Catalog feature coverage	509 → 920 rows (55% → 100%) after the feature backfill
EOL devices previously returning empty results	Verified populated for IBR350, RUT240, IBR900, IBR450, COR IBR1100, ES450, MP70 and more — plus the 411-row backfill set as a class
Production bugs found + fixed in sprint	5 distinct (tool nesting, missing RAG corpus, replacement-scope gap, compare-schema, feature-data gap)
Deploys	9 sequential, 100% success; ~12 net-new end-to-end real-DB regression tests added
User-facing answer accuracy	Not yet measured — collecting in the pending rollout window; will replace these build-sprint signals with harder data

TRADEOFFS AND LIMITS

Hosted LLM, not local. Internal partner tool; no end-customer PII passes through the model. v2 adds an optional local-model path for deployments that cannot egress.

387 of 411 backfilled feature rows use a family-level default. Correct for the bulk of fleet questions; per-SKU correction is cheap when flagged. v2: targeted datasheet pass on highest-traffic SKUs.

No formal extraction-accuracy benchmark yet; single deploy, no canary. Canary A/B infrastructure exists in code, feature-gated off; a held-out test set with an answer-grading rubric is the v2 move.

Spec compare doesn't yet model certifications, operating temp, or antenna ports. The schema doesn't normalize them yet; the persona prompt sets the agent's expectations correctly.

CONSULTING TAKEAWAY

The bug was not where it looked like it was. Every theory was data loss — pointers, missing files, a deploy regression. The actual root cause was a tool-implementation bug: a lookup read the wrong nesting level of a perfectly fine API response, and the unit tests stubbed a fictional response shape that masked it. The highest-leverage move of the whole engagement was adding end-to-end real-database regression tests.

AI workflows fail in unfamiliar ways. The agent layer obscures bugs as “the LLM answered poorly” when the real problem lives in the seam between the tool contract and the data contract. If you only test one thing, test that seam.