Independent Research Desk — Technical View (Example)

Status: Conceptual, illustrative. This is Level 3 of three: the architecture blueprint set the principles, the systems design showed the end-to-end behaviour. This document is **one example instantiation** of both — a concrete technology choice, sized and costed, so an operator can see the actual shape, weight, and cost before committing to anything. Any other tech stack that honours Sections 1–5 of the architecture blueprint would satisfy the same design equally well; nothing here is the only correct choice.

All figures below are **illustrative** — round numbers meant to show the shape of the cost, not a quote. An operator should replace the placeholder rates with whatever their chosen provider actually charges at build time.

1. Example Technology Choices

Deliberately ordinary, deliberately swappable — chosen because each piece is boring, well-understood, and cheap to run at this scale:

Layer	Example choice	Why this one
Language/runtime	Python	Large ecosystem, cheap for a coding agent to scaffold and test, no build toolchain to manage
Storage	SQLite (single file)	Zero operational overhead, handles this data volume comfortably, trivial to back up (copy the file)
Scheduling	OS-level cron	No orchestration system needed at one-desk scale
Reasoning	A general-purpose frontier LLM API, provider-agnostic	Architecture doesn't depend on which one — see architecture blueprint Section 4
Interface	Command-line tool	Fastest to build; a local web view is an optional later upgrade, not a dependency
Compute	A single machine — a personal laptop, or one small cloud instance	No component here does heavy local computation; the LLM provider carries the reasoning load

2. Component List & Specs

The same five reasoning/process components as the architecture blueprint (Section 3 there), plus the data layer and the CLI that make them runnable day to day — seven rows below, with concrete build size and runtime footprint attached. "Build size" is an order-of-magnitude estimate for planning purposes — how much there is to build and maintain, not a line-count target.

Component	Build size (approx.)	Runtime footprint	Notes
Screener	Small (a few hundred lines)	Negligible — a list and a text field	No LLM call
Research module	Medium	One LLM call per company; largest input (raw source documents)	Dominates token cost — see Section 4
Investment Thesis module	Medium	One LLM call per company; moderate input (Research Report + Claims)	
Adversarial Report module	Medium	One LLM call per company; re-reads Claims directly, not just the Thesis	
Executive Report module	Medium	One LLM call per company; reads Thesis + Adversarial Report as context, Claims directly for its own analysis	Widest context window of the four
Data layer (SQLite schema + access)	Small–Medium	Always resident, trivial size at this scale	Four record types per architecture blueprint Section 2.2
CLI	Small	Negligible	Orchestrates the five components above in sequence

Realistically: a small, well-scoped codebase — closer to a long weekend-to-few-weeks build for a coding agent working from the architecture and systems-design documents, than a multi-month engineering project. The complexity is almost entirely in getting the no-synthesis discipline (architecture blueprint Section 1.5) right, not in the amount of code.

---

3. Deployment Shape

Inside one machine (a laptop, or one small cloud instance): the CLI orchestrates everything, reading and writing the SQLite file; cron wakes the CLI on a schedule for anything that should run on its own.

Outside that machine, exactly two things: the CLI calls out to the reasoning provider's API, and it fetches from the primary source feeds. Both are external and read/call-only — nothing writes back into either one. The operator's only touchpoint is the CLI itself, for approval and review.

Everything the desk owns fits on one machine. The only two things it talks to outside itself are the primary source feeds (read-only) and the reasoning provider (stateless, swappable). Nothing about this shape requires a server that's always on beyond what cron needs to fire — most of the time the machine is doing nothing.

---

4. Weight: Data Volume

Rough sizing at a small, realistic operating pace:

Item	Illustrative size
One company's Source Documents	A handful of filings, typically low tens of MB in total (raw documents, not extracted text)
One company's Claims	Roughly dozens to a couple hundred discrete assertions
One company's four-report set	A few hundred KB of text, total
Database at 25 companies covered	Comfortably under 1 GB, likely well under
Database at 100 companies covered	Still a single-digit-GB SQLite file — no infrastructure change needed

The data layer does not become a real engineering problem at any scale a single desk is likely to reach. This is one of the advantages of staying deliberately small (architecture blueprint, Section 0, non-goals).

---

5. Cost Model

Two components: a small fixed infrastructure cost, and a variable reasoning cost that scales with how many companies move through the pipeline. The reasoning cost dominates — this is fundamentally a per-cycle API cost, not an infrastructure cost.

5.1 Infrastructure (fixed)

Item	Illustrative monthly cost
Compute	$0 (own laptop) to roughly $10–20 (small cloud instance)
Storage/backup	Negligible at this data volume
Fixed total	**~$0–20/month**, regardless of volume

5.2 Reasoning (variable, per company cycle)

An illustrative token model for one full cycle (one company, all four reports). Actual token counts depend heavily on how long the primary source documents are; the ratios between stages are the useful part, not the absolute numbers:

Stage	Illustrative input tokens	Illustrative output tokens
Research	50,000–150,000 (raw source documents)	3,000–5,000
Investment Thesis	5,000–10,000 (Research Report + Claims)	2,000–3,000
Adversarial Report	10,000–20,000 (Thesis + re-read Claims)	2,000–3,000
Executive Report	15,000–25,000 (Thesis + Adversarial as context, Claims directly)	2,000–3,000

The formula that stays durable regardless of provider or pricing changes:

```
Cycle cost ≈ Σ (stage input tokens × input rate) + (stage output tokens × output rate),
             summed across Research, Investment Thesis, Adversarial Report, Executive Repor
```

One fully worked example, so the shape of the number is visible — substitute the operator's own provider's current pricing before relying on it. At illustrative rates of ~$3 per million input tokens and ~$15 per million output tokens (roughly current frontier-model pricing at time of writing, not a quote), and using the midpoint of each range in the table above: total input across the four stages is roughly 142,500 tokens, total output roughly 11,500 tokens. That works out to approximately **$0.60 for one complete cycle** — dominated by the Research stage's input volume, as expected.

5.3 Total, worked as a scenario

For a desk running, say, 10 new company cycles a month:

```
Monthly cost ≈ Fixed infrastructure ($0–20)
             + (10 × Cycle cost)
             ≈ $0–20 + (10 × ~$0.60)
             ≈ $6–26 total, at the illustrative rates above
```

Reasoning cost — roughly **$6 for ten cycles** at these illustrative rates — is the only line item that scales with actual research output, and at this volume it stays smaller than the infrastructure it runs on. That's the right place for the cost to live, since it's a direct measure of work actually done, not idle capacity paid for either way.

---

6. What Makes This Cheap

Worth stating plainly, since it's the point of designing it this way: nothing here requires paying for infrastructure that sits idle. A company not being researched costs nothing. A month with fewer cycles costs less. The only fixed cost is the machine itself, and even that's optional if the operator already owns one. This is what "smaller scale, different platform" (architecture blueprint, Section 6) is supposed to buy — a desk an operator can actually afford to run before they've proven anything, rather than one that requires committed infrastructure spend up front.
