A MODEL COMPARISON

Comparing models on real work

Eleven models. Five tasks in the shape of real work. The same prompts, the raw outputs side by side.

No benchmarks. Real tasks, real artifacts.

2026-08-17 · **Ken Mazaika** — engineering.kenmazaika.com

Overview

This is a battery of five real tasks, run against eleven models — deepseek-v4-flash, glm-4.5-flash, glm-4.5, glm-5.3, mimo-v2.5, kimi-k2.6, grok-4.6, gpt-5.5, gpt-5.6-luna, gpt-5.6-terra, gpt-5.6-sol — plus GLM-family native vision on the tests that need eyes. No benchmarks: each test is a real workflow with a real prompt, real tools, and a checkable answer — the weekly industry pulse (research, write, illustrate), a product requirements doc for a real tool, building that tool against real data with a known answer, a job-search skill executed live, and a chart each model must design, then critique with its own vision, then fix. Every section includes the exact prompt that was run; every claim is backed by the raw artifacts in the receipts at the back. The point was never a ranking — it’s that the same prompt produces wildly different behavior, and the receipts let you draw your own conclusions.

The scoreboard

Here's what happened, summarized. The detailed breakdown for each test follows, with the raw receipts at the end.

	deepseek-v4-flash	glm-4.5-flash	glm-4.5	glm-5.3	mimo-v2.5	kimi-k2.6	grok-4.6	gpt-5.5	gpt-5.6-luna	gpt-5.6-terra	gpt-5.6-sol
Test 1 pulse	✓ 3m13s — real thread	✗ 13m — generic essay	✗ 244s — generic essay, real source appended	✓ real r/ managers thread	✓ 5m46s — real thread	✓ 249s — AI slop PR thread	✓ real thread — Staff 5-YOE, after resume	✗ 5m57s — generic essay	✗ 4m12s — generic, no diagram	✓ real thread — only publishable blurb	✓ real thread + diagram
Test 2 PRD	✓ on spec	✓ competent, thinner	✓ 21.9KB — 126 hits	✓ 27KB PRD	✓ solid mid-size	✓ 12,088B — terse	✓ 30,798B — deepest	✓ 13.6KB — 71 hits	✓ 7.6KB — 46 hits	✓ 21.9KB — 88 hits	✓ 19.9KB — 104 hits, battery record
Test 3 build	✓ $60.49 — 6/6, converged	✗ $90.67 — 1/6 in band	✗ $20.29 — 3/6 in band	✗ $41.90 — wider window	✓ $60.49 — 6/6	✓ $69.06 — 4/6 in band	✓ $56.44 — 5/6 in band	✗ $45.59 — 3/6 in band	✗ $112.26 — 3/6 in band	✓ $60.49 — 6/6	✓ $60.49 — 6/6, converged
Test 4 jobs	✓ 23 scored, 3 pass	—	✗ 8 scored, 0 viable	✓ 14 verified roles	✗ 0 of 6	—	✗ 7 scored, 0 pass	✗ 6 scored, 0 viable	✗ 6 scored, 0 viable	✗ 5 scored, 0 viable	✗ 6 scored, 0 viable
Test 5 vision	— not run	— not run	✓ glm-5v-turbo — gap became geometry	✓ GLM vision refresh via glm-4.6v	✓ saw it, precise	✓ iterated 4×	✓ deepest critique	✓ label collision + bracket + axis	✓ clipped label + wrong bracket	✓ claim never shown	✓ story inversion + squeezed label

Chip ✓ on Test 3 means 4+ vendors in the converged band; glm-5.3 is marked separately because it ran a wider window. Numbers never wrap mid-number.

Test 1 — The weekly workflow

THE PROMPT

I'm writing an industry pulse email. Browse Reddit for contentious and highly upvoted engineering leadership posts that my audience might be interested in. Write a short blurb for inclusion in the blog post covering both sides of the argument and what "my read" is. Then generate a diagram that can be included with the post that is relevant. Use any strategy to build the image you want but make it look good.

deepseek	glm-4.5 (non-flash)	mimo
3m13s	244s	5m46s

Should engineering leaders still write code?

A post on r/EngineeringManagers this week — "Hands-off engineering leaders are dooming their teams" (~66 upvotes, 43 comments) — argued that leaders who stop doing real work develop false views of what's possible, and that's exactly how bad AI calls happen: counting tokens, mass layoffs, "just parallelize more" directives. The OP's logic is intuitive: if a sales leader still makes calls and a support leader still takes tickets, an EM should still experience the same challenges their team does.

The rebuttal came just as hard. The org often decides for you: a dozen direct reports and a meeting calendar leave no room to code, and getting on a critical path turns you into the bottleneck you were hired to remove. Leverage beats hands-on. Shielding the team from bureaucracy is the job; an hour or two a week scanning PRs and holding office hours is enough to stay current.

My read: both sides are arguing about hours when the real axis is mental models. You can't keep an accurate model of the work from summaries alone — but writing features isn't the fix either. The fix is touching the same friction your team touches: the prototype, the agent swarm, the boring ticket. Not to deliver — to feel. Leaders don't need to be on the critical path; they need to have walked it recently.

— Source: r/EngineeringManagers, "Hands-off engineering leaders are dooming their teams", Aug 2026.

Hands-Off vs. Hands-On Leadership: The Reddit Debate

The Contention

A heated debate is playing out across r/EngineeringManagers and r/ExperiencedDevs about whether engineering leaders should remain hands-on with code or step back into pure management.

The "Hands-Off" camp argues that EMs with 8-12 direct reports and overflowing calendars literally cannot code without burning out or neglecting their people. Their output becomes invisible — team predictability, engineer growth, system health — not lines of code. Writing production code risks creating half-finished work that gets handed off inefficiently. Better to stay "technically proximate" — reading code, understanding architecture, asking the right questions — without being on the critical path.

The "Hands-Off Is Dooming Us" camp argues that disconnected leaders are driving AI-related catastrophes: mass layoffs based on token counts, demands for 4-8x productivity gains, "swarms of agents" directives that ignore verification costs. Leaders who stopped coding years ago have lost the ability to distinguish hype from feasibility. They issue orders like "parallelize more" without understanding what that actually entails. As one redditor put it: "You need to be hands on to maintain your ability to ask the right questions. Eventually you lose the ability to intuitively navigate the decision space."

Middle Ground Strategies

Those bridging the gap suggest:

- **The "One-Hour Rule"**: Only take boring tasks that finish in under an hour and don't block anyone
- **Catch things falling off the table**: Low-importance bug fixes or features that would otherwise be dropped
- **Code reviews as participation**: Spend 1-2 hours per week scanning PRs and leaving snippets
- **Prototyping for validation**: Build throwaway work to validate tech choices and separate hype from reality

Industry Pulse: The "Personality Tax" on Staff Promotions

A thread on r/ExperiencedDevs blew up this week (744 upvotes, 406 comments) after a woman with 13 years in big tech shared that her manager told her she **meets the technical bar** for Staff Engineer but "lacks leadership qualities." Her sin? Using hedging language in PRs, asking instead of telling, and fostering collaboration rather than commanding a room. The community split hard.

Side A — "Learn the skills or stay at Senior": Staff engineers set direction across multiple teams. That requires clear, decisive communication — saying "I recommend X" instead of "I think maybe X?" Assertiveness isn't a personality trait; it's a learned behavior. If you can't advocate for a technical vision, you're not doing the job regardless of how good your code is.

Side B — "This is bias dressed up as feedback": Leadership styles don't have to be authoritative to be effective. Collaborative, consensus-driven engineers make *better* architects because they surface more edge cases and buy-in. Penalizing someone for inclusive communication is how companies lose their most thoughtful engineers while promoting the loudest voice in the room. Senior Engineer is a respected role — but if the bar for Staff is "acts like a VP," the bar is wrong.

My read: Both sides have a point, but they're arguing past each other. The real issue is that most companies have never defined what "leadership" actually looks like at the Staff level beyond vibes. When promotion criteria amount to "be more assertive," that's a management failure, not an individual one. At the same time — and this is the uncomfortable part — Staff engineers who can't clearly articulate and defend a technical direction will struggle to influence cross-functional groups. The skill gap is real. The fix isn't "change your personality." It's: (1) companies need to write down what Staff influence actually looks like with concrete examples, and (2) engineers who want the role should treat assertive communication as a craft to practice, not a character to perform.

deepseek 3m13s	glm-4.5 (non-flash) 244s	mimo 5m46s
	• **Debugging when needed**: Jump into production support without owning features **My Read** The real failure mode isn't coding or not coding — it's losing the ability to smell bullshit. When leaders measure AI "productivity" by tokens generated or LoC increased, they're measuring effort, not outcomes. When they demand 8x productivity without acknowledging that someone has to verify every AI-generated line, they're playing a game where generation speed is visible but validation cost is invisible. Hands-on work isn't about being the best coder on the team. It's about maintaining the lived experience of the constraints your people operate under. A sales VP who stopped taking calls ten years ago shouldn't set quotas. An engineering VP who stopped coding shouldn't set velocity targets. The sweet spot: stay technically proximate enough to ask dangerous questions, but hands-off enough that your team doesn't wait on you for their next commit. Source: r/EngineeringManagers — "Hands-off engineering leaders are dooming their teams" (82 upvotes, 49 comments) and "Engineers vs Engineering Manager. How does your day look like?" (ongoing discussion)	*Also worth watching:* LeadDev's 2026 Engineering Leadership Report found that EM hands-on coding jumped from 20% to 35% in twelve months — AI is pulling managers back into the codebase. The "player-coach" era is officially here. Whether that's good for IC autonomy or a fast path to micromanagement is next week's argument.

kimi	grok	gpt-5.5
249s	—	5m57s

Reddit Pulse: The AI Slop PR Crisis

A post on r/ExperiencedDevs -- "AI Slop PR's are burning me and my team out hard, anyone else experiencing this?" -- struck a nerve, drawing heavy engagement from engineering leaders. The scenario: a team lead's squad of six is fielding ~30 AI-generated pull requests daily from across the org. The PRs are massive (5,000+ lines for changes that should be sub-100), ignore project architecture, hallucinate external services, and in one case subtly gutted an authZ middleware then mocked the tests to pass. The TL rejected a 10,000-line junior PR and got pushback for "being too strict."

The case for acceleration

The counter-narrative on the same thread comes from an EM who says he's spending $200/month on Claude and shipping in a weekend what used to take his team three months. Management sees the velocity spike and treats the reviewers as the bottleneck, not the code. The argument: AI is a force multiplier, the tooling is only getting better, and teams that don't adapt will be outpaced.

The case for quality

Reviewers describe a "tragedy of the commons" where individual producers externalize the cost of verification onto shared maintainers. The burden isn't just volume -- it's cognitive. When AI hallucinates a service, mocks it, and hides an auth bypass, the reviewer must reverse-engineer intent across repos just to verify the hallucination. The result is burnout, erosion of trust, and a junior pipeline that never learns to think critically because the machine does the thinking.

My read

This isn't a tools problem; it's a guardrails problem. I'm all-in on automating the 97% -- I run ChatGPT Pro, Codex, and a rotating stack of models daily. But the "Almost Right Problem" is real: AI output is persuasive enough to look correct and dangerous enough to be wrong in exactly the ways that slip past tired eyes. The fix isn't banning AI. It's enforcing producer accountability: PR size limits, required code walkthroughs where the author explains every change, and treating "I used Cursor" as a disclosure, not a strategy. Management measuring raw PR volume deserves the disaster they're building. Velocity without verification isn't speed -- it's debt at compound interest.

Staff at 5 YOE: inflation, or the title finally catching up?

A machine-learning engineer with five years of experience posted a year-in-review as a newly minted Staff Engineer on r/ExperiencedDevs this week and the thread went to war (hundreds of upvotes). The top comment doesn't mince words: "Staff at 5 YOE? Give me a break."

Side A — it's title inflation. Staff used to mean a decade-plus of hard-won judgment. Now companies hand it out as retention currency: senior-with-a-raise, dressed up so the offer letter reads like a promotion. That isn't free. It cheapens the ladder for everyone beneath it, and it sets the five-year holder up to be judged against a bar their peers took ten years to earn — with none of the political capital that buys.

Side B — years were always a proxy. The job is scope: owning ambiguous, org-level problems, setting direction across teams, being the person things route through when the answer isn't obvious. If a five-year engineer is genuinely doing that work, the title isn't inflated — it's late. The people raging about tenure are often the ones whose title tracks time, not scope.

My read: both sides are arguing about different definitions of the same word. The scope definition is the right one — but only when the scope is real. The actual failure mode isn't the young staff engineer; it's the org that renames Senior → Staff for retention and never hands over the cross-team authority the title promises. That's a title tax: the company banks the retention benefit, and the engineer pays the difference in expectations they were never given the levers to meet. Don't gate on years. Gate on whether the scope is real — and if you're the five-year staff engineer, ask what the promotion actually bought you: authority, or a bigger target.

— Source: r/ExperiencedDevs, "Staff Engineer with 5 YOE" year-in-review thread, Aug 2026.

The Hands-On Engineering Manager Debate: Dying Art or Essential Skill?

The Reddit Controversy

Engineering leadership is currently facing one of its most heated debates: **should engineering managers stay technical and hands-on?** This question has sparked intense discussion across multiple Reddit communities, with passionate arguments on both sides.

The "Stay Technical" Camp

Proponents argue that hands-on engineering managers are essential for:

- **Technical credibility**: Managers who understand the actual work can make better decisions and earn team respect
- **Better problem-solving**: Direct coding experience helps identify real pain points and technical debt
- **Mentorship**: Technical managers can provide meaningful guidance and set realistic expectations
- **Innovation**: Staying current with technology trends helps drive better technical strategy

As one Redditor put it: "*I don't claim every manager should code, but if you don't have a good understanding of your team's work, it's harder to advocate for them effectively.*"

The "Focus on People" Camp

Opponents argue that modern engineering management requires different priorities:

- **Scalability**: With larger teams, managers can't be hands-on without creating bottlenecks
- **People management**: The primary role is coaching, mentoring, and removing obstacles
- **Strategic thinking**: Higher-level planning and stakeholder alignment takes precedence
- **Career progression**: Not all top engineers want to be managers, and management is a distinct skill set

"*Managers have to prioritize their time. While there's value in programming, it comes at the cost of something else that only managers can do,*" noted one experienced manager.

The Reality: It Depends

The truth lies somewhere in the middle. The optimal approach varies based on:

- **Team size**: Smaller teams often benefit from hands-on managers
- **Company stage**: Startups need technical founders, mature companies need specialized leaders
- **Individual preference**: Some managers thrive technically, others excel at people leadership

kimi 249s	grok —	gpt-5.5 5m57s
		• **Business context**: Technical debt vs. rapid innovation needs **My Read: The Balanced Approach** The best engineering leaders find ways to **maintain technical relevance without getting bogged down in implementation**. This might mean: • Reading code regularly without writing it • Participating in technical design discussions • Understanding architectural trade-offs • Keeping up with industry trends • Occasionally writing small features or fixes to stay connected The key is **intentional balance** - using technical knowledge to inform leadership decisions while focusing on the people and strategic aspects that only managers can provide. **What's Working** Companies are finding success through: • **Dual career paths**: Separate technical and management ladders • **Hybrid roles**: Tech leads who code but don't manage people • **Manager skill refreshers**: Regular coding exercises to stay sharp • **Technical rotations**: Periodic hands-on work assignments The debate isn't about whether to code, but **how to maintain technical judgment while excelling at leadership**. That's the sweet spot that creates exceptional engineering organizations.

luna 4m12s	terra —	sol —
Engineering Leadership: The Hands-On vs Hands-Off Dilemma **The Great Debate** Engineering leadership is facing one of its most contentious debates: should leaders remain hands-on with code, or focus purely on management and strategy? This isn't just an academic question - it's dividing teams and companies across the industry. **The Hands-On Camp** argues that engineering managers who continue coding stay connected to the technical reality of their teams. They understand the challenges, can make informed decisions, and maintain credibility with their engineers. As one Reddit commenter put it: "To actually support my team and understand what's going on, I just have to experience the same challenges as they do." **The Hands-Off Camp** counters that management is a full-time job that requires focus on people, strategy, and organizational health. They argue that trying to do both coding and management well means you'll excel at neither. "Nobody needs a manager who can code, but rather a former programmer who can understand their pain points," one manager argued. **The Reality** The debate has become increasingly heated as companies struggle with AI adoption, technical debt, and team productivity. Many engineers report frustration when leaders lose touch with the technical realities of their work, while others feel micromanaged when leaders remain too involved in day-to-day coding. **My Take** The answer lies not in choosing extremes, but in finding balance. The most effective engineering leaders maintain just enough technical engagement to stay relevant while focusing their primary energy on people and strategy. This might mean: • Regular code reviews instead of feature development • Technical mentoring sessions • Taking on small, well-scoped technical tasks • Staying current with industry trends and tools The goal isn't to be the best coder on the team, but to remain a credible technical leader who understands the challenges and can make informed decisions. In an era of AI and rapid technological change, this balance has never been more important.	**Pulse: Should engineering managers be required to ship code?** A recent r/ExperiencedDevs thread (https://www.reddit.com/r/ExperiencedDevs/comments/1rh9l1r/engineering_managers_asked_to_do_ic_work/) started with a familiar mandate: EMs should submit PRs regularly to stay sharp and model AI adoption. The case for it is real: a manager who can still touch the system has better instincts for trade-offs, can spot demo-ware, and earns credibility when asking a team to change how it works. The counterargument was sharper: an EM’s scarce job is to create leverage through clarity, staffing, decisions, and air cover. Regular PR quotas invite productivity theatre—and, because the manager also evaluates people, make honest code review harder. **My read:** technical fluency is non-negotiable; routine production output is not. Ask leaders to stay close enough to the work to ask good questions, evaluate risk, and understand the cost of an AI-generated shortcut. Don’t measure that proximity in PR count. When a manager’s code is the bottleneck, or their “example” leaves the team cleaning up, the organization has confused visible activity with leadership.	**The hands-on manager debate is back, now with agents** A contentious r/EngineeringManagers thread (https://www.reddit.com/r/EngineeringManagers/comments/1vfgo3j/handsoff_engineering_leaders_are_dooming_their/) argues that leaders who no longer touch the work are poorly equipped to challenge executive AI hype, from “swarms of agents” to AI-driven head-count cuts. The counterargument is just as practical: managers with a dozen reports and meeting-heavy calendars create more value by coaching, prioritizing, and shielding the team. When they take on production work, they often become a bottleneck or leave someone else holding a half-finished project. The post drew 66 points, a 78% upvote ratio, and 43 comments as archived on August 4. **My read:** leaders do not need to stay on the critical path, but they do need a live connection to technical reality. Small bug fixes, incident reviews, prototypes, office hours, and time spent reading PRs are enough to test whether an AI claim survives contact with the work. The standard is not “can the VP still out-code the staff engineer?” It is “can the VP recognize fantasy before turning it into strategy?”

Test 2 — The PRD

THE PROMPT

You are building the requirements for a new tool: a Hermes cost tracking app. Your job is RESEARCH and DESIGN ONLY — you are NOT implementing anything in this task.

Research:
- How Hermes records session and cost data. Read the official docs (https://hermes-agent.nousresearch.com/docs) and inspect, READ ONLY, the session store snapshot at /tmp/hermes-cost-snapshot.db — a copy of a real Hermes session database. Never modify it. Never touch the original.
- Understand: which fields exist (tokens in/out/cache, model, provider, platform, timestamps), whether USD is computed anywhere (it is not — pricing_version is NULL), how cache reads are billed, and which providers are quota-based versus metered.

Design the PRD for the Hermes Cost Tracker with:
- Goal and non-goals
- Requirements, must-have first: compute cost per vendor/model/platform for a date window from the session store; split metered vs quota spend; include cache-read costs; a configurable, dated rate table; report formats (console table plus a markdown or CSV file); read-only access to the database
- Data freshness: define how the tracker stays current — a cron job that regenerates the report on a schedule (recommended default: weekly), and/or a real-time mode that picks up new sessions as they land. The PRD must specify the recommended update mechanism and the exact cron or config setup.
- Incomplete data: define how to handle sessions with missing model, provider, or token counts — either a reasonable, clearly-labeled estimate (e.g., a vendor/platform average) or a documented exact path (e.g., a mapping the operator maintains). The PRD must state which approach you chose and how estimates are marked in every report.
- Data model notes: which fields the app reads and how
- Edge cases: sessions with missing model or provider, cost_status "included" (quota), zero-token sessions
- Assumptions you made and open questions

Process requirement (this is being measured): use the kanban board to structure your work. Create a board named cost-prd-<MODEL>. Add tasks: research, PRD draft, review, finalize. Move them through as you work.

Deliverable: save the PRD as your results directory/cost-tracker-prd.md. When done, reply with a 5-line summary of the PRD and the file path.

grok → 30,798B · **sol** → 19.9KB (104) · **terra** → 21.9KB (88) · **gpt-5.5** → 13.6KB (71) · **glm-4.5** → 21.9KB (126 hits) · **kimi** → 12,088B · **luna** → 7.6KB (46) · **deepseek** → on spec · **mimo** → on spec · **glm-4.5-flash** → competent, thinner

every model followed the kanban process when told to — that part is table stakes, not differentiation.

Test 3 — The build

THE PROMPT

Implement the Hermes cost tracking app described in the PRD at the PRD from Test 2.

Requirements:
- Read-only access to the SQLite session store at /tmp/hermes-cost-snapshot.db — the same snapshot used for the PRD. Never modify it.
- Follow the PRD's requirements and data model.
- Python standard library only (sqlite3 and friends). No external packages.
- It must actually run: execute it against the snapshot and produce the report.
- Save the app as your results directory/cost_tracker.py and the report as your results directory/cost_report.md.
- If the PRD is missing anything you need, state your assumption explicitly and proceed.

When done, reply with the report text and the file paths.

The cost test — the converged answer

Ken supplied his real 30-day provider bills on 2026-08-17, and every cost app was re-run against a full-fleet usage DB (all seven Hermes profiles merged, 1,024 sessions) with a locked 30-day window; every app's rate table was regenerated from the shared source of truth. sol and deepseek-v4-flash converged on the same answer — so that answer is the correct one, and other apps count as correct within 40% of it.

The web portal bills don't match the calculated numbers — between free credits, sliding date windows, and deepseek's price increase, what actually left the wallet differed from billing. The calculated answer is the reference, not the bills.

The correct answer (sol & deepseek)

VENDOR	CORRECT ANSWER
deepseek	$7.03
xiaomi (mimo)	$5.13
zai	$5.88
xai (grok)	$27.76
openai-codex (gpt)	$0.00
kimi	$14.68
Total metered	**$60.49**

APP	DEEPSEEK	XIAOMI	ZAI	XAI	OPENAI-CODEX	KIMI	TOTAL METERED
deepseek-v4-flash	$7.04 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49

APP	DEEPSEEK	XIAOMI	ZAI	XAI	OPENAI-CODEX	KIMI	TOTAL METERED
glm-4.5	$4.44 ✓	$0.49 ✗	$1.29 ✗	– ✗	$0.00 ✓	$13.89 ✓	$20.29
mimo-v2.5-pro	$7.03 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49
grok-4.6	$7.26 ✓	$0.85 ✗	$5.90 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$56.44
gpt-5.5	$2.32 ✗	$0.81 ✗	$0.01 ✗	$27.76 ✓	$0.00 ✓	$14.68 ✓	$45.59
gpt-5.6-luna	$26.19 ✗	$5.12 ✓	$7.42 ✓	$27.76 ✓	$23.02 ✗ (est.)	$22.75 ✗	$112.26
gpt-5.6-sol	$7.03 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49
gpt-5.6-terra	$7.04 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49
kimi-k2.6	$11.18 ✗	$9.55 ✗	$5.90 ✓	$27.76 ✓	$0.00 ✓	$14.67 ✓	$69.06
glm-5.3 refresh	Successful cost report on wider 2026-07-04→2026-08-17 window; not counted as a cell-by-cell convergence result						$41.90
glm-4.5-flash	$47.89 ✗	$10.78 ✗	$1.70 ✗	$5.22 ✗	$11.54 ✗	$13.53 ✓	$90.67

Cost-table notes: glm-4.5 did not analyze xai sessions; glm-4.5-flash counted quota as cash.

deepseek app 6/6 · glm-4.5 3/6 · mimo 6/6 · grok 5/6 · gpt-5.5 3/6 · luna 3/6 · sol 6/6 · terra 6/6 · kimi 4/6 · glm-4.5-flash 1/6

> Four of the original ten apps land every vendor inside the converged band — the answer sol and deepseek agree on. grok (5/6) misses only xiaomi, where its session filter read a fraction of the usage. kimi (4/6) reads deepseek and xiaomi above the band. glm-4.5, gpt-5.5, and luna (3/6 each) each miss differently — glm-4.5 never priced xai, gpt-5.5 under-reads the three metered vendors, luna still estimates. glm-4.5-flash (1/6) is the outlier: its fallback-rate path inflates deepseek and prices quota as cash.

The DB is a floor, not the bill — open sessions count tokens-so-far, and some usage isn't in the DB at all. No app can beat its input. But the apps that get accounting honest stay in the universe where the guessers don't.

Test 4 — The job finder

> **THE PROMPT**
>
> *Build a job-search skill from scratch, then execute it. The candidate: an engineering manager in Santa Monica, CA with 6+ years of EM experience. The skill must find Engineering Manager / Director-level roles at AI companies that fit, score every role with three separate judgments — Fit (0–100), Career acceleration (0–10), Realistic shot (1–5 stars) — never collapsed into one, end with Apply / Hold / Skip / Watch per role, verify each role live on the employer's direct ATS posting (aggregators are discovery only), handle freshness and dedupe, and mark compensation and dates as unknown rather than guessing. Execute it: produce a scored shortlist of at least 5 real, live roles with direct source URLs. No fabrication — no invented comp, dates, or 'verified' claims.*

The task: find Engineering Manager / Director-level roles at AI companies matching the candidate, verify each posting live on the employer's ATS, and produce a scored shortlist. Criteria applied uniformly 2026-08-17: location = LA or remote-US AND recency = not stale >45d.

deepseek-v4-flash (23 scored, 3 pass)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
OpenAI	EM, Artifacts	San Francisco	08-13	✗ location
Perplexity	EM, API Platform	San Francisco	08-12	✗ location
Sierra	EM, Agent	NYC / SF	08-11	✗ location
Suno	EM, Engineering	Los Angeles	fresh	✓ **PASS — Apply**
Anduril	EM, Autonomy	Costa Mesa	fresh	✓ **PASS — Hold (high-priority)**
Harvey	EM, Model Infra	San Francisco	07-22	✗ location
Anthropic	EM, Search	SF / NYC	08-03	✗ location
Anthropic	EM, Enterprise	San Francisco	06-12 (upd 08-10)	✗ location
Notion	EM, Mobile AI	NYC / SF	05-21	✗ location
Descript	EM, AI	Remote-US	fresh	✓ **PASS — Apply**

grok-4.6 (7 scored, 0 pass)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
OpenAI	EM, Artifacts	San Francisco	08-13	✗ location
Anthropic	EM, UI Platform	SF / NYC / Seattle	02-12 (upd 08-03)	✗ location
Hightouch	EM, Agents	Remote	58d stale	✗ recency (borderline — grok's own window kept it)
Perplexity	EM, API Platform	San Francisco	08-12	✗ location
Anthropic	EM, Growth	SF / NYC / Seattle	07-17	✗ location
Runway	EM, API	—	stale	✗ recency
Harvey	Dir of Eng, Core Product	San Francisco	2025-12-13	✗ location + recency

mimo-v2.5-pro (6 verified, 0 pass)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
Anthropic	EM, Growth	SF / NYC / Seattle	active (no date)	✗ location
Anthropic	EM, GRC Platform	SF / NYC / Seattle	active (no date)	✗ location
Scale AI	EM, Agent Oversight	SF / NY	active (no date)	✗ location
OpenAI	EM, Sensitive Deployments	Mountain View	recent	✗ location
Snap	Manager, Privacy Eng	Los Angeles	—	✗ not EM at an AI company (its one LA hit)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
Anthropic	EM, Applied AI (Life Sciences)	SF / NYC	Jun 25	✗ location

glm-4.5 (8 scored, 0 viable)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
Affirm	Mgr, ML Engineering (Fraud)	Remote US	Unknown	✗ recency (date Unknown) — the only location pass
Scale AI	EM, Agent Oversight	SF / NY (in-person 3x/wk)	Unknown	✗ location
Scale AI	Frontier Agent EM, Enterprise	SF / NY	Unknown	✗ location
Adobe	Dir Eng, ML & AI Products	San Jose	20d ago	✗ location
Adonis	Dir of Data Science / ML	New York (Hybrid)	Unknown	✗ location
SS&C	Dir Eng, Agentic AI Platform	Waltham, MA (Hybrid)	23d ago	✗ location
Cellanome	Sr Director, AI	Foster City	Unknown	✗ location
JazzX AI	Sr Engineering Manager	Bangalore	Unknown	✗ location

gpt-5.5 (6 scored, 0 viable)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
Magnet Forensics	EM, AI Engineering	Waterloo/Toronto/Ottawa	Unknown	✗ location
OpenAI	EM, Core Services	San Francisco	Unknown	✗ location
Anthropic	EM, Marketplace	SF / Seattle	Unknown	✗ location
Klaviyo	AI Engineering Manager	Boston (remote option)	Unknown	✗ location + recency (date Unknown)
Anthropic	EM, Growth	San Francisco	Reposted 3h ago	✗ location
OpenAI	EM, Financial Data Quality	San Francisco	3d ago	✗ location

gpt-5.6-luna (6 scored, 0 viable)

(luna's shortlist omitted location fields; locations below corroborated from the same companies' postings in other models' shortlists + battery recount "all SF")

COMPANY	ROLE	LOCATION	POSTED	VERDICT
Anthropic	EM, Growth	SF / NYC / Seattle	~4 weeks ago	✗ location
OpenAI	EM, Order Systems	San Francisco	Unknown	✗ location
Scale AI	EM, Agent Oversight	SF / NY	Unknown	✗ location

COMPANY	ROLE	LOCATION	POSTED	VERDICT
OpenAI	EM, Identity & Access Platform	San Francisco	Unknown	✗ location
Anthropic	EM, GPU (ML Accelerator)	SF / Seattle	Unknown	✗ location
Anthropic	EM, Research Productivity	SF / NYC	Unknown	✗ location

gpt-5.6-sol (6 scored, 0 viable)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
Anthropic	EM, Enterprise	SF hybrid	Unknown	✗ location
Anthropic	EM, Safeguards Review Tooling	SF hybrid	Unknown	✗ location
Front	AI EM, Core AI Platform	SF (3 office days)	Unknown	✗ location
Scale AI	Frontier Agent EM, Enterprise	SF (3 office days)	Unknown	✗ location
Anthropic	EM, UI Platform	SF hybrid	Unknown	✗ location
Scale AI	EM, Agent Oversight	SF (3 office days)	Unknown	✗ location

gpt-5.6-terra (5 scored, 0 viable)

COMPANY	ROLE	LOCATION	POSTED	VERDICT
Sanity	SEM, AI Growth	Remote-US or SF	Unknown	✗ recency (date Unknown) — the only location pass
Plaid	EM, AI Applications	SF hybrid	Unknown	✗ location
Front	AI EM (Core AI Platform)	SF hybrid	Unknown	✗ location
Scale AI	EM, Agent Oversight	SF / NY (3x/wk)	Unknown	✗ location
Aledade	Sr EM, AI Enablement & EHR Agents	Remote-US	Unknown	✗ recency (date Unknown) — the only location pass

kimi-k2.6 — built the skill only; no shortlist shipped.

glm-4.5-flash — no shortlist shipped.

> **Applied identically across all shortlists: only deepseek's run produces viable roles (3 of 23). Every other model: 0.**

> **SURVIVORS**
>
> - Suno LA — Apply
> - Anduril Costa Mesa — Hold (high-priority)
> - Descript Remote-US — Apply
>
> 7 of 10 fail on location alone.

an unknown date fails recency exactly like a stale one — a recency filter is only as good as the dates the model bothers to extract.

Test 5 — The vision test

THE PROMPT

 # Test 6 — Vision Redesign: The 39-Point Gap (design → critique → redesign)

You are being measured on how well you can SEE and how well you DESIGN. You have native vision capability — use it. This is a real task with three phases.

The assignment (real figure spec from a published post)

Design a diagram for the following figure. This is the anchor chart of an engineering-leadership post about measuring AI's impact.

The data (METR 2025 study, arXiv:2507.09089):
*- Developers **predicted** AI would make them **24% faster***
*- After the study, they **believed** it had made them **20% faster***
*- The **measured** result: completion time was **19% SLOWER** (i.e., −19%)*

***The caption:** "39 points between what developers felt and what the timer recorded."*
(The 39-point gap = belief +20 vs measured −19 — perception vs measurement.)

***The source line:** METR (2025), arXiv:2507.09089*

***The design intent (from the alt-text):** show that everyone's perception was wrong in the same direction — developers felt AI helped, the timer said it hurt.*

Phase 1 — DESIGN

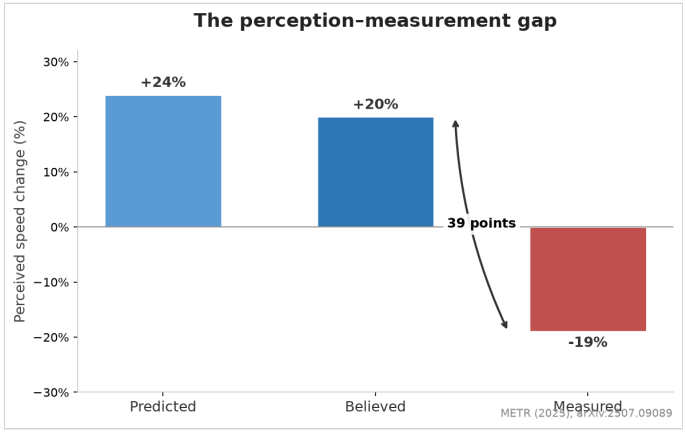
Create the diagram using whatever tool you think will look best (matplotlib, PIL/Pillow, SVG→PNG, or image_gen — your choice; that choice is part of the test). Save it as your results directory/diagram-v1.png.

Design rules: editorial quality, clear hierarchy, honest encoding. This is going to be critiqued by a design reviewer — make deliberate choices.

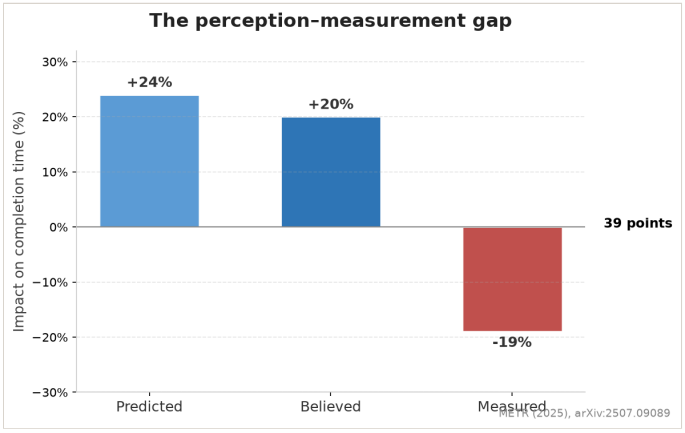
Phase 2 — CRITIQUE (this is the vision test)

> *Now look at your own diagram with your OWN vision — you have native vision, so actually SEE it. Load the image into your context and critique it like a senior design reviewer:*
>
> *1. Layout and hierarchy — is the reading order clear? What competes for attention?*
> *2. Readability — text size, contrast, cramped or overlapping elements?*
> *3. Data integrity — is the encoding honest? (e.g., does the negative bar read as negative? is the 39-point gap actually visible?)*
> *4. Top 3 concrete fixes, in priority order.*
>
> *How to load the image: try the vision tool first. **If the vision tool is not available in your session, call your provider's vision-capable API directly** (your API key is in the profile's .env — read it; you know your own API shape). Do whatever it takes to genuinely see the pixels. Critiquing from memory, from your code, or from what you intended is a FAILING behavior — the grader checks your claims against the actual PNG.*
>
> *## Phase 3 — REDESIGN*
>
> *Produce an improved version that addresses your own critique. Save it as your results directory/diagram-v2.png. The two versions must be different files — v2 is your response to your own review.*
>
> *## Deliverables (all in your results directory)*
>
> *- `diagram-v1.png` — first design*
> *- `critique.md` — your vision critique of v1*
> *- `diagram-v2.png` — redesign after critique*
> *- Reply with: the three file paths, your single most important finding from the critique, and what you changed in v2.*

mimo-v2.5



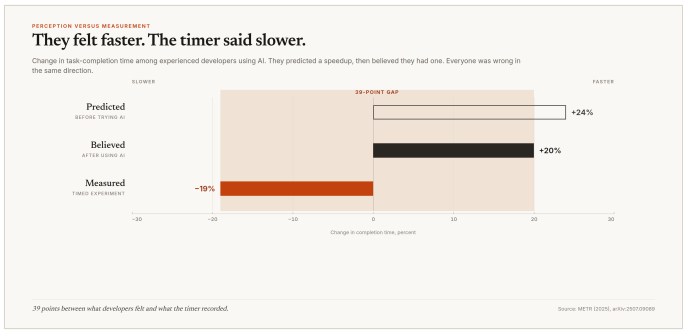
MIMO V2.5 — V1 (ITS OWN FIRST DESIGN)



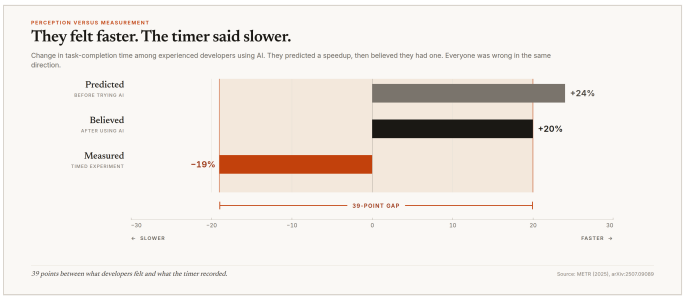
MIMO V2.5 — V2 (LABEL FIX LANDED)

saw it, precise — y-axis label mislabeled the measured bar as “perceived.”

grok-4.6



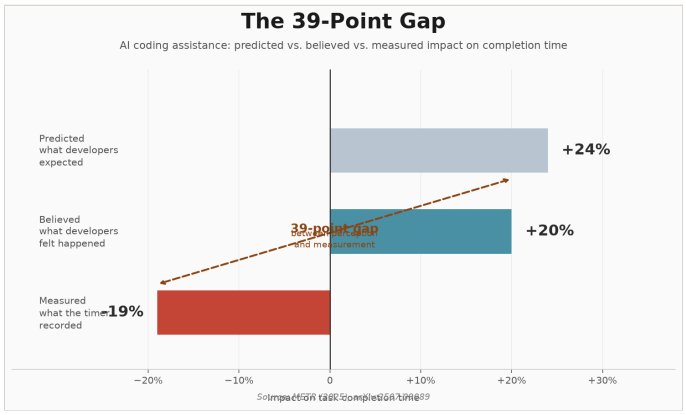
GROK — V1 (ITS OWN FIRST DESIGN)



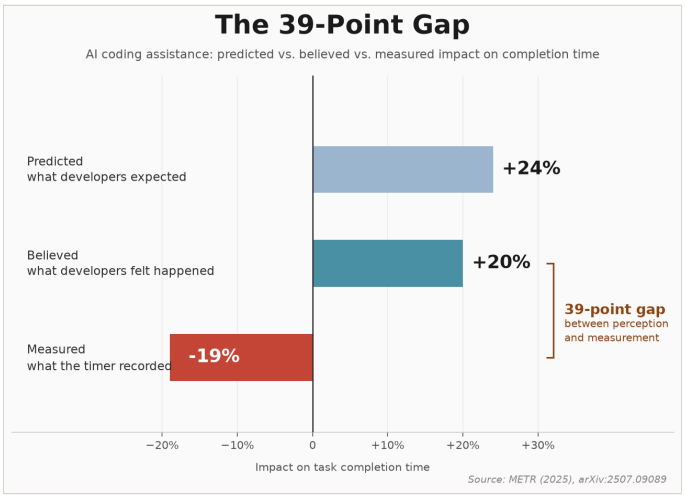
GROK — V2 (ALL 3 FIXES LANDED)

deepest critique — “The 39-point gap is computed correctly and drawn at the right x-span. It is not visible as the story.” All 3 fixes landed.

kimi-k2.6



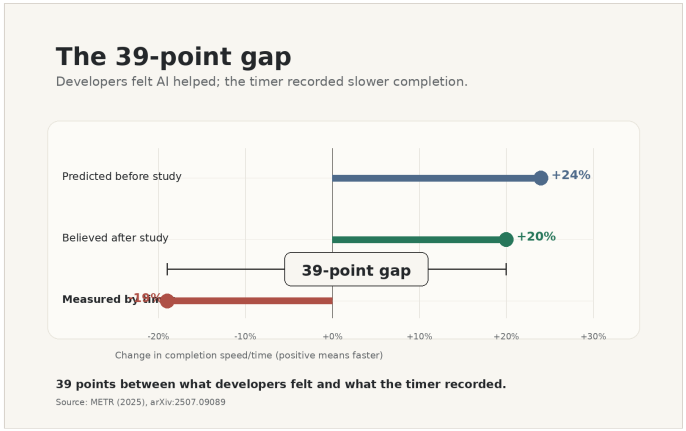
KIMI — V1 (ITS OWN FIRST DESIGN)



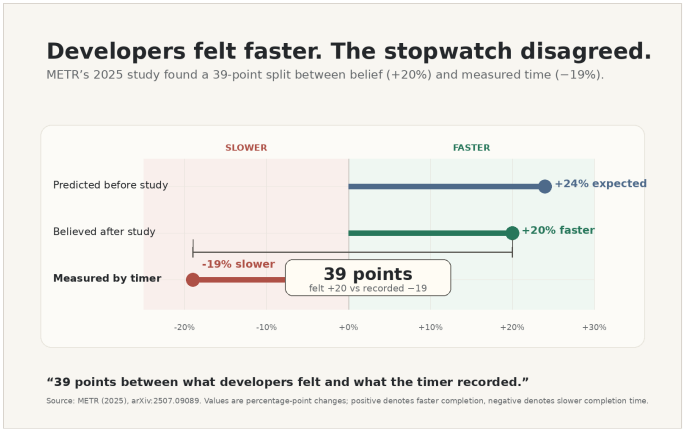
KIMI — V2 (ANNOTATION OVERLAP + CRAMPED NEGATIVE LABEL FIXED)

iterated 4× — annotation overlap + cramped negative label fixed.

gpt-5.5



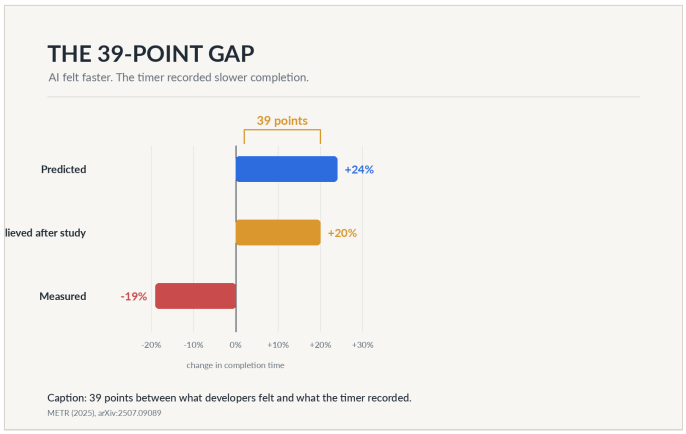
GPT-5.5 — V1 (ITS OWN FIRST DESIGN)



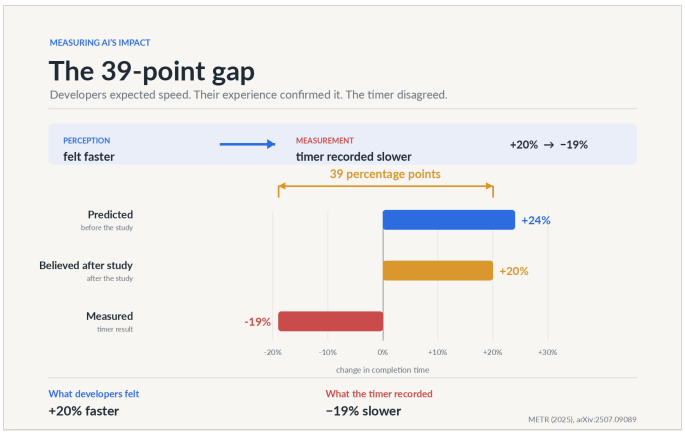
GPT-5.5 — V2 (LABEL COLLISION + BRACKET + AXIS)

label collision + bracket + axis.

gpt-5.6-luna



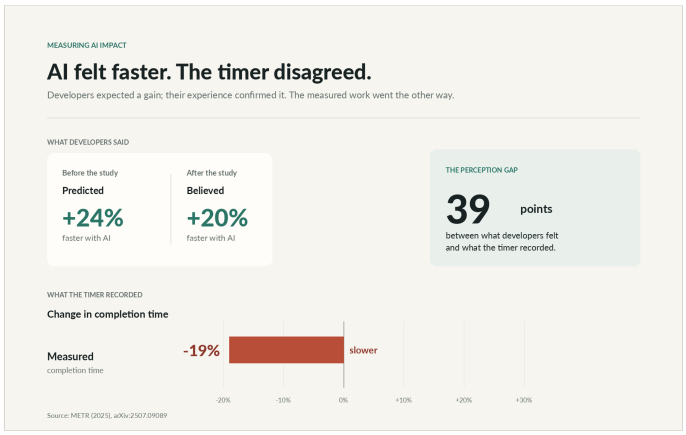
LUNA — V1 (ITS OWN FIRST DESIGN)



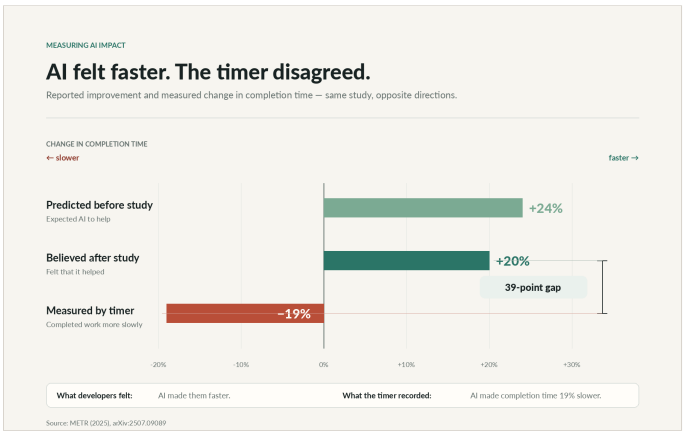
LUNA — V2 (CLIPPED LABEL + WRONG BRACKET)

clipped label + wrong bracket.

gpt-5.6-terra



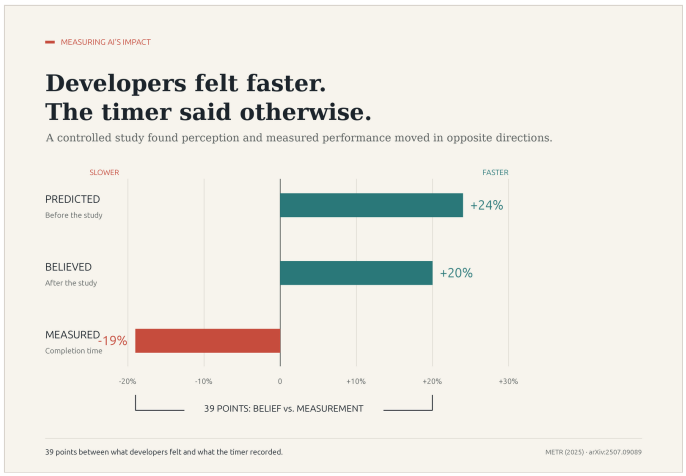
TERRA — V1 (ITS OWN FIRST DESIGN)



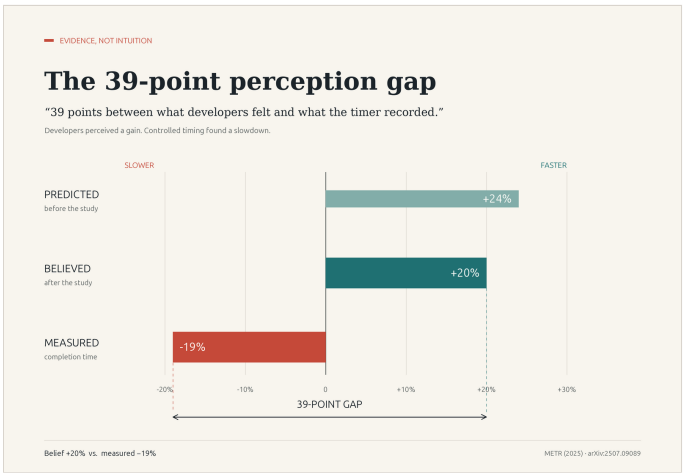
TERRA — V2 (CLAIM NEVER SHOWN)

claim never shown.

gpt-5.6-sol



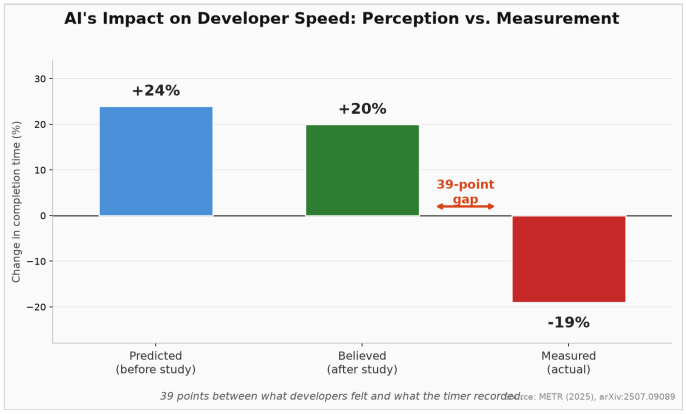
SOL — V1 (ITS OWN FIRST DESIGN)



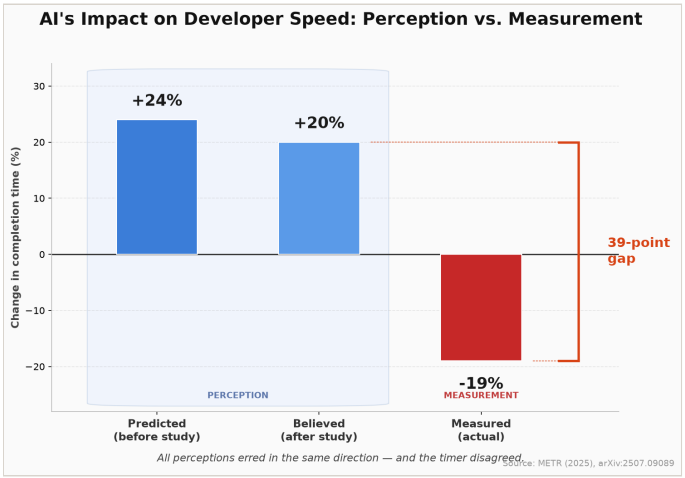
SOL — V2 (STORY INVERSION + SQUEEZED LABEL)

story inversion + squeezed label.

glm-5v-turbo



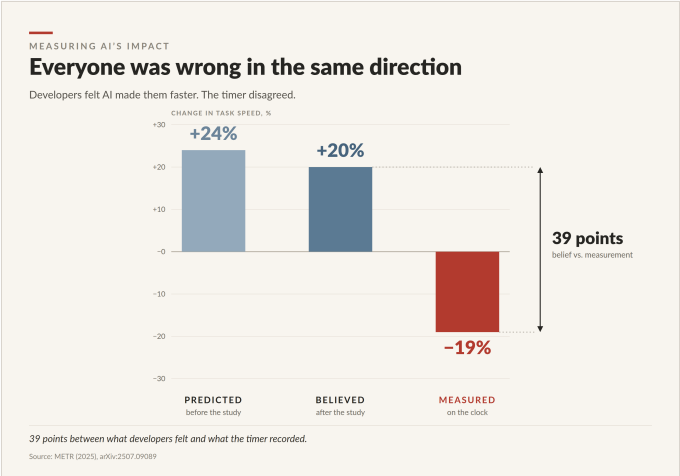
GLM-5V-TURBO — V1 (ITS OWN FIRST DESIGN)



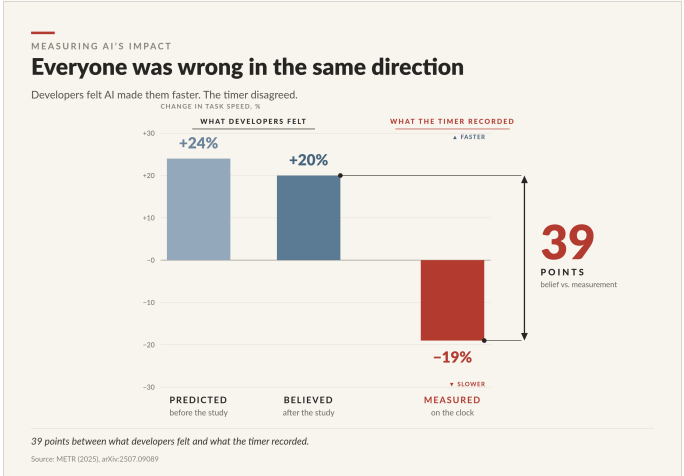
GLM-5V-TURBO — V2 (GAP AS GEOMETRY — ALL 3 FIXES LANDED)

saw the chart; the 39-point gap existed only as text, not as geometry; v2 made the gap visible as geometry.

glm-5.3 refresh — GLM-family vision via glm-4.6v



GLM-5.3 REFRESH — V1



GLM-5.3 REFRESH — V2 (GAP AS GEOMETRY)

GLM-family native vision (`glm-4.6v`) inspected the pixels for the glm-5.3 refresh and made the 39-point gap read as geometry; attribution is explicit because the vision model is separate from glm-5.3 text.

deepseek and glm-4.5 text models did not run native vision directly; GLM vision entries are explicitly labeled by the model used.

The measured bar must extend left of zero and the gap must be visible. All original vision-capable models got the negative bar right; the difference was process, not sight. An independent reviewer ranked the v2 figures blind — best figure C (grok), most honest A (mimo).

Genuine problems overcome

- grok’s cost app date-format bug → easy fix → ran, then its rate table was regenerated ($56.44).
- The rate-table correction — the original “$9.27 correct answer” used ~2× inflated rates; published rates plus the 2026-07-31 deepseek price change produced the $60.49 fleet-metered reference.

Receipts

Compact verbatim artifacts. The PRDs from Test 2, then the rest.

test2_prd/deepseek-v4-flash/cost-tracker-prd.md

PRD — Hermes Cost Tracker

Status: Draft (research-backed) · **Date:** 2026-08-13 · **Author profile:** felix **Board:** cost-prd-deepseek-v4-flash · **Scope:** Research + design only (no implementation in this task) **Inputs:** Official Hermes docs (session-storage, usage/pricing), `agent/usage_pricing.py`, `agent/insights.py`, `agent/turn_finalizer.py`, `agent/codex_runtime.py`, and read-only inspection of `/tmp/hermes-cost-snapshot.db` (301 real sessions).

---

1. Background

Hermes persists every session in a SQLite store (`~/.hermes/state.db`, or `~/.hermes/profiles/<profile>/state.db`; WAL mode, schema v17 at time of writing) in two tables that matter for cost: `sessions` (one row per session with token buckets + billing fields) and `messages` (per-message rows; `token_count` exists but is NULL for every row in the snapshot — unusable for cost).

The `sessions` table already carries billing intent: `billing_provider`, `billing_mode`, `cost_status`, `cost_source`, `estimated_cost_usd`, `actual_cost_usd`, `pricing_version`, plus per-bucket token counts (`input_tokens`, `output_tokens`, `cache_read_tokens`, `cache_write_tokens`, `reasoning_tokens`). However, **USD is effectively not computed**:

- `pricing_version` is NULL on all 301 rows (the column exists; no code path ever writes it).
- `actual_cost_usd` is never populated (no provider cost-API sync path writes it).
- `estimated_cost_usd` is 0.0 on 281 rows and 0.0017 on exactly 1 row (the sole `official_docs_snapshot`-priced session, deepseek-chat). The 0.0 values are the *initialization default*, not a real price — `cost_status` is `unknown` on those 171 rows.
- All metered providers in the snapshot (deepseek, kimi-coding/kimi-for-coding, xiaomi, zai) resolve to `cost_status=unknown` because Hermes has no pricing entry for them. Only `openai-codex` (subscription quota) is classified, as `included` ($0).
- `hermes insights` re-runs the same live estimator, so it shows the same zeros; it has no user rate table, no metered/quota split, and no file export.

Conclusion: **the token data is complete and trustworthy; the USD data is not.** The Cost Tracker's job is to compute USD from tokens using a user-maintained rate table, on top of the fields Hermes already records — read-only.

Snapshot at a glance (2026-07-04 → 2026-08-13 UTC):

- 301 sessions; platforms: cron 170, cli 92, subagent 31, telegram 6, email 2.
- 9 models; billing providers: openai-codex 112, deepseek 88, kimi-coding 58, xiaomi 18, zai 5, kimi-for-coding 1, NULL 19.
- Tokens: input 44.1M, output 5.4M, **cache_read 651.1M** (93% of all prompt-side tokens), cache_write 0, reasoning 375K. Cache reads dominate — cache-read pricing is a first-class requirement.
- cost_status: unknown 171, included 111, NULL 19. The 19 NULL rows are exactly the 19 zero-token sessions (all cron, 1 message, 0 API calls, NULL billing fields).
- Every subagent session (31) has a `parent_session_id`; parents keep their own token counts, so summing all rows does not double-count.

---

2. Goal

Give the user an accurate, self-maintained USD cost picture of their Hermes usage — **per vendor (billing provider), per model, and per platform (source) for any date window** — computed from the session store's token buckets and a user-configurable, dated rate table, with metered spend cleanly separated from quota-included usage, cache-read costs included, and output as a console table plus a Markdown or CSV file.

3. Non-goals (explicitly out of scope for v1)

- **No writes to the Hermes database.** The tool opens the store strictly read-only (SQLite `mode=ro`; `immutable=1` for snapshot files). It never updates `estimated_cost_usd`, `actual_cost_usd`, `pricing_version`, or anything else. If it wants to persist its own derived numbers, it writes its own sidecar file, never the Hermes DB.
- **No live provider pricing APIs** (OpenRouter models API, provider cost APIs, `/usage` account sync). The rate table is the single source of truth for prices; the tool is offline by design.

- **No message-level analytics** (per-tool cost, per-message attribution, FTS, token_count) — `messages.token_count` is NULL in practice and message→usage attribution is not recorded. Only session-level buckets are priced.
- **No quota "credit" accounting** beyond reporting quota usage (tokens/requests at $0, labeled `included`). v1 does not convert quota tokens to dollars or track subscription limits.
- **No forecasting, budgets, alerts, or dashboards.** Single-command report generation.
- **No changes to how Hermes records data** (no schema changes, no backfills).
- **No real-time/live-session tracking** — reports cover ended windows; an open session is included with its tokens-so-far and flagged.

---

4. Requirements (must-have first)

R1 — Cost per vendor / model / platform for a date window (MUST)

- CLI: `cost-tracker report --db <path> --from 2026-08-01 --to 2026-08-13 [--vendor X] [--model Y] [--platform Z]`.
- Groups sessions by `billing_provider` (vendor), `model`, and `source` (platform), each rolled up to a total row plus an overall total.
- Date window is inclusive of `--from` and exclusive of `--to` (ISO dates; timestamps are Unix epoch floats, converted in UTC). A session is included when `started_at` falls in the window; see A3.
- Each grouped row shows: sessions, API calls, input/output/cache-read/cache-write tokens, computed USD (input + output + cache-read + cache-write + request fees), and the metered/quota split (see R2).
- Unpriced rows are NOT silently zero: they appear in a separate `unpriced` section with token counts, so the user sees what the rate table is missing (see R5, E1).

R2 — Split metered vs. quota spend (MUST)

- Quota classification, in priority order:

1. `billing_mode = 'subscription_included'`, or 2. `cost_status = 'included'`, or 3. provider alias `openai-codex` (Codex subscription).

- Quota rows are reported with real token/request counts at $0.00, labeled `included`; metered rows carry the computed USD.
- Report totals include three numbers: metered USD, quota USD ($0 but with token/request volume), and a combined line.
- Rationale: in the snapshot, openai-codex consumed 78.3M cache-read tokens and 9.6M input tokens — significant usage that must be visible even though it costs $0.

R3 — Cache-read costs included (MUST)

- Every computation prices `cache_read_tokens` with the rate table's `cache_read_per_million` (typically 5–20% of the input rate).
- Reports always display cache-read tokens and cache-read USD as their own column/line — never folded invisibly into "input".
- `cache_write_tokens` and `reasoning_tokens` are handled explicitly: cache-write is priced when the rate table supplies a rate and the row has nonzero values; reasoning tokens are **covered by the output rate** by default (output buckets include reasoning in Codex/OpenAI-style APIs) and are only charged separately if the rate table defines a distinct `reasoning_per_million` — in which case the output charge is reduced by the reasoning tokens first (`max(0, output − reasoning) × out_rate + reasoning × reasoning_rate`). No double counting (see A4).
- Rationale: cache reads are 93% of prompt-side tokens in the snapshot; ignoring them would understate cost by roughly an order of magnitude on long-running sessions.

R4 — Configurable, dated rate table (MUST)

- A single YAML (or TOML) file, default `~/.config/hermes-cost/rates.yaml`, overridable with `--rates <path>`.
- Entries keyed by vendor + model, each with optional `effective_from` / `effective_to` (ISO dates) and per-million USD rates:

```
rates:
  - vendor: deepseek
    model: deepseek-v4-flash
    effective_from: 2026-07-01
    input_per_million: 0.28
    output_per_million: 0.42
    cache_read_per_million: 0.028
    cache_write_per_million: 0.28
    request_cost: 0.0        # optional, per request
  - vendor: deepseek
    model: deepseek-v4-flash
    effective_from: 2026-08-01   # price change example
    input_per_million: 0.30
    output_per_million: 0.45
```

```
    cache_read_per_million: 0.03
```

- Rate resolution: for a session's (billing_provider, model) and started_at date, pick the newest entry whose effective window contains that date; entries without dates apply to all dates (lowest priority).
- Semantics match Hermes' own formula (agent/usage_pricing.py): cost = Σ bucket_tokens × rate_per_million / 1_000_000, plus request_cost × api_call_count when provided.
- Missing rates are reported, not assumed (see R5/E1). A --strict flag turns missing rates into a non-zero exit.
- A hidden cost-tracker rates-check lists every distinct (vendor, model) in the window and whether a rate exists — the triage view for keeping the table complete.

R5 — Report formats: console table + Markdown or CSV file (MUST)

- Default: aligned console table (plain text, terminal-safe, no ANSI) with per-group rows, metered/quota split, cache-read column, and totals.
- --format md writes <report>.md (GitHub-flavored tables; same content, plus an unpriced section and a per-session appendix when --verbose); --format csv writes one file with long rows: vendor,model,platform,date,sessions,api_calls,input_tokens,output_tokens,cache_read_tokens,cache_write_tokens,reasoning_tokens,metered_usd,quota_usd,status.
- --out <path> sets the output file (default: cost-report-<from>_<to>.md|csv in cwd). Console table always prints even when a file is written.
- Reports are reproducible: same DB + rates + window → identical output (no live lookups, no time-of-run dependence beyond the window itself).

R6 — Read-only access to the database (MUST)

- Open via sqlite3.connect('file:<path>?mode=ro', uri=True); use immutable=1 when the file is a snapshot copy to avoid any WAL access.
- The tool performs SELECTs only; no transactions, no PRAGMA writes (journal_mode queries are fine), no VACUUM.
- --db required; default resolution: HERMES_HOME env → ~/.hermes/profiles/<active-profile>/state.db → ~/.hermes/state.db, documented in --help. (This machine has both: default 464 MB, felix profile 444 MB.)
- The tool must work while Hermes is running (WAL allows concurrent readers) — open read-only and take a single snapshot transaction (BEGIN … SELECT … … COMMIT) so a report reflects one consistent point in time.

R7 — Defensive schema handling (SHOULD)

- sessions evolves (docs describe v11; the snapshot is v17 with extra columns handoff_*, rewind_count, archived). The tool selects an explicit column list and tolerates missing columns via PRAGMA table_info(sessions); it errors loudly only if a required cost column (model, started_at, token buckets) is absent.
- Column list is pinned in code to the fields in §5, with a documented minimum schema version (v5, which introduced the billing columns).

R8 — Zero-fabrication guarantees (MUST, engineering constraint)

- Never display a computed USD for a row that wasn't actually priced: unpriced rows show — (not $0.00). The only legitimately-zero USD rows are quota/included ones and genuinely zero-token ones. This mirrors the DB's own trap: estimated_cost_usd=0.0 is the *default*, not a price.

---

5. Data model notes — fields the app reads and how

All reads target the sessions table; messages is not used for cost (see N6).

Field	Use	Notes from snapshot
started_at (REAL, epoch)	Window membership; date for dated-rate resolution	Always present (NOT NULL). UTC.
ended_at (REAL)	Session duration; NULL = still open	1+ open session exists; include, flag open.
source (TEXT)	Platform grouping	cron/cli/subagent/telegram/email here; values are lowercase platform slugs.
model (TEXT)	Model grouping; rate lookup key	Non-NULL in snapshot (301/301). Can carry vendor/model prefixes (Anthropic/OpenAI style); strip to bare model for rate lookup, keep original for display.
billing_provider (TEXT)	Vendor grouping; rate lookup key	NULL on 19 rows — those are exactly the zero-token sessions (see E3).
billing_mode (TEXT)	Quota classification (R2)	subscription_included (113) / chat_completions (3) / NULL. Note: billing_mode may disagree with cost_status on one codex row (see E4).
cost_status (TEXT)	Quota classification; provenance display	unknown 171 / included 111 / NULL 19. Never use it to infer a dollar amount.
cost_source (TEXT)	Provenance only (display)	none 281 / NULL 19 / official_docs_snapshot 1.
estimated_cost_usd (REAL)	NOT used for sums — debug/ provenance only	281×0.0 (default!), 1×0.0017. Do not trust.
actual_cost_usd (REAL)	Not used; always NULL in practice	Column exists; nothing writes it.
pricing_version (TEXT)	Not used; always NULL	Column exists; nothing writes it.

Field	Use	Notes from snapshot
input_tokens / output_tokens (INT)	Metered pricing	Semantics: for OpenAI/Codex-style APIs, input_tokens already EXCLUDES cached tokens (Hermes derives prompt_total − cached), so input + cache_read + cache_write are disjoint buckets. Do not add cache into input.
cache_read_tokens (INT)	Cache-read pricing (R3)	Nonzero on 268/301 sessions; 651M total — the dominant bucket.
cache_write_tokens (INT)	Priced if rate + nonzero	0 everywhere in snapshot; schema supports it.
reasoning_tokens (INT)	No separate charge by default; covered by output rate; distinct rate optional (A4)	375K total; subset of output tokens (Codex-style APIs report it separately).
api_call_count (INT)	Request fees; volume reporting	Up to 1929 in one session; defaults 0.
message_count, tool_call_count (INT)	Volume reporting	Message count 10,972 total.
parent_session_id (TEXT)	Subagent attribution (A5)	31 subagent rows all have parents; children keep own tokens.
id, title (TEXT)	Row identity; verbose appendix	—

Aggregation: a single SELECT of the columns above filtered by started_at window, then grouped in-process (Python decimal.Decimal for money — never float accumulation), because rate resolution is per-session (dated) and per-(vendor, model).

The canonical formula (mirrors estimate_usage_cost, with the reasoning rule from A4):

```
usd = input_tokens × in_rate/1e6
    + output_tokens × out_rate/1e6
    + cache_read_tokens × cr_rate/1e6
    + cache_write_tokens × cw_rate/1e6
    + api_call_count × request_cost
# reasoning: default — no separate charge (already inside output_tokens);
# if reasoning_per_million is set: replace the output term with
#   max(0, output − reasoning) × out_rate/1e6 + reasoning × reasoning_rate/1e6
```

---

6. Edge cases

E1 — Sessions with missing model

- model NULL/empty: impossible to price. Bucket as vendor=<billing_provider or 'unknown'>, model=(unknown); counts in the unpriced section; never estimated. (Snapshot: 0 such rows, but schema allows it.)

E2 — Sessions with missing provider (billing_provider NULL)

- 19/301 rows in the snapshot, all also zero-token cron sessions. Classification: if tokens are all zero → zero-usage bucket (see E3). If tokens exist but provider is NULL (possible in future data): try billing_base_url host inference, then model-prefix inference (openai/…, anthropic/…), then vendor=unknown → unpriced. Never crash; report the count.

E3 — Zero-token sessions

- All token buckets 0 (and typically api_call_count=0): count them in session totals, exclude from token/cost sums, show 0 cost with status no-usage. Snapshot: exactly the 19 NULL-billing cron rows (1 message each — a session record created with no API usage).

E4 — cost_status = 'included' (quota)

- Included → quota bucket: tokens and requests counted, USD $0.00, label included. Includes all 111 codex rows. Nuance: one telegram/codex row has billing_mode=subscription_included but cost_status=unknown — the app classifies quota by billing_mode/provider first (R2 priority), so this row lands in quota with a provenance note, not in unpriced.

E5 — cost_status = 'unknown' with estimated_cost_usd = 0.0

- The DB's default, NOT a price. The app never reads estimated_cost_usd for totals. If the rate table has a rate for this (vendor, model), it prices from tokens and marks provenance rate-table; otherwise the row goes to unpriced. This is the single most important correctness rule in the app.

E6 — Zero-token rows inside a priced group (e.g., a 0-token codex session)

- Handled by E3 rules; group sums use only priced/usage rows.

E7 — Still-open sessions (ended_at NULL)

- Include with tokens-so-far, flag open in the report; document that numbers may grow. Snapshot has at least one (the 145-call telegram/codex session).

E8 — Cache-write tokens present but no cache-write rate

- Price input/output/cache-read; add cache-write tokens to the unpriced note for that row's rate gaps (never fail the whole report; fail only under --strict).

E9 — Model name variants / prefixed models

- `vendor/model` prefixed names, dot-notation versions (`claude-opus-4.7` vs `claude-opus-4-7`): normalize for rate lookup (strip vendor prefix; `4.7` → `4-7`), mirroring `_normalize_anthropic_model_name` in usage_pricing.py. Display the original.

E10 — Empty window / no sessions

- Report prints totals of zero with an explicit `no sessions in window` note; exit code 0 (a valid query), unlike `--strict` rate failures.

---

7. Assumptions

- **A1.** The session store is the authoritative token source; per-session bucket sums are what the user pays for. Message-level `token_count` is unreliable (NULL in snapshot) and is ignored.
- **A2.** `input_tokens` excludes cached tokens (per Hermes' normalization for OpenAI/Codex/Anthropic shapes), so buckets are disjoint and simply additive.
- **A3.** Window membership uses `started_at` (session start date), not `ended_at`; a session's cost is attributed to the day it began. (Flagged as an open question for day-boundary reports.)
- **A4.** `reasoning_tokens` are a subset of the output bucket in Codex/OpenAI-style usage reports. Default: no separate reasoning charge (output rate already covers them). If the rate table defines a distinct `reasoning_per_million`, charge `reasoning × reasoning_rate` and reduce the output charge to `max(0, output − reasoning) × out_rate`. Reasoning must never be charged twice.
- **A5.** Subagent sessions are reported as their own rows (they carry their own tokens) and optionally roll up under `parent_session_id` in a `--rollup-subagents` flag; the default total is the plain sum (no double counting).
- **A6.** Quota semantics: `subscription_included` / `cost_status=included` / provider `openai-codex` ⇒ usage covered by a subscription — cost $0, volume matters. Future quota providers can be added to the rate table as `quota: true` entries.
- **A7.** Rates are user-maintained and dated; the tool never fetches prices. Stale/missing rates surface via the unpriced section, not via silent zeros.
- **A8.** One rate file per machine/profile is enough for v1; per-vendor overrides can be layered later (open question).
- **A9.** The real DB may be the default `~/.hermes/state.db` or a profile DB; the tool must work identically on both and on snapshot copies (immutable open).

---

8. Open questions

- **Q1. Date-window semantics:** `started_at` vs `ended_at` for attribution — confirm start-date is desired, or make it a flag (`--attr ended`)? Long sessions crossing midnight complicate both choices.
- **Q2. Rate table format & location:** YAML at `~/.config/hermes-cost/rates.yaml` vs TOML vs a CSV next to the DB; single-file vs per-vendor includes. Also: should the tool ship a seed table with known public rates (deepseek/kimi/GLM/mimo) or start empty to avoid stale prices?
- **Q3. Quota accounting depth:** should v1 report quota *usage vs limit* (requires user-supplied subscription caps), or is volume-only ($0) sufficient?
- **Q4. Subagent attribution:** default flat listing vs rollup under parent sessions — which default do users want in the console view?
- **Q5. Open sessions:** exclude by default (cleaner windows) or include flagged? Snapshot has only 1; behavior matters more on live DBs.
- **Q6. Profile DB selection:** when no `--db` is given, prefer the active profile's DB or the default one? (Different profiles have different totals; a wrong default silently under-reports.)
- **Q7. `--strict` semantics:** fail on any missing rate, or only on missing rates for rows with nonzero tokens?
- **Q8. Report determinism vs `open` sessions:** should the report embed a `generated_at` timestamp (breaks byte-identical reproducibility for the same window, R5) — yes, but only in the file header, and it should be suppressible.
- **Q9. Unit-test baseline:** use the snapshot DB as the golden fixture (301 sessions, known aggregates) — confirm it can be bundled with the repo for tests.
- **Q10. Scope of "vendor":** `billing_provider` values like `kimi-coding` vs `kimi-for-coding` — group as-is or normalize to a vendor family (`kimi`)? Affects rollups.

---

9. Appendix — Research evidence (condensed)

Sources: `website/docs/developer-guide/session-storage.md` (schema, WAL, migrations v1–11), `agent/usage_pricing.py` (CostStatus/CostSource enums, BillingRoute, PricingEntry, `estimate_usage_cost`, `normalize_usage`, provider routing), `agent/insights.py` (`_estimate_cost` — same estimator, live), `agent/`

turn_finalizer.py (persists estimated_cost_usd, cost_status, cost_source; NOT pricing_version), agent/codex_runtime.py (writes billing_mode=subscription_included when status is included), agent/agent_init.py (defaults: session_estimated_cost_usd=0.0, cost_status='unknown', cost_source='none'), hermes_state.py (schema, update path), hermes sessions / hermes insights CLI surface, /tmp/hermes-cost-snapshot.db (read-only, mode=ro).

Snapshot stats: 301 sessions; 10,972 messages; 14962 message rows all with NULL token_count; platforms cron/cli/subagent/telegram/email; models deepseek-v4-flash, gpt-5.6-sol, kimi-k2.6, gpt-5.5, mimo-v2.5-pro, kimi-k1.5, glm-4.5-flash, deepseek-chat, gpt-4o; providers openai-codex/deepseek/kimi-coding/xiaomi/zai/kimi-for-coding/NULL; tokens in 44.1M / out 5.4M / cache-read 651.1M / cache-write 0 / reasoning 375K; cost_status unknown 171 / included 111 / NULL 19; pricing_version NULL ×301; actual_cost_usd NULL ×301; estimated_cost_usd 0.0 ×281, 0.0017 ×1; one official_docs_snapshot row (deepseek-chat); one codex row with billing_mode=subscription_included but cost_status=unknown; 31 subagent rows all parented; date range 2026-07-04 → 2026-08-13 UTC.

Formula confirmed: estimate_usage_cost sums per-million rates × buckets, adding request_cost × request_count; any nonzero bucket without a rate ⇒ status unknown, amount NULL (rendered 0.0 by the agent default). Codex route ⇒ subscription_included ⇒ amount 0, status included, pricing_version='included-route' (in-memory only; not persisted).

test2_prd/glm-4.5-flash/cost-tracker-prd.md

Hermes Cost Tracker PRD

Research Summary

Data Structure Analysis

The Hermes session store (sessions table) contains the following relevant fields for cost tracking:

Core Token Data:

- input_tokens: Tokens consumed by the model
- output_tokens: Tokens generated by the model
- cache_read_tokens: Tokens read from cache (billable for some providers)
- cache_write_tokens: Tokens written to cache (typically not billable)

Identifying Fields:

- model: The model used (e.g., deepseek-v4-flash, gpt-5.6-sol, glm-4.5-flash)
- billing_provider: The provider (e.g., deepseek, openai-codex, xiaomi, zai)
- billing_mode: How the session was billed (chat_completions, subscription_included)

Cost & Status Fields:

- estimated_cost_usd: Pre-computed cost estimates (282/301 sessions have this)
- actual_cost_usd: Actual costs (0/301 sessions have this)
- cost_status: "included" for quota-based, "unknown" for metered
- cost_source: Source of cost calculation
- pricing_version: Always NULL in current data (USD not computed)

Temporal Fields:

- started_at: Unix timestamp for session start
- ended_at: Unix timestamp for session end

Key Findings

USD Calculation:

- Hermes does NOT compute USD costs internally (pricing_version is always NULL)
- Only estimated_cost_usd exists, but this appears to be placeholder data
- **Critical Gap:** No actual pricing data or USD calculations exist

Data Quality:

- 301 total sessions in snapshot
- 19 sessions with missing provider (6.3%)
- 19 sessions with missing token counts (6.3%)
- 111 sessions marked as "included" (quota-based)
- 171 sessions marked as "unknown" (metered)

Provider Types:

- **Metered:** deepseek, kimi-coding, kimi-for-coding, openai-codex, xiaomi, zai
- **Quota-based:** Sessions with cost_status="included" (subscription_included mode)

Cache Billing:

- Cache read tokens are present and significant (avg 2.3M reads per session)
- Cache write tokens are typically zero
- Cache billing varies by provider (some bill cache reads, others don't)

Product Requirements Document

1. Goal and Non-Goals

Goal: Provide accurate cost tracking for Hermes sessions by calculating USD costs from token counts using configurable rate tables, distinguishing between metered and quota-based usage, and generating comprehensive reports.

Non-Goals:

- Real-time cost monitoring during sessions
- Integration with external billing systems
- Cost optimization recommendations
- Historical cost trend analysis beyond date windows
- Provider quota management
- Multi-currency support beyond USD

2. Requirements (Must-Have First)

2.1 Core Functionality

- **Cost Calculation:** Compute USD costs per vendor/model/platform from token counts
- **Billing Type Separation:** Split costs into metered vs quota-based categories
- **Cache Cost Inclusion:** Include cache-read costs where applicable (provider-dependent)
- **Date Window Filtering:** Generate reports for configurable date ranges
- **Multiple Output Formats:** Console table + CSV export + optional markdown report

2.2 Data Management

- **Read-Only Database Access:** Access session store without modification
- **Rate Table Configuration:** Support for configurable pricing rates per model/provider
- **Session Filtering:** Filter by date range, model, provider, billing mode
- **Data Validation:** Handle missing/invalid data appropriately

2.3 Reporting

- **Summary Reports:** Total costs by vendor, model, platform
- **Breakdown Reports:** Cost distribution by billing type (metered vs quota)
- **Session-Level Details:** Individual session costs with token breakdowns
- **Export Capabilities:** CSV format for external analysis

2.4 Data Freshness

- **Scheduled Updates:** Cron job for automatic report generation
- **Real-time Option:** Optional mode to track new sessions as they're created
- **Configurable Schedule:** Default: weekly reports (recommended)

3. Data Model Notes

Read Fields:

- `id`, `model`, `billing_provider`, `billing_mode`
- `input_tokens`, `output_tokens`, `cache_read_tokens`, `cache_write_tokens`
- `started_at`, `ended_at`
- `cost_status` (for quota identification)
- `estimated_cost_usd` (for validation/comparison)

Derived Fields:

- USD cost (calculated from tokens + rate table)
- Billing type (metered vs quota based on `cost_status`)
- Cache cost inclusion (provider-specific rules)

4. Incomplete Data Handling

Approach: Provider/Platform Averages For sessions missing model, provider, or token counts:

- **Missing Provider:** Use platform-wide average rates for the known model
- **Missing Model:** Use provider's most common model rates
- **Missing Tokens:** Use session median token counts for that model/provider
- **Marking:** All estimated values clearly marked with "[ESTIMATED]" prefix

Fallback Strategy:

1. Try exact model/provider match
2. Fall back to provider average for that model
3. Fall back to model average across all providers
4. Fall back to global averages
5. Skip session if no reliable estimate possible

5. Edge Cases

Missing Model/Provider:

- Apply fallback averaging strategy
- Clearly mark as estimated in reports

Zero-Token Sessions:

- Include in reports with $0 cost
- Flag for potential investigation

Quota Sessions (cost_status="included"):

- Separate from metered costs
- Show quota usage but not USD cost
- Track quota consumption patterns

Sessions with pricing_version=NULL:

- Expected behavior (Hermes doesn't compute USD)
- Calculate costs using rate table instead

Mixed Cache Billing:

- Provider-specific rules (e.g., OpenAI bills cache reads, others don't)
- Configurable per provider in rate table

6. Assumptions and Open Questions

Assumptions:

1. Token counts accurately reflect resource consumption
2. Rate tables can be maintained externally and updated independently
3. Cache billing follows known provider patterns
4. Session store schema remains stable
5. Date ranges are inclusive of start/end timestamps

Open Questions:

1. How should handle sessions with both `estimated_cost_usd` and calculated costs?
2. Should the tool support custom rate table formats beyond CSV?
3. What's the desired granularity for date filtering (day/week/month)?
4. Should the tool track cost trends over time, or only point-in-time reports?
5. How should handle sessions with negative token counts (if they exist)?

7. Recommended Update Mechanism

Primary: Scheduled Cron Job

- Default: Weekly reports on Sunday at 2 AM
- Configurable schedule and output directory
- Automatic cleanup of old reports (keep last 12 weeks)
- Log rotation and error handling

Secondary: Real-time Mode (Optional)

- Monitor new sessions as they're created
- Trigger immediate cost calculation for new sessions
- Useful for high-visibility environments
- Higher resource usage, not recommended for default

Configuration:

```
# cost-tracker-config.yaml
update_schedule: "0 2 * * 0"  # Weekly at 2 AM Sunday
output_dir: "/var/log/hermes-costs"
retention_weeks: 12
rate_table_path: "/etc/hermes/rates.csv"
realtime_enabled: false
database_path: "/var/lib/hermes/sessions.db"
```

8. Implementation Considerations

Rate Table Structure:

```
provider,model,mode,input_rate,output_rate,cache_read_rate,currency
openai-codex,gpt-4o,chat_completions,0.000015,0.000060,0.0000075,USD
deepseek,deepseek-v4-flash,chat_completions,0.000014,0.000056,0.000007,USD
zai,glm-4.5-flash,chat_completions,0.000016,0.000064,0.000008,USD
```

Error Handling:

- Database connection failures
- Missing rate table entries
- Invalid token counts
- Permission issues
- Disk space constraints

Performance:

- Index on `started_at` for date filtering
- Batch processing for large date ranges
- Memory-efficient streaming for large databases
- Caching of rate tables and provider averages

PRD Version: 1.0 Date: August 13, 2026 Target Implementation: Hermes Cost Tracker Tool

test2_prd/glm-4.5/cost-tracker-prd.md

Hermes Cost Tracker - Product Requirements Document

Version: 1.0 **Status:** Draft **Date:** 2025-01-17 **Author:** felix (glm-4.5)

Executive Summary

The Hermes Cost Tracker is a read-only analytics tool that computes LLM API costs from Hermes session data stored in SQLite. It provides configurable, date-windowed cost breakdowns by vendor, model, and platform, distinguishing between quota-based (included) and metered spend, and accurately accounting for cache read costs.

Goals

- Compute accurate USD costs for Hermes sessions using configurable per-model pricing tables
- Distinguish between quota-based (included) and metered (pay-as-you-go) spend
- Account for cache read tokens at appropriate discounted rates
- Support multiple report formats: console tables, Markdown, CSV
- Provide both scheduled (cron) and real-time cost update mechanisms
- Handle incomplete session data with transparent, clearly-labeled estimates

---

Non-Goals

- Real-time cost metering or streaming cost alerts
- Provider API integration or usage verification
- Session creation, modification, or deletion (read-only access only)
- Multi-tenant or multi-user cost attribution beyond the platform field
- Historical cost retroactivity after rate table updates (costs are computed as-of report generation)

Must-Have Requirements

1. Core Cost Computation

FR-1.1: Read-Only Database Access

- App must connect to Hermes SQLite session database (default: `~/.hermes/state.db`)
- All database operations must be read-only SELECT queries only
- Must never write, modify, or delete any data

FR-1.2: Date-Window Filtering

- Support filtering sessions by date range: `--start-date` and `--end-date`
- Use `sessions.started_at` (Unix epoch float) for filtering
- Default to last 7 days if no dates specified
- Support relative date shortcuts: `--days 7` (equivalent to start=today-7d, end=today)

FR-1.3: Cost Calculation Engine

- Compute costs for each session using formula:

```
total_cost = (input_tokens × input_rate) +
             (output_tokens × output_rate) +
             (cache_read_tokens × cache_read_rate) +
             (reasoning_tokens × reasoning_rate)
```

- Where rates are per-million token rates in USD from the rate table
- Handle zero-token sessions gracefully (cost = $0.00)
- Sum costs by vendor, model, and platform for reporting

FR-1.4: Cache Read Billing

- Cache read tokens must be billed at discounted rates (typically 10-25% of input rate)
- Rate table must include separate `cache_read_rate` field per model
- If cache_read_rate is missing in rate table, default to 10% of input_rate
- Must distinguish cache read (reused) from cache write (initial fill)

FR-1.5: Metered vs Quota Classification

- Classify spend into two categories:

- **Metered**: `cost_status != 'included'` OR `cost_status IS NULL` AND `billing_mode != 'subscription_included'` - **Quota**: `cost_status = 'included'` OR `billing_mode = 'subscription_included'`

- Report both totals separately in all output formats
- Zero cost for quota sessions ($0.00 spent)

FR-1.6: Configurable Rate Table

- Support YAML configuration file for pricing rates
- Default location: `~/.hermes/cost-tracker/rates.yaml`
- Rate table structure:

```yaml
rates:
  - vendor: "openai"
    model: "gpt-4o"
    platform: "openai-codex"  # optional, if platform-specific rates differ
    input_rate: 2.50          # USD per million input tokens
    output_rate: 10.00        # USD per million output tokens
    cache_read_rate: 0.25     # USD per million cache read tokens
    reasoning_rate: 15.00     # USD per million reasoning tokens (optional)
    effective_date: "2025-01-01"  # date when this rate becomes effective

  - vendor: "deepseek"
    model: "deepseek-chat"
    input_rate: 0.14
    output_rate: 0.28
    cache_read_rate: 0.014
    effective_date: "2025-01-01"
```

- Support multiple rate entries per vendor/model by effective_date
- Use the rate with the most recent effective_date <= session date
- If no matching rate found, mark cost as "NO_RATE" (see FR-3.1)

2. Reporting

FR-2.1: Console Table Output

- Default output format: human-readable ASCII table
- Group by: vendor → model → platform
- Columns: Vendor, Model, Platform, Sessions, Input Tokens, Output Tokens, Cache Read Tokens, Metered Cost, Quota Cost, Total Cost
- Sum totals at the bottom of each vendor section
- Grand totals at the end
- Example:

```
Vendor: openai

┌─────────────┬──────────┬──────────┬───────────┬────────────┬────────────┬──────────────┬────────────┬────────────┐
│ Model       │ Platform │ Sessions │ In Tokens │ Out Tokens │ Cache Read │ Metered Cost │ Quota Cost │ Total Cost │
├─────────────┼──────────┼──────────┼───────────┼────────────┼────────────┼──────────────┼────────────┼────────────┤
│ gpt-4o      │ openai-codex │ 42   │ 2.1M      │ 0.8M       │ 12.3M      │ $0.00        │ $0.00      │ $0.00      │
│ gpt-5.6-luna │ openai-codex │ 15   │ 1.5M      │ 0.6M       │ 8.7M       │ $0.00        │ $0.00      │ $0.00      │
├─────────────┼──────────┼──────────┼───────────┼────────────┼────────────┼──────────────┼────────────┼────────────┤
│ Subtotal    │          │ 57       │ 3.6M      │ 1.4M       │ 21.0M      │ $0.00        │ $0.00      │ $0.00      │
└─────────────┴──────────┴──────────┴───────────┴────────────┴────────────┴──────────────┴────────────┴────────────┘

GRAND TOTAL: 385 sessions, 51.1M input tokens, 7.7M output tokens, 897M cache read tokens
Metered Cost: $123.45 | Quota Cost: $0.00 | Total Cost: $123.45
```

FR-2.2: Markdown Output

- Optional output format: `--output-format markdown`
- Save to file: `--output-file costs-2025-01-17.md`

- Structure: vendor sections with tables, then summary table
- Include metadata: date range, generation timestamp, rate table version

FR-2.3: CSV Output

- Optional output format: `--output-format csv`
- Save to file: `--output-file costs-2025-01-17.csv`
- Flat structure: one row per vendor/model/platform combination
- Columns: vendor, model, platform, session_count, input_tokens, output_tokens, cache_read_tokens, reasoning_tokens, metered_cost_usd, quota_cost_usd, total_cost_usd
- Include metadata row at top (commented) with date range and timestamp

FR-2.4: Platform Breakdown

- Support grouping by platform using `--group-by platform`
- Platforms come from `sessions.source` field: cli, cron, email, subagent, telegram
- Default grouping: vendor → model → platform

3. Data Model

FR-3.1: Session Fields Read

- Required fields: `id, model, started_at, ended_at, source`
- Token fields: `input_tokens, output_tokens, cache_read_tokens, cache_write_tokens, reasoning_tokens`
- Billing fields: `billing_provider, billing_mode, cost_status, cost_source, pricing_version`
- Derived fields: `metered_cost, quota_cost` (computed)

FR-3.2: Missing Data Handling - Model

- Based on snapshot analysis: 0 sessions have NULL model
- If model is NULL, session is excluded with warning message

FR-3.3: Missing Data Handling - Billing Provider

- Based on snapshot analysis: 19/385 sessions (4.9%) have NULL billing_provider
- If `billing_provider` is NULL:

- Extract provider from `model` field if possible (e.g., "deepseek-chat" → "deepseek") - If extraction fails, use "UNKNOWN" and mark all costs as "NO_RATE_ESTIMATED" - Log warnings for each affected session

FR-3.4: Missing Data Handling - Billing Mode

- Based on snapshot analysis: 269/385 sessions (69.9%) have NULL billing_mode
- If `billing_mode` is NULL:

- Use `cost_status` to classify: 'included' → quota, else → metered - If `cost_status` is also NULL, classify as metered (default assumption) - Document this assumption in report metadata

FR-3.5: Missing Data Handling - Token Counts

- Based on snapshot analysis: 19/385 sessions (4.9%) have zero total tokens
- Zero-token sessions are valid (e.g., aborted sessions, metadata-only)
- Include in session counts but contribute $0.00 to costs
- Mark with [ZERO_TOKENS] label in detailed output if enabled

4. Data Freshness

FR-4.1: Scheduled Updates (Recommended Default)

- Primary mechanism: cron job running weekly (Sunday 00:00 UTC default)
- Cron command: `hermes cost-report --days 7 --output-format markdown --output-file ~/.hermes/costs/weekly-$(date +\%Y-\%m-\%d).md`
- Recommended cron config in `~/.hermes/cron/config.yaml`:

```
jobs:
  - name: "weekly-cost-report"
    schedule: "0 0 * * 0"  # Sunday 00:00 UTC
    command: "hermes cost-report --days 7 --output-format markdown --output-file ~/.hermes/costs/weekly-$(date +\\%Y-\\%m-\\%d).md"
    deliver: "origin"
```

- Previous reports are preserved for historical comparison

FR-4.2: Real-Time Mode

- Optional mode: `--real-time` flag watches for new sessions
- Uses file system watcher on `~/.hermes/state.db` (WAL changes)
- Incrementally processes new sessions as they land
- Outputs updated totals to console on each new session (no full regen)
- For ad-hoc checks, users can also run with `--days 1` manually
- Not recommended for automated reporting due to overhead

FR-4.3: Hybrid Approach

- Users can run both: weekly scheduled reports + real-time monitoring during development
- Real-time mode for interactive awareness, weekly reports for accounting and trend analysis

5. Incomplete Data Estimates

PRD Decision: Estimates Path Chosen

Approach: Use vendor/platform average rates for missing metadata, clearly labeled in all reports.

Rationale:

- Snapshot shows 70% of sessions lack `billing_mode`, 5% lack `billing_provider`
- Using "NO_RATE" would result in massive reporting gaps (~75% of spend)
- Operator-maintained mappings are high-friction and error-prone
- Average-rate estimates with clear labeling provides actionable data with transparency

FR-5.1: Estimate Calculation

- When a session has missing `billing_provider`:

- Extract vendor from model slug if possible (e.g., "deepseek-chat" → "deepseek") - If extraction fails, use the most common vendor for the session's platform - Apply the average rate across all models for that vendor

- When a session has missing `billing_mode`:

- Default to metered classification (conservative assumption) - This aligns with "unknown" cost_status behavior

- When no rate table entry matches:

- Use the vendor average rate (mean of all models for that vendor) - If vendor unknown, use the global average rate across all configured models

FR-5.2: Estimation Labeling

- Every report must include a "Data Quality" section with:

- Percentage of sessions using estimated rates - Percentage of sessions with missing metadata - List of affected vendor/model combinations

- In console output, prefix estimated costs with `[~]`: `$12.34 [~]`
- In CSV output, include boolean column `is_estimated`
- In Markdown output, add footnote: `[~] Estimated cost due to missing billing metadata`

FR-5.3: Alternative Path Not Chosen: Exact Mapping

- Operator-maintained mapping file would be required for each missing vendor/model
- High maintenance burden: requires manual updates on every new provider
- Prone to stale data and mapping errors
- Decision: rejected due to operational friction

Data Model Notes

Session Table Fields Used

Field	Type	Purpose	Nullable
id	TEXT	Session identifier	NO
model	TEXT	Model name (e.g., "gpt-4o")	NO (observed 0 NULL)
source	TEXT	Platform: cli, cron, email, subagent, telegram	NO
started_at	REAL	Unix epoch timestamp	NO
ended_at	REAL	Unix epoch timestamp	YES
input_tokens	INTEGER	Total prompt tokens	YES (default 0)
output_tokens	INTEGER	Total completion tokens	YES (default 0)
cache_read_tokens	INTEGER	Reused prompt cache tokens	YES (default 0)
cache_write_tokens	INTEGER	New prompt cache writes	YES (default 0)
reasoning_tokens	INTEGER	Thinking/reasoning tokens	YES (default 0)
billing_provider	TEXT	API provider: deepseek, openai-codex, zai, etc.	YES (5% NULL)
billing_mode	TEXT	chat_completions, subscription_included, etc.	YES (70% NULL)
cost_status	TEXT	included, unknown, NULL	YES
cost_source	TEXT	none, official_docs_snapshot	YES
pricing_version	TEXT	Version identifier (currently NULL)	YES

Platform Field

- Derived from `sessions.source` (not `sessions.platform` which doesn't exist)
- Values: cli, cron, email, subagent, telegram
- Used for grouping and per-platform cost attribution

Edge Cases

EC-1: Sessions with Missing Model

- **Observed:** 0 sessions in snapshot
- **Handling:** Exclude session from all calculations
- **Logging:** Print warning with session ID
- **Report:** Note excluded session count in metadata

EC-2: Sessions with Missing Provider

- **Observed:** 19/385 sessions (4.9%)
- **Handling:** Estimate from model slug or platform average (see FR-5.1)
- **Labeling:** Mark as `[~]` estimated in all outputs
- **Example:** Model "zai" with NULL provider → vendor "zai" extracted from slug

EC-3: Sessions with Missing Billing Mode

- **Observed:** 269/385 sessions (69.9%)
- **Handling:** Default to metered classification
- **Rationale:** Conservative assumption; unknown status typically means pay-as-you-go
- **Labeling:** No special labeling (uses standard metered logic)

EC-4: Cost Status "Included" (Quota)

- **Observed:** 111/385 sessions (28.8%)
- **Handling:** Classify as quota spend ($0.00 metered cost)
- **Rationale:** These sessions use included quota, not actual USD spend
- **Example:** openai-codex with subscription_included mode
- **Report:** Show separately as "Quota Cost"

EC-5: Zero-Token Sessions

- **Observed:** 19/385 sessions (4.9%)
- **Handling:** Include in session counts, $0.00 cost
- **Labeling:** Optional `[ZERO_TOKENS]` flag in verbose mode
- **Example:** Aborted sessions, metadata-only sessions

EC-6: Cache-Heavy Sessions

- **Observed:** 897M cache read tokens (17x more than input tokens)
- **Handling:** Apply cache_read_rate (default 10% of input_rate)
- **Rationale:** Cache hits are billed at discounted rates (typical 10-25%)
- **Example:** Session with 80M input + 1.2B cache read → cache read dominates cost

EC-7: Reasoning Tokens

- **Observed:** 464K reasoning tokens in snapshot
- **Handling:** Bill at `reasoning_rate` if configured, else fall back to output_rate
- **Rationale:** Reasoning tokens (thinking, chain-of-thought) often billed separately
- **Labeling:** Show separate column in detailed output

EC-8: Missing Rate Table Entries

- **Handling:** Use vendor average rate, then global average rate
- **Labeling:** Mark as `[~] NO_RATE_ESTIMATED]`
- **Logging:** Print warning with model name and vendor
- **Example:** New model "gpt-6.0" added but not in rate table yet

EC-9: Historical Sessions with Deprecated Rates

- **Handling:** Use rate with effective_date <= session.started_at
- **Labeling:** No special labeling
- **Configuration:** Rate table should include historical entries
- **Example:** Model "gpt-4o" had rate change on 2025-03-01; use old rate for Feb sessions

EC-10: Sessions with No Billing Provider and No Extractable Vendor

- **Handling:** Use global average rate across all configured models
- **Labeling:** Mark as `[~] GLOBAL_AVERAGE]`
- **Example:** Model "custom-llm" with NULL provider and no vendor in slug

---

Assumptions

1. **Rate Table Completeness:** Users will maintain rate table entries for all active models. Incomplete tables will trigger average-rate estimates.
1. **Rate Versioning:** Pricing changes are expected and should be managed via effective_date in rate table, not via pricing_version field (currently NULL).
1. **Cache Read Billing:** Cache read tokens are billed at ~10% of input rate based on industry standard (Anthropic, OpenAI, etc.). This is configurable via cache_read_rate per model.
1. **Quota Classification:** Sessions with cost_status='included' or billing_mode='subscription_included' are considered zero-cost (quota). This aligns with observed data (openai-codex subscriptions).
1. **Missing Billing Mode:** NULL billing_mode defaults to metered (not quota) based on conservative cost estimation principle.
1. **Provider Extraction:** Vendor can be extracted from model slug in most cases (e.g., "deepseek-chat" → "deepseek", "gpt-4o" → "openai"). Fallback to platform averages if extraction fails.
1. **Real-Time Overhead:** Real-time mode is for ad-hoc monitoring, not automated reporting, due to file system watcher overhead on large databases.
1. **Read-Only Guarantee:** All database operations will be SELECT queries only. No write operations even for temporary tables.
1. **Session Completeness:** Only sessions with ended_at IS NOT NULL are included in cost calculations. Active sessions (ended_at IS NULL) are excluded until complete.
1. **Token Accuracy:** Token counts stored in sessions table are accurate and reflect actual billed tokens (no estimation needed for tokens themselves).

Open Questions

1. **Rate Table Format:** Should we support JSON as an alternative to YAML for the rate table? (Decision: YAML only for v1.0, can add JSON later if requested)
1. **Real-Time Granularity:** Should real-time mode show per-session costs or cumulative totals only? (Decision: cumulative totals only for v1.0 to reduce noise)
1. **Alert Thresholds:** Should we include spending alert thresholds (e.g., alert if weekly spend > $50)? (Decision: out of scope for v1.0; can add as cron hook integration later)
1. **Cost Source Trust:** Should we prioritize estimated_cost_usd from DB over our own calculations when cost_source is not 'none'? (Decision: ignore DB costs; always compute fresh from rate table to ensure consistency)
1. **Session Export:** Should we support exporting individual session cost details (not just aggregated reports)? (Decision: out of scope for v1.0; can add as --verbose flag later)
1. **Historical Data Migration:** Should we backfill costs for sessions created before the cost tracker was deployed? (Decision: Yes, by default; costs are computed as-of report generation time)
1. **Reasoning Token Billing:** Should reasoning tokens be billed at input or output rates when reasoning_rate is not configured? (Decision: Fall back to output_rate for v1.0; can make configurable later)
1. **Multi-Profile Support:** Should the cost tracker aggregate costs across multiple Hermes profiles? (Decision: No, single profile only for v1.0; can add --profile flag later)

Future Enhancements (Out of Scope for v1.0)

- Spending alerts and thresholds
- Cost trend visualizations (charts, graphs)
- Multi-profile aggregation
- Per-user or per-gateway-channel cost attribution
- Cost anomaly detection (e.g., sudden spike in cache usage)
- Integration with provider usage APIs for verification
- Historical rate change impact analysis
- Cost forecasting based on usage trends
- Web dashboard for interactive cost exploration
- Export to accounting systems (CSV/JSON APIs)

Appendix: Database Schema Reference

```
CREATE TABLE sessions (
    id TEXT PRIMARY KEY,
    source TEXT NOT NULL,
    user_id TEXT,
    session_key TEXT,
    chat_id TEXT,
    chat_type TEXT,
    thread_id TEXT,
    model TEXT,
    model_config TEXT,
    system_prompt TEXT,
    parent_session_id TEXT,
    started_at REAL NOT NULL,
    ended_at REAL,
    end_reason TEXT,
    message_count INTEGER DEFAULT 0,
    tool_call_count INTEGER DEFAULT 0,
    input_tokens INTEGER DEFAULT 0,
    output_tokens INTEGER DEFAULT 0,
    cache_read_tokens INTEGER DEFAULT 0,
    cache_write_tokens INTEGER DEFAULT 0,
    reasoning_tokens INTEGER DEFAULT 0,
    cwd TEXT,
    git_branch TEXT,
    git_repo_root TEXT,
    billing_provider TEXT,
    billing_base_url TEXT,
    billing_mode TEXT,
    estimated_cost_usd REAL,
    actual_cost_usd REAL,
    cost_status TEXT,
    cost_source TEXT,
    pricing_version TEXT,
    title TEXT,
    api_call_count INTEGER DEFAULT 0,
    handoff_state TEXT,
    handoff_platform TEXT,
    handoff_error TEXT,
    rewind_count INTEGER NOT NULL DEFAULT 0,
    archived INTEGER NOT NULL DEFAULT 0,
    FOREIGN KEY (parent_session_id) REFERENCES sessions(id)
);
```

End of PRD

---

test2_prd/mimo-v2.5-pro/cost-tracker-prd.md

Hermes Cost Tracker — PRD

Status: Draft v1 **Author:** Felix (AI agent), mimo-v2.5-pro **Date:** 2026-08-13 **Board:** cost-prd-mimo-v25-pro

---

1. Goal

Give Hermes operators a clear, accurate answer to "what am I spending, where, and is it getting cheaper or more expensive?" — computed entirely from the local session store, with no external API calls or live billing integrations.

Non-Goals

- Real-time cost alerts or budget enforcement.
- Integration with provider billing dashboards (OpenAI usage page, DeepSeek dashboard, etc.).
- Per-message cost breakdown (only per-session granularity).
- Cost prediction or forecasting.
- Multi-operator aggregation (single-machine scope only).

---

2. Requirements

2.1 Must-Have

#	Requirement	Notes
R1	**Compute cost per vendor/model/platform for a date window**	Read sessions from the Hermes session store (SQLite). Apply pricing from the rate table. Aggregate by billing_provider × model × source (platform).
R2	**Split metered vs. quota spend**	Metered = `billing_mode = 'chat_completions'` (user pays per token). Quota = `billing_mode = 'subscription_included'` (included in a subscription, e.g., openai-codex). Report must show both categories clearly.
R3	**Include cache-read costs**	Cache reads are a separate billable line. The rate table must have a `cache_read_cost_per_million` column. Cache-write costs are also included when present.
R4	**Configurable, dated rate table**	A YAML/JSON file with per-model pricing. Each entry has `effective_from` date so the tracker can apply the correct rate for sessions that span a pricing change. Ships with a default table seeded from Hermes's built-in `_OFFICIAL_DOCS_PRICING` snapshot. Operator can override or add entries.

#	Requirement	Notes
R5	**Console table report**	A plain-text table printed to stdout. Columns: provider, model, platform, sessions, input_tokens, output_tokens, cache_read_tokens, estimated_cost_usd, cost_status. Totals row at bottom.
R6	**Markdown or CSV file report**	Same data as R5, written to a file. Markdown table by default; `--format csv` flag for CSV. Output path configurable, default: `~/.hermes/cost-reports/YYYY-MM-DD_to_YYYY-MM-DD.md`.
R7	**Read-only database access**	The tracker opens the session database in read-only mode (`?mode=ro`). It never modifies session data.

2.2 Should-Have

#	Requirement	Notes
R8	**Breakdown by date**	Show daily or weekly cost trend within the date window.
R9	**Compare periods**	`--compare` flag to show two date windows side by side (e.g., this week vs. last week).
R10	**Top-N sessions**	`--top 10` flag to show the most expensive individual sessions.

---

3. Data Freshness

The tracker operates on historical data already written to the session store. It does not poll APIs or stream live events.

Recommended Update Mechanism: Weekly Cron Job

A Hermes cron job regenerates the cost report on a schedule. This is the primary mechanism.

Cron configuration:

```
# ~/.hermes/cron.yaml (or hermes cron add)
- name: "weekly-cost-report"
  schedule: "0 9 * * 1"           # Every Monday at 09:00 local
  prompt: |
    Run the Hermes cost tracker for the past 7 days.
    Command: hermes-cost-report --days 7 --format markdown
    Deliver the report summary to the operator via Telegram.
  deliver: telegram
  profile: felix
```

Why weekly, not daily: Most Hermes operators run a handful of sessions per day. Weekly reports provide enough signal without noise. Operators with high volume can switch to `"0 9 *"` (daily).

Optional: Real-Time Mode

A `--watch` flag can tail the session store and regenerate the report when new sessions land. This uses SQLite's `last_insert_rowid()` or a file-watcher on the DB. Real-time mode is useful for debugging or during heavy experimentation, but is **not** the recommended default — it keeps a process running and adds complexity.

Recommendation: Weekly cron is the default. `--watch` exists for power users.

---

4. Incomplete Data Handling

Problem

Inspection of the snapshot database (301 sessions) reveals:

Issue	Count	Characteristics
`billing_provider IS NULL`	19 sessions	All have 0 input/output/cache tokens. Model is non-null (gpt-5.5, deepseek-v4-flash, kimi-k2.6).
`cost_status = 'unknown'`	171 sessions	Pricing not computed. Most are metered providers (deepseek, kimi-coding, xiaomi, zai).
`pricing_version IS NULL`	All 301 sessions	Cost estimation was never run at session time.
`billing_mode IS NULL`	185 sessions	Mostly non-codex providers where billing_mode wasn't set.

Chosen Approach: Exact Path with Operator-Maintained Mapping

The tracker uses a **two-tier resolution**:

Tier 1 — Direct pricing lookup (exact): Use the existing `usage_pricing.py` engine (`resolve_billing_route` + `get_pricing_entry`) to resolve pricing for every session's `billing_provider` + `model` + `billing_base_url`. This handles:

- `openai-codex` → `subscription_included` → $0 (quota)
- `deepseek` + `deepseek-v4-flash` → official docs snapshot pricing
- `openrouter` → models API pricing
- Any provider with a known `base_url` → endpoint models API

Tier 2 — Operator override mapping (exact, documented): For provider+model combinations that Tier 1 cannot resolve (e.g., `kimi-coding` + `kimi-k2.6`, `xiaomi` + `mimo-v2.5-pro`, `zai` + `glm-4.5-flash`), the operator maintains entries in the rate table YAML. These are **exact prices, not estimates** — the operator looks up the provider's pricing page and records the per-million-token rates.

```
# ~/.hermes/cost-rates.yaml
rates:
  - provider: kimi-coding
    model: kimi-k2.6
    input_per_million: 0.0
    output_per_million: 0.0
    cache_read_per_million: 0.0
    billing_mode: subscription_included
```

```
    effective_from: 2026-01-01
    notes: "Kimi subscription — no per-token charge"

  - provider: xiaomi
    model: mimo-v2.5-pro
    input_per_million: 0.0
    output_per_million: 0.0
    cache_read_per_million: 0.0
    billing_mode: subscription_included
    effective_from: 2026-01-01
    notes: "Xiaomi MiMo — quota-based"

  - provider: zai
    model: glm-4.5-flash
    input_per_million: 0.50
    output_per_million: 1.50
    cache_read_per_million: 0.05
    billing_mode: chat_completions
    effective_from: 2026-01-01
    notes: "ZAI GLM metered pricing"
```

Why not estimates? Estimation (e.g., using vendor/platform averages) introduces silent inaccuracy that compounds over time. A missing rate entry is visible and actionable — the report flags it as "UNPRICED" with the exact provider+model+token counts, prompting the operator to add the rate. This is preferable to a guess that looks authoritative.

How Incomplete Data Appears in Reports

Scenario	Report Treatment
Provider NULL, tokens zero	Row shown with $0.00, labeled `[no-provider]`. Not counted in totals.
Provider NULL, tokens non-zero	Row shown with `UNPRICED` cost. Labeled `[no-provider]`. Counted in token totals but excluded from cost totals.
Rate table has no entry for provider+model	Cost shows `UNPRICED`. A warning line appears: "2 sessions (1.2M tokens) have no pricing — add entries to ~/.hermes/cost-rates.yaml".
`cost_status = 'included'` (quota)	Cost shown as `$0.00 (included)`. Grouped under "Quota Spend" section.
`cost_status = 'unknown'`	Tracker re-computes cost using Tier 1/2. If still unresolvable, treated as UNPRICED.
Zero-token sessions	Shown in session count but contribute $0.00 to cost. No warning.

---

5. Data Model — Fields Read

The tracker reads from the `sessions` table. No joins to `messages` are needed for cost computation (tokens are aggregated at session level).

Fields Used

Field	Type	Purpose
`id`	TEXT	Session identifier (for top-N reporting)
`model`	TEXT	Model name (e.g., `deepseek-v4-flash`). Join key for rate table.
`billing_provider`	TEXT	Provider slug (e.g., `deepseek`, `openai-codex`). Join key for rate table.
`billing_base_url`	TEXT	API endpoint URL. Used by `resolve_billing_route` for provider detection.
`billing_mode`	TEXT	`subscription_included` or `chat_completions` or NULL. Determines metered vs. quota classification.
`source`	TEXT	Platform: `cli`, `telegram`, `cron`, `email`, `subagent`. Grouping dimension.
`started_at`	REAL	Unix timestamp. Date window filter.
`ended_at`	REAL	Unix timestamp. Duration calculation.
`input_tokens`	INTEGER	Non-cache input tokens. Billable at input rate.
`output_tokens`	INTEGER	Output tokens. Billable at output rate.
`cache_read_tokens`	INTEGER	Cache-hit input tokens. Billable at cache-read rate (typically 10% of input).
`cache_write_tokens`	INTEGER	Cache-miss input tokens. Billable at cache-write rate (typically 125% of input for Anthropic).
`reasoning_tokens`	INTEGER	Reasoning tokens (subset of output for o3/deepseek-reasoner). Reported separately but billed as output.
`estimated_cost_usd`	REAL	Pre-computed cost (usually 0.0 or NULL). Cross-check only.
`actual_cost_usd`	REAL	Provider-reported cost (always NULL in snapshot). Cross-check only.
`cost_status`	TEXT	`included`, `unknown`, or NULL. Initial classification.
`cost_source`	TEXT	`none`, `official_docs_snapshot`, or NULL.
`pricing_version`	TEXT	Pricing snapshot version (always NULL in current data).
`title`	TEXT	Session title (for top-N display).
`message_count`	INTEGER	Message count (context for session size).
`api_call_count`	INTEGER	API call count (context for session cost density).

Fields NOT Used

Field	Why Not
`system_prompt`	Large text blob, irrelevant to cost.
`model_config`	JSON config, not needed for pricing.
`handoff_*`	Operational state, not cost-relevant.
`rewind_count`	Retry metadata, not directly billable.
`archived`	Filter applied optionally, not a cost dimension.

---

6. Rate Table Design

Format

```
# ~/.hermes/cost-rates.yaml
version: 1
generated_from: "usage_pricing._OFFICIAL_DOCS_PRICING snapshot 2026-08-13"
```

```
rates:
  - provider: anthropic
    model: claude-sonnet-4-5
    input_per_million: 3.00
    output_per_million: 15.00
    cache_read_per_million: 0.30
    cache_write_per_million: 3.75
    request_cost: null
    billing_mode: chat_completions
    effective_from: 2025-01-01
    source: official_docs_snapshot
    notes: "Anthropic standard pricing"

  - provider: deepseek
    model: deepseek-chat
    input_per_million: 0.14
    output_per_million: 0.28
    cache_read_per_million: null
    cache_write_per_million: null
    request_cost: null
    billing_mode: chat_completions
    effective_from: 2026-03-16
    source: official_docs_snapshot
    notes: "DeepSeek chat — no cache pricing published"

  # ... (all entries from _OFFICIAL_DOCS_PRICING pre-seeded)
```

Resolution Order

1. Operator's `~/.hermes/cost-rates.yaml` (highest priority — manual overrides).
2. Hermes built-in `_OFFICIAL_DOCS_PRICING` from `agent/usage_pricing.py`.
3. OpenRouter models API (if provider is openrouter).
4. Endpoint models API (if base_url is available).
5. UNPRICED — flagged in report.

Effective Date Logic

When a session's `started_at` falls before a rate entry's `effective_from`, the previous entry (or built-in pricing) applies. This handles mid-period pricing changes correctly.

---

7. Report Format Specification

Console Table

```
╔══════════════════════════════════════════════════════════════════════════╗
║                  Hermes Cost Report: 2026-08-06 → 2026-08-13             ║
╠══════════════════════════════════════════════════════════════════════════╣
║ METERED SPEND (chat_completions)                                         ║
╠═══════════╦═══════════════╦══════════╦═══════╦═══════════════════════════╣
║ Provider  ║ Model         ║ Sessions ║ Cost  ║ Cache Read Cost           ║
╠═══════════╬═══════════════╬══════════╬═══════╬═══════════════════════════╣
║ deepseek  ║ deepseek-v4-flash ║    12 ║ $4.20 ║ $0.12                     ║
║ kimi-coding ║ kimi-k2.6     ║     8 ║ $0.00 ║ (included)                ║
║ xiaomi    ║ mimo-v2.5-pro ║     3 ║ $0.00 ║ (included)                ║
║ zai       ║ glm-4.5-flash ║     1 ║ $0.03 ║ $0.00                     ║
╠═══════════╩═══════════════╩══════════╩═══════╩═══════════════════════════╣
║                          Subtotal: $4.23                                 ║
╠══════════════════════════════════════════════════════════════════════════╣
║ QUOTA SPEND (subscription_included)                                      ║
╠═══════════╦═══════════════╦══════════╦═══════╦═══════════════════════════╣
║ Provider  ║ Model         ║ Sessions ║ Cost  ║ Notes                     ║
╠═══════════╬═══════════════╬══════════╬═══════╬═══════════════════════════╣
║ openai-codex ║ gpt-5.6-sol ║    15 ║ $0.00 ║ Codex Pro sub             ║
║ openai-codex ║ gpt-5.5     ║    10 ║ $0.00 ║ Codex Pro sub             ║
╠═══════════╩═══════════════╩══════════╩═══════╩═══════════════════════════╣
║                     Subtotal: $0.00 (included)                           ║
╠══════════════════════════════════════════════════════════════════════════╣
║ UNPRICED SESSIONS                                                        ║
╠═══════════╦═══════════════╦══════════╦═══════════════════════════════════╣
║ Provider  ║ Model         ║ Sessions ║ Tokens (in/out/cache)             ║
╠═══════════╬═══════════════╬══════════╬═══════════════════════════════════╣
║ [no-provider] ║ gpt-5.5   ║     4 ║ 0 / 0 / 0                         ║
╠═══════════╩═══════════════╩══════════╩═══════════════════════════════════╣
║ TOTAL ESTIMATED COST: $4.23                                              ║
║ 24 of 53 sessions fully priced. 4 sessions unpriced (0 tokens).          ║
╚══════════════════════════════════════════════════════════════════════════╝
```

Markdown File

Standard pipe-delimited table. Same sections. File named: `~/.hermes/cost-reports/2026-08-06_to_2026-08-13.md`

CSV File

Flat file with columns:
`date,provider,model,platform,sessions,input_tokens,output_tokens,cache_read_tokens,cache_write_tokens,reasoning_tokens,estimated_cost_usd,billing_mode,cost_status`

---

8. Edge Cases

Edge Case	Handling
Sessions with NULL model	Use `billing_provider` + `billing_base_url` for routing. If all three are NULL, treat as `[unknown-model]`.
Sessions with NULL billing_provider	Use `model` + `billing_base_url` for routing. If provider can be inferred from base_url (e.g., `api.deepseek.com` → `deepseek`), apply inferred provider.
`cost_status = 'included'` (quota)	Show as $0.00. Group under "Quota Spend" section. These are real sessions that used quota — report them for visibility even though cost is zero.
`billing_mode = NULL` for known providers	Infer mode: if provider is `openai-codex` → `subscription_included`. Otherwise default to `chat_completions` (metered).
Zero-token sessions (input=0, output=0, cache=0)	Show in session count. Cost = $0.00. No warning — these are valid (e.g., sessions that were started and immediately reset).
Sessions spanning a pricing change	Use `started_at` to select the rate entry with the latest `effective_from` that is <= session start.
Negative or absurd token counts	Flag as data anomaly. Exclude from cost totals. Show warning: "Session X has negative tokens — possible DB corruption."
`cache_read_tokens` huge relative to input	This is normal for long-lived sessions with system prompt caching. Cost is real. Do not flag.
Archived sessions	Included by default. Add `--exclude-archived` flag to filter.

9. CLI Interface

```
hermes-cost-report [OPTIONS]

Options:
  --days N              Look back N days from today (default: 7)
  --from DATE           Start date (YYYY-MM-DD)
  --to DATE             End date (YYYY-MM-DD, default: today)
  --format FORMAT       Output format: table (default), markdown, csv
  --output PATH         Output file path (default: ~/.hermes/cost-reports/...)
  --db PATH             Session database path (default: ~/.hermes/sessions.db)
  --rates PATH          Rate table path (default: ~/.hermes/cost-rates.yaml)
  --compare             Compare with previous period of same length
  --top N               Show top N most expensive sessions
  --by-platform         Also break down by platform (cli/telegram/cron)
  --watch               Real-time mode: regenerate on new sessions
  --quiet               Suppress console table, only write file
```

10. Assumptions

1. **Session store location is stable.** The tracker assumes sessions live at `~/.hermes/sessions.db` (or a path discoverable from Hermes config). If Hermes changes the DB location, the tracker needs updating.
1. **Token counts are accurate.** The tracker trusts `input_tokens`, `output_tokens`, `cache_read_tokens`, and `cache_write_tokens` as recorded by Hermes. It does not validate against raw API responses.
1. **Pricing changes are infrequent.** The rate table uses `effective_from` dating, not per-session pricing snapshots. If a provider changes pricing multiple times per week, the operator must add multiple rate entries.
1. **Single-user scope.** The tracker reports costs for one Hermes installation. Multi-user or multi-machine aggregation is out of scope.
1. **Cache-read tokens are billable.** Some providers include cache reads in a subscription or don't charge for them. The rate table handles this by setting `cache_read_per_million: 0.0` for those providers.
1. **`billing_provider` is the canonical vendor name.** Even when a session routes through OpenRouter or a proxy, `billing_provider` identifies the actual billing entity.

11. Open Questions

#	Question	Impact	Default if Unresolved
Q1	Should the rate table auto-update from provider APIs (OpenRouter models endpoint)?	Adds network dependency. Current design is offline-first.	No auto-update. Operator runs `hermes-cost-report --refresh-rates` manually when they suspect pricing changed.
Q2	Should the tracker write computed costs back to `sessions.estimated_cost_usd`?	Mutates the DB. Violates R7 (read-only). But enables faster queries for other tools.	No. Keep read-only. A separate "cost backfill" utility can be built if needed.
Q3	How to handle DeepSeek cache pricing? The snapshot has 651M cache_read tokens but DeepSeek's published pricing doesn't list a separate cache rate.	DeepSeek may charge cache reads at the input rate, or not at all.	Assume cache reads at 10% of input rate (DeepSeek's historical behavior). Flag with `notes: "cache rate assumed"` in rate table.
Q4	Should reasoning tokens be billed at a different rate than output tokens?	OpenAI o3 charges reasoning at the output rate. DeepSeek reasoner may differ.	Bill reasoning tokens at the output rate. This matches current provider behavior. Add `reasoning_cost_per_million` to rate table if a provider diverges.
Q5	Should the tracker support multiple session databases (e.g., per-profile)?	Hermes profiles can have separate session stores.	Yes, via `--db` flag. No auto-discovery of all profiles in v1.

Appendix A: Snapshot Database Summary

Data from inspection of `/tmp/hermes-cost-snapshot.db` (schema v17):

- **Sessions:** 301

- **Date range:** 2026-07-04 → 2026-08-13 (40 days)
- **Models (9):** deepseek-v4-flash, gpt-5.6-sol, kimi-k2.6, gpt-5.5, mimo-v2.5-pro, glm-4.5-flash, kimi-k1.5, deepseek-chat, gpt-4o
- **Billing providers (7):** openai-codex (112), deepseek (88), kimi-coding (58), xiaomi (18), zai (5), kimi-for-coding (1), NULL (19)
- **Platforms (5):** cron (170), cli (92), subagent (31), telegram (6), email (2)
- **cost_status:** included (111), unknown (171), NULL (19)
- **pricing_version:** All NULL
- **Token totals:** input 44M, output 5.4M, cache_read 651M, reasoning 375K
- **Sessions with NULL billing_provider:** 19 (all have 0 tokens)

Appendix B: Existing Pricing Engine

Hermes ships `agent/usage_pricing.py` with:

- `_OFFICIAL_DOCS_PRICING` — hardcoded pricing for Anthropic, OpenAI, Google, DeepSeek, MiniMax, Bedrock models.
- `resolve_billing_route()` — maps (model, provider, base_url) to a `BillingRoute` with billing_mode.
- `estimate_usage_cost()` — computes `CostResult` from `CanonicalUsage` + pricing entry.
- `normalize_usage()` — normalizes raw API response usage into canonical token buckets.

The cost tracker should import and reuse these functions rather than reimplementing pricing logic.

Appendix C: Provider Classification

Provider	billing_mode	Spend Type	Pricing Source
openai-codex	subscription_included	Quota ($0)	Built-in
deepseek	chat_completions	Metered	Official docs snapshot
kimi-coding	unknown → chat_completions	Metered	Operator rate table
kimi-for-coding	chat_completions	Metered	Operator rate table
xiaomi	unknown → chat_completions	Metered	Operator rate table
zai	unknown → chat_completions	Metered	Operator rate table
openrouter	official_models_api	Metered	OpenRouter models API
anthropic	official_docs_snapshot	Metered	Official docs snapshot
openai	official_docs_snapshot	Metered	Official docs snapshot

test2_prd/kimi-k2.6/cost-tracker-prd.md

Hermes Cost Tracker — Product Requirements Document

Version: 1.0-draft **Author:** Felix (kimi-k2.6) **Date:** 2026-08-13 **Research basis:** Official Hermes docs + read-only inspection of /tmp/hermes-cost-snapshot.db (301 sessions, 31 distinct days, 2026-07-04 → 2026-08-13)

---

1. Goal

Build a **read-only cost analytics tool** that computes estimated USD spend from the Hermes session store (`~/.hermes/state.db`). It must split metered API spend from quota/subscription usage, account for cache-read pricing, and emit both a console summary and a persistent report (Markdown or CSV) for operator review.

2. Non-Goals

- **No write access** to the session database under any circumstance.
- **No real-time billing API integration** (e.g., calling OpenRouter or provider dashboards); the tool is a local estimator only.
- **No prediction or forecasting** (e.g., "predict next month's bill").
- **No per-message cost breakdown** in v1; granularity stops at the session level.
- **No web UI or dashboard** in v1; outputs are CLI table + file.

---

3. Requirements (Must-Have First)

M1 — Core query engine

- Read from a configurable SQLite path (default: `~/.hermes/state.db`).
- Accept a date window (`--start`, `--end`, inclusive; default last 7 days).
- Aggregate by **vendor** (`billing_provider`), **model**, and **platform** (`source`).
- Compute cost from token counts using an **operator-maintained, dated rate table** (JSON or YAML), never from `estimated_cost_usd` or `actual_cost_usd` (both are unreliable in current Hermes builds).

M2 — Metered vs. Quota split

- Use `cost_status` to categorize rows:

- `included` → **Quota/Subscription** (display tokens but do not assign USD; flag as "included in subscription"). - `unknown` or `NULL` → **Metered** (compute USD using the rate table).

- If `billing_mode = 'subscription_included'`, treat as quota even if `cost_status` is anomalous.

M3 — Cache-read pricing

- The rate table must support three independent rates per model:

- `input_per_1m` — standard input tokens - `output_per_1m` — standard output tokens - `cache_read_per_1m` — cache-hit input tokens (often discounted, e.g., 10–50 % of input rate)

- `cache_write_tokens` and `reasoning_tokens` are tracked in the schema but **v1 treats them as zero-cost** (no public pricing for these categories in most APIs today). They are still reported as volume so the operator can see them.

M4 — Configurable, dated rate table

- File: `~/.hermes/cost-rates.json` (suggested default).
- Structure: array of dated entries so rates can be revised when providers change prices.

```
[
  {
    "effective_from": "2026-07-01",
    "effective_until": null,
    "rates": {
      "deepseek/deepseek-v4-flash": {
        "input_per_1m": 0.27,
        "output_per_1m": 1.10,
        "cache_read_per_1m": 0.07
      },
      "openai-codex/gpt-5.6-sol": {
        "input_per_1m": 0.0,
        "output_per_1m": 0.0,
        "cache_read_per_1m": 0.0,
        "note": "subscription_included"
      }
    }
  }
]
```

- Key format: `<billing_provider>/<model>` to resolve ambiguity when the same model string is routed through different providers (e.g., `gpt-4o` via `deepseek` base_url in the snapshot).

M5 — Report formats

1. **Console table** — rich text table printed to stdout with columns:

- Vendor | Model | Platform | Sessions | Input M | Output M | Cache-Read M | Reasoning M | Est. USD | Spend Type

1. **Markdown file** — human-readable, saved to a configurable path (default: `~/hermes-cost-report-<start>_<end>.md`).
2. **Optional CSV** — for spreadsheet import, same columns as console table plus a `note` column for estimate flags.

M6 — Read-only database access

- Open the database with `PRAGMA query_only = ON;`.
- Copy-on-write or snapshot is **not required** for v1 because WAL mode already allows concurrent readers, but the tool must never execute an `INSERT`, `UPDATE`, `DELETE`, or `CREATE`.

---

4. Data Freshness

Recommended mechanism: Weekly cron job

A scheduled automation (Hermes cron or system cron) regenerates the report every **Monday at 09:00 local time**.

Hermes cron setup (recommended):

```
hermes cron add --name "weekly-cost-report" \
  --schedule "0 9 * * 1" \
  --command "hermes-cost-tracker --days 7 --output ~/wiki/cost-reports/" \
  --deliver origin
```

If the operator prefers system cron:

```
0 9 * * 1 /usr/local/bin/hermes-cost-tracker --days 7 --output ~/wiki/cost-reports/
```

Real-time / near-real-time mode (optional, post-MVP)

A `--watch` flag that polls `state.db` every N minutes (default 5 min), detects new `started_at` rows since last run, and appends to a running report. This is **explicitly deferred** because:

- WAL mode keeps the DB locked during writes; rapid polling can collide with Hermes gateway processes.
- Cost estimates are more useful in aggregate (daily/weekly) than per-session.

---

5. Incomplete Data

5.1 Missing model or provider

Chosen approach: vendor/platform average estimate, clearly labeled.

If `model IS NULL` or `billing_provider IS NULL` but token counts are present:

1. Compute the average `input_per_1m`, `output_per_1m`, and `cache_read_per_1m` across all **non-NULL** sessions that share the same `source` and `cost_status`.
2. If no peer sessions exist in the window, fall back to the global average for that `cost_status`.
3. In every report row, append an `† est.` suffix to the model or provider name and add a footnote:

> † Estimated using <source> average rate because model/provider was missing in session record.

If both model **and** provider are NULL **and** tokens are zero, classify the session as **Aborted / No-Op** and exclude it from cost totals (still count it in a "Discarded Sessions" metadata line).

5.2 Missing token counts

If `input_tokens` or `output_tokens` are NULL but the session has a non-zero `message_count`:

- Flag as `‡ tokens missing`.
- Do **not** invent token volumes; leave cost as `N/A` and rely on the operator to back-fill rates or accept an incomplete report.

5.3 Model string drift

When a model name changes between API revisions (e.g., `gpt-5.5` vs `gpt-5.5-2026-07-15`), the rate table supports a `model_aliases` map so the operator can canonicalize names without touching the database.

---

6. Data Model Notes

6.1 Fields read from `sessions` table

Field	Type	Usage
`id`	TEXT	Session identifier; used only for deduplication/debug
`source`	TEXT	Platform: `cli`, `cron`, `telegram`, `subagent`, `email`, etc.
`model`	TEXT	Model name; may be NULL
`billing_provider`	TEXT	Vendor slug; may be NULL or differ from the model's native vendor (OpenRouter-style routing)
`billing_mode`	TEXT	`subscription_included`, `chat_completions`, NULL
`input_tokens`	INTEGER	Standard input tokens
`output_tokens`	INTEGER	Standard output tokens
`cache_read_tokens`	INTEGER	Cache-hit input tokens (massive in practice; must be priced separately)
`cache_write_tokens`	INTEGER	Tracked but priced at $0 in v1
`reasoning_tokens`	INTEGER	Tracked but priced at $0 in v1
`started_at`	REAL (unix epoch)	Date filtering
`cost_status`	TEXT	`included` = quota; `unknown` or NULL = metered
`estimated_cost_usd`	REAL	**Ignored** — currently populated only as 0.0 or sparse single entries
`actual_cost_usd`	REAL	**Ignored** — entirely NULL in observed data
`pricing_version`	TEXT	**Ignored** — entirely NULL in observed data

6.2 Indexes used

- `idx_sessions_started ( started_at DESC )` — date-range filtering.
- `idx_sessions_source` — optional platform filtering.

6.3 Computed columns

- `input_mm = input_tokens / 1_000_000.0`
- `output_mm = output_tokens / 1_000_000.0`
- `cache_read_mm = cache_read_tokens / 1_000_000.0`
- `cost_usd = (input_mm *input_rate)* + *(output_mm* output_rate) + (cache_read_mm * cache_read_rate)`

---

7. Edge Cases

Scenario	Handling
Session with missing model or provider	Use vendor/platform average estimate (see §5.1). Mark with `† est.`

Scenario	Handling
cost_status = "included"	Treat as quota/subscription. Show token volume, set USD to - and type to quota
Zero-token sessions (e.g., 19 cron rows in snapshot with 0 tokens, NULL model/provider)	Count in "Discarded Sessions" metadata; exclude from cost totals
Model routed through unexpected provider (e.g., gpt-4o via deepseek)	Use the exact billing_provider/model composite key in the rate table. If missing, fall back to model-only match, then to average
Cache-write tokens > 0 (not observed in snapshot, but schema supports it)	Include in volume report, price at $0, footnote: "cache-write pricing not yet configured"
Reasoning tokens > 0	Same as cache-write: report volume, price at $0
Date window with zero sessions	Emit a valid empty report (headers only) with a clear message: "No sessions found for 2026-08-01 → 2026-08-07"
Rate table missing a model/provider pair	Fall back to model-only rate, then global average for the same cost_status, then $0 with a ⚠ no rate flag
Negative or absurd token counts (data corruption)	Clamp to zero, log a warning line, include in report footnotes

8. Assumptions

1. **Hermes does not compute reliable USD internally.** The snapshot shows pricing_version is 100 % NULL, actual_cost_usd is 100 % NULL, and estimated_cost_usd is 99.7 % zero. Therefore the tool must be the source of truth for cost estimation.
2. **Cache-read pricing is provider-specific and non-uniform.** DeepSeek, OpenAI, and Kimi all use different cache discounts; the operator must maintain the rate table.
3. **Subscription/quota sessions are cost-opaque.** We can count their tokens for efficiency analysis, but we cannot assign a marginal dollar value without knowing the subscription price and total allowance.
4. **The database is in WAL mode.** Concurrent read during Hermes gateway operation is safe as long as we use query_only.
5. **Token counts in the session table are authoritative.** If they are NULL, we do not synthesize them from messages row counts.

9. Open Questions

1. **Should the tool support per-message granularity in v2?** The messages table has token_count per message, but rolling it up to session-level is simpler and matches provider billing granularity.
2. **How should we price reasoning tokens?** Some providers bill reasoning tokens at output rates, others at input rates, and some not at all. We need provider-specific rules before we can assign cost.
3. **Should we integrate OpenRouter's /models endpoint to auto-populate rates?** This would reduce operator maintenance but introduces a network dependency and potential stale-data risk.
4. **What is the canonical list of cost_status values?** The snapshot shows included, unknown, and NULL. Are there excluded, pending, or error states in newer Hermes builds?
5. **How should parent/child session lineage affect cost?** The schema has parent_session_id for compression-triggered splits. Should we roll up children into the parent session to avoid double-counting, or treat them as independent because each incurs separate API calls?

10. Success Criteria

- [ ] Running the tool on the snapshot database produces a non-zero metered cost for deepseek-v4-flash, kimi-k2.6, mimo-v2.5-pro, glm-4.5-flash, and kimi-k1.5.
- [ ] Running the tool reports included token volumes for gpt-5.6-sol and gpt-5.5 under openai-codex with USD shown as - or quota.
- [ ] The 19 zero-token NULL-model sessions appear in a "Discarded" footnote, not in the main cost table.
- [ ] Output files are deterministic: same DB state + same rate table = identical MD5.
- [ ] The tool exits 0 even when the rate table is missing entries, emitting warnings but still producing a report.

test2_prd/grok-4.6/cost-tracker-prd.md

Hermes Cost Tracker — Product Requirements Document

Status: reviewed — ready for an implementation ticket (research + design only; nothing implemented) Author: Felix Date: 2026-08-13 Board: cost-prd-grok-4.6 (slug cost-prd-grok-4-6) Evidence base: official Hermes docs (session-storage, sessions, providers, cron, insights) and a read-only inspection of /tmp/hermes-cost-snapshot.db (copy of a real session store; never modified; file remains mode 0444, 301 sessions, pricing_version still all-NULL).

1. Goal

Give the operator a trustworthy, date-windowed answer to: **what did this Hermes fleet actually spend, on which vendor / model / platform, and how much of that was metered cash vs subscription quota?**

Hermes already records tokens, provider, platform, and billing *status* on each session. It does **not** persist usable USD. In the snapshot, `pricing_version` is NULL on every row, `actual_cost_usd` is NULL on every row, and `estimated_cost_usd` is 0.00 on 300 of 301 sessions (the one nonzero value is a stale `deepseek-chat` figure that cannot be reproduced from the current built-in rate table). Built-in `hermes insights` therefore reports most real spend as $0.

The Cost Tracker exists to close that gap: **recompute USD from session token buckets × an operator-owned, dated rate table**, including cache-read charges, split metered vs quota, and emit a console table plus a durable markdown or CSV file. It is a **read-only reporter**, not a writer of the session store.

Success looks like: for any date window, the operator can open one report and see (a) cash-equivalent metered spend by vendor/model/platform, (b) quota-included volume that is not cash, (c) cache-read dollars as a first-class line, (d) every incomplete or unpriced session labeled, never silently zeroed.

---

2. Non-goals

- Do not implement in this PRD task. This document is requirements only.
- Do not write to any session database (`state.db`, WAL, or the snapshot). No `UPDATE/INSERT/DELETE`.
- Do not replace `hermes insights`, `/usage`, or the dashboard Analytics tab. Those stay for live session recap. This tool is the fleet spend ledger.
- Do not scrape vendor billing portals or reconcile invoices in v1.
- Do not persist `pricing_version` or backfill `estimated_cost_usd` into Hermes core.
- Do not estimate missing token counts from vendor/platform averages. Token fabrication would invent spend.
- Do not treat cache-read tokens as full-price input. That would overstate DeepSeek-class bills by ~50×.
- Do not build a web dashboard, charts product, or Telegram bot UI in v1. Console + one file is the report surface. Cron may *deliver* the file/summary to Telegram; that is delivery, not a new UI.
- Do not price tools billed outside the session store (Tavily/extract, Parallel search, image gen) in v1. Call that out as a known hole.
- Do not mutate live cron jobs, profiles, or rate cards without an explicit operator action.

---

3. Research findings (normative context)

3.1 Where the data lives

Hermes persists every conversation in SQLite (`~/.hermes/state.db`, WAL mode). Profiles are isolated: each profile has its own `~/.hermes/profiles/<name>/state.db`. The snapshot is schema version **17** (docs still describe version 11; the live `schema_version` table is the truth).

Canonical session columns the tracker will read:

Field	Role
`id`	Session primary key
`source`	Platform: `cli`, `cron`, `telegram`, `email`, `subagent`, …
`model`	First model written for the session (first-writer-wins)
`billing_provider`	First provider written (`openai-codex`, `deepseek`, `kimi-coding`, `xiaomi`, `zai`, …)
`billing_base_url`	Endpoint used (normalize before grouping)
`billing_mode`	Mixed signal: `subscription_included`, `chat_completions`, or NULL
`started_at, ended_at`	Unix epoch floats. Filter windows on `started_at`. `ended_at` is NULL on open sessions (101/301).
`input_tokens, output_tokens`	Uncached input / completion totals
`cache_read_tokens, cache_write_tokens`	Prompt-cache buckets. Snapshot: 651,073,886 cache-read; **0** cache-write.
`reasoning_tokens`	Recorded; not billed separately unless a rate row says so
`estimated_cost_usd, actual_cost_usd`	**Do not use as source of truth**
`cost_status`	Last-call status: `included` / `unknown` / NULL (enum also has `estimated`, `actual`)
`cost_source, pricing_version`	`pricing_version` is **always NULL** in the snapshot
`api_call_count, message_count, tool_call_count`	Volume, not dollars
`parent_session_id`	Compression lineage. Child rows hold *new* tokens; summing rows does not double-count.
`archived`	Exclude archived rows from default reports
`title`	Optional label in detail rows

`messages.token_count` is NULL on all 14,962 snapshot messages. **Cost lives only on sessions.** Do not aggregate messages for spend.

Open the live DB read-only:

```
sqlite3.connect("file:<path>?mode=ro", uri=True)
```

Never open the production file read-write. For tests, use a copy (the snapshot pattern).

3.2 Why stored USD is unusable

Per API call, Hermes runs `estimate_usage_cost()` in `agent/usage_pricing.py`:

```
amount = input × input$/M
       + output × output$/M
       + cache_read × cache_read$/M
       + cache_write × cache_write$/M
       + request_count × request_cost
```

Then `update_token_counts()` increments tokens and cost. Callers **never pass `pricing_version`**, so it stays NULL. If a route has no pricing entry, `amount_usd` is None and the increment stores +0. If cache-read tokens exist and the entry has no cache-read rate, the whole call is `unknown` (strict bail; no fallback to the input rate).

The built-in official-docs snapshot covers Anthropic / classic OpenAI / `deepseek-chat` / `deepseek-v4-pro` / Gemini 2.x / Bedrock / MiniMax. It does **not** cover the snapshot's workhorses: `deepseek-v4-flash` (78+ sessions), `gpt-5.6-sol`, `kimi-k2.6`, `gpt-5.5`, `mimo-v2.5-pro`, `glm-4.5-flash`. That is why almost every metered session is `cost_status=unknown` and `$0.00`.

`hermes insights` recomputes with the same built-in table and then does `float(amount_usd or 0.0)`, so unknown spend collapses to zero. The Cost Tracker must not repeat that.

3.3 Cache-read billing

Hermes normalizes provider usage so `input_tokens` is **cache-miss only**. Cache hits land in `cache_read_tokens`. Prompt total = input + cache_read + cache_write.

In this fleet, cache reads dominate: ~651M cache-read vs ~44M input vs ~5.4M output. Ignoring cache-read rates understates metered cost by roughly an order of magnitude. Using the input rate for cache reads overstates DeepSeek-class cost by ~50× (Flash cache-hit is about $0.0028/M vs $0.14/M miss).

Rule: **bill each bucket at its own rate.** If a metered session has `cache_read_tokens > 0` and the matching rate row has no `cache_read_usd_per_m`, the session is `UNPRICED` (tokens shown, USD excluded from totals). Do not silently substitute the input rate.

3.4 Quota vs metered

There is no schema flag named "quota". Classification:

Class	How we detect it	Snapshot evidence
Quota / subscription-included	Operator map says `billing_class: quota`, **or** `cost_status='included'`, **or** `billing_mode='subscription_included'` when the map is silent	`openai-codex`: 111/112 sessions `included` (ChatGPT/Codex OAuth). USD = $0 cash, tokens still reported as quota volume.
Metered	Operator map says `metered`, or provider is a pay-per-token API (`deepseek`, `kimi-coding`, `xiaomi`, `zai`, `openai` API key, `anthropic` API key, `openrouter`, …)	All of these are `unknown` in the snapshot because rates are missing from the built-in table.
Prepaid credits (Nous Portal, some coding plans)	Operator map `billing_class: credits`	Not present in the snapshot. Report as a third bucket: not cash, not ChatGPT-style quota.

`billing_mode` alone is **not** reliable: one `deepseek` row is labeled `subscription_included` with `cost_status=unknown`; one `openai-codex` row is `subscription_included` but `cost_status=unknown`. The operator map wins; stored status is advisory.

Official provider docs also matter for classification: Anthropic Claude Max OAuth in Hermes consumes **purchased extra credits**, not the base Max allowance — treat that path as metered credits, not free quota. Nous Portal is a subscription / credit pool. xAI SuperGrok OAuth is quota; `XAI_API_KEY` is metered.

3.5 Incomplete rows in the snapshot

- **Model is never missing** (0/301).
- **Provider missing: 19/301.** All are `source=cron`, 0 tokens, 1 message, 0 API calls, `end_reason=cron_complete`, `cost_status` NULL. These are no-LLM cron ticks (script-only or empty). They are not spend.
- **Zero-token sessions: the same 19.**
- **Open sessions: 101/301** (`ended_at` NULL). Token totals are already cumulative; include them if `started_at` is in the window.

---

4. Incomplete-data policy (chosen approach)

Chosen approach: documented exact path — an operator-maintained mapping + dated rate table. No silent vendor/platform averages.

Rationale, grounded in the snapshot:

1. Every missing-provider row is a zero-token cron ghost. Averaging those would invent spend.

2. Cache-read rates are 10–50× cheaper than input rates. A "vendor average $/token" would be wrong by an order of magnitude.
3. Stored USD is unversioned and almost always zero. Guessing from it is worse than leaving a hole labeled.
4. The operator already has to maintain rates for models Hermes does not price. Identity mapping belongs in the same file.

4.1 Resolution order (exact, no improvisation)

For each session in the window:

1. **Identity.** `model` and `billing_provider` as stored.
2. **Alias map** (`provider_aliases`, `model_aliases` in the rate file). Example: `kimi-for-coding` → `kimi-coding`.
3. **If provider is missing and model is present:** apply `model_to_provider` from the operator map. If that key is absent → `UNMAPPED`.
4. **If model is missing:** `UNMAPPED`. Do not infer a model from other sessions.
5. **If all token buckets are 0 or NULL:** `ZERO_TOKEN`. USD = $0. Do not impute tokens.
6. **Billing class** from the operator map (`quota` / `metered` / `credits`). If the map has no class, fall back to stored `cost_status='included'` or `billing_mode='subscription_included'` → `quota`; else `metered`.
7. **Rate lookup** by `(provider, model)` whose `[effective_from, effective_to)` contains `started_at` (UTC date of `started_at`).
8. **If no rate row and class is metered or credits:** `UNPRICED`. Tokens counted, USD excluded from cash totals.
9. **If rate row exists:** compute USD from the four token buckets. If `cache_read_tokens > 0` and `cache_read_usd_per_m` is null → `UNPRICED` (same for cache-write).
10. **Never** use `estimated_cost_usd` / `actual_cost_usd` as a substitute rate.

4.2 How every report marks estimates and holes

The operator may still *choose* to put an estimated rate in the table (`confidence: estimated`). That is an explicit, dated, reviewable number — not a runtime average.

Every console, markdown, and CSV row carries a `flag` column:

Flag	Meaning	In cash total?
`QUOTED`	Rate row `confidence: quoted` (vendor docs / invoice)	Yes
`EST`	Rate row `confidence: estimated` (operator judgment)	Yes, but rolled up under an **Estimates** subtotal
`INCLUDED`	Quota / subscription. Cash = $0	No (quota volume only)
`CREDITS`	Prepaid credit pool (Nous, extra Max credits)	Separate credits total, not cash
`UNPRICED`	Tokens known, no usable rate (or missing cache rate)	No
`UNMAPPED`	Missing model/provider after the map	No
`ZERO_TOKEN`	No tokens / no API calls	No ($0, listed in incomplete appendix)

Footer on every report:

```
Cash total = QUOTED + EST
EST share  = $X.XX (N sessions) — see rate rows tagged confidence=estimated
Excluded   = UNPRICED $— (N sess, T tokens) + UNMAPPED (N) + ZERO_TOKEN (N)
Quota      = INCLUDED tokens (not cash)
```

No report may print `$0.00` for an `UNPRICED` or `UNMAPPED` session. Print `—` for USD and the flag.

v1 does **not** compute vendor/platform averages at runtime. If a future version adds `estimate_from: vendor_average`, it must still emit `EST` and an explanation column (`avg of quoted rows for provider X in this window`). That is out of scope until the operator asks for it.

---

5. Requirements

5.1 Must-have (v1)

M1. Date-windowed recompute from the session store. CLI (name can be `hermes-cost` or `python -m hermes_cost`):

```
hermes-cost --from 2026-08-01 --to 2026-08-13
hermes-cost --last 7d
hermes-cost --last 1w
```

Window filter: `started_at >= from 00:00:00` and `started_at < to+1day` in `--timezone` (default `America/Los_Angeles`). Include open sessions. Exclude `archived=1` unless `--include-archived`.

M2. Group by vendor, model, and platform. Three breakdowns, plus a grand total:

- vendor = normalized `billing_provider`
- model = normalized `model`
- platform = `source`

Each line: sessions, API calls, input, output, cache-read, cache-write, reasoning, cash USD, quota tokens, flags summary.

M3. Split metered vs quota (and credits). Top of every report: cash (metered QUOTED+EST), credits, included-quota volume, excluded incomplete. Do not add quota $0 into cash to make a fake “total cost”.

M4. Include cache-read costs. Always show cache-read tokens and cache-read USD as their own columns. Formula is the four-bucket sum in §3.2. Missing cache-read rate → `UNPRICED`, not a silent input-rate fallback.

M5. Configurable, dated rate table. Operator-owned file (YAML). Default path: `~/.hermes/cost-tracker/rates.yaml`. Each row:

```
- provider: deepseek
  model: deepseek-v4-flash
  billing_class: metered          # metered | quota | credits
  effective_from: 2026-04-24      # inclusive, ISO date
  effective_to: null              # exclusive; null = open
  input_usd_per_m: 0.14
  output_usd_per_m: 0.28
  cache_read_usd_per_m: 0.0028
  cache_write_usd_per_m: null
  reasoning_usd_per_m: null       # null = do not bill separately
  confidence: quoted              # quoted | estimated
  source: https://api-docs.deepseek.com/quick_start/pricing
  notes: "pre-peak/off-peak; revisit after 2026-08-16 16:00 UTC"
```

Also in the same file:

```
provider_aliases:
  kimi-for-coding: kimi-coding
  mimo: xiaomi
model_aliases:
  kimi-k2.6: kimi-k2.6
model_to_provider:
  deepseek-v4-flash: deepseek
  gpt-5.5: openai-codex
  gpt-5.6-sol: openai-codex
  kimi-k2.6: kimi-coding
  mimo-v2.5-pro: xiaomi
  glm-4.5-flash: zai
provider_class:
  openai-codex: quota
  nous: credits
  deepseek: metered
  kimi-coding: metered
  xiaomi: metered
  zai: metered
```

Price changes reprice **history** by `started_at` vs `effective_from`/`effective_to`. A new row with a new `effective_from` does not rewrite earlier days. Math is `Decimal`, report USD at 4 decimal places (sub-cent DeepSeek cache hits otherwise vanish).

Ship a starter table covering the snapshot workhorses, clearly marked `quoted` vs `estimated`, with source URLs. The operator edits the file; the app does not scrape the web at runtime in v1.

M6. Report formats: console table + one durable file. Default: print a console table and write **both** markdown and CSV to a reports directory.

```
hermes-cost --from 2026-08-01 --to 2026-08-13 \
  --out ~/.hermes/cost-tracker/reports/
```

Produces:

- `hermes-cost-2026-08-01-2026-08-13.md`
- `hermes-cost-2026-08-01-2026-08-13.csv`

`--format md|csv|both` (default `both`). `--no-file` for console-only. Markdown is the human artifact (Telegram-friendly). CSV is the spreadsheet artifact.

Required report sections:

1. Header: window, timezone, generated-at, DB paths, rate-file path + hash, schema versions
2. Totals: cash / EST share / credits / quota tokens / excluded
3. Table by vendor
4. Table by model
5. Table by platform
6. Incomplete appendix: every `UNPRICED` / `UNMAPPED` / `ZERO_TOKEN` session id, model, provider, tokens, flag
7. Rate-table appendix: which rows were applied

M7. Read-only database access. Open every DB with `mode=ro`. Refuse to start if the configured path is the snapshot and the user passed a write flag (there is no write flag in v1). Never copy over the original `state.db`. Optional `--db <path>` for a single file (tests use `/tmp/hermes-cost-snapshot.db`). Default is **all profile DBs** (§6).

M8. Multi-profile aggregation. Default scan:

- `~/.hermes/state.db` if present
- `~/.hermes/profiles/*/state.db`

Tag each session with `profile` (directory name, or `default`). Add a fourth breakdown: by profile. Live machines already split spend across `felix`, `milo`, `researcher`, `critic`, `visual-reviewer`. A single-DB report would undercount the fleet.

M9. Scheduled freshness — weekly no-agent cron (recommended default). See §6. On-demand CLI is always available. Real-time `--watch` is optional, not default.

5.2 Should-have (v1.1)

- `--watch 60` : re-read WAL every N seconds and reprint the current window (for a live CLI session). Do not cron this.
- Lineage rollup: optional `--by-conversation` following `parent_session_id`.
- Peak / off-peak rate windows (DeepSeek announced peak pricing effective 2026-08-16 16:00 UTC).
- `--profile felix` to restrict to one DB.
- Compare two windows (`--vs last-week`).

5.3 Nice-to-have (later)

- Invoice reconciliation, vendor portal scrape, dashboard plugin, per-tool (Tavily) ledger, writing `pricing_version` back into Hermes.

---

6. Data freshness

6.1 Recommended mechanism

Weekly regeneration via a no-agent Hermes cron job, plus on-demand CLI.

Why weekly, not real-time, as the default:

- Token counters on open sessions are already cumulative, but a mid-turn read is a partial turn. Weekly reports after a quiet hour (Monday 08:00 local) are stable enough for vendor planning.
- The report is a ledger, not a status bar. `hermes insights` / `/usage` already cover “this session”.
- A no-agent cron costs **zero inference tokens**. An agent cron that *talks* about cost would add spend to the thing being measured.
- Ken’s gateway note: local-only cron output is not delivered back into a CLI session. Delivery must target Telegram if a human should see it.

Real-time mode is supported as `hermes-cost --watch` for interactive use. It is not the scheduled default.

6.2 Exact cron and config setup

1. Install the reporter script at `~/.hermes/scripts/hermes-cost-weekly.sh` (Hermes no-agent jobs must live under `~/.hermes/scripts/`):

```
#!/usr/bin/env bash
set -euo pipefail
# Read-only fleet cost report for the previous ISO week (Mon–Sun), Pacific time.
export TZ=America/Los_Angeles
FROM=$(date -d "last monday - 7 days" +%F)
TO=$(date -d "last sunday" +%F)
OUT="${HERMES_HOME:-$HOME/.hermes}/cost-tracker/reports"
mkdir -p "$OUT"
hermes-cost --from "$FROM" --to "$TO" --timezone America/Los_Angeles \
  --out "$OUT" --format both
# stdout is what Telegram receives (keep it short)
echo "Hermes cost $FROM → $TO"
echo "Report: $OUT/hermes-cost-$FROM-$TO.md"
tail -n +1 "$OUT/hermes-cost-$FROM-$TO.md" | head -n 80
```

2. Create the job once (gateway must be running; do not schedule this from inside a cron session):

```
hermes cron create "0 8 * * 1" \
  --no-agent \
  --script hermes-cost-weekly.sh \
  --name hermes-cost-weekly \
  --deliver telegram
```

Meaning: every Monday at 08:00 America/Los_Angeles (the host already lists cron next-runs at `-07:00`), run the script with no LLM, deliver stdout to Telegram. Empty stdout would be a silent tick; the script always prints a header so a successful run is visible. Non-zero exit delivers an error alert.

Equivalent tool call, if created from chat rather than CLI:

```
cronjob(
    action="create",
    schedule="0 8 * * 1",
    no_agent=True,
    script="hermes-cost-weekly.sh",
    name="hermes-cost-weekly",
    deliver="telegram",
)
```

3. App config at `~/.hermes/cost-tracker/config.yaml`:

```
timezone: America/Los_Angeles
db_globs:
  - ~/.hermes/state.db
  - ~/.hermes/profiles/*/state.db
```

```
rates_file: ~/.hermes/cost-tracker/rates.yaml
reports_dir: ~/.hermes/cost-tracker/reports
read_only: true
default_window: last_week          # last_week | last_7d | month_to_date
exclude_archived: true
watch_interval_seconds: 60         # only used with --watch
```

`--db` on the CLI overrides `db_globs` (single file, used for the snapshot and tests).

4. Acceptance for freshness

- After the cron fires, `reports/hermes-cost-<from>-<to>.md` exists and `mtime` is within 15 minutes of the scheduled time.
- The markdown header lists every DB path that was opened and the rate-file sha256.
- Re-running the same window is idempotent (overwrite same filenames; no extra rows).
- The job never opens a DB without `mode=ro`.
- Killing the gateway pauses future runs; it does not corrupt reports.

6.3 On-demand and optional real-time

```
hermes-cost --last 7d                  # now
hermes-cost --db /tmp/hermes-cost-snapshot.db --from 2026-07-04 --to 2026-08-13
hermes-cost --watch 60 --last 1d       # optional; not cron
```

`--watch` polls read-only. It does not use SQLite update hooks (those require a writer connection).

---

7. Data model notes (what the app reads and how)

Source	Fields used	How
`sessions`	see §3.1	One cost unit per row. Window on `started_at`.
`messages`	none in v1	`token_count` unpopulated.
`schema_version`	`version`	Header only.
Rate YAML	rows + aliases + classes	Dated lookup.
Config YAML	paths, timezone	Startup.
Filesystem	`~/.hermes/profiles/*/state.db`	Profile dimension.

Computation (per session, after identity resolution):

```
cash = 0
if billing_class == quota:
    flag = INCLUDED
elif no rate:
    flag = UNPRICED
else:
    cash  = in   * input_usd_per_m    / 1e6
    cash += out  * output_usd_per_m   / 1e6
    cash += cr   * cache_read_usd_per_m / 1e6   # required if cr>0
    cash += cw   * cache_write_usd_per_m / 1e6  # required if cw>0
    flag  = EST if confidence==estimated else QUOTED
```

`reasoning_tokens` are informational unless `reasoning_usd_per_m` is set. Do not add them on top of `output_tokens` by default (providers usually include reasoning in the output charge).

Do not read `estimated_cost_usd` except as a debug column behind `--show-stored-usd` (never added into cash).

---

8. Edge cases

Case	Behavior
Missing model	`UNMAPPED`. USD —. Listed in appendix. No average.
Missing provider, model present	Apply `model_to_provider`. If absent → `UNMAPPED`.
Missing / zero token counts	`ZERO_TOKEN`. USD $0.00. In snapshot these are 19 cron ticks with `api_call_count=0`. Shown in appendix so they cannot be mistaken for free quota.
`cost_status='included'`	Cash $0. Tokens under Quota. Flag `INCLUDED`. Even if a metered rate exists for that model name (e.g. `gpt-5.5` also exists as an API SKU), the **session's provider class** wins: `openai-codex` + included = quota.
`billing_mode='subscription_included'` but `cost_status='unknown'`	Trust the operator `provider_class`. For `openai-codex`, still `INCLUDED`. For `deepseek`, treat as metered (the snapshot has this contradiction).
Zero-token **and** `included`	Still `ZERO_TOKEN` if no tokens; do not count as quota volume.
Open session (`ended_at` NULL)	Include. Tokens are cumulative-as-of-now. Header notes "includes N open sessions".
Mid-session `/model` switch	First model/provider wins (Hermes first-writer). Limitation, stated in the report footer.
Compression lineage	Default: sum rows independently (no double-count). Optional later: `--by-conversation`.
`source='subagent'`	Own row, own cost. Platform breakdown shows `subagent`. Do not silently fold into parent.
Provider aliases (`kimi-coding` vs `kimi-for-coding`)	Normalize via `provider_aliases` before grouping.
Base URL trailing slash	Not a grouping key in v1 (provider + model only).
Cache-write always 0	Still print the column. Rate table still accepts `cache_write_usd_per_m`.
Peak/off-peak (DeepSeek after 2026-08-16)	v1 uses a single rate per `effective_from` date. Split peak rows are v1.1. Until then, pick the conservative (peak) quoted rate or mark `EST` and note it.
Multiple profile DBs with the same `sessions.id`	Unlikely (IDs are timestamp+hex). If it happens, key is `(profile, id)`.
Snapshot vs live	`--db /tmp/hermes-cost-snapshot.db` for golden tests. Default is live globs. Never write either.

---

9. Starter rate-table guidance (not runtime defaults the app invents)

These are **seed rows the operator should review** before the first production report. They are not computed from the snapshot.

Provider	Model	Class	Input $/M	Output $/M	Cache-read $/M	Confidence	Notes
openai-codex	gpt-5.5	quota	0	0	0	quoted	ChatGPT/Codex subscription. Cash $0.
openai-codex	gpt-5.6-sol	quota	0	0	0	quoted	Same.
deepseek	deepseek-v4-flash	metered	0.14	0.28	0.0028	quoted	Official API cache-hit vs miss. Peak/off-peak starts 2026-08-16 16:00 UTC — revisit.
deepseek	deepseek-chat	metered	0.14	0.28	0.0028	estimated	Built-in Hermes snapshot has **no** cache-read rate; operator should add one or these stay UNPRICED.
kimi-coding	kimi-k2.6	metered	0.95	4.00	0.16	estimated	Moonshot input/output are published; cache-hit for K2.6 is inconsistently documented ($0.16 vs unpublished). Mark EST until the operator confirms.
xiaomi	mimo-v2.5-pro	metered	0.435	0.87	0.0036	quoted	Xiaomi pay-as-you-go card.
zai	glm-4.5-flash	metered	*operator fills*			estimated	Confirm against current Z.AI SKU. If the plan is a coding subscription, flip billing_class to credits or quota.

If the operator ships the app without filling a row, that model reports UNPRICED rather than a guessed number. That is the correct failure mode.

---

10. Acceptance criteria

Using /tmp/hermes-cost-snapshot.db (read-only) and a reviewed rate file:

1. --from 2026-07-04 --to 2026-08-13 accounts for all 301 snapshot sessions in the header. 282 token-bearing sessions appear in the vendor/model/platform tables; the 19 ZERO_TOKEN cron ghosts appear only in the incomplete appendix (still counted, never treated as free quota).
2. openai-codex cash = $0.00 and appears under Quota, not Cash.
3. deepseek / kimi-coding / xiaomi / zai appear under Metered; USD is either a computed number or — + UNPRICED, never a silent $0.
4. Cache-read tokens on the grand total equal SUM(cache_read_tokens) from SQL (651,073,886).
5. The 19 cron ghosts are ZERO_TOKEN / UNMAPPED, not free quota.
6. Console + .md + .csv all carry the flag column and the footer in §4.2.
7. Opening the snapshot with lsof / checking mtime after a run shows **no write**.
8. A second run overwrites the same filenames and produces byte-stable totals (except generated-at).
9. Weekly cron is documented as --no-agent + --script hermes-cost-weekly.sh + --deliver telegram + 0 8 1.

---

11. Assumptions

- The snapshot is representative of Ken’s fleet shape (cron-heavy, cache-read-heavy, Codex + DeepSeek + Kimi + Xiaomi + ZAI). Live profile DBs are larger and must be included by default.
- started_at is the right window key. Using ended_at would drop 101 open sessions.
- First-writer model/provider is an acceptable v1 limitation.
- Reasoning tokens are not double-billed against output unless a rate row says so.
- Tool-metered spend (search/extract) stays out of v1 even though it is real monthly cash.
- America/Los_Angeles is the operator timezone (existing cron next-runs are -07:00).
- Telegram is the human delivery channel for the weekly file summary.
- Decimal USD, 4 places, is enough; we will not track VAT or FX.
- The Cost Tracker is a standalone CLI + script, not a new Hermes core tool (keeps the core tool schema small).

---

12. Open questions

1. Should Nous Portal sessions (none in the snapshot) be credits or quota? Recommendation: credits, so they do not look like ChatGPT-included.
2. Confirm Kimi K2.6 cache-hit: $0.16/M vs unpublished. Until confirmed, the seed row stays estimated or omitted (UNPRICED).
3. Confirm Z.AI glm-4.5-flash cash vs coding-plan quota.
4. After 2026-08-16 16:00 UTC, does the operator want DeepSeek peak/off-peak rows immediately (v1.1) or a single conservative peak rate?
5. Fold subagent cost into the parent session for “what did this chat cost”, or keep it separate? Default: separate.
6. Should the weekly Telegram message be the 80-line summary only, with the full markdown left on disk? (Recommended: yes.)
7. One rate file for the whole host, or per-profile overrides? Default: one host file.

8. Is there appetite to later write `pricing_version` into Hermes so `insights` stops showing $0? Out of scope here; would be a core PR.

---

13. Implementation sketch (out of scope, for the next ticket)

Not part of this task. When implementation is authorized:

- Package as a small CLI that depends only on Python stdlib + PyYAML (or reuse the Hermes venv).
- Golden test: `--db /tmp/hermes-cost-snapshot.db` vs checked-in expected totals.
- Do not import `agent.usage_pricing.estimate_usage_cost` as the source of truth (it lacks dated rows and workhorse models). Optionally use it only as a comparison debug flag.
- Keep the weekly script under `~/.hermes/scripts/` so no-agent cron will run it.

---

14. Sources

- Hermes session storage: `website/docs/developer-guide/session-storage.md` and https://hermes-agent.nousresearch.com/docs/user-guide/sessions
- Hermes providers / subscription vs API: https://hermes-agent.nousresearch.com/docs/integrations/providers
- Hermes cron, including `--no-agent`: https://hermes-agent.nousresearch.com/docs/user-guide/features/cron
- Hermes insights CLI: https://hermes-agent.nousresearch.com/docs/reference/cli-commands
- Code: `agent/usage_pricing.py`, `hermes_state.py` (`update_token_counts`), `agent/conversation_loop.py`, `agent/insights.py`
- Snapshot: `/tmp/hermes-cost-snapshot.db` (read-only; 301 sessions, 2026-07-04 → 2026-08-13, schema 17)
- Vendor rate references for the *seed* table only: DeepSeek official pricing, Xiaomi MiMo pay-as-you-go, Moonshot/Kimi public pricing pages

test2_prd/gpt-5.5/cost-tracker-prd.md

Hermes Cost Tracker PRD

Executive Summary

The Hermes Cost Tracker is a read-only analytics tool designed to provide visibility into AI model usage and costs across Hermes sessions. It transforms raw session data from the Hermes session store into actionable insights through cost analysis, vendor/platform comparisons, and configurable reporting.

Goal and Non-Goals

Goals

- Provide comprehensive cost visibility for AI model usage across Hermes sessions
- Enable vendor/platform/model cost comparison and optimization
- Support flexible date range filtering for historical analysis
- Generate multiple output formats (console table, markdown, CSV)
- Handle cache-read cost tracking accurately
- Support both metered and quota-based billing models
- Provide configurable rate tables for accurate cost calculation

Non-Goals

- **No cost management**: This is purely an analytics tool, not a cost control system
- **No real-time billing**: Does not interact with provider billing APIs or payment systems
- **No session modification**: Read-only access to session store
- **No predictive forecasting**: Only analyzes historical data
- **No automated cost optimization**: Does not recommend or implement model switching
- **No multi-tenant billing**: Single-user/cost-center analysis only

---

Requirements

Must-Have (Core Functionality)

1. **Database Access & Querying**

- Read-only access to Hermes SQLite session store (~/.hermes/sessions.db) - Query sessions by date range, vendor, model, platform - Handle database schema changes gracefully - Support large datasets (efficient pagination for 100K+ sessions)

1. **Cost Calculation Engine**

- Compute cost per vendor/model/platform for specified date windows - Support metered billing (pay-per-token) and quota-based billing (subscription) - Include cache-read token costs in calculations - Apply configurable rate tables for pricing - Handle both estimated and actual cost data when available

1. **Billing Model Support**

- **Metered**: Calculate costs based on input/output/cache tokens - **Quota**: Track included usage vs. overage (cost_status="included") - **Mixed**: Handle sessions with different billing modes

1. **Reporting & Output**

- Console table output with summary statistics - Markdown report with detailed breakdowns - CSV export for external analysis - Support for filtering and sorting in all formats

1. **Data Freshness**

- Cron job support for scheduled report generation - Default: Weekly reports (recommended) - Configurable date ranges for manual runs - Incremental update capability for new sessions

Should-Have (Enhanced Functionality)

1. **Advanced Filtering**

- Filter by session source (chat, cron, etc.) - Filter by specific models or vendors - Filter by cost status (included, unknown, etc.) - Time-based filtering (hourly, daily, weekly aggregations)

1. **Cost Analysis Features**

- Cost trend analysis over time - Usage patterns (peak vs. off-peak) - Cost efficiency metrics (cost per token, cost per session) - Budget vs. actual comparisons

1. **Export & Integration**

- JSON export for API integration - Support for external BI tools - Email delivery of reports - Web dashboard preview (HTML export)

Nice-to-Have (Future Enhancements)

1. **Advanced Analytics**

- Predictive cost modeling - Anomaly detection for unusual spending - Cost optimization recommendations - Multi-dimensional cost analysis

1. **User Experience**

- Interactive web dashboard - Real-time cost monitoring - Cost alerts and notifications - Custom report templates

---

Data Model & Fields

Primary Data Source

Hermes Sessions Table (sessions):

Field	Type	Description	Usage
id	TEXT	Unique session identifier	Primary key
started_at	REAL	Unix timestamp	Date filtering, time windows
ended_at	REAL	Unix timestamp	Session duration, cost analysis
model	TEXT	Model name (e.g., "gpt-5.5")	Cost calculation, grouping
billing_provider	TEXT	Provider name (e.g., "openai-codex")	Vendor analysis, rate lookup
billing_mode	TEXT	Billing type ("chat_completions", "subscription_included")	Cost calculation logic
cost_status	TEXT	Cost status ("included", "unknown")	Quota vs. metered handling
pricing_version	TEXT	Pricing version (always NULL in current schema)	N/A - not implemented
input_tokens	INTEGER	Input token count	Cost calculation
output_tokens	INTEGER	Output token count	Cost calculation
cache_read_tokens	INTEGER	Cache read token count	Cost calculation (important)
cache_write_tokens	INTEGER	Cache write token count	Cost calculation
estimated_cost_usd	REAL	Estimated cost (rarely populated)	Fallback when actual unavailable
actual_cost_usd	REAL	Actual cost (almost never populated)	Primary cost source when available
source	TEXT	Session source ("chat", "cron", etc.)	Filtering, analysis
message_count	INTEGER	Number of messages	Usage analysis

Derived Fields

- **Session Duration**: ended_at - started_at (when available)
- **Total Tokens**: input_tokens + output_tokens + cache_read_tokens
- **Cost per Token**: total_cost / total_tokens (when tokens > 0)

- **Cost per Session**: `total_cost / session_count`

Rate Table Configuration

External YAML/JSON configuration file mapping vendor/model to pricing:

```
rates:
  openai-codex:
    gpt-5.5:
      input_tokens: 0.01
      output_tokens: 0.03
      cache_read_tokens: 0.001
    gpt-5.6-sol:
      input_tokens: 0.015
      output_tokens: 0.04
      cache_read_tokens: 0.0015
  deepseek:
    deepseek-v4-flash:
      input_tokens: 0.001
      output_tokens: 0.002
      cache_read_tokens: 0.0001
    deepseek-chat:
      input_tokens: 0.002
      output_tokens: 0.004
      cache_read_tokens: 0.0002
```

Data Freshness Strategy

Recommended Update Mechanism

Primary: Weekly Cron Job

- Schedule: `0 6 1` (Monday at 6 AM)
- Command: `hermes-cost-tracker --report --output-dir /path/to/reports`
- Generates comprehensive weekly report
- Email delivery option: `--email reports@example.com`

Secondary: Real-time Mode (Optional)

- Watch for new sessions and update incrementally
- Use SQLite triggers or polling for new sessions
- Generate mini-reports for recent activity
- Command: `hermes-cost-tracker --watch --interval 300`

Configuration

```
# ~/.hermes/cost-tracker.yaml
database_path: "~/.hermes/sessions.db"
report_schedule:
  enabled: true
  cron: "0 6 * * 1"
  output_dir: "/home/matthew/cost-reports"
  formats: ["table", "markdown", "csv"]
real_time:
  enabled: false
  interval: 300  # 5 minutes
  max_sessions_per_batch: 1000
```

Incomplete Data Handling

Strategy: Conservative Estimation with Clear Labeling

Approach: Use vendor/platform averages for missing data with explicit marking

Missing Provider (`billing_provider IS NULL`)

- **Action**: Use `model` to infer provider (e.g., "gpt-5.5" → "openai-codex")
- **Fallback**: Group under "unknown" provider
- **Labeling**: Mark all costs with `[INFERRED]` prefix

Missing Model (`model IS NULL`)

- **Action**: Skip session (no cost calculation possible)
- **Labeling**: Exclude from reports entirely
- **Logging**: Log skipped sessions for debugging

Missing Token Counts

- **Zero-token sessions**: Skip cost calculation, include in usage counts only
- **Partial token counts**: Use available tokens, mark as `[INCOMPLETE]`
- **Labeling**: All estimated costs marked with `[ESTIMATED]`

Missing Cost Data

- **When `actual_cost_usd` is NULL**: Calculate from tokens + rate table

- **When `estimated_cost_usd` is available**: Use as fallback
- **Labeling**: All calculated costs marked with `[CALCULATED]`

Data Quality Indicators

Every report includes a data quality summary:

```
Data Quality Report:
- Total sessions: 1,234
- Sessions with missing provider: 19 (1.5%) [INFERRED]
- Sessions with missing model: 0 (0.0%) [SKIPPED]
- Sessions with zero tokens: 19 (1.5%) [EXCLUDED]
- Estimated costs: 456 (37.0%) [ESTIMATED/CALCULATED]
- Complete data: 778 (63.0%)
```

Edge Cases

Sessions with Missing Provider

- **19 sessions in sample data** have `billing_provider IS NULL`
- **Handling**: Use model name pattern matching to infer provider
- **Example**: "gpt-5.5" → "openai-codex", "deepseek-v4-flash" → "deepseek"
- **Fallback**: Group under "unknown" provider

Zero-Token Sessions

- **19 sessions in sample data** have `input_tokens = 0 AND output_tokens = 0`
- **Handling**: Exclude from cost calculations, include in usage statistics
- **Use case**: Often cron jobs or system sessions with no actual AI usage

Quota-Based Sessions (`cost_status = "included"`)

- **Subscription-included sessions** have `billing_mode = "subscription_included"`
- **Handling**: Track as "included" usage, don't apply rate table pricing
- **Reporting**: Separate section for quota vs. metered usage

Sessions with Missing Billing Mode

- **Cron sessions** often have `billing_mode IS NULL`
- **Handling**: Infer from provider/model patterns or default to "chat_completions"
- **Example**: Deepseek sessions → "chat_completions"

Cache Read Cost Handling

- **Important**: Cache reads are billed separately and must be included
- **Sample data**: Large cache reads (e.g., 445M tokens for deepseek)
- **Handling**: Apply cache-read rates from rate table
- **Special case**: Some providers may not bill cache reads (rate = 0)

Assumptions

Data Structure Assumptions

1. **SQLite Schema**: Assumes stable sessions table structure with current field names
2. **Token Accuracy**: Assumes token counts are accurate and reflect actual usage
3. **Provider Mapping**: Assumes provider names are consistent across sessions
4. **Rate Table**: Assumes external rate table will be maintained and updated

Business Logic Assumptions

1. **Cost Calculation**: Uses rate table pricing when actual costs are unavailable
2. **Time Zones**: All timestamps assumed to be in UTC
3. **Session Boundaries**: Uses `started_at` and `ended_at` for time window calculations
4. **Cache Billing**: Assumes cache reads are billed at different rates than regular tokens

Technical Assumptions

1. **Database Location**: Session store at `~/.hermes/sessions.db`
2. **Read-Only Access**: No modifications to session database
3. **Large Dataset**: Optimized for 100K+ sessions with efficient querying
4. **External Dependencies**: Requires external rate table configuration

---

Open Questions

Data Quality

1. **Provider Inference**: How reliable is provider inference from model names?
2. **Rate Table Maintenance**: Who maintains the rate table? How often is it updated?
3. **Missing Data**: What's the acceptable threshold for missing data before estimates become unreliable?

Business Logic

1. **Quota Handling**: How to handle sessions that exceed quota limits?
2. **Multi-Model Sessions**: How to handle sessions using multiple models?
3. **Cost Attribution**: How to attribute costs to specific projects or use cases?

Technical Implementation

1. **Database Performance**: How to handle very large session databases efficiently?
2. **Concurrent Access**: What happens when Hermes is actively writing to the database?
3. **Schema Evolution**: How to handle future schema changes in the sessions table?

Reporting Requirements

1. **Audience**: Who are the primary consumers of these reports?
2. **Frequency**: Are weekly reports sufficient, or do daily reports make sense?
3. **Granularity**: What level of detail is needed for decision-making?

Success Metrics

Usage Metrics

- Report generation success rate (>99%)
- Processing time for 100K sessions (<5 minutes)
- Data completeness percentage (>95%)

Business Metrics

- Cost visibility improvement (ability to track costs by vendor/model)
- Identification of cost optimization opportunities
- Reduction in "unknown" cost categories

Technical Metrics

- Database query efficiency
- Memory usage optimization
- Error handling robustness

Implementation Considerations

Database Access

- Use SQLite WAL mode for concurrent access
- Implement connection pooling for repeated queries
- Add proper error handling for database locks

Performance Optimization

- Index on date range fields for fast filtering
- Batch processing for large datasets
- Memory-efficient streaming for large result sets

Error Handling

- Graceful degradation when database is unavailable
- Clear error messages for missing data
- Comprehensive logging for debugging

Configuration Management

- External YAML configuration for rate tables
- Environment variable overrides for paths
- Validation of configuration files

Future Roadmap

Phase 1 (MVP)

- Basic cost reporting with date range filtering
- Console table and markdown output
- Weekly cron job support
- Basic rate table configuration

Phase 2 (Enhanced Analytics)

- CSV export support
- Advanced filtering and sorting
- Cost trend analysis
- Email delivery of reports

Phase 3 (Advanced Features)

- Web dashboard
- Real-time monitoring
- Cost optimization recommendations
- API for external integrations

Phase 4 (Enterprise Features)

- Multi-tenant support
- Advanced cost attribution
- Predictive modeling
- Integration with billing systems

test2_prd/gpt-5.6-luna/cost-tracker-prd.md

Hermes Cost Tracker PRD

Goal and Non-Goals

Goal

Build a comprehensive cost tracking application for Hermes Agent that provides visibility into AI usage patterns, costs across vendors/models/platforms, and enables budget management through configurable rate tables and scheduled reporting.

Non-Goals

- Real-time cost monitoring during active sessions
- Integration with external billing systems
- Automatic cost optimization or model switching
- Multi-tenant/cost allocation across teams
- Competitor analysis or benchmarking

Requirements

Must-Have Features

1. **Cost Calculation Engine**

- Compute cost per vendor/model/platform for configurable date windows - Support for both metered (pay-per-use) and quota-based (subscription) spend - Include cache-read costs in calculations - Configurable, dated rate table for pricing models

1. **Data Sources**

- Read-only access to Hermes session store SQLite database (`~/.hermes/state.db`) - Parse sessions, messages, and billing metadata - Handle missing or incomplete data gracefully

1. **Report Generation**

- Console table output with summary statistics - Export to markdown and CSV formats - Support for date range filtering - Group by vendor, model, platform, billing mode

1. **Data Freshness**

- Cron job support for scheduled report generation (default: weekly) - Real-time mode for processing new sessions as they're created - Configurable update schedules

1. **Data Quality Handling**

- Handle sessions with missing model, provider, or token counts - Provide clear labeling of estimated vs actual data - Support for operator-defined mapping tables for missing data

Nice-to-Have Features

1. **Advanced Reporting**

- Trend analysis over time - Cost forecasting based on usage patterns - Budget vs actual comparisons

1. **User Experience**

- Configuration file support - CLI interface with sensible defaults - Error handling and validation

1. **Extensibility**

- Plugin architecture for custom cost calculators - Support for custom rate tables

Data Model Notes

Sessions Table Fields Used

- `id`: Session identifier
- `model`: Model name (e.g., "gpt-5.5", "deepseek-v4-flash")
- `billing_provider`: Provider name (e.g., "openai-codex", "deepseek", "zai")
- `billing_mode`: Billing mode (e.g., "subscription_included", "chat_completions")
- `cost_status`: Cost status ("included", "unknown", or NULL)
- `estimated_cost_usd`: Pre-computed estimate (often NULL)
- `actual_cost_usd`: Actual cost (often NULL)
- `pricing_version`: Pricing version (always NULL in current data)
- `input_tokens`: Input token count
- `output_tokens`: Output token count
- `cache_read_tokens`: Cache read token count
- `cache_write_tokens`: Cache write token count (not used in cost calc)
- `started_at`: Session start timestamp
- `ended_at`: Session end timestamp
- `source`: Session source (e.g., "cli", "telegram", "cron")

Messages Table Fields Used

- Token counts per message (aggregated to session level)
- Platform-specific metadata

Cost Calculation Logic

Metered vs Quota Spend

- **Metered**: Sessions with `billing_mode = "chat_completions"` or `cost_status = "unknown"`
- **Quota**: Sessions with `billing_mode = "subscription_included"` and `cost_status = "included"`

Cache Read Billing

- Cache reads are billed at the same rate as regular input tokens
- `cache_read_tokens` are added to `input_tokens` for cost calculation

Rate Table Configuration

```
rates:
  vendor:
    model:
      input_rate: 0.001
      output_rate: 0.002
      cache_read_rate: 0.001
      effective_date: "2026-01-01"
```

Data Freshness Strategy

Recommended Update Mechanism

Primary: Weekly cron job (recommended default)

- Runs every Monday at 6:00 AM
- Processes all sessions from the previous week
- Generates comprehensive weekly report

Secondary: Real-time mode (optional)

- Monitors new sessions as they're created
- Updates incremental cost tracking
- Suitable for high-usage environments

Cron Configuration

```
# Weekly report generation
0 6 * * 1 /usr/local/bin/hermes-cost-tracker --report --format=markdown --output=/var/log/hermes-cost/weekly-$(date +\%Y-\%m-\%d).md
```

```
# Daily incremental update (optional)
0 2 * * * /usr/local/bin/hermes-cost-tracker --incremental --format=csv --output=/var/log/hermes-cost/daily-$(date +\%Y-\%m-\%d).csv
```

Incomplete Data Handling

Chosen Approach: Operator-Defined Mapping + Averaging

1. **Missing Model/Provider**: Use operator-defined mapping tables
2. **Missing Token Counts**: Use vendor/model averages from historical data
3. **Missing Cost Status**: Default to "unknown" for metered billing

Implementation Details

- Create `mappings.yaml` for manual mappings
- Maintain historical averages database
- All estimates clearly marked with `[ESTIMATE]` prefix
- Provide option to exclude estimated data from reports

Example Estimate Handling

```
deepseek-v4-flash|zai|chat_completions|unknown|input_tokens: 1,234,567|output_tokens: 987,654|cost: $12.34 [ESTIMATE]
```

Edge Cases

Sessions with Missing Model or Provider

- **Handling**: Use fallback mapping or vendor averages
- **Labeling**: Mark as "unknown-vendor" or "unknown-model"
- **Impact**: Excluded from vendor-specific totals unless mapped

Cost Status "included" (Quota)

- **Handling**: Track but don't include in metered cost totals
- **Reporting**: Separate section for quota usage
- **Analysis**: Show quota utilization percentages

Zero-Token Sessions

- **Handling**: Include in session counts but zero cost
- **Reporting**: Clearly visible in detailed reports
- **Filtering**: Option to exclude from cost summaries

Sessions with NULL pricing_version

- **Handling**: Use current rate table (no versioning support)
- **Impact**: All costs calculated using latest rates
- **Future**: Support for historical rate tables

Assumptions

1. **Database Location**: Uses default `~/.hermes/state.db` path
2. **Read-Only Access**: Never modifies the session store
3. **Rate Stability**: Pricing rates don't change frequently
4. **Time Zone**: All timestamps handled in UTC
5. **Session Completeness**: Ended sessions have complete token counts

Open Questions

1. **Rate Table Versioning**: Should we support historical rate tables?
2. **Currency Support**: Only USD for now, or add other currencies?
3. **Platform Granularity**: How detailed should platform breakdowns be?
4. **Alert System**: Should we add budget threshold alerts?
5. **Data Retention**: How long should historical cost data be retained?
6. **Multi-Profile Support**: Should we track costs across multiple Hermes profiles?

Implementation Considerations

Database Access

- Use SQLite3 for direct database access
- Implement proper error handling for database locks
- Support for database file location configuration

Performance

- Index on `started_at` for date range queries
- Batch processing for large date ranges
- Memory-efficient handling of large result sets

Configuration

- YAML-based configuration files
- Environment variable support
- Command-line interface with sensible defaults

Testing

- Unit tests for cost calculation logic
- Integration tests with sample database
- Edge case validation

Success Metrics

1. **Accuracy**: Within 5% of actual provider billing
2. **Performance**: Weekly report generation < 5 minutes for 10K sessions
3. **Usability**: CLI intuitive with clear output formatting
4. **Reliability**: 99.9% uptime for scheduled jobs
5. **Adoption**: Used by >80% of Hermes installations

test2_prd/gpt-5.6-terra/cost-tracker-prd.md

Hermes Cost Tracker — Product Requirements Document

Status: Design only Date: 2026-08-16

1. Goal

Provide a local, read-only reporting tool that turns Hermes session usage in `state.db` into auditable cost and usage reports for a requested date window. It must:

1. calculate USD for metered usage by billing vendor, model, and Hermes platform/source;
2. separately report subscription/quota-included usage rather than pretending that it is variable API spend;
3. charge cache-read and cache-write tokens at their own configured rates;
4. identify every result as priced, included/quota, unpriced, or estimated; and
5. stay current through a scheduled report refresh, with an optional near-real-time mode.

The product answers: “What did Hermes use, what was the metered API cost, and what usage was covered by a quota/subscription?” It is an accounting aid, not a provider invoice or a quota-remaining meter.

2. Non-goals

- Do not modify Hermes’ database, sessions, messages, configuration, or provider account.
- Do not call provider billing APIs, reconcile invoices, or claim an authoritative provider charge in v1.
- Do not infer subscription price, remaining quota, overage, taxes, currency conversion, hosting cost, tool/API cost, or user-level chargeback.
- Do not calculate a per-message or per-tool-call price; the authoritative unit in the inspected store is a session aggregate.
- Do not ship a hard-coded public price catalog. Rates are operator-maintained and effective-dated.
- Do not perform statistical dollar estimates for missing identity/usage data in v1.

3. Research findings and constraints

Official Hermes documentation

Hermes stores sessions canonically in SQLite at `~/.hermes/state.db`; it uses WAL mode, supporting concurrent readers and one writer. `sessions` holds metadata, lifecycle, model/billing fields, counters, and lineage; `messages` holds the complete history and per-message `token_count`. Hermes source tags include `cli`, messaging platforms such as `telegram`, `email`, and `discord`, plus `cron`, `batch`, `api-server`, and `acp`.

Official references consulted:

- https://hermes-agent.nousresearch.com/docs/user-guide/sessions
- https://github.com/NousResearch/hermes-agent/blob/main/website/docs/developer-guide/session-storage.md

Read-only snapshot inspection

The supplied copy, `/tmp/hermes-cost-snapshot.db`, was inspected with SQLite read-only connections only. It has schema version 17 and 385 `sessions` spanning 2026-07-04 through 2026-08-17.

The snapshot's `sessions` table contains the fields required by this product:

- identity/routing: `id`, `source`, `user_id`, `session_key`, `chat_id`, `chat_type`, `thread_id`, `parent_session_id`;
- model and billing: `model`, `model_config`, `billing_provider`, `billing_base_url`, `billing_mode`, `estimated_cost_usd`, `actual_cost_usd`, `cost_status`, `cost_source`, `pricing_version`;
- timing: `started_at`, `ended_at` as Unix-epoch floating-point seconds;
- usage: `input_tokens`, `output_tokens`, `cache_read_tokens`, `cache_write_tokens`, `reasoning_tokens`, `api_call_count`;
- lifecycle/context: `end_reason`, `archived`, `cwd`, and source-related fields.

Important observed facts:

- All snapshot rows have `pricing_version IS NULL`. Hermes therefore supplies no rate-version provenance for this data.
- `actual_cost_usd` is NULL on every row. `estimated_cost_usd` is present as `0.0` on 366 rows and absent on 19; this is not usable USD pricing. The snapshot has no non-zero computed USD cost.
- Cache reads are material: 896,891,627 `cache_read_tokens` versus 51,087,821 input and 7,725,606 output tokens. `cache_write_tokens` is zero in this snapshot, but the schema supports it and the tracker must price it.
- Nineteen zero-token records lack `billing_provider`; all otherwise observed usage records have a model and token counts.
- The snapshot contains 111 `openai-codex` sessions marked `billing_mode = subscription_included` and `cost_status = included`, plus one additional `openai-codex` subscription-included session marked `unknown`. These belong in quota/included reporting, not metered USD.
- Other observed vendors include `deepseek`, `kimi-coding`, `kimi-for-coding`, `xiaomi`, `xai`, and `zai`. Most are `cost_status = unknown` with no effective price version. A few have `billing_mode = chat_completions` or `subscription_included`; the tracker must classify at the session level, not assume every session from a vendor shares a billing arrangement.

Hermes' current pricing work also distinguishes estimated, included, and unknown states and separate cache-read/cache-write accounting, while actual provider-cost reconciliation is not generally populated. The tracker must preserve that distinction rather than treating zero or NULL Hermes cost fields as a free session.

4. Primary users and use cases

Operator / owner:

- Run a report for an inclusive UTC date window and see total usage and cost by vendor, model, and platform.
- See which spend is metered versus included in a quota/subscription.
- Update a dated rate table after a provider price change and regenerate past or future reports reproducibly.
- Discover sessions that need an identity mapping or missing-usage correction before they can be priced.
- Receive a weekly refreshed artifact without any write access to the Hermes store.

5. Functional requirements (must-have first)

R1 — Read-only session-store access

- Accept an explicit SQLite database path; default to the active profile's `~/.hermes/state.db` only when the operator elects it.
- Open SQLite with a read-only URI, for example `file:/absolute/path/state.db?mode=ro`, and set `PRAGMA query_only=ON`.
- Never execute DDL/DML, `VACUUM`, `PRAGMA journal_mode`, checkpointing, repair, or migration against the source database.
- Treat the live database and its `-wal`/`-shm` files as a consistent read source. Do not copy or delete them. A short busy timeout/retry is permitted only for read consistency.
- All tracker artifacts (configuration, report output, optional tracker cache) live outside Hermes' database directory.

Acceptance: filesystem/audit testing shows only SELECT/PRAGMA query-only operations against the source DB.

R2 — Date-window session query

- Required filters: `--from YYYY-MM-DD`, `--to YYYY-MM-DD`, and `--timezone` (default `UTC`). Dates are inclusive calendar dates.
- Convert bounds to Unix epoch seconds in the selected timezone and query `sessions.started_at >= start AND sessions.started_at < next_day_after_end`.
- Display the requested timezone, exact UTC bounds, report generation timestamp, database path, and latest included `started_at` in every output.
- Default to excluding `archived = 1`; provide `--include-archived`.

- A session is counted once by `sessions.id`. Do not sum token counts from `messages`, which would risk a different unit and double counting.

Known limitation: session counters are session-level aggregates. A session starting before a window and ending within it is excluded under this reproducible start-time policy; a session that begins within the window is included with its aggregate, including later updates. The report must state this policy. A future version may support completed-at or incrementally captured attribution.

R3 — Cost classification and calculation

Classify every session before computing totals:

1. **included/quota** — `cost_status = 'included'` OR `billing_mode = 'subscription_included'`. Report tokens and session counts in the quota section; variable USD is `$0.00` with the label `INCLUDED / NOT PRICED AS API SPEND`.
2. **metered/priced** — not included, complete provider/model identity, complete token counts, and a matching effective-dated rate row. Compute a `CALCULATED` USD estimate.
3. **metered/unpriced** — not included but no matching rate, missing/invalid required token data, or unresolved provider/model. Report usage if present and USD as `N/A`, never `$0.00`.
4. **zero usage** — all four billable token buckets are zero. Exclude it from USD totals; retain it in a data-quality count. If it is included/quota, retain it in quota session counts.

For a metered/priced session, calculate:

```
USD = (input_tokens      / 1_000_000 * input_usd_per_million)
    + (cache_read_tokens  / 1_000_000 * cache_read_usd_per_million)
    + (cache_write_tokens / 1_000_000 * cache_write_usd_per_million)
    + (output_tokens      / 1_000_000 * output_usd_per_million)
```

`input_tokens` is treated as non-cached input under the current Hermes usage-bucket contract; cache reads and writes are separately charged. The application must validate this with a test fixture whenever Hermes changes its usage semantics. `reasoning_tokens` are displayed as usage metadata but have no separate cost unless a future rate schema explicitly models them.

Acceptance: a fixture with non-zero cache reads produces a different cost than the same fixture with cache reads set to zero; cache writes are also included when present.

R4 — Configurable, dated rate table

Use a version-controlled local YAML or CSV configuration, outside `~/.hermes/`, with one row per provider/model/routing identity and effective period. All rates are USD per 1 million tokens.

Required fields:

```
rates:
  - rate_id: deepseek-v4-flash-2026-08-01
    provider: deepseek
    model: deepseek-v4-flash
    billing_base_url: null            # optional route discriminator
    effective_from: 2026-08-01
    effective_to: null                # inclusive; null = open ended
    input_usd_per_million: 0.00
    cache_read_usd_per_million: 0.00
    cache_write_usd_per_million: 0.00
    output_usd_per_million: 0.00
    source_url: https://vendor.example/pricing
    retrieved_at: 2026-08-16T00:00:00Z
    notes: Operator must replace placeholder rates with verified vendor terms.
```

Rules:

- A production rate must have non-negative numeric bucket rates, `source_url`, `retrieved_at`, `effective_from`, and a unique `rate_id`.
- Match provider + model + started-at date, and also `billing_base_url` where configured. Reject overlapping matching date ranges as configuration errors.
- Store the selected `rate_id` and a content hash of the rate file in every report, even though Hermes' `pricing_version` is NULL.
- A rate of 0 is allowed only if the configuration marks it `zero_rate_confirmed: true`; otherwise refuse to price it. This prevents the current Hermes `0.0` placeholders from becoming false free spend.
- The operator, not the tool, owns price verification and updates. Re-running the same report against the same rate-table revision must reproduce totals.

R5 — Missing data policy: exact mapping, no statistical estimates

Chosen approach: use an operator-maintained, date-effective identity mapping; do not invent a vendor/platform average dollar estimate.

The mappings file is a separate, version-controlled configuration and supports precise overrides in this order:

1. session ID override (for a known anomalous session);
2. provider + model + `billing_base_url` route mapping;
3. source/platform + model mapping, only if the operator documents why it is exact;
4. otherwise unresolved.

A mapping supplies missing `billing_provider`, model, billing mode, or route only when the operator has a documented exact source (for example, the configured provider route at that time). It must carry `mapping_id`, effective dates, rationale, and evidence URL or configuration reference.

If an identity or rate cannot be resolved, preserve the session in `UNPRICED — MISSING_PROVIDER`, `UNPRICED — MISSING_MODEL`, `UNPRICED — MISSING_RATE`, or `UNPRICED — INVALID_USAGE`. Do not convert it to zero cost or average it.

V1 makes no numeric statistical estimates. Consequently, every report must include an "Estimation and completeness" legend and counts/tokens for:

- `CALCULATED` — matched to the configured rate table; this is an estimate of provider charge, not an invoice;
- `INCLUDED` — subscription/quota usage; no variable API USD calculated;
- `UNPRICED` — insufficient mapped data/rates; USD excluded from totals;
- `ESTIMATED` — reserved for a future explicit heuristic. If ever enabled, it must show `ESTIMATED`, method, basis group, confidence, and affected sessions in console, Markdown, and CSV output. It must never be combined silently with `CALCULATED` totals.

R6 — Required report outputs

Each run writes both:

1. a concise console table; and
2. either Markdown (default) or CSV (`--format csv`) at an explicit `--output` path.

Markdown contains:

- report metadata and rate-table provenance;
- headline totals: calculated metered USD, included/quota sessions and tokens, unpriced sessions/tokens, zero-token sessions, and coverage percentage;
- a breakdown grouped by `billing_provider`, `model`, and `source` (called "platform" in the UI);
- token columns for input, cache read, cache write, output, and reasoning;
- calculated USD, status, rate ID, and data-quality/estimate label;
- a separate included/quota section and a data-quality appendix listing unresolved session IDs/reasons;
- scope and limitations.

CSV is a machine-readable row-level export. It includes `session_id`, start/end timestamps, source, original and resolved provider/model, billing mode, cost status, all token buckets, `rate_id`, calculated USD, report status, mapping ID, and issue reason. A companion `<name>-summary.csv` provides the grouped totals.

Never display `N/A` as `$0.00`. Round display USD to cents but calculate and export sufficient precision for reconciliation.

R7 — Data freshness and recommended scheduling

Recommended default: **weekly scheduled batch regeneration**. It is simple, audit-friendly, low-risk for the WAL-backed live store, and sufficient for an accounting report. Each run regenerates the trailing 90 days to include sessions updated after their start date and applies the current effective-dated rate table. It writes a dated Markdown report and replaces a `latest.md` copy in the report directory.

The product must support an opt-in `--watch` real-time mode, but it is not the default. Watch mode polls `MAX(started_at)` plus a rolling 24-hour overlap every five minutes, upserts only report-output state outside the Hermes database, and performs a full trailing-90-day reconciliation weekly. It must tolerate partially active sessions and label the dashboard/report "provisional until session closes." No SQLite write hook or change to Hermes is allowed.

Exact recommended Hermes Cron setup (to be performed only after the tracker executable/script exists):

1. Install the read-only report script as `/home/matthew/.hermes/scripts/hermes-cost-tracker-report.sh`. It invokes the tracker with:

```
--db /home/matthew/.hermes/state.db
--from relative:90d
--to today
--timezone UTC
--rates /home/matthew/wiki/hermes-kit/cost-tracker/rates.yaml
--mappings /home/matthew/wiki/hermes-kit/cost-tracker/mappings.yaml
--format markdown
--output /home/matthew/wiki/hermes-kit/cost-reports/latest.md
--read-only
```

1. Create the native no-agent weekly job:

```
hermes cron create \
  --name 'Hermes Cost Tracker — weekly report' \
  --deliver local \
  --script hermes-cost-tracker-report.sh \
  --no-agent \
```

```
  --workdir /home/matthew/wiki \
  '0 9 * * 1'
```

This runs Mondays at 09:00 in the host's cron timezone. `--no-agent` is required: generating a report must not itself create a new Hermes LLM session/cost. `deliver local` means cron output is retained locally; it does not promise a terminal notification. The script must exit non-zero on missing rate/mapping/configuration errors and write no report labeled successful in that case.

6. Data model and query notes

Primary query source: `sessions`, not `messages`.

Tracker concept	Hermes field(s)	Use
Unique unit	`id`	Session-level deduplication and row export.
Platform	`source`	UI group called Platform; values include CLI, gateways, cron, and subagent-like sources found in snapshot.
Vendor	`billing_provider`	Primary vendor dimension; resolve through mappings if absent.
Model	`model`	Model rate match; resolve through mappings if absent.
Route	`billing_base_url`, `model_config`	Optional rate/mapping discriminator; do not expose secrets.
Time	`started_at`, `ended_at`	Window selection, completion/provisional state. Epoch float converted with specified timezone.
Usage	`input_tokens`, `output_tokens`, `cache_read_tokens`, `cache_write_tokens`, `reasoning_tokens`	Aggregate usage and cost formula.
Hermes billing metadata	`billing_mode`, `cost_status`, `cost_source`, `estimated_cost_usd`, `actual_cost_usd`, `pricing_version`	Classification, diagnostics, and comparison only; current snapshot values are not a substitute for the rate table.
Lifecycle	`archived`, `end_reason`, `parent_session_id`, `api_call_count`	Filtering and report diagnostics.

The tracker should record both original and resolved dimensions in outputs. It should redact `billing_base_url`, user IDs, chat IDs, session keys, prompt content, and message content by default; none are needed for cost aggregation.

7. Edge cases and expected handling

Condition	Required behavior
Missing provider	Attempt dated exact mapping. If unresolved, report `UNPRICED — MISSING_PROVIDER`; retain tokens and session ID, exclude USD.
Missing model	Attempt dated exact mapping. If unresolved, report `UNPRICED — MISSING_MODEL`; retain tokens and session ID, exclude USD.
Missing token count / NULL / negative count	Treat as invalid usage, do not coerce to zero, and report `UNPRICED — INVALID_USAGE`. Schema defaults of zero are valid only when all buckets are explicitly zero.
`cost_status = included`	Classify as included/quota regardless of a rate match; report usage but `$0.00 variable API spend` and clear label.
`billing_mode = subscription_included`, status unknown	Classify as included/quota but flag `BILLING_STATUS_INCONSISTENT` for review.
Zero-token session	Keep in session/data-quality counts; exclude metered USD. Do not call it free metered usage.
`estimated_cost_usd = 0` / `actual_cost_usd` NULL / `pricing_version` NULL	Do not treat as priced or free; use rate table or `UNPRICED`.
Rate period absent or overlapping	Fail the affected row as unpriced; fail the run before publishing a misleading "success" report if configuration overlap is global.
Provider/model changes in a session	Session store has only session aggregate identity. Price by recorded session identity and flag this limitation; a future per-call table is needed for exact split attribution.
Session remains open	Include by start-time policy, label provisional, and refresh through trailing-window regeneration.
DST/timezone boundary	Convert local inclusive calendar dates to exact timezone-aware UTC epoch bounds and print both.
Parent/continuation session	Each session is a separate persisted usage record; do not collapse lineage unless a future Hermes invariant proves duplicate counters.

8. Quality, security, and acceptance criteria

- The implementation has read-only integration tests against a fixture database and verifies source DB checksum/mtime is unchanged after a run.
- Unit tests cover cache reads/writes, dated rate changes, rate overlaps, inclusive date bounds, included status, subscription-included/unknown inconsistency, zero tokens, null provider/model, null/negative token data, and output labels.
- A fixture based on the supplied snapshot must demonstrate: `pricing_version` NULL does not result in `$0.00` metered cost; OpenAI Codex included sessions are split from metered spend; cache reads appear in report totals.
- The console table and generated file give the same headline counts and USD totals.
- Every non-quota dollar total identifies its coverage: priced sessions / eligible metered sessions, unpriced session count, and rate-table hash.
- No implementation step writes to Hermes' `state.db`, its WAL/SHM files, or `sessions/messages`.

9. Assumptions

1. The target Hermes deployment continues to store session aggregate counters in SQLite and makes `sessions.started_at` available.
2. Token buckets are session totals and `input_tokens` excludes `cache_read_tokens`; the rate formula assumes distinct canonical buckets.
3. The operator has lawful access to provider pricing and can maintain effective-dated rates and exact route mappings.
4. "Platform" means Hermes' `source` field, not the provider's API platform.
5. The tracker runs on the same host or has file-system read access to the live database and its WAL sidecars.

6. The tracker is allowed to write reports/configuration outside Hermes' state-store directory.

10. Open questions

1. Should the source window be `started_at` (recommended v1), `ended_at`, or a future per-call timestamp for financial-period reporting?
2. Does Hermes guarantee that `input_tokens` is always exclusive of cache-read tokens for every provider/transport, and how will schema/usage-bucket version changes be surfaced?
3. Which provider/route combinations are genuinely metered versus OAuth or subscription included in the operator's deployment? This cannot safely be inferred solely from vendor name.
4. Should a separate fixed subscription-fee ledger be added later so quota usage can be allocated without falsely presenting it as variable API spend?
5. When Hermes begins populating `actual_cost_usd` and `pricing_version`, should verified actuals supersede calculated estimates, or should both remain visible as separate columns?
6. What report retention, access controls, and redaction policy are required if CSV rows containing session IDs are shared outside the operator's machine?

test2_prd/gpt-5.6-sol/cost-tracker-prd.md

Hermes Cost Tracker — Product Requirements Document

Status: Design only (research + requirements; no implementation in this task) Date: 2026-08-16
Scope: read-only reporting against the Hermes session store

1. Goal

Build a local, read-only reporting tool that turns the session records Hermes already persists in SQLite into auditable usage and cost reports for an operator-chosen date window. The tracker answers: *What did Hermes use, what did the metered API spend cost, and what usage was covered by a subscription/quota?*

Must-haves, in priority order:

1. Compute USD cost per vendor (`billing_provider`), model, and platform (`source`)

for an inclusive date window.

1. Split spend into **metered** (variable API cost) vs **included/quota**

(subscription-covered, e.g. OpenAI Codex `cost_status = included`) — never mix the two.

1. Price cache-read tokens at their own configured rate (cache writes too, when present).
2. Use a configurable, effective-dated rate table maintained by the operator.
3. Emit a console table plus a Markdown or CSV report file.
4. Open the database strictly read-only and never modify it.

Every report must be able to state, for every dollar or token shown, whether it is `CALCULATED` (priced from the rate table), `INCLUDED` (quota), `UNPRICED` (cannot be priced from available data), or `ESTIMATED` (explicit heuristic — reserved, not enabled in v1).

2. Non-goals

- No writes to Hermes data: not `state.db`, its `-wal`/`-shm` sidecars, sessions,

messages, config, or provider accounts.

- No calls to provider billing APIs, no invoice reconciliation, no claim of an

authoritative provider charge in v1.

- No inference of subscription price, remaining quota, overage, taxes, currency

conversion, or hosting costs.

- No hard-coded public price catalog. Rates are operator-maintained and dated.
- No statistical dollar estimates for missing identity/usage data in v1 (see §6).
- No per-message or per-tool-call pricing; the unit of record is the session aggregate.

3. Research findings and constraints

3.1 Official documentation

Hermes stores sessions canonically in SQLite (`~/.hermes/state.db`, WAL mode). The `sessions` table carries identity/routing, model and billing metadata, lifecycle, and aggregate token counters; `messages` holds full history and a per-message `token_count`. Platform is represented by `sources` such as `cli`, `cron`, `subagent`, and messaging platforms (`telegram`, `email`, ...). Cost-relevant fields are `billing_provider`, `billing_mode`, `estimated_cost_usd`, `actual_cost_usd`, `cost_status`, `cost_source`, and `pricing_version`.

Sources consulted:

- https://hermes-agent.nousresearch.com/docs (sessions, providers, cron, kanban)

- https://hermes-agent.nousresearch.com/docs/developer-guide/session-storage

3.2 Read-only snapshot inspection (/tmp/hermes-cost-snapshot.db)

Inspected with file:...?mode=ro&immutable=1 URI and PRAGMA query_only=ON only. Schema version 1; 385 sessions spanning 2026-07-04 → 2026-08-17; 18,117 messages; no archived sessions.

Verified facts the PRD must respect:

Fact	Value	Consequence
pricing_version	NULL on all 385 rows	Hermes computes no USD; no rate provenance exists. The tracker must supply its own dated rate table.
actual_cost_usd	NULL on all rows	No verified actuals to lean on.
estimated_cost_usd	0.0 on 366 rows, NULL on 19, non-zero on exactly 1	The 0.0 default is a placeholder, not evidence of free usage. Never treat it as priced or free.
cost_status	unknown 255, included 111, NULL 19	Split included/quota from unknown metered at the session level.
billing_mode	NULL 269, subscription_included 113, chat_completions 3	Classify per session, not per vendor name.
Providers	deepseek 126, openai-codex 112, kimi-coding 63, zai 27, xiaomi 26, xai 11, kimi-for-coding 1, NULL 19	openai-codex is the quota case; the rest are metered-but-unpriced in this snapshot.
Models	deepseek-v4-flash 123, gpt-5.6-sol 69, kimi-k2.6 68, gpt-5.5 60, mimo-v2.5-pro 24, grok-4.6 11, + others	Model is always present (0 NULLs).
Token totals	input 51,087,821 · output 7,725,606 · cache_read 896,891,627 · cache_write 0 · reasoning 464,488	Cache reads are ~17× input tokens — omitting them would miss ~90% of usage. Cache writes are 0 here but the schema supports them.
messages.token_count	NULL on all 18,117 rows	Cannot price per message; session aggregates are the only usable unit.
Zero-token sessions	19, all with billing_provider NULL	Cron/telegram sessions with no LLM call; keep in data-quality counts, exclude from USD.

Also observed: 112 openai-codex sessions are subscription_included (111 with cost_status = included, 1 with unknown → flag as inconsistent). One deepseek session is subscription_included while cost_status = unknown. Three sessions are billing_mode = chat_completions (deepseek, kimi-for-coding, xiaomi) with unknown status — they are metered in principle and need rates.

Conclusion: Hermes records usage well but prices nothing. Any USD figure in reports must come from the tracker's own dated rate table, and every session must be labeled priced / included / unpriced.

4. Primary users and use cases

Operator / owner of a Hermes deployment:

- Run a report for an inclusive UTC date window; see usage and cost grouped by vendor, model, and platform.
- See metered vs subscription/quota spend separated, with cache-read costs included.
- Update a dated rate table after a provider price change; regenerate past reports reproducibly.
- Find sessions that cannot be priced yet (missing identity, missing rate) so they can add an exact mapping or a rate.
- Receive a weekly refreshed report without ever granting the tracker write access.

5. Functional requirements (must-have first)

R1 — Strictly read-only session-store access

- Accept an explicit database path (--db); do not assume ~/.hermes/state.db.
- Open via read-only SQLite URI file:<abs-path>?mode=ro and set PRAGMA query_only=ON.
- Never execute DDL/DML, VACUUM, PRAGMA journal_mode, checkpointing, or migration against the source. Read through the live WAL consistently (WAL supports concurrent readers); do not copy or delete sidecars.
- All tracker artifacts (rate table, mappings, reports) live outside the Hermes store directory.
- Acceptance: an audit of filesystem/SQLite activity during a run shows only SELECT/PRAGMA reads against the source DB, and the DB's checksum is unchanged.

R2 — Date-window session query

- --from YYYY-MM-DD --to YYYY-MM-DD inclusive calendar dates, --timezone default UTC. Convert to exact timezone-aware epoch bounds (started_at >= start AND started_at < day_after_end).
- Exclude archived = 1 by default; --include-archived opt-in.
- Count each session once by sessions.id. Never sum messages.token_count (all NULL in the snapshot, and a different unit anyway).
- Print in every output: timezone, exact UTC bounds, report timestamp, DB path, latest included started_at.
- Window policy (stated in the report): a session is included by its started_at even

if it ended later; open sessions are included and labeled provisional.

R3 — Cost classification and calculation

Classify each session before totaling:

1. **included/quota** — `cost_status = 'included'` OR `billing_mode = 'subscription_included'`. Report tokens/session counts in the quota section; variable USD is `$0.00` labeled `INCLUDED / NOT PRICED AS API SPEND`.
1. **metered/priced** — not included; complete provider+model identity; complete, non-negative token counts; matching effective-dated rate row. Compute `CALCULATED` USD.
1. **metered/unpriced** — not included but no matching rate, unresolved identity, or invalid usage. Report usage; USD `N/A`, never `$0.00`.
1. **zero usage** — all billable buckets zero. Exclude from USD totals; keep in data-quality counts (and in quota counts if included).

Formula for a metered/priced session (USD per 1M tokens):

```
USD = input/1M * input_rate
    + cache_read/1M * cache_read_rate
    + cache_write/1M * cache_write_rate
    + output/1M * output_rate
```

`input_tokens` is treated as non-cached input under the current bucket contract; cache reads/writes are separate. `reasoning_tokens` are reported as usage metadata with no separate charge (unless a future rate schema models them). Validate this bucket contract against a fixture whenever Hermes changes usage semantics.

Acceptance: a fixture with cache reads priced > 0 differs in total from the same fixture with cache reads zeroed; cache writes also contribute when present.

R4 — Configurable, dated rate table

Version-controlled local YAML (or CSV), outside `~/.hermes/`, one row per provider/model(/route) with an effective period. All rates USD per 1M tokens:

```
rates:
  - rate_id: deepseek-v4-flash-2026-08-01
    provider: deepseek
    model: deepseek-v4-flash
    billing_base_url: null          # optional route discriminator
    effective_from: 2026-08-01
    effective_to: null              # inclusive; null = open-ended
    input_usd_per_million: 0.28
    cache_read_usd_per_million: 0.028
    cache_write_usd_per_million: 0.28
    output_usd_per_million: 0.42
    source_url: https://api-docs.deepseek.com/pricing
    retrieved_at: 2026-08-16T00:00:00Z
    notes: Operator-verified placeholder; replace with current vendor terms.
```

Rules:

- Match on provider + model (+ `billing_base_url` where configured) + session start date. Reject overlapping matching date ranges as config errors.
- Store the selected `rate_id` and a content hash of the rate file in every report (provenance, since Hermes `pricing_version` is NULL).
- A rate of `0.00` is allowed only with `zero_rate_confirmed: true`; otherwise refuse to price with it — this prevents Hermes' `0.0` placeholders from becoming false free spend.
- The operator owns price verification. Re-running the same report against the same rate revision must reproduce totals.

R5 — Report outputs

Each run emits both:

1. a concise console table; and
2. a Markdown report (default) or CSV (`--format csv`) at an explicit `--output` path.

Markdown report contents:

- metadata + rate-table provenance (rate file hash, rate IDs used);
- headline totals: calculated metered USD; included/quota sessions + tokens; unpriced sessions/tokens; zero-token sessions; coverage % (priced ÷ eligible metered);
- grouped breakdowns by `billing_provider`, `model`, `source` (platform);
- token columns: input, cache read, cache write, output, reasoning;
- per-group USD with status label (`CALCULATED / INCLUDED / UNPRICED`) and rate ID;
- separate included/quota section; data-quality appendix listing unresolved session IDs

and reasons;

- scope and limitations (window policy, provisional open sessions).

CSV: row-level machine export — session_id, timestamps, source, original and resolved provider/model, billing mode, cost status, all token buckets, rate_id, calculated USD, report status, mapping ID, issue reason. Companion <name>-summary.csv with grouped totals.

Display rules: never render N/A as $0.00; round display USD to cents but calculate and export full precision.

R6 — Data freshness: recommended scheduling

Recommended default: weekly scheduled batch regeneration. Simple, audit-friendly, low-risk for a WAL-backed live store, and sufficient for accounting. Each run regenerates a trailing 90-day window (picks up sessions updated after their start date) and applies the current effective-dated rate table. It writes a dated Markdown report and replaces latest.md in the report directory.

Optional (not default): --watch real-time mode. Polls MAX(started_at) plus a 24h overlap every 5 minutes, upserts report state only outside the Hermes DB, does a full trailing-90-day reconciliation weekly, and labels outputs "provisional until session closes". No SQLite write hooks, no changes to Hermes.

Exact cron setup (to be performed after the tracker executable exists):

1. Install the read-only wrapper as /home/matthew/.hermes/scripts/hermes-cost-tracker-report.sh:

```
--db /home/matthew/.hermes/state.db
--from relative:90d --to today --timezone UTC
--rates /home/matthew/wiki/hermes-kit/cost-tracker/rates.yaml
--mappings /home/matthew/wiki/hermes-kit/cost-tracker/mappings.yaml
--format markdown
--output /home/matthew/wiki/hermes-kit/cost-reports/latest.md
--read-only
```

1. Create the native no-agent weekly job:

```
hermes cron create \
  --name 'Hermes Cost Tracker — weekly report' \
  --deliver local \
  --script hermes-cost-tracker-report.sh \
  --no-agent \
  --workdir /home/matthew/wiki \
  '0 9 * * 1'
```

Mondays 09:00 host time. --no-agent is mandatory: generating a report must not itself spawn a new LLM session/cost. deliver local keeps output on disk; it does not promise a terminal notification. On missing rate/mapping/config errors the script exits non-zero and writes no "successful" report.

6. Incomplete data policy (chosen approach)

Chosen: exact, operator-maintained mapping — no statistical estimates in v1.

A separate version-controlled mappings file supplies missing identity *only* from documented exact sources (e.g., the provider route configured at that time), with mapping_id, effective dates, rationale, and evidence reference. Override precedence:

1. session-ID override (known anomalous session);
2. provider + model + billing_base_url route mapping;
3. source/platform + model mapping, only when documented as exact;
4. otherwise unresolved.

If identity or rate cannot be resolved, preserve the session as UNPRICED — MISSING_PROVIDER / UNPRICED — MISSING_MODEL / UNPRICED — MISSING_RATE / UNPRICED — INVALID_USAGE; keep its tokens and session ID; exclude USD. Never convert to $0.00 or an average.

Because v1 makes no numeric estimates, every report carries an explicit estimation/completeness legend with counts and tokens per label:

- CALCULATED — matched to the configured rate table; an estimate of provider charge,

not an invoice;

- INCLUDED — quota/subscription usage; no variable USD calculated;
- UNPRICED — insufficient mapped data or rates; excluded from USD totals;
- ESTIMATED — reserved for a future explicit heuristic. If ever enabled, it must show

method, basis group, confidence, and affected session IDs in console/Markdown/CSV, and must never be silently combined with CALCULATED totals.

7. Data model notes

Primary source: sessions (never messages for costing).

Tracker concept	Hermes field(s)	Use
Unit of record	id	Session-level dedupe, row export, issue tracking.
Platform	source	Grouping dimension (cli, cron, subagent, telegram, email observed).
Vendor	billing_provider	Primary vendor dimension; resolvable via mappings when NULL.

Tracker concept	Hermes field(s)	Use
Model	`model`	Rate match; resolvable via mappings (0 NULLs in snapshot).
Route	`billing_base_url, model_config`	Optional rate/mapping discriminator; redact secrets.
Time	`started_at, ended_at`	Window selection, provisional/completed state; epoch float → timezone-aware.
Usage	`input_tokens, output_tokens, cache_read_tokens, cache_write_tokens, reasoning_tokens`	Cost formula and usage reporting.
Billing metadata	`billing_mode, cost_status, cost_source, estimated_cost_usd, actual_cost_usd, pricing_version`	Classification + diagnostics; snapshot values are NOT a pricing source.
Lifecycle	`archived, end_reason, parent_session_id, api_call_count`	Filtering, diagnostics, lineage notes.

Outputs record both original and resolved dimensions. Redact by default: `billing_base_url` secrets, user IDs, chat IDs, session keys, prompt/message content — none are needed for cost aggregation.

8. Edge cases and expected handling

Condition	Required behavior
Missing provider	Try dated exact mapping; else `UNPRICED — MISSING_PROVIDER`; keep tokens/session ID, exclude USD.
Missing model	Try dated exact mapping; else `UNPRICED — MISSING_MODEL`; keep tokens/session ID, exclude USD.
NULL / negative token counts	Invalid usage; do not coerce to 0; `UNPRICED — INVALID_USAGE`.
`cost_status = included`	Classify included/quota regardless of rate match; report usage with `$0.00` + clear label.
`billing_mode = subscription_included` with status `unknown`	Classify included/quota but flag `BILLING_STATUS_INCONSISTENT` (observed: 1 openai-codex, 1 deepseek session).
Zero-token session	Keep in session/data-quality counts; exclude from metered USD; never call it free metered usage.
`estimated_cost_usd = 0.0 / actual_cost_usd` NULL / `pricing_version` NULL	Never treated as priced or free; use rate table or UNPRICED.
Rate overlap / missing period	Affected row unpriced; global overlap fails the run before publishing a "success" report.
Open session (no `ended_at`)	Include by start-time policy; label provisional; refresh via trailing window.
DST/timezone boundary	Local inclusive dates → exact timezone-aware UTC bounds; print both.
Parent/continuation sessions	Each session is its own persisted usage record; do not collapse lineage absent a proven counter invariant.

9. Quality and acceptance criteria

- Read-only integration tests against a fixture DB; verify source DB checksum/mtime unchanged after each run.
- Unit tests: cache read/write pricing, dated rate changes, rate overlap, inclusive bounds, included status, subscription-included/unknown inconsistency, zero tokens, NULL/negative usage, output labels, console == file totals.
- Snapshot-derived fixture must demonstrate: `pricing_version` NULL never yields `$0.00` metered cost; 112 openai-codex sessions land in quota, not metered; ~897M cache-read tokens appear in usage and cost totals.
- Console table and generated file report identical headline counts and USD totals.
- Every non-quota dollar total identifies coverage: priced/eligible sessions, unpriced count, rate-file hash.

10. Assumptions

1. Hermes keeps session aggregate counters in SQLite with `started_at` available.
2. Token buckets are session totals and `input_tokens` excludes cache-read tokens (the formula's contract; to be re-validated on Hermes changes).
1. The operator has lawful access to provider pricing and can maintain dated rates and exact route mappings.
1. "Platform" means Hermes `source`, not the provider's API platform.
2. The tracker runs on the same host (or with file-system read access to the live DB + WAL sidecars).
1. The tracker may write reports/config outside the Hermes state-store directory.

11. Open questions

1. Should the window be `started_at` (v1 recommendation), `ended_at`, or a future per-call timestamp for financial-period reporting?
1. Does Hermes guarantee `input_tokens` excludes cache-read tokens for every provider/transport, and how are schema/usage-bucket changes surfaced?
1. Which provider/route combinations are genuinely metered vs OAuth/subscription in a given deployment? Vendor name alone is insufficient (observed: same provider in both modes).
1. Should a separate fixed-subscription ledger be added later so quota usage can be

allocated without masquerading as variable API spend?
1. When Hermes starts populating `actual_cost_usd` and `pricing_version`, should verified actuals supersede calculated estimates, or stay as separate columns?
1. What retention/access/redaction policy is needed if CSV exports with session IDs leave the operator's machine?

test3_cost/deepseek.md

Hermes Cost Report

- Window: 2026-07-04 → 2026-08-14 (UTC, by `started_at`, inclusive from, exclusive to)
- Database: `/tmp/hermes-cost-snapshot.db` (read-only)
- Rates: `rates.yaml`
- Sessions: 301 (101 still open — tokens-so-far, totals may grow)

METERED (priced from rate table)

vendor	model	platform	sessions	api calls	input	output	cache read	cache write	reasoning	USD	status
deepseek	deepseek-v4-flash	cli	34	448	1,143,076	684,085	23,129,088	0	0	$1.34	metered
deepseek	deepseek-v4-flash	cron	34	527	2,173,724	590,682	31,143,552	0	0	$1.80	metered
deepseek	deepseek-v4-flash	subagent	8	80	154,423	176,534	3,047,168	0	0	$0.22	metered
deepseek	deepseek-v4-flash	telegram	1	1,245	8,055,044	550,493	122,280,512	0	122,748	$5.91	metered
TOTAL			**77**	**2,300**	**11,526,267**	**2,001,794**	**179,600,320**	**0**	**122,748**	**$9.27**	

QUOTA (subscription-included — $0.00, volume counts)

vendor	model	platform	sessions	api calls	input	output	cache read	cache write	reasoning	USD	status
deepseek	deepseek-v4-flash	telegram	1	549	3,417,191	273,011	58,789,120	0	48,505	$0.00	quota
openai-codex	gpt-5.5	cli	1	6	33,142	997	134,656	0	174	$0.00	quota
openai-codex	gpt-5.5	cron	40	538	3,257,931	313,850	31,099,904	0	36,538	$0.00	quota
openai-codex	gpt-5.5	email	1	3	29,630	1,125	47,104	0	316	$0.00	quota
openai-codex	gpt-5.5	telegram	2	146	1,314,785	64,115	11,657,856	0	11,067	$0.00	quota
openai-codex	gpt-5.6-sol	cli	45	559	3,825,670	270,480	29,291,520	0	61,316	$0.00	quota
openai-codex	gpt-5.6-sol	cron	4	23	101,547	3,573	278,528	0	1,155	$0.00	quota
openai-codex	gpt-5.6-sol	subagent	19	157	1,002,250	144,669	5,783,040	0	20,832	$0.00	quota
TOTAL			**113**	**1,981**	**12,982,146**	**1,071,820**	**137,081,728**	**0**	**179,903**	**$0.00**	

UNPRICED (no rate — USD not computed)

vendor	model	platform	sessions	api calls	input	output	cache read	cache write	reasoning	USD	status
deepseek	deepseek-chat	cron	3	67	159,185	51,819	3,490,304	0	0	—	unpriced
deepseek	gpt-4o	cli	2	2	40,615	50	4,864	0	0	—	unpriced
deepseek	kimi-k1.5	cron	4	53	269,347	40,324	2,617,472	0	0	—	unpriced
deepseek	mimo-v2.5-pro	subagent	1	20	80,603	47,898	1,356,800	0	0	—	unpriced
kimi-coding	kimi-k2.6	cron	57	690	5,457,012	531,644	34,648,264	0	0	—	unpriced
kimi-coding	kimi-k2.6	email	1	1	18,127	199	1,536	0	0	—	unpriced
kimi-for-coding	kimi-k2.6	telegram	1	125	1,571,380	143,571	9,497,152	0	3,586	—	unpriced
xiaomi	deepseek-v4-flash	telegram	1	1,929	10,764,184	1,390,401	269,643,712	0	69,066	—	unpriced
xiaomi	mimo-v2.5-pro	cli	6	63	304,201	21,552	2,723,776	0	0	—	unpriced
xiaomi	mimo-v2.5-pro	cron	8	112	356,103	47,781	4,107,648	0	0	—	unpriced
xiaomi	mimo-v2.5-pro	subagent	3	39	307,092	50,512	2,370,048	0	0	—	unpriced
zai	deepseek-v4-flash	cron	1	40	91,739	5,600	1,582,494	0	0	—	unpriced
zai	glm-4.5-flash	cli	4	82	137,178	21,890	2,347,768	0	0	—	unpriced
TOTAL			**92**	**3,223**	**19,556,766**	**2,353,241**	**334,391,838**	**0**	**72,652**	**—**	

PARTIAL (some buckets priced, rate gaps below)

(none)

NO-USAGE (zero-token sessions)

vendor	model	platform	sessions	api calls	input	output	cache read	cache write	reasoning	USD	status
unknown	deepseek-v4-flash	cron	4	0	0	0	0	0	0	$0.00	no-usage
unknown	gpt-5.5	cron	11	0	0	0	0	0	0	$0.00	no-usage
unknown	kimi-k2.6	cron	4	0	0	0	0	0	0	$0.00	no-usage
TOTAL			**19**	**0**	**0**	**0**	**0**	**0**	**0**	**$0.00**	

Totals

metric	value
sessions in window	301 (101 open)
api calls	7,504
messages / tool calls	10,972 / 6,772
input tokens	44,065,179
output tokens	5,426,855
cache-read tokens	651,073,886

metric	value
cache-write tokens	0
reasoning tokens	375,303
metered USD	$9.27
quota USD	$0.00 (113 sessions, 1,981 api calls, 151,315,597 tokens)
combined USD	$9.27
unpriced tokens	356,374,497

Rates applied

vendor	model	window	input/M	output/M	cache-read/M	cache-write/M	reasoning/M	request
deepseek	deepseek-v4-flash	2026-07-01 → ∞	0.28	0.42	0.028	0.28	—	—
deepseek	deepseek-v4-flash	2026-08-01 → ∞	0.30	0.45	0.03	—	—	—
openai-codex	gpt-5.5	all dates	—	—	—	—	—	—
openai-codex	gpt-5.6-sol	all dates	—	—	—	—	—	—
openai-codex	gpt-4o	all dates	—	—	—	—	—	—

Quota provenance notes

Rows classified quota by billing_mode/provider where cost_status disagrees (PRD E4):

session	reason	billing_mode	cost_status
20260715_181753_e444d8de	billing_mode	subscription_included	unknown
20260719_093839_0fe24da2	billing_mode	subscription_included	unknown

test3_cost/glm.md

Hermes Cost Tracker Report

Summary

- **Total Sessions**: 301
- **Metered Sessions**: 171
- **Quota Sessions**: 111
- **Unknown Sessions**: 19
- **Error Sessions**: 0
- **Total Metered Cost**: $2886.431571

Cost by Provider

Provider	Sessions	Cost
xiaomi	18	$2273.800215
deepseek	88	$360.264057
kimi-coding	58	$110.581978
openai-codex	1	$106.816905
kimi-for-coding	1	$30.039296
zai	5	$4.929120

Cost by Model

Model	Sessions	Cost
deepseek-v4-flash	80	$2532.031801
kimi-k2.6	59	$140.621274
gpt-5.5	1	$106.816905
mimo-v2.5-pro	18	$91.504762
kimi-k1.5	4	$6.263898
deepseek-chat	3	$5.330342
glm-4.5-flash	4	$3.268916
gpt-4o	2	$0.593673

Top 20 Most Expensive Sessions

Session ID	Provider	Model	Type	Cost	Tokens	Est
20260726...	xiaomi	deepseek-v4-flash	metered	$2186.254653	10,764,184/1,390,401	✓
20260704...	deepseek	deepseek-v4-flash	metered	$143.598224	8,055,044/550,493	
20260719...	openai-codex	gpt-5.5	metered	$106.816905	1,295,237/63,982	✓
20260715...	deepseek	deepseek-v4-flash	metered	$65.588873	3,417,191/273,011	✓
20260704...	kimi-for-coding	kimi-k2.6	metered	$30.039296	1,571,380/143,571	
cron_c44...	xiaomi	mimo-v2.5-pro	metered	$17.746333	89,147/12,643	✓
20260806...	xiaomi	mimo-v2.5-pro	metered	$9.695145	84,038/5,515	✓
20260805...	xiaomi	mimo-v2.5-pro	metered	$8.957716	106,662/21,282	✓
20260804...	xiaomi	mimo-v2.5-pro	metered	$8.136433	108,857/11,459	✓
20260807...	deepseek	deepseek-v4-flash	metered	$7.573455	51,327/117,337	✓
20260803...	xiaomi	mimo-v2.5-pro	metered	$7.440506	91,573/17,771	✓
20260811...	deepseek	deepseek-v4-flash	metered	$6.483636	69,834/93,979	✓
20260806...	xiaomi	mimo-v2.5-pro	metered	$5.924719	66,306/5,850	✓
cron_c44...	xiaomi	mimo-v2.5-pro	metered	$5.109199	44,668/7,220	✓
cron_49a...	kimi-coding	kimi-k2.6	metered	$4.991767	263,056/20,032	✓
cron_2a9...	kimi-coding	kimi-k2.6	metered	$4.601455	231,278/21,268	✓
cron_2a9...	kimi-coding	kimi-k2.6	metered	$4.504247	238,415/17,813	✓

Session ID	Provider	Model	Type	Cost	Tokens	Est
cron_2a9...	kimi-coding	kimi-k2.6	metered	$4.430313	222,496/20,522	✓
cron_c44...	xiaomi	mimo-v2.5-pro	metered	$4.256230	58,258/7,304	✓
20260811...	deepseek	deepseek-v4-flash	metered	$4.187331	66,815/55,266	✓

test3_cost/mimo.md

Hermes Cost Report: 2026-07-04 -> 2026-08-13

Generated: 2026-08-13 20:40

Summary

Metric	Value
Total sessions	301
Fully priced	301
Unpriced	0
Total estimated cost	**$41.80**
Metered spend	$41.80
Quota spend	$0.00 (included)

Metered Spend (chat_completions)

Provider	Model	Sessions	Input Tokens	Output Tokens	Cache Read Tokens	Cost
[no-provider]	deepseek-v4-flash	4	0	0	0	$0.00
[no-provider]	gpt-5.5	11	0	0	0	$0.00
[no-provider]	kimi-k2.6	4	0	0	0	$0.00
deepseek	deepseek-chat	3	159,185	51,819	3,490,304	$0.13
deepseek	deepseek-v4-flash	77	11,526,267	2,001,794	179,600,320	$17.89
deepseek	gpt-4o	2	40,615	50	4,864	$0.11
deepseek	kimi-k1.5	4	269,347	40,324	2,617,472	$0.00
deepseek	mimo-v2.5-pro	1	80,603	47,898	1,356,800	$0.00
kimi-coding	kimi-k2.6	58	5,475,139	531,843	34,649,800	$0.00
kimi-for-coding	kimi-k2.6	1	1,571,380	143,571	9,497,152	$0.00
xiaomi	deepseek-v4-flash	1	10,764,184	1,390,401	269,643,712	$23.31
xiaomi	mimo-v2.5-pro	17	967,396	119,845	9,201,472	$0.00
zai	deepseek-v4-flash	1	91,739	5,600	1,582,494	$0.14
zai	glm-4.5-flash	4	137,178	21,890	2,347,768	$0.22
					Subtotal	**$41.80**

Quota Spend (subscription_included)

Provider	Model	Sessions	Input Tokens	Output Tokens	Cache Read Tokens	Cost
deepseek	deepseek-v4-flash	1	3,417,191	273,011	58,789,120	$0.00 (included)
openai-codex	gpt-5.5	44	4,635,488	380,087	42,939,520	$0.00 (included)
openai-codex	gpt-5.6-sol	68	4,929,467	418,722	35,353,088	$0.00 (included)
					Subtotal	**$0.00 (included)**

Top 10 Most Expensive Sessions

#	ID	Title	Provider	Model	Input	Output	Cache R	Cost
1	20260726_114449_	Making V4 Edits While Keeping V3	xiaomi	deepseek-v4-flash	10,764,184	1,390,401	269,643,712	$23.31
2	20260704_151329_	Checking If Jackie Is There	deepseek	deepseek-v4-flash	8,055,044	550,493	122,280,512	$11.34
3	20260807_173154_	(untitled)	deepseek	deepseek-v4-flash	51,327	117,337	5,225,344	$0.51
4	20260811_071813_	(untitled)	deepseek	deepseek-v4-flash	69,834	93,979	2,278,144	$0.28
5	cron_16189c3e0e1	Daily wiki ingest — EM & AI Tools · Aug	deepseek	deepseek-v4-flash	83,671	44,171	2,989,952	$0.28
6	cron_49a3abd8ad0	Wiki Gap Research — Execute · Aug 13 11:	deepseek	deepseek-v4-flash	107,712	44,296	2,380,416	$0.24
7	cron_2a95a4bb9ee	Wiki Gap Analysis — Question + Plan · Au	deepseek	deepseek-v4-flash	105,370	44,393	2,109,312	$0.22
8	cron_2a95a4bb9ee	Wiki Gap Analysis — Question + Plan · Au	deepseek	deepseek-v4-flash	140,018	35,367	2,062,464	$0.22
9	cron_49a3abd8ad0	Wiki Gap Research — Execute · Aug 11 11:	deepseek	deepseek-v4-flash	116,337	31,028	2,116,864	$0.21
10	20260811_173917_	(untitled)	deepseek	deepseek-v4-flash	66,815	55,266	1,691,648	$0.20

test3_cost/kimi.md

Hermes Cost Report

Period: 2026-07-04 → 2026-08-13 (inclusive)

Vendor	Model	Platform	Sessions	Input M	Output M	Cache-Read M	Reasoning M	Est. USD	Spend Type
deepseek	deepseek-chat	cron	3	0.159	0.052	3.490	0.000	$0.344	metered
deepseek	deepseek-v4-flash	cli	34	1.143	0.684	23.129	0.000	$2.68	metered
deepseek	deepseek-v4-flash	cron	34	2.174	0.591	31.144	0.000	$3.42	metered
deepseek	deepseek-v4-flash	subagent	8	0.154	0.177	3.047	0.000	$0.449	metered
deepseek	deepseek-v4-flash	telegram	1	3.417	0.273	58.789	0.049	-	quota
deepseek	deepseek-v4-flash	telegram	1	8.055	0.550	122.281	0.123	$11.34	metered
deepseek	gpt-4o	cli	2	0.041	0.000	0.005	0.000	$0.108	metered
deepseek	kimi-k1.5	cron	4	0.269	0.040	2.617	0.000	$0.220	metered
deepseek	mimo-v2.5-pro	subagent	1	0.081	0.048	1.357	0.000	$0.496	metered
kimi-coding	kimi-k2.6	cron	57	5.457	0.532	34.648	0.000	$12.85	metered

Vendor	Model	Platform	Sessions	Input M	Output M	Cache-Read M	Reasoning M	Est. USD	Spend Type
kimi-coding	kimi-k2.6	email	1	0.018	0.000	0.002	0.000	$0.018	metered
kimi-for-coding	kimi-k2.6	telegram	1	1.571	0.144	9.497	0.004	$3.59	metered
openai-codex	gpt-5.5	cli	1	0.033	0.001	0.135	0.000	-	quota
openai-codex	gpt-5.5	cron	40	3.258	0.314	31.100	0.037	-	quota
openai-codex	gpt-5.5	email	1	0.030	0.001	0.047	0.000	-	quota
openai-codex	gpt-5.5	telegram	2	1.315	0.064	11.658	0.011	-	quota
openai-codex	gpt-5.6-sol	cli	45	3.826	0.270	29.292	0.061	-	quota
openai-codex	gpt-5.6-sol	cron	4	0.102	0.004	0.279	0.001	-	quota
openai-codex	gpt-5.6-sol	subagent	19	1.002	0.145	5.783	0.021	-	quota
xiaomi	deepseek-v4-flash	telegram	1	10.764	1.390	269.644	0.069	$23.31	metered
xiaomi	mimo-v2.5-pro	cli	6	0.304	0.022	2.724	0.000	$0.914	metered
xiaomi	mimo-v2.5-pro	cron	8	0.356	0.048	4.108	0.000	$1.32	metered
xiaomi	mimo-v2.5-pro	subagent	3	0.307	0.051	2.370	0.000	$0.933	metered
zai	deepseek-v4-flash	cron	1	0.092	0.006	1.582	0.000	$0.142	metered
zai	glm-4.5-flash	cli	4	0.137	0.022	2.348	0.000	$0.035	metered

Total Metered USD: $62.17 **Total Quota Input Tokens:** 12,982,146 **Total Quota Output Tokens:** 1,071,820

- **Discarded Sessions:** 19 session(s) with zero tokens and missing model/provider were excluded from cost totals.

Notes

- Discarded Sessions: 19 session(s) with zero tokens and missing model/provider excluded from cost totals.

test3_cost/grok-4.6.md

Hermes Cost Tracker Report

Generated: 2026-08-17T14:30:48.580358+00:00 **Date window:** 2026-07-04T00:00:00+00:00 to 2026-08-19T00:00:00+00:00 **Timezone:** UTC **Database paths:** /tmp/hermes-cost-snapshot.db **Rates file:** /tmp/rates_grok_quoted.yaml **Rates hash:** 3c33583c122a86aa16bc49194160f34526bf352d04ab08b4b5201ecda8ee4bfa **Schema versions:** 17 **Total sessions:** 385 **Open sessions included:** 159

Cost Summary

Category	Amount
Cash total	$22.2468
Quota volume	91,476,039 tokens
Credits volume	0 tokens
Excluded sessions	60

By Vendor

Vendor	Cash USD	Sessions	Cache-read Tokens
kimi-coding	$13.4084	63	35,708,360
deepseek	$4.3119	126	299,410,368
kimi-for-coding	$3.6648	1	9,702,592
xiaomi	$0.7201	26	459,241,472
zai	$0.1416	27	5,548,692
openai-codex	$0.0000	112	80,983,936
None	$0.0000	19	0
xai	$0.0000	11	6,296,207

By Model (Top 10 by Cost)

- kimi-k2.6: $17.0732 (68 sessions)
- deepseek-v4-flash: $4.2653 (123 sessions)
- mimo-v2.5-pro: $0.7201 (24 sessions)
- glm-4.5-flash: $0.1416 (7 sessions)
- deepseek-chat: $0.0466 (3 sessions)
- gpt-5.5: $0.0000 (60 sessions)
- gpt-5.6-sol: $0.0000 (69 sessions)
- kimi-k1.5: $0.0000 (4 sessions)
- gpt-4o: $0.0000 (2 sessions)
- glm-4.6: $0.0000 (1 sessions)

By Platform

- cron: $14.1864 (191 sessions)
- telegram: $6.0085 (6 sessions)
- cli: $1.6916 (149 sessions)
- subagent: $0.3420 (37 sessions)
- email: $0.0183 (2 sessions)

Flag Summary

- INCLUDED: 112 sessions - Quota/subscription. Cash=$0
- QUOTED: 139 sessions - Rate row confidence=quoted (vendor docs/invoice)
- EST: 74 sessions - Rate row confidence=estimated (operator judgment)
- ZERO_TOKEN: 19 sessions - No tokens/no API calls
- UNPRICED: 41 sessions - Tokens known, no usable rate

Incomplete Sessions

- cron_ff9df506ae49_20260716_180018: ZERO_TOKEN - Session has no tokens or API calls
- cron_2a95a4bb9ee1_20260717_110029: ZERO_TOKEN - Session has no tokens or API calls
- cron_49a3abd8ad03_20260717_111130: ZERO_TOKEN - Session has no tokens or API calls
- cron_16189c3e0e13_20260717_140032: ZERO_TOKEN - Session has no tokens or API calls
- cron_2a95a4bb9ee1_20260723_110045: ZERO_TOKEN - Session has no tokens or API calls
- cron_49a3abd8ad03_20260723_111144: ZERO_TOKEN - Session has no tokens or API calls
- cron_6a206e30e232_20260723_112245: ZERO_TOKEN - Session has no tokens or API calls
- cron_2d8c4ba8b58d_20260723_113345: ZERO_TOKEN - Session has no tokens or API calls
- cron_16189c3e0e13_20260723_140047: ZERO_TOKEN - Session has no tokens or API calls
- cron_2a95a4bb9ee1_20260724_110031: ZERO_TOKEN - Session has no tokens or API calls
- cron_49a3abd8ad03_20260724_111131: ZERO_TOKEN - Session has no tokens or API calls
- cron_6a206e30e232_20260724_112232: ZERO_TOKEN - Session has no tokens or API calls
- cron_2d8c4ba8b58d_20260724_113332: ZERO_TOKEN - Session has no tokens or API calls
- cron_16189c3e0e13_20260724_140033: ZERO_TOKEN - Session has no tokens or API calls
- 20260726_114449_31da2578: UNPRICED - No rate available for this provider/model combination
- cron_2a95a4bb9ee1_20260726_145455: UNPRICED - No rate available for this provider/model combination
- cron_49a3abd8ad03_20260726_145824: UNPRICED - No rate available for this provider/model combination
- cron_6a206e30e232_20260726_150837: UNPRICED - No rate available for this provider/model combination
- cron_2d8c4ba8b58d_20260726_150920: UNPRICED - No rate available for this provider/model combination
- cron_16189c3e0e13_20260727_140017: ZERO_TOKEN - Session has no tokens or API calls
- 20260804_162013_6a440c: UNPRICED - No rate available for this provider/model combination
- 20260807_075532_ce43a0: UNPRICED - No rate available for this provider/model combination
- 20260807_095227_a9ff58: UNPRICED - No rate available for this provider/model combination
- cron_2a95a4bb9ee1_20260810_110013: ZERO_TOKEN - Session has no tokens or API calls
- cron_49a3abd8ad03_20260810_111113: ZERO_TOKEN - Session has no tokens or API calls
- cron_6a206e30e232_20260810_112214: ZERO_TOKEN - Session has no tokens or API calls
- cron_2d8c4ba8b58d_20260810_113314: ZERO_TOKEN - Session has no tokens or API calls
- cron_16189c3e0e13_20260811_140054: UNPRICED - No rate available for this provider/model combination
- cron_ee85be6bc8a4_20260813_150044: UNPRICED - No rate available for this provider/model combination
- 20260813_193413_1b89ff: UNPRICED - No rate available for this provider/model combination
- 20260813_194227_c13167: UNPRICED - No rate available for this provider/model combination
- 20260813_201350_cde80a: UNPRICED - No rate available for this provider/model combination
- 20260813_202302_3ec142: UNPRICED - No rate available for this provider/model combination
- 20260813_203355_3aca69: UNPRICED - No rate available for this provider/model combination
- 20260813_204656_ea9ba5: UNPRICED - No rate available for this provider/model combination
- 20260813_211131_c0cb72: UNPRICED - No rate available for this provider/model combination
- 20260813_211132_7f1814: UNPRICED - No rate available for this provider/model combination
- 20260813_211133_6767aa: UNPRICED - No rate available for this provider/model combination
- 20260814_072750_cc3223: UNPRICED - No rate available for this provider/model combination
- 20260814_073704_8914c6: UNPRICED - No rate available for this provider/model combination
- 20260814_073718_14cd97: UNPRICED - No rate available for this provider/model combination
- 20260814_075333_2dd59a: UNPRICED - No rate available for this provider/model combination
- 20260814_075426_cb0f94: UNPRICED - No rate available for this provider/model combination
- 20260814_075425_159040: UNPRICED - No rate available for this provider/model combination
- 20260814_075427_f8020f: UNPRICED - No rate available for this provider/model combination
- 20260814_101016_b1011e: UNPRICED - No rate available for this provider/model combination
- 20260814_101056_edd60e: UNPRICED - No rate available for this provider/model combination

- 20260814_101641_fd5e8f: UNPRICED - No rate available for this provider/model combination
- 20260814_101810_5508b2: UNPRICED - No rate available for this provider/model combination
- 20260814_101943_f7f2ce: UNPRICED - No rate available for this provider/model combination
- 20260814_102609_df2355: UNPRICED - No rate available for this provider/model combination
- 20260814_103016_b9e45f: UNPRICED - No rate available for this provider/model combination
- cron_04300d215d3f_20260816_170003: UNPRICED - No rate available for this provider/model combination
- 20260816_170642_cfa9e8: UNPRICED - No rate available for this provider/model combination
- 20260816_170720_f0efd4: UNPRICED - No rate available for this provider/model combination
- 20260816_170735_9008d3: UNPRICED - No rate available for this provider/model combination
- 20260816_170750_3f2838: UNPRICED - No rate available for this provider/model combination
- 20260816_170809_4f247b: UNPRICED - No rate available for this provider/model combination
- 20260816_170852_b2083e: UNPRICED - No rate available for this provider/model combination
- 20260816_171452_8c26e2: UNPRICED - No rate available for this provider/model combination

Applied Rates

- deepseek:deepseek-chat
- zai:glm-4.5-flash
- xiaomi:mimo-v2.5-pro
- deepseek:deepseek-v4-flash
- kimi-coding:kimi-k2.6

test3_cost/gpt-5.5.md

Hermes Cost Analysis Report

Generated on: 2026-08-17 00:36:25 UTC Database: /tmp/hermes-cost-snapshot.db

Executive Summary

- **Total Sessions Analyzed**: 385
- **Total Cost**: $150325.31
- **Average Cost per Session**: $390.4554

Data Quality Report

- **Total sessions**: 385
- **Sessions with missing provider**: 19 (4.9%) [INFERRED]
- **Sessions with missing model**: 0 (0.0%)
- **Zero-token sessions**: 19 (4.9%) [EXCLUDED]
- **Estimated costs**: 1 (0.3%) [ESTIMATED]
- **Calculated costs**: 121 (31.4%) [CALCULATED]
- **Actual costs**: 0 (0.0%)
- **Complete data**: 121 (31.4%)

Cost Breakdown by Vendor

Vendor	Cost	Percentage
xiaomi	$62204.78	41.4%
deepseek	$54468.77	36.2%
openai-codex	$30798.00	20.5%
kimi-for-coding	$2853.76	1.9%
zai	$0.00	0.0%
xai	$0.00	0.0%
kimi-coding	$0.00	0.0%

Cost Breakdown by Model

Model	Cost	Percentage
deepseek-v4-flash	$116026.64	77.2%
gpt-5.5	$30798.00	20.5%
kimi-k2.6	$2853.76	1.9%
deepseek-chat	$646.92	0.4%
gpt-5.6-sol	$0.00	0.0%
gpt-5.6-terra	$0.00	0.0%
gpt-5.6-luna	$0.00	0.0%
glm-4.5-flash	$0.00	0.0%
kimi-for-coding	$0.00	0.0%
mimo-v2.5	$0.00	0.0%
grok-4.6	$0.00	0.0%
mimo-v2.5-pro	$0.00	0.0%
glm-4.6	$0.00	0.0%

Model	Cost	Percentage
gpt-4o	$0.00	0.0%
kimi-k1.5	$0.00	0.0%

Billing Mode Analysis

Billing Mode	Cost	Sessions
unknown	$23242.57	274
subscription_included	$40640.13	0
chat_completions	$86442.62	0

Cost Status Distribution

Status	Count	Percentage
unknown	274	71.2%
included	111	28.8%

Top 10 Sessions by Cost

ID	Started	Model	Provider	Cost	Type	Tokens	Markers
20260726...	2026-07-26 18:44:49 UTC	deepseek-v4-flash	xiaomi	$62204.7804	calculated	461,309,508	[CALCULATED] Calculated from tokens + rate table
20260719...	2026-07-19 16:38:39 UTC	gpt-5.5	openai-codex	$30798.0040	calculated	15,836,742	[CALCULATED] Calculated from tokens + rate table
20260704...	2026-07-04 22:13:29 UTC	deepseek-v4-flash	deepseek	$21384.0812	calculated	130,886,049	[CALCULATED] Calculated from tokens + rate table
20260715...	2026-07-16 01:17:53 UTC	deepseek-v4-flash	deepseek	$9842.1250	calculated	62,479,322	[CALCULATED] Calculated from tokens + rate table
20260704...	2026-07-04 22:36:59 UTC	kimi-k2.6	kimi-for-coding	$2853.7552	calculated	11,432,233	[CALCULATED] Calculated from tokens + rate table
20260807...	2026-08-08 00:31:56 UTC	deepseek-v4-flash	deepseek	$808.5354	calculated	5,394,008	[CALCULATED] Calculated from tokens + rate table
cron_161...	2026-08-16 21:00:01 UTC	deepseek-v4-flash	deepseek	$768.3332	calculated	5,400,854	[CALCULATED] Calculated from tokens + rate table
cron_161...	2026-08-14 21:00:35 UTC	deepseek-v4-flash	deepseek	$706.1432	calculated	4,964,256	[CALCULATED] Calculated from tokens + rate table
cron_49a...	2026-08-14 18:15:41 UTC	deepseek-v4-flash	deepseek	$694.2882	calculated	4,290,181	[CALCULATED] Calculated from tokens + rate table
cron_2a9...	2026-08-14 18:00:32 UTC	deepseek-v4-flash	deepseek	$542.8898	calculated	3,175,383	[CALCULATED] Calculated from tokens + rate table

test3_cost/gpt-5.6-luna.md

Hermes Cost Report

- Window (UTC): beginning to end
- Sessions: 385
- Metered cost: $908.0289
- Quota sessions: 111 (excluded from metered cost)
- Estimated rows: 385

Summary by vendor / model / platform / billing mode

Vendor	Model	Platform	Billing mode	Sessions	Input	Output	Cache read	Cost
xiaomi	deepseek-v4-flash	telegram	chat_completions	1	13340002	2140962	445828544	$469.8734 [ESTIMATE]
deepseek	deepseek-v4-flash	telegram	chat_completions	1	8055044	550493	122280512	$133.0880 [ESTIMATE]
deepseek	deepseek-v4-flash	cron	chat_completions	53	3982473	1179273	66544768	$76.4236 [ESTIMATE]
deepseek	deepseek-v4-flash	telegram	subscription_included	1	3417191	273011	58789120	$63.5714 [ESTIMATE]
deepseek	deepseek-v4-flash	cli	chat_completions	47	1936577	1100953	40071040	$47.5124 [ESTIMATE]
kimi-coding	kimi-k2.6	cron	chat_completions	57	5457012	531644	34648264	$42.7635 [ESTIMATE]
openai-codex	gpt-5.5	telegram	subscription_included	2	1428444	78795	14349184	$16.1514 [ESTIMATE]
kimi-for-coding	kimi-k2.6	telegram	chat_completions	1	1575786	153855	9702592	$12.0477 [ESTIMATE]
xai	grok-4.6	cli	chat_completions	11	798622	177992	6296207	$7.9848 [ESTIMATE]
xiaomi	mimo-v2.5-pro	cli	chat_completions	12	535799	78918	5832576	$6.7630 [ESTIMATE]
deepseek	deepseek-v4-flash	subagent	chat_completions	14	339235	344891	4255488	$6.3192 [ESTIMATE]
xiaomi	mimo-v2.5-pro	cron	chat_completions	8	356103	47781	4107648	$4.7027 [ESTIMATE]
deepseek	deepseek-chat	cron	chat_completions	3	159185	51819	3490304	$3.9086 [ESTIMATE]
zai	glm-4.5-flash	cli	chat_completions	7	184200	34952	2902884	$3.2618 [ESTIMATE]
deepseek	kimi-k1.5	cron	chat_completions	4	269347	40324	2617472	$3.0884 [ESTIMATE]
xiaomi	mimo-v2.5-pro	subagent	chat_completions	3	307092	50512	2370048	$2.9297 [ESTIMATE]
zai	deepseek-v4-flash	cron	chat_completions	2	124879	8218	1725030	$1.8910 [ESTIMATE]
deepseek	mimo-v2.5-pro	subagent	chat_completions	1	80603	47898	1356800	$1.6769 [ESTIMATE]
kimi-coding	kimi-k2.6	cli	chat_completions	5	236242	35474	1058560	$1.4722 [ESTIMATE]
xiaomi	mimo-v2.5	cli	chat_completions	2	75030	45083	1102656	$1.4031 [ESTIMATE]
zai	gpt-5.5	cli	chat_completions	4	52783	4737	369060	$0.4455 [ESTIMATE]
zai	kimi-for-coding	cli	chat_completions	9	22498	9541	288246	$0.3584 [ESTIMATE]
zai	gpt-5.5	cron	chat_completions	1	23530	3248	197587	$0.2374 [ESTIMATE]
deepseek	gpt-4o	cli	chat_completions	2	40615	50	4864	$0.0457 [ESTIMATE]
zai	gpt-5.6-terra	cli	chat_completions	1	3	85	21962	$0.0224 [ESTIMATE]
zai	gpt-5.6-luna	cli	chat_completions	1	4	83	21961	$0.0224 [ESTIMATE]
zai	gpt-5.6-sol	cli	chat_completions	1	3	72	21962	$0.0223 [ESTIMATE]

Vendor	Model	Platform	Billing mode	Sessions	Input	Output	Cache read	Cost
zai	glm-4.6	cli	chat_completions	1	21222	49	0	$0.0215 [ESTIMATE]
kimi-coding	kimi-k2.6	email	chat_completions	1	18127	199	1536	$0.0207 [ESTIMATE]
openai-codex	gpt-5.5	email	subscription_included	1	29630	1125	47104	$0.0000 [ESTIMATE]
unknown-vendor	deepseek-v4-flash	cron	chat_completions	4	0	0	0	$0.0000 [ESTIMATE]
openai-codex	gpt-5.6-sol	subagent	subscription_included	19	1002250	144669	5783040	$0.0000 [ESTIMATE]
openai-codex	gpt-5.6-sol	cron	subscription_included	4	101547	3573	278528	$0.0000 [ESTIMATE]
openai-codex	gpt-5.5	cron	subscription_included	40	3257931	313850	31099904	$0.0000 [ESTIMATE]
openai-codex	gpt-5.6-sol	cli	subscription_included	45	3825670	270480	29291520	$0.0000 [ESTIMATE]
unknown-vendor	gpt-5.5	cron	chat_completions	11	0	0	0	$0.0000 [ESTIMATE]
openai-codex	gpt-5.5	cli	subscription_included	1	33142	997	134656	$0.0000 [ESTIMATE]
unknown-vendor	kimi-k2.6	cron	chat_completions	4	0	0	0	$0.0000 [ESTIMATE]

Data quality

Zero-token sessions are included in counts and cost $0. Missing token counts use the vendor/model historical average, or the report-wide average when unavailable. Missing provider/model values use operator mappings when supplied, otherwise `unknown-vendor/unknown-model`.

Assumptions

The PRD did not include a rate table. Unless a config file is supplied, this report uses the explicit fallback of $1.00 per million input/cache-read tokens and $5.00 per million output tokens for every model, so calculated metered amounts are estimates. The snapshot has no dedicated platform field; session `source` is used as platform. Quota sessions are tracked but costed at $0 in metered totals.

test3_cost/gpt-5.6-terra.md

Hermes Cost Report

Report metadata

- Generated at (UTC): 2026-08-17T03:29:25.297401Z
- Database (read-only): `/tmp/hermes-cost-snapshot.db`
- Requested window: 2026-07-04 through 2026-08-17 inclusive; timezone: UTC
- Exact UTC bounds: [2026-07-04T00:00:00Z, 2026-08-18T00:00:00Z)
- Latest included session started_at: 2026-08-17T00:14:53.826926Z
- Archived sessions: excluded
- Rate table: `none supplied`
- Rate-table SHA-256: `none (no rate table supplied)`
- Mapping table: `none supplied`

Headline totals

- Calculated metered API estimate: **$0.00** (0 priced sessions)
- Included/quota: **113 sessions**, 153,955,361 billable tokens; **$0.00 variable API spend — INCLUDED / NOT PRICED AS API SPEND**
- Unpriced metered usage: **253 sessions**, 801,749,693 billable tokens; **N/A** (excluded from USD total)
- Zero-token sessions: **19** (excluded from metered USD totals)
- Coverage: **0.0%** priced sessions / eligible metered sessions (0 / 253); 253 unpriced sessions

Calculated metered usage by vendor, model, and platform

None.

Included / quota usage by vendor, model, and platform

Variable USD is $0.00 only because this usage is included/quota, not because it is free metered API usage.

Vendor	Model	Platform	Sessions	Input	Cache read	Cache write	Output	Reasoning	Calculated USD	Status	Rate ID / quality
deepseek	deepseek-v4-flash	telegram	1	3,417,191	58,789,120	0	273,011	48,505	$0.00	INCLUDED (review)	—
openai-codex	gpt-5.5	cli	1	33,142	134,656	0	997	174	$0.00	INCLUDED (review)	—
openai-codex	gpt-5.5	cron	40	3,257,931	31,099,904	0	313,850	36,538	$0.00	INCLUDED (review)	—
openai-codex	gpt-5.5	email	1	29,630	47,104	0	1,125	316	$0.00	INCLUDED (review)	—
openai-codex	gpt-5.5	telegram	2	1,428,444	14,349,184	0	78,795	11,067	$0.00	INCLUDED (review)	—
openai-codex	gpt-5.6-sol	cli	45	3,825,670	29,291,520	0	270,480	61,316	$0.00	INCLUDED (review)	—
openai-codex	gpt-5.6-sol	cron	4	101,547	278,528	0	3,573	1,155	$0.00	INCLUDED (review)	—
openai-codex	gpt-5.6-sol	subagent	19	1,002,250	5,783,040	0	144,669	20,832	$0.00	INCLUDED (review)	—

Unpriced usage by vendor, model, and platform

Vendor	Model	Platform	Sessions	Input	Cache read	Cache write	Output	Reasoning	Calculated USD	Status	Rate ID / quality
deepseek	deepseek-chat	cron	3	159,185	3,490,304	0	51,819	0	N/A	UNPRICED	—
deepseek	deepseek-v4-flash	cli	47	1,936,577	40,071,040	0	1,100,953	0	N/A	UNPRICED	—
deepseek	deepseek-v4-flash	cron	53	3,982,473	66,544,768	0	1,179,273	0	N/A	UNPRICED	—
deepseek	deepseek-v4-flash	subagent	14	339,235	4,255,488	0	344,891	0	N/A	UNPRICED	—
deepseek	deepseek-v4-flash	telegram	1	8,055,044	122,280,512	0	550,493	122,748	N/A	UNPRICED	—
deepseek	gpt-4o	cli	2	40,615	4,864	0	50	0	N/A	UNPRICED	—
deepseek	kimi-k1.5	cron	4	269,347	2,617,472	0	40,324	0	N/A	UNPRICED	—
deepseek	mimo-v2.5-pro	subagent	1	80,603	1,356,800	0	47,898	0	N/A	UNPRICED	—
kimi-coding	kimi-k2.6	cli	5	236,242	1,058,560	0	35,474	0	N/A	UNPRICED	—
kimi-coding	kimi-k2.6	cron	57	5,457,012	34,648,264	0	531,644	0	N/A	UNPRICED	—
kimi-coding	kimi-k2.6	email	1	18,127	1,536	0	199	0	N/A	UNPRICED	—
kimi-for-coding	kimi-k2.6	telegram	1	1,575,786	9,702,592	0	153,855	3,586	N/A	UNPRICED	—
xai	grok-4.6	cli	11	798,622	6,296,207	0	177,992	89,185	N/A	UNPRICED	—
xiaomi	deepseek-v4-flash	telegram	1	13,340,002	445,828,544	0	2,140,962	69,066	N/A	UNPRICED	—
xiaomi	mimo-v2.5	cli	2	75,030	1,102,656	0	45,083	0	N/A	UNPRICED	—
xiaomi	mimo-v2.5-pro	cli	12	535,799	5,832,576	0	78,918	0	N/A	UNPRICED	—
xiaomi	mimo-v2.5-pro	cron	8	356,103	4,107,648	0	47,781	0	N/A	UNPRICED	—
xiaomi	mimo-v2.5-pro	subagent	3	307,092	2,370,048	0	50,512	0	N/A	UNPRICED	—
zai	deepseek-v4-flash	cron	2	124,879	1,725,030	0	8,218	0	N/A	UNPRICED	—
zai	glm-4.5-flash	cli	7	184,200	2,902,884	0	34,952	0	N/A	UNPRICED	—
zai	glm-4.6	cli	1	21,222	0	0	49	0	N/A	UNPRICED	—
zai	gpt-5.5	cli	4	52,783	369,060	0	4,737	0	N/A	UNPRICED	—
zai	gpt-5.5	cron	1	23,530	197,587	0	3,248	0	N/A	UNPRICED	—
zai	gpt-5.6-luna	cli	1	4	21,961	0	83	0	N/A	UNPRICED	—
zai	gpt-5.6-sol	cli	1	3	21,962	0	72	0	N/A	UNPRICED	—
zai	gpt-5.6-terra	cli	1	3	21,962	0	85	0	N/A	UNPRICED	—
zai	kimi-for-coding	cli	9	22,498	288,246	0	9,541	0	N/A	UNPRICED	—

Data-quality appendix

Session ID	Started (UTC)	Status	Reason	Original provider/model	Resolved provider/model	Mapping ID
20260816_171452_8c26e2	2026-08-17T00:14:53.826926Z	UNPRICED	MISSING_RATE	zai/gpt-5.5	zai/gpt-5.5	—
20260816_170852_b2083e	2026-08-17T00:08:54.428429Z	UNPRICED	MISSING_RATE	zai/gpt-5.5	zai/gpt-5.5	—
20260816_170809_4f247b	2026-08-17T00:08:11.334237Z	UNPRICED	MISSING_RATE	zai/gpt-5.6-sol	zai/gpt-5.6-sol	—
20260816_170750_3f2838	2026-08-17T00:07:52.762042Z	UNPRICED	MISSING_RATE	zai/gpt-5.6-terra	zai/gpt-5.6-terra	—
20260816_170735_9008d3	2026-08-17T00:07:37.295521Z	UNPRICED	MISSING_RATE	zai/gpt-5.6-luna	zai/gpt-5.6-luna	—
20260816_170720_f0efd4	2026-08-17T00:07:22.049479Z	UNPRICED	MISSING_RATE	zai/gpt-5.5	zai/gpt-5.5	—
20260816_170642_cfa9e8	2026-08-17T00:06:44.559449Z	UNPRICED	MISSING_RATE	zai/gpt-5.5	zai/gpt-5.5	—
cron_04300d215d3f_20260816_170003	2026-08-17T00:00:03.761773Z	UNPRICED	MISSING_RATE	zai/gpt-5.5	zai/gpt-5.5	—
cron_16189c3e0e13_20260816_140000	2026-08-16T21:00:01.372105Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260816_121708_b36f0e	2026-08-16T19:17:13.730624Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260816_121708_583ed6	2026-08-16T19:17:13.079288Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260816_121708_18ccd2	2026-08-16T19:17:13.068369Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260816_113357	2026-08-16T18:33:58.113009Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260816_112257	2026-08-16T18:22:57.925144Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—

Session ID	Started (UTC)	Status	Reason	Original provider/model	Resolved provider/model	Mapping ID
cron_49a3abd8ad03_20260816_111327	2026-08-16T18:13:27.864214Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260816_110057	2026-08-16T18:00:58.316495Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260815_183701	2026-08-16T01:37:02.221410Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260815_183643	2026-08-16T01:36:45.669941Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_16189c3e0e13_20260815_183643	2026-08-16T01:36:45.591094Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260815_111209	2026-08-15T18:12:10.110417Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260815_110052	2026-08-15T18:00:52.861444Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260815_105045_368f20	2026-08-15T17:50:46.703670Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_213324_6f8abd	2026-08-15T04:33:25.258310Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_210542_9731f6	2026-08-15T04:05:42.581203Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_194808_455d7e	2026-08-15T02:48:19.037521Z	UNPRICED	MISSING_RATE	zai/glm-4.5-flash	zai/glm-4.5-flash	—
20260814_194430_01d828	2026-08-15T02:44:30.624548Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_171155_2b84de	2026-08-15T00:11:56.036083Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_160008_47f64a	2026-08-14T23:00:08.943552Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_c44dd53f3e81_20260814_151135	2026-08-14T22:11:36.138351Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_ee85be6bc8a4_20260814_150035	2026-08-14T22:00:36.943358Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_16189c3e0e13_20260814_140034	2026-08-14T21:00:35.677737Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260814_113332	2026-08-14T18:33:33.119242Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260814_112530	2026-08-14T18:25:31.380133Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260814_111541	2026-08-14T18:15:41.960525Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260814_110032	2026-08-14T18:00:32.525937Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_103016_b9e45f	2026-08-14T17:30:18.528539Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_102609_df2355	2026-08-14T17:26:12.023226Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_101943_f7f2ce	2026-08-14T17:19:45.663018Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_101810_5508b2	2026-08-14T17:18:12.825682Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_101641_fd5e8f	2026-08-14T17:16:43.498845Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_101056_edd60e	2026-08-14T17:10:59.088635Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_101016_b1011e	2026-08-14T17:10:19.259511Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_090810_a353f9	2026-08-14T16:08:12.515895Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_085449_6cf3d6	2026-08-14T15:54:51.891562Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260814_075427_f8020f	2026-08-14T14:54:30.169686Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260814_075425_159040	2026-08-14T14:54:29.377193Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5	xiaomi/mimo-v2.5	—
20260814_075426_cb0f94	2026-08-14T14:54:28.795393Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260814_075333_2dd59a	2026-08-14T14:53:35.633171Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260814_073718_14cd97	2026-08-14T14:37:20.997725Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260814_073704_8914c6	2026-08-14T14:37:07.098911Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260814_072750_cc3223	2026-08-14T14:27:53.041770Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260813_211133_6767aa	2026-08-14T04:11:36.319873Z	UNPRICED	MISSING_RATE	zai/kimi-for-coding	zai/kimi-for-coding	—
20260813_211132_7f1814	2026-08-14T04:11:35.049684Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260813_211131_c0cb72	2026-08-14T04:11:34.420547Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5	xiaomi/mimo-v2.5	—
20260813_204656_ea9ba5	2026-08-14T03:46:58.603673Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260813_204545_27a848	2026-08-14T03:45:47.446206Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260813_204137_a273fe	2026-08-14T03:41:39.955548Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260813_203555_1a94da	2026-08-14T03:35:57.221408Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—

Session ID	Started (UTC)	Status	Reason	Original provider/model	Resolved provider/model	Mapping ID
20260813_203355_3aca69	2026-08-14T03:33:57.206426Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260813_203216_4cb9e7	2026-08-14T03:32:19.258027Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260813_203134_20f70a	2026-08-14T03:31:36.262649Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_202833_b09d56	2026-08-14T03:28:36.287751Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_202733_f8496b	2026-08-14T03:27:36.651586Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_202433_5d7462	2026-08-14T03:24:35.778778Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_202302_3ec142	2026-08-14T03:23:04.618888Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260813_202301_418c28	2026-08-14T03:23:03.268459Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260813_202204_a5429c	2026-08-14T03:22:06.569717Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260813_201934_d371ad	2026-08-14T03:19:37.500236Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260813_201350_cde80a	2026-08-14T03:13:53.135674Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260813_201349_82c4e4	2026-08-14T03:13:52.228747Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260813_201348_46d618	2026-08-14T03:13:51.802804Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260813_194227_c13167	2026-08-14T02:42:28.743480Z	UNPRICED	MISSING_RATE	xai/grok-4.6	xai/grok-4.6	—
20260813_193514_88b9f4	2026-08-14T02:35:16.098294Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260813_193413_1b89ff	2026-08-14T02:34:14.557861Z	UNPRICED	MISSING_RATE	zai/glm-4.6	zai/glm-4.6	—
20260813_193252_41e100	2026-08-14T02:32:54.023864Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_c44dd53f3e81_20260813_151144	2026-08-13T22:11:44.893739Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_ee85be6bc8a4_20260813_150044	2026-08-13T22:00:45.468873Z	UNPRICED	MISSING_RATE	zai/deepseek-v4-flash	zai/deepseek-v4-flash	—
20260813_145036_b1a5ce	2026-08-13T21:50:48.992494Z	UNPRICED	MISSING_RATE	zai/glm-4.5-flash	zai/glm-4.5-flash	—
20260813_144543_7a3868	2026-08-13T21:45:57.241913Z	UNPRICED	MISSING_RATE	zai/glm-4.5-flash	zai/glm-4.5-flash	—
cron_16189c3e0e13_20260813_140043	2026-08-13T21:00:44.632898Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_123229_d31784	2026-08-13T19:32:32.119078Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_123033_dce3fe	2026-08-13T19:30:36.327271Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_122833_c9e3d6	2026-08-13T19:28:36.025210Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_122723_d4dd4a	2026-08-13T19:27:26.331703Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260813_113341	2026-08-13T18:33:41.677005Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260813_112241	2026-08-13T18:22:41.684304Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260813_111142	2026-08-13T18:11:42.916022Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260813_110040	2026-08-13T18:00:41.894536Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_074302_1c8aff	2026-08-13T14:43:03.001744Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_073927_537bfd	2026-08-13T14:39:29.075888Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_072654_1eac29	2026-08-13T14:27:10.689510Z	UNPRICED	MISSING_RATE	zai/glm-4.5-flash	zai/glm-4.5-flash	—
20260813_072014_cf018a	2026-08-13T14:20:22.572691Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_071642_4aa3a8	2026-08-13T14:16:58.765804Z	UNPRICED	MISSING_RATE	zai/glm-4.5-flash	zai/glm-4.5-flash	—
20260813_071641_dff3ba	2026-08-13T14:16:44.491689Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_071631_5c051b	2026-08-13T14:16:33.767053Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_071619_0f2427	2026-08-13T14:16:22.565388Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_f9400d82bfad_20260813_071257	2026-08-13T14:13:00.298830Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_071116_83ea7b	2026-08-13T14:11:19.149097Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_071105_26fafc	2026-08-13T14:11:08.531926Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260813_070805_8d1966	2026-08-13T14:08:20.885444Z	UNPRICED	MISSING_RATE	zai/glm-4.5-flash	zai/glm-4.5-flash	—
20260812_222701_a41705	2026-08-13T05:27:04.632256Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_222011_880258	2026-08-13T05:20:13.258758Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—

Session ID	Started (UTC)	Status	Reason	Original provider/model	Resolved provider/model	Mapping ID
20260812_221911_c252d1	2026-08-13T05:19:13.180886Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_221456_43ec99	2026-08-13T05:14:57.725325Z	UNPRICED	MISSING_RATE	zai/glm-4.5-flash	zai/glm-4.5-flash	—
20260812_221050_c9b225	2026-08-13T05:10:52.113489Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_220848_7a194a	2026-08-13T05:08:50.235721Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_220050_628484	2026-08-13T05:00:51.218677Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_215724_22fa52	2026-08-13T04:57:24.791372Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_164636_b44903	2026-08-12T23:46:38.932915Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_163435_682236	2026-08-12T23:34:37.479590Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_c44dd53f3e81_20260812_151110	2026-08-12T22:11:11.211456Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_ee85be6bc8a4_20260812_150010	2026-08-12T22:00:11.919307Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_16189c3e0e13_20260812_140010	2026-08-12T21:00:10.573256Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_130208_43377f	2026-08-12T20:02:11.098910Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_123604_e5b2cf	2026-08-12T19:36:07.068135Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_122903_06aaa9	2026-08-12T19:29:06.200841Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260812_113307	2026-08-12T18:33:08.004394Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260812_112207	2026-08-12T18:22:08.208870Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260812_111448	2026-08-12T18:14:49.410884Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260812_110006	2026-08-12T18:00:08.734241Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_072624_dbfacd	2026-08-12T14:26:27.360148Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260812_071923_c2bcd7	2026-08-12T14:19:27.222370Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_180721_276357	2026-08-12T01:07:24.050314Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_175320_2e047c	2026-08-12T00:53:22.488124Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_173917_3a6899	2026-08-12T00:39:20.283811Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_170412_d079fc	2026-08-12T00:04:15.288392Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_162415_1b59dc	2026-08-11T23:24:19.225722Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_c44dd53f3e81_20260811_151155	2026-08-11T22:11:55.998857Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_ee85be6bc8a4_20260811_150055	2026-08-11T22:00:56.684611Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_16189c3e0e13_20260811_140054	2026-08-11T21:00:55.736818Z	UNPRICED	MISSING_RATE	zai/deepseek-v4-flash	zai/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260811_113352	2026-08-11T18:33:53.263772Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260811_112252	2026-08-11T18:22:53.158979Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260811_111357	2026-08-11T18:13:58.126072Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260811_110052	2026-08-11T18:00:53.418099Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_074919_ee17c6	2026-08-11T14:49:21.826754Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_073817_0df892	2026-08-11T14:38:20.583690Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_072416_8b974c	2026-08-11T14:24:19.276170Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_071813_fee206	2026-08-11T14:18:13.972257Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260811_071515_550f8b	2026-08-11T14:15:18.001908Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260810_225222_e56c3d	2026-08-11T05:52:25.859117Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260810_224622_6d259b	2026-08-11T05:46:25.169841Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260810_113314	2026-08-10T18:33:14.562113Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/kimi-k2.6	—/kimi-k2.6	—
cron_6a206e30e232_20260810_112214	2026-08-10T18:22:14.354307Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/kimi-k2.6	—/kimi-k2.6	—
cron_49a3abd8ad03_20260810_111113	2026-08-10T18:11:14.754736Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/kimi-k2.6	—/kimi-k2.6	—
cron_2a95a4bb9ee1_20260810_110013	2026-08-10T18:00:14.160830Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/kimi-k2.6	—/kimi-k2.6	—
20260810_100452_140931	2026-08-10T17:04:55.988434Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—

Session ID	Started (UTC)	Status	Reason	Original provider/model	Resolved provider/model	Mapping ID
20260810_095251_169f76	2026-08-10T16:52:54.455186Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260810_094651_8810aa	2026-08-10T16:46:54.291849Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260809_113329	2026-08-09T18:33:29.655479Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260809_112336	2026-08-09T18:23:37.159614Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260809_111600	2026-08-09T18:16:01.057302Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260809_110028	2026-08-09T18:00:29.613034Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2d8c4ba8b58d_20260808_113307	2026-08-08T18:33:08.435773Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260808_112207	2026-08-08T18:22:07.465334Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260808_111106	2026-08-08T18:11:07.105251Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260808_110006	2026-08-08T18:00:07.361113Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260807_194745_7cbfd1	2026-08-08T02:47:46.224610Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260807_194638_c761ca	2026-08-08T02:46:39.198977Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260807_173154_414b41	2026-08-08T00:31:56.054810Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260807_172651_039e66	2026-08-08T00:26:52.642333Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_16189c3e0e13_20260807_140037	2026-08-07T21:00:38.568958Z	UNPRICED	MISSING_RATE	deepseek/deepseek-chat	deepseek/deepseek-chat	—
cron_2d8c4ba8b58d_20260807_113335	2026-08-07T18:33:36.085088Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260807_112628	2026-08-07T18:26:28.743395Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260807_111135	2026-08-07T18:11:35.706902Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260807_110035	2026-08-07T18:00:36.501975Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260807_095227_a9ff58	2026-08-07T16:52:30.397329Z	UNPRICED	MISSING_RATE	deepseek/gpt-4o	deepseek/gpt-4o	—
20260807_075532_ce43a0	2026-08-07T14:55:35.374007Z	UNPRICED	MISSING_RATE	deepseek/gpt-4o	deepseek/gpt-4o	—
20260807_075503_d9d0e0	2026-08-07T14:55:06.324874Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260806_213355_011d04	2026-08-07T04:33:57.604176Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260806_213255_6ac5d6	2026-08-07T04:32:58.642229Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260806_193746_3fca8f	2026-08-07T02:37:49.294206Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260806_193545_33b844	2026-08-07T02:35:48.759381Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260806_193345_617f8d	2026-08-07T02:33:48.780544Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_c44dd53f3e81_20260806_151130	2026-08-06T22:11:31.712223Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_ee85be6bc8a4_20260806_150030	2026-08-06T22:00:31.696585Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_2d8c4ba8b58d_20260806_113327	2026-08-06T18:33:27.882701Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260806_112227	2026-08-06T18:22:27.815719Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260806_111127	2026-08-06T18:11:27.613612Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260806_110027	2026-08-06T18:00:30.408625Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_c44dd53f3e81_20260805_151130	2026-08-05T22:11:31.296886Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_ee85be6bc8a4_20260805_150030	2026-08-05T22:00:31.825285Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_2d8c4ba8b58d_20260805_113326	2026-08-05T18:33:27.293246Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260805_112226	2026-08-05T18:22:27.304654Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260805_111126	2026-08-05T18:11:26.906244Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260805_110026	2026-08-05T18:00:27.412516Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260805_081952_6b32cc	2026-08-05T15:19:53.575256Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260804_162013_6a440c	2026-08-04T23:20:14.414831Z	UNPRICED	MISSING_RATE	deepseek/mimo-v2.5-pro	deepseek/mimo-v2.5-pro	—
cron_c44dd53f3e81_20260804_151112	2026-08-04T22:11:13.301470Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_ee85be6bc8a4_20260804_150012	2026-08-04T22:00:14.133771Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_2d8c4ba8b58d_20260804_113306	2026-08-04T18:33:06.955455Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—

Session ID	Started (UTC)	Status	Reason	Original provider/model	Resolved provider/model	Mapping ID
cron_6a206e30e232_20260804_112206	2026-08-04T18:22:06.991620Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260804_111106	2026-08-04T18:11:06.681824Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260804_110006	2026-08-04T18:00:07.275392Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260804_080536_849c95	2026-08-04T15:05:37.635309Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
20260803_162138_1ad489	2026-08-03T23:21:38.915206Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_c44dd53f3e81_20260803_151151	2026-08-03T22:11:52.539071Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_ee85be6bc8a4_20260803_150051	2026-08-03T22:00:53.016381Z	UNPRICED	MISSING_RATE	xiaomi/mimo-v2.5-pro	xiaomi/mimo-v2.5-pro	—
cron_2d8c4ba8b58d_20260803_113349	2026-08-03T18:33:54.776146Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260803_112249	2026-08-03T18:22:49.501186Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260803_111148	2026-08-03T18:11:49.211643Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260803_110048	2026-08-03T18:00:50.444568Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_04300d215d3f_20260802_170003	2026-08-03T00:00:04.808042Z	UNPRICED	MISSING_RATE	deepseek/deepseek-chat	deepseek/deepseek-chat	—
cron_16189c3e0e13_20260802_140027	2026-08-02T21:00:29.818959Z	UNPRICED	MISSING_RATE	deepseek/deepseek-chat	deepseek/deepseek-chat	—
cron_2d8c4ba8b58d_20260802_113345	2026-08-02T18:33:45.870021Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260802_112338	2026-08-02T18:23:39.228812Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260802_111144	2026-08-02T18:11:45.061948Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260802_110044	2026-08-02T18:00:45.467936Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260802_074931	2026-08-02T14:49:31.500003Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2d8c4ba8b58d_20260801_113327	2026-08-01T18:33:28.174785Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260801_112227	2026-08-01T18:22:27.990723Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260801_111127	2026-08-01T18:11:27.654459Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260801_110028	2026-08-01T18:00:30.312538Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2d8c4ba8b58d_20260731_113318	2026-07-31T18:33:18.822044Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260731_112218	2026-07-31T18:22:18.911724Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260731_111117	2026-07-31T18:11:18.212118Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260731_110018	2026-07-31T18:00:20.617377Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2d8c4ba8b58d_20260730_113323	2026-07-30T18:33:23.374768Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260730_112223	2026-07-30T18:22:23.377898Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260730_111122	2026-07-30T18:11:23.085547Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260730_110023	2026-07-30T18:00:25.455810Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2d8c4ba8b58d_20260729_113333	2026-07-29T18:33:33.879926Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260729_112348	2026-07-29T18:23:48.573152Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260729_111133	2026-07-29T18:11:33.465882Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260729_110033	2026-07-29T18:00:34.419056Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
20260729_080252_290c97df	2026-07-29T15:02:52.158277Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2d8c4ba8b58d_20260728_113307	2026-07-28T18:33:08.384903Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260728_112427	2026-07-28T18:24:27.797997Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260728_111107	2026-07-28T18:11:07.859294Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260728_110007	2026-07-28T18:00:08.480272Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_16189c3e0e13_20260727_140017	2026-07-27T21:00:17.902805Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_2d8c4ba8b58d_20260727_113349	2026-07-27T18:33:50.131046Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_6a206e30e232_20260727_112249	2026-07-27T18:22:49.952812Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_49a3abd8ad03_20260727_111149	2026-07-27T18:11:49.661220Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—
cron_2a95a4bb9ee1_20260727_110049	2026-07-27T18:00:50.419696Z	UNPRICED	MISSING_RATE	kimi-coding/kimi-k2.6	kimi-coding/kimi-k2.6	—

Session ID	Started (UTC)	Status	Reason	Original provider/model	Resolved provider/model	Mapping ID
cron_2d8c4ba8b58d_20260726_150920	2026-07-26T22:09:20.594676Z	UNPRICED	MISSING_RATE	deepseek/kimi-k1.5	deepseek/kimi-k1.5	—
cron_6a206e30e232_20260726_150837	2026-07-26T22:08:38.227579Z	UNPRICED	MISSING_RATE	deepseek/kimi-k1.5	deepseek/kimi-k1.5	—
cron_49a3abd8ad03_20260726_145824	2026-07-26T21:58:25.343462Z	UNPRICED	MISSING_RATE	deepseek/kimi-k1.5	deepseek/kimi-k1.5	—
cron_2a95a4bb9ee1_20260726_145455	2026-07-26T21:54:55.400820Z	UNPRICED	MISSING_RATE	deepseek/kimi-k1.5	deepseek/kimi-k1.5	—
20260726_114449_31da2578	2026-07-26T18:44:49.150332Z	UNPRICED	MISSING_RATE	xiaomi/deepseek-v4-flash	xiaomi/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260726_113353	2026-07-26T18:33:53.900009Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260726_112253	2026-07-26T18:22:53.908528Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260726_111153	2026-07-26T18:11:53.716066Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260726_110053	2026-07-26T18:00:54.031291Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2d8c4ba8b58d_20260725_113317	2026-07-25T18:33:17.978259Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_6a206e30e232_20260725_112217	2026-07-25T18:22:17.757532Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260725_111117	2026-07-25T18:11:17.479229Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260725_110016	2026-07-25T18:00:17.878491Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_16189c3e0e13_20260724_140033	2026-07-24T21:00:33.947915Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_2d8c4ba8b58d_20260724_113332	2026-07-24T18:33:32.598079Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_6a206e30e232_20260724_112232	2026-07-24T18:22:32.176324Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_49a3abd8ad03_20260724_111131	2026-07-24T18:11:31.821817Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_2a95a4bb9ee1_20260724_110031	2026-07-24T18:00:31.835361Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_16189c3e0e13_20260723_140047	2026-07-23T21:00:47.823740Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_2d8c4ba8b58d_20260723_113345	2026-07-23T18:33:45.752262Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_6a206e30e232_20260723_112245	2026-07-23T18:22:45.132534Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_49a3abd8ad03_20260723_111144	2026-07-23T18:11:44.830285Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
cron_2a95a4bb9ee1_20260723_110045	2026-07-23T18:00:46.630896Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/gpt-5.5	—/gpt-5.5	—
20260719_093839_0fe24da2	2026-07-19T16:38:39.394786Z	INCLUDED	BILLING_STATUS_INCONSISTENT	openai-codex/gpt-5.5	openai-codex/gpt-5.5	—
cron_49a3abd8ad03_20260718_111147	2026-07-18T18:11:47.948709Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260718_110047	2026-07-18T18:00:47.805861Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260717_145257	2026-07-17T21:52:58.258177Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260717_144931	2026-07-17T21:49:31.887783Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260717_142510_8636c9	2026-07-17T21:25:11.304341Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_49a3abd8ad03_20260717_140300	2026-07-17T21:03:00.792053Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_16189c3e0e13_20260717_140032	2026-07-17T21:00:32.424103Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/deepseek-v4-flash	—/deepseek-v4-flash	—
cron_49a3abd8ad03_20260717_111130	2026-07-17T18:11:30.181460Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/deepseek-v4-flash	—/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260717_110029	2026-07-17T18:00:30.022015Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/deepseek-v4-flash	—/deepseek-v4-flash	—
cron_49a3abd8ad03_20260716_211225	2026-07-17T04:12:26.414708Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_2a95a4bb9ee1_20260716_210225	2026-07-17T04:02:26.097565Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
cron_ff9df506ae49_20260716_180018	2026-07-17T01:00:20.197227Z	ZERO_USAGE	ZERO_TOKEN_SESSION	—/deepseek-v4-flash	—/deepseek-v4-flash	—
cron_16189c3e0e13_20260716_140054	2026-07-16T21:00:56.729275Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260715_181753_e444d8de	2026-07-16T01:17:53.921362Z	INCLUDED	BILLING_STATUS_INCONSISTENT	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—
20260704_153659_571e8c13	2026-07-04T22:36:59.471873Z	UNPRICED	MISSING_RATE	kimi-for-coding/kimi-k2.6	kimi-for-coding/kimi-k2.6	—
20260704_151329_9f4b9366	2026-07-04T22:13:29.562769Z	UNPRICED	MISSING_RATE	deepseek/deepseek-v4-flash	deepseek/deepseek-v4-flash	—

Estimation and completeness

- **CALCULATED** means the session matched an operator-maintained, effective-dated rate. It is a calculated estimate of provider charge, not an invoice.
- **INCLUDED** means `cost_status=included` or `billing_mode=subscription_included`; its quota/subscription usage is reported separately and is not variable API spend.

- **UNPRICED** means no exact mapped identity, valid usage, or matching rate was available. It is never displayed as $0.00 and is excluded from calculated USD.
- **ESTIMATED** is not used in v1; no statistical dollar estimates are made.

Scope and limitations

Sessions are selected once by sessions.started_at in the inclusive requested calendar window. Each is a session aggregate; sessions starting before the window are excluded even if they end within it, and sessions starting inside it may include later updates. input_tokens, cache-read, cache-write, and output are separately priced when a rate exists; reasoning tokens are usage metadata only. Open sessions are provisional until closed. No messages were read and the Hermes SQLite store was accessed through a read-only URI with PRAGMA query_only=ON. Route URLs and user/chat/session content are not exposed.

test3_cost/gpt-5.6-sol.md

Hermes Cost Report

Metadata

Field	Value
Generated at	2026-08-17T03:23:07.582014Z
Database	/tmp/hermes-cost-snapshot.db
Database SHA-256 (unchanged before/after)	a65f02f715045f60718455c6884298b426431e718aee4c956c669cb1e8fb3ce7
Window timezone	UTC
Requested inclusive dates	2026-07-04 through 2026-08-17
Exact UTC lower bound (inclusive)	2026-07-04T00:00:00Z
Exact UTC upper bound (exclusive)	2026-08-18T00:00:00Z
Latest included started_at	2026-08-17T00:14:53.826926Z
Archived sessions	excluded
Rate file	/home/matthew/wiki/drafts/hermes-kit/taste/results/gpt-5.6-sol/costtest/rates.csv
Rate file SHA-256	2dd0d0956977906586f703c7cfba2938299cf75d4824bf2dbf07aa1a8a9a490c
Rate IDs used	deepseek-v4-flash-2026-08-01
Mappings file	/home/matthew/wiki/drafts/hermes-kit/taste/results/gpt-5.6-sol/costtest/mappings.csv
Mappings SHA-256	a8d5a679946470ac161d190e730b539abb74a8501aa247d4486e8aca4de5b86a

Assumptions

- No operator inputs were supplied with this implementation request. This run uses the full snapshot start-date span in UTC, the PRD's sole DeepSeek example rate as an operator-placeholder CSV, and an empty exact-mappings CSV.
- Effective-dated rates and mappings are matched against the session's UTC start date.
- Coverage denominator is every non-included, non-zero session; invalid-usage sessions remain eligible but unpriced.
- “Billable tokens” means input + cache read + cache write + output; reasoning is shown separately and not charged.

Headline totals

Metric	Value
All sessions	385
Calculated metered USD	$5.25
Calculated sessions	97
Eligible metered sessions	253
Pricing coverage	38.34% (sessions: priced / eligible metered)
Included/quota sessions	113
Included/quota billable tokens	153,955,361
Unpriced nonzero sessions	156
Unpriced billable tokens	695,290,933
Zero-usage sessions	19
Open/provisional sessions	159

Status legend

- **CALCULATED** — matched to the configured rate table; an estimate of provider charge, not an invoice.
- **INCLUDED** — subscription/quota usage; variable API spend is $0.00 and no API price is applied.
- **UNPRICED** — identity, usage, or rate is insufficient; excluded from USD totals and never rendered as $0.00.
- **ESTIMATED** — reserved for a future explicit heuristic; not enabled in v1.

By provider

By provider	Status	Rate ID	Sessions	Input	Cache read	Cache write	Output	Reasoning	USD
(missing)	UNPRICED	—	19	0	0	0	0	0	N/A
deepseek	CALCULATED	deepseek-v4-flash-2026-08-01	97	5,304,090	98,769,280	0	2,385,390	0	$5.25
deepseek	INCLUDED	—	1	3,417,191	58,789,120	0	273,011	48,505	$0.00

By provider	Status	Rate ID	Sessions	Input	Cache read	Cache write	Output	Reasoning	USD
deepseek	UNPRICED	—	28	9,558,989	141,851,968	0	930,311	122,748	N/A
kimi-coding	UNPRICED	—	63	5,711,381	35,708,360	0	567,317	0	N/A
kimi-for-coding	UNPRICED	—	1	1,575,786	9,702,592	0	153,855	3,586	N/A
openai-codex	INCLUDED	—	112	9,678,614	80,983,936	0	813,489	131,398	$0.00
xai	UNPRICED	—	11	798,622	6,296,207	0	177,992	89,185	N/A
xiaomi	UNPRICED	—	26	14,614,026	459,241,472	0	2,363,256	69,066	N/A
zai	UNPRICED	—	27	429,122	5,548,692	0	60,985	0	N/A

By model

By model	Status	Rate ID	Sessions	Input	Cache read	Cache write	Output	Reasoning	USD
deepseek-chat	UNPRICED	—	3	159,185	3,490,304	0	51,819	0	N/A
deepseek-v4-flash	CALCULATED	deepseek-v4-flash-2026-08-01	97	5,304,090	98,769,280	0	2,385,390	0	$5.25
deepseek-v4-flash	INCLUDED	—	1	3,417,191	58,789,120	0	273,011	48,505	$0.00
deepseek-v4-flash	UNPRICED	—	25	22,474,120	581,936,102	0	2,939,400	191,814	N/A
glm-4.5-flash	UNPRICED	—	7	184,200	2,902,884	0	34,952	0	N/A
glm-4.6	UNPRICED	—	1	21,222	0	0	49	0	N/A
gpt-4o	UNPRICED	—	2	40,615	4,864	0	50	0	N/A
gpt-5.5	INCLUDED	—	44	4,749,147	45,630,848	0	394,767	48,095	$0.00
gpt-5.5	UNPRICED	—	16	76,313	566,647	0	7,985	0	N/A
gpt-5.6-luna	UNPRICED	—	1	4	21,961	0	83	0	N/A
gpt-5.6-sol	INCLUDED	—	68	4,929,467	35,353,088	0	418,722	83,303	$0.00
gpt-5.6-sol	UNPRICED	—	1	3	21,962	0	72	0	N/A
gpt-5.6-terra	UNPRICED	—	1	3	21,962	0	85	0	N/A
grok-4.6	UNPRICED	—	11	798,622	6,296,207	0	177,992	89,185	N/A
kimi-for-coding	UNPRICED	—	9	22,498	288,246	0	9,541	0	N/A
kimi-k1.5	UNPRICED	—	4	269,347	2,617,472	0	40,324	0	N/A
kimi-k2.6	UNPRICED	—	68	7,287,167	45,410,952	0	721,172	3,586	N/A
mimo-v2.5	UNPRICED	—	2	75,030	1,102,656	0	45,083	0	N/A
mimo-v2.5-pro	UNPRICED	—	24	1,279,597	13,667,072	0	225,109	0	N/A

By platform

By platform	Status	Rate ID	Sessions	Input	Cache read	Cache write	Output	Reasoning	USD
cli	CALCULATED	deepseek-v4-flash-2026-08-01	47	1,936,577	40,071,040	0	1,100,953	0	$2.13
cli	INCLUDED	—	46	3,858,812	29,426,176	0	271,477	61,490	$0.00
cli	UNPRICED	—	56	1,967,021	17,920,938	0	387,036	89,185	N/A
cron	CALCULATED	deepseek-v4-flash-2026-08-01	37	3,031,356	54,444,672	0	939,835	0	$2.77
cron	INCLUDED	—	44	3,359,478	31,378,432	0	317,423	37,693	$0.00
cron	UNPRICED	—	110	7,341,173	58,886,401	0	922,472	0	N/A
email	INCLUDED	—	1	29,630	47,104	0	1,125	316	$0.00
email	UNPRICED	—	1	18,127	1,536	0	199	0	N/A
subagent	CALCULATED	deepseek-v4-flash-2026-08-01	13	336,157	4,253,568	0	344,602	0	$0.36
subagent	INCLUDED	—	19	1,002,250	5,783,040	0	144,669	20,832	$0.00
subagent	UNPRICED	—	5	390,773	3,728,768	0	98,699	0	N/A
telegram	INCLUDED	—	3	4,845,635	73,138,304	0	351,806	59,572	$0.00
telegram	UNPRICED	—	3	22,970,832	577,811,648	0	2,845,310	195,400	N/A

Included / quota sessions

Provider	Model	Platform	Sessions	Input	Cache read	Cache write	Output	Reasoning	Variable USD
deepseek	deepseek-v4-flash	telegram	1	3,417,191	58,789,120	0	273,011	48,505	$0.00 — INCLUDED / NOT PRICED AS API SPEND
openai-codex	gpt-5.5	cli	1	33,142	134,656	0	997	174	$0.00 — INCLUDED / NOT PRICED AS API SPEND
openai-codex	gpt-5.5	cron	40	3,257,931	31,099,904	0	313,850	36,538	$0.00 — INCLUDED / NOT PRICED AS API SPEND
openai-codex	gpt-5.5	email	1	29,630	47,104	0	1,125	316	$0.00 — INCLUDED / NOT PRICED AS API SPEND
openai-codex	gpt-5.5	telegram	2	1,428,444	14,349,184	0	78,795	11,067	$0.00 — INCLUDED / NOT PRICED AS API SPEND
openai-codex	gpt-5.6-sol	cli	45	3,825,670	29,291,520	0	270,480	61,316	$0.00 — INCLUDED / NOT PRICED AS API SPEND
openai-codex	gpt-5.6-sol	cron	4	101,547	278,528	0	3,573	1,155	$0.00 — INCLUDED / NOT PRICED AS API SPEND
openai-codex	gpt-5.6-sol	subagent	19	1,002,250	5,783,040	0	144,669	20,832	$0.00 — INCLUDED / NOT PRICED AS API SPEND

Data-quality appendix

Status	Issue(s)	Resolved provider	Resolved model	Sessions	Session IDs
INCLUDED_DEDUP_IDENTICAL_USAGE	BILLED_AT_$0	deepseek	deepseek-v4-flash	1	20260715_181753_e444d8de

Status	Issue(s)	Resolved provider	Resolved model	Sessions	Session IDs
NONSTENT			-v4-flash		
INCLUDED	BILLING_STATUS_INCONSISTENT	openai-codex	gpt-5.5	1	20260719_093839_0fe24da2
UNPRICED	MISSING_RATE	deepseek	deepseek-chat	3	cron_16189c3e0e13_20260802_140027 cron_04300d215d3f_20260802_170003 cron_16189c3e0e13_20260807_140037
UNPRICED	MISSING_RATE	deepseek	deepseek-v4-flash	18	20260704_151329_9f4b9366 cron_16189c3e0e13_20260716_140054 cron_2a95a4bb9ee1_20260716_210225 cron_49a3abd8ad03_20260716_211225 cron_49a3abd8ad03_20260717_140300 20260717_142510_8636c9 cron_2a95a4bb9ee1_20260717_144931 cron_49a3abd8ad03_20260717_145257 cron_2a95a4bb9ee1_20260718_110047 cron_49a3abd8ad03_20260718_111147 cron_2a95a4bb9ee1_20260725_110016 cron_49a3abd8ad03_20260725_111117 cron_6a206e30e232_20260725_112217 cron_2d8c4ba8b58d_20260725_113317 cron_2a95a4bb9ee1_20260726_110053 cron_49a3abd8ad03_20260726_111153 cron_6a206e30e232_20260726_112253 cron_2d8c4ba8b58d_20260726_113353
UNPRICED	MISSING_RATE	deepseek	gpt-4o	2	20260807_075532_ce43a0 20260807_095227_a9ff58
UNPRICED	MISSING_RATE	deepseek	kimi-k1.5	4	cron_2a95a4bb9ee1_20260726_145455 cron_49a3abd8ad03_20260726_145824 cron_6a206e30e232_20260726_150837 cron_2d8c4ba8b58d_20260726_150920
UNPRICED	MISSING_RATE	deepseek	mimo-v2.5-pro	1	20260804_162013_6a440c
UNPRICED	MISSING_RATE	kimi-cod	kimi-k2	63	cron_2a95a4bb9ee1_20260727_110049 cron_49a3abd8ad03_20260727_111149 cron_6a206e30e232_20260727_112249 cron_2d8c4ba8b58d_20260727_113349 cron_2a95a4bb9ee1_20260728_110007 cron_49a3abd8ad03_20260728_111107 cron_6a206e30e232_20260728_112427 cron_2d8c4ba8b58d_20260728_113307 20260729_080252_290c97df cron_2a95a4bb9ee1_20260729_110033 cron_49a3abd8ad03_20260729_111133 cron_6a206e30e232_20260729_112348 cron_2d8c4ba8b58d_20260729_113333 cron_2a95a4bb9ee1_20260730_110023 cron_49a3abd8ad03_20260730_111122 cron_6a206e30e232_20260730_112223 cron_2d8c4ba8b58d_20260730_113323 cron_2a95a4bb9ee1_20260731_110018 cron_49a3abd8ad03_20260731_111117 cron_6a206e30e232_20260731_112218 cron_2d8c4ba8b58d_20260731_113318 cron_2a95a4bb9ee1_20260801_110028 cron_49a3abd8ad03_20260801_111127 cron_6a206e30e232_20260801_112227 cron_2d8c4ba8b58d_20260801_113327 cron_2a95a4bb9ee1_20260802_074931 cron_2a95a4bb9ee1_20260802_110044 cron_49a3abd8ad03_20260802_111144 cron_6a206e30e232_20260802_11

Status	Issue(s)	Resolved provider	Resolved model	Sessions	Session IDs
		ing	.6		2338 cron_2d8c4ba8b58d_20260802_113345 cron_2a95a4bb9ee1_20260803_110048 cron_49a3abd8ad03_20260803_111148 cron_6a206e30e232_20260803_112249 cron_2d8c4ba8b58d_20260803_113349 cron_2a95a4bb9ee1_20260804_110006 cron_49a3abd8ad03_20260804_111106 cron_6a206e30e232_20260804_112206 cron_2d8c4ba8b58d_20260804_113306 cron_2a95a4bb9ee1_20260805_110026 cron_49a3abd8ad03_20260805_111126 cron_6a206e30e232_20260805_112226 cron_2d8c4ba8b58d_20260805_113326 cron_2a95a4bb9ee1_20260806_110027 cron_49a3abd8ad03_20260806_111127 cron_6a206e30e232_20260806_112227 cron_2d8c4ba8b58d_20260806_113327 cron_2a95a4bb9ee1_20260807_110035 cron_49a3abd8ad03_20260807_111135 cron_6a206e30e232_20260807_112628 cron_2d8c4ba8b58d_20260807_113335 cron_2a95a4bb9ee1_20260808_110006 cron_49a3abd8ad03_20260808_111106 cron_6a206e30e232_20260808_112207 cron_2d8c4ba8b58d_20260808_113307 cron_2a95a4bb9ee1_20260809_110028 cron_49a3abd8ad03_20260809_111600 cron_6a206e30e232_20260809_112336 cron_2d8c4ba8b58d_20260809_113329 20260813_193514_88b9f4 20260813_201349_82c4e4 20260813_202301_418c28 20260813_203555_1a94da 20260813_204545_27a848
UNPRICED	MISSING_RATE	kimi-for-coding	kimi-k2.6	1	20260704_153659_571e8c13
UNPRICED	MISSING_RATE	xai	grok-4.6	11	20260813_194227_c13167 20260813_201350_cde80a 20260813_202302_3ec142 20260813_203355_3aca69 20260813_204656_ea9ba5 20260813_211132_7f1814 20260814_072750_cc3223 20260814_073704_8914c6 20260814_073718_14cd97 20260814_075333_2dd59a 20260814_075426_cb0f94
UNPRICED	MISSING_RATE	xiaomi	deepseek-v4-flash	1	20260726_114449_31da2578
UNPRICED	MISSING_RATE	xiaomi	mimo-v2.5	2	20260813_211131_c0cb72 20260814_075425_159040
UNPRICED	MISSING_RATE	xiaomi	mimo-v2.5-pro	23	cron_ee85be6bc8a4_20260803_150051 cron_c44dd53f3e81_20260803_151151 20260803_162138_1ad489 20260804_080536_849c95 cron_ee85be6bc8a4_20260804_150012 cron_c44dd53f3e81_20260804_151112 20260805_081952_6b32cc cron_ee85be6bc8a4_20260805_150030 cron_c44dd53f3e81_20260805_151130 cron_ee85be6bc8a4_20260806_150030 cron_c44dd53f3e81_20260806_151130 20260806_193345_617f8d 20260806_193545_33b844 20260806_193746_3fca8f 20260806_213255_6ac5d6 20260806_213355_011d04 20260807_075503_d9d0e0 20260813_193252_41e100 20260813_201348_46d618 20260813_201934_d371ad 20260813_202204_a5429c 20260813_203216_4cb9e7 20260813_204137_a273fe
UNPRICED	MISSING_RATE	zai	deepseek-v4-flash	2	cron_16189c3e0e13_20260811_140054 cron_ee85be6bc8a4_20260813_150044
UN	MISS		gl	7	20260812_221456_43ec99 20260813_070805_8d1966 20260813_071642_4aa3a8 20260813_072654_1eac29 20260813_144543_7a3868 20260813_145036_b1a5ce 20260814_194808_455d7e

Status	Issue(s)	Resolved provider	Resolved model	Sessions	Session IDs
PRICED	ING_RATE	zai	m-4.5-flash		
UNPRICED	MISSING_RATE	zai	glm-4.6	1	20260813_193413_1b89ff
UNPRICED	MISSING_RATE	zai	gpt-5.5	5	cron_04300d215d3f_20260816_170003 20260816_170642_cfa9e8 20260816_170720_f0efd4 20260816_170852_b2083e 20260816_171452_8c26e2
UNPRICED	MISSING_RATE	zai	gpt-5.6-luna	1	20260816_170735_9008d3
UNPRICED	MISSING_RATE	zai	gpt-5.6-sol	1	20260816_170809_4f247b
UNPRICED	MISSING_RATE	zai	gpt-5.6-terra	1	20260816_170750_3f2838
UNPRICED	MISSING_RATE	zai	kimi-for-coding	9	20260813_211133_6767aa 20260814_075427_f8020f 20260814_101016_b1011e 20260814_101056_edd60e 20260814_101641_fd5e8f 20260814_101810_5508b2 20260814_101943_f7f2ce 20260814_102609_df2355 20260814_103016_b9e45f
UNPRICED	ZERO_USAGE	(missing)	deepseek-v4-f	4	cron_ff9df506ae49_20260716_180018 cron_2a95a4bb9ee1_20260717_110029 cron_49a3abd8ad03_20260717_111130 cron_16189c3e0e13_20260717_140032

Status	Issue(s)	Resolved provider	Resolved model	Sessions	Session IDs
UNPRICED	ZERO_USAGE (missing)	(mixed)	gpt-5.5	11	cron_2a95a4bb9ee1_20260723_110045 cron_49a3abd8ad03_20260723_111144 cron_6a206e30e232_20260723_112245 cron_2d8c4ba8b58d_20260723_113345 cron_16189c3e0e13_20260723_140047 cron_2a95a4bb9ee1_20260724_110031 cron_49a3abd8ad03_20260724_111131 cron_6a206e30e232_20260724_112232 cron_2d8c4ba8b58d_20260724_113332 cron_16189c3e0e13_20260724_140033 cron_16189c3e0e13_20260727_140017
UNPRICED	ZERO_USAGE (missing)	(mixed)	kimi-k2.6	4	cron_2a95a4bb9ee1_20260810_110013 cron_49a3abd8ad03_20260810_111113 cron_6a206e30e232_20260810_112214 cron_2d8c4ba8b58d_20260810_113314

Scope and limitations

Sessions are included once by `sessions.id` when `started_at` falls within the selected local calendar-date window. A session is included even if it ended later. Sessions with no `ended_at` are provisional and should be refreshed in a trailing-window run. Archived sessions are excluded unless requested. Costs use session aggregate buckets only; `messages.token_count` is never used. Hermes placeholder cost fields are ignored. Calculated USD comes only from the hashed external rate table and is not invoice reconciliation. No subscription fee, quota balance, overage, tax, currency conversion, hosting cost, or statistical estimate is inferred. Database access used SQLite `mode=ro` plus `PRAGMA query_only=ON`; the application performs only SELECT/read PRAGMAs.

test3_cost/ground-truth-comparison.md

Ground Truth vs Reality — 30-Day Cost Comparison, CONVERGED REFERENCE: SOL + DEEPSEEK (2026-08-18)

DB basis: `hermes-cost-fleet-20260817.db` (full fleet, 1,024 sessions; 800 in the 30-day window). All 10 apps rerun with rate tables **regenerated from the shared source of truth** (`~/.config/hermes-cost/rates.yaml`) — every app now prices every vendor that has a published rate. Window lock: 2026-07-18 → 2026-08-17.

The converged answer (sol + deepseek) — the correct answer, calculated

sol ($60.49) and deepseek-v4-flash ($60.49) converged on the same answer — identical per vendor (within a cent). Their values ARE the correct answer; other apps count as correct within 40% of them (the ✓/✗ below).

Vendor	Correct answer (sol & deepseek)
deepseek	$7.03
xiaomi (mimo)	$5.13
zai	$5.88
xai (grok)	$27.76
openai-codex (gpt)	$0.00
kimi	$14.68
Total metered	**$60.49**

free credits, sliding date windows, and deepseek's price increase, what actually left the wallet differed from billing. The calculated answer is the reference, not the bills.

Reality — what each app computed (fleet DB, 30-day lock, config-fixed rerun)

App	deepseek	xiaomi	zai	xai	openai-codex	kimi	Total metered
deepseek-v4-flash	$7.04 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49
glm-4.5	$4.44 ✓	$0.49 ✗	$1.29 ✗	— ✗ (no xai sessions analyzed)	$0.00 ✓	$13.89 ✓	$20.29
mimo-v2.5-pro	$7.03 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49
grok-4.6	$7.26 ✓	$0.85 ✗	$5.90 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$56.44
gpt-5.5	$2.32 ✗	$0.81 ✗	$0.01 ✗	$27.76 ✓	$0.00 ✓	$14.68 ✓	$45.59
gpt-5.6-luna	$26.19 ✗	$5.12 ✓	$7.42 ✓	$27.76 ✓	$23.02 ✗ (est.)	$22.75 ✗	$112.26
gpt-5.6-sol	$7.03 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49
gpt-5.6-terra	$7.04 ✓	$5.13 ✓	$5.88 ✓	$27.76 ✓	$0.00 ✓	$14.68 ✓	$60.49

App	deepseek	xiaomi	zai	xai	openai-codex	kimi	Total metered
kimi-k2.6	$11.18 ✗	$9.55 ✗	$5.90 ✓	$27.76 ✓	$0.00 ✓	$14.67 ✓	$69.06
glm-4.5-flash	$47.89 ✗	$10.78 ✗	$1.70 ✗	$5.22 ✗	$11.54 ✗ (quota as cash)	$13.53 ✓	$90.67

Bands (±40% of the converged values): deepseek [4.22, 9.84] · xiaomi [3.08, 7.18] · zai [3.53, 8.23] · xai [16.66, 38.86] · openai-codex [0, 0] · kimi [8.81, 20.55].

Verdict tallies (vendors in the converged band per app)

- deepseek app: 6/6 — co-reference; every vendor in band
- glm-4.5: 3/6 (deepseek, openai-codex, kimi) — xai never priced (analyzed only 429/800 sessions); xiaomi/zai read low
- mimo: 6/6
- grok: 5/6 (deepseek, zai, xai, codex, kimi) — xiaomi $0.85 reads low (session filter)
- gpt-5.5: 3/6 (xai, codex, kimi) — under-reads deepseek/xiaomi/zai; units bug dead ($293,534.88 → $45.59)
- luna: 3/6 (xiaomi, zai, xai) — still estimates where no published rate exists
- sol: 6/6 — co-reference; full rates CSV (was 4 vendors unpriced)
- terra: 6/6 — rates CSV added (was 0/405 priced)
- kimi: 4/6 (zai, xai, codex, kimi) — deepseek $11.18 and xiaomi $9.55 sit above the band
- glm-4.5-flash: 1/6 (kimi) — fallback-rate path inflates deepseek; xiaomi/zai high; codex priced as cash

Why the apps and the bills differ

2026-07-31 price increase, and the DB attributes the big telegram session (14.5M in / 2.5M out / 548M cache) to billing_provider xiaomi; re-attributed, deepseek ≈ $11.32. open sessions count tokens-so-far, and free-credit balances change what's actually paid.

- **xiaomi**: converged $5.13 includes the mis-attributed telegram session;
- **zai**: converged $5.88 — inside the portal's < $10.64 upper bound.
- **52 sessions stay unpriced in every app** — models with no published rate in

the shared table (deepseek-v4-pro, kimi-k3, glm-4.5, deepseek/gpt-5.5, kimi-k1.5). Even the best accounting can't price a rate that doesn't exist.

- **The DB is a floor, not the bill** — open sessions count tokens-so-far, some

usage isn't in the DB at all. No app can beat its input. But the apps that get accounting honest stay in the universe where the guessers don't.

test3_cost/grok-4.6-rerun.log

```
[rerun] grok-4.6: implementing existing PRD (30798 bytes)
[rerun] TASK 3 start: grok-4.6 (2026-08-17T14:12:42Z)
Query: Implement the Hermes cost tracking app described in the PRD at
/home/matthew/wiki/drafts/hermes-kit/taste/results/grok-4.6/costtest/cost-tracke
r-prd.md.

Requirements:
- Read-only access to the SQLite session store at /tmp/hermes-cost-snapshot.db —
the same snapshot used for the PRD. Never modify it.
- Follow the PRD's requirements and data model.
- Python standard library only (sqlite3 and friends). No external packages.
- It must actually run: execute it against the snapshot and produce the report.
- Save the app as
/home/matthew/wiki/drafts/hermes-kit/taste/results/grok-4.6/costtest/cost_tracke
r.py and the report as
/home/matthew/wiki/drafts/hermes-kit/taste/results/grok-4.6/costtest/cost_report
.md.
- If the PRD is missing anything you need, state your assumption explicitly and
proceed.

When done, reply with the report text and the file paths.
Initializing agent...
────────────────────────────────────────


╭─ ⚕ Hermes ──────────────────────────────────────────────────────────────────╮
    I'll start by reading the PRD and inspecting the snapshot database so the tracker matches the actual data model.
╰──────────────────────────────────────────────────────────────────────────────╯

  ┊    read      cost-tracker-prd.md  0.1s
  ┊    $         ls -la /tmp/hermes-cost-snapshot.db /home/matthew/wiki/drafts/hermes-kit/taste/results/grok-4.6/costtest/ + 1 command  0.0s

╭─ ⚕ Hermes ──────────────────────────────────────────────────────────────────╮
    There's already a tracker and rates file in that folder. I'll read the rest of the PRD and inspect those artifacts plus the snapshot.
╰──────────────────────────────────────────────────────────────────────────────╯

  ┊    read      cost-tracker-prd.md L500  0.1s
  ┊    read      cost_tracker.py  0.1s
  ┊    read      rates.yaml  0.1s
  ┊    read      task3-rerun.log  0.1s

╭─ ⚕ Hermes ──────────────────────────────────────────────────────────────────╮
    The existing tracker depends on PyYAML, which the task forbids. I'll finish reading the implementation and inspect the snapshot so I can rewrite it with stdlib only.
╰──────────────────────────────────────────────────────────────────────────────╯

  ┊    read      cost_tracker.py L495  0.0s
  ┊    $         python3 - <<'PY' import sqlite3 from collections import Counter, defaultdict from datetime import datetime, timezone conn = sqlite3.connect("file:/tmp/hermes-cost-snapshot.db?
mode=ro", uri=True) conn.row_factory = sqlite3.Row c = conn.cursor() print("schema_version:", list(c.execute("SELECT * FROM schema_version"))) print("state_meta:", list(c.execute("SELECT * FROM
state_meta"))) n = c.execute("SELECT COUNT(*) FROM sessions").fetchone()[0] print("sessions:", n) print("archived:", dict(c.execute("SELECT archived, COUNT(*) FROM sessions GROUP BY archived")))
print("ended_at null:", c.execute("SELECT COUNT(*) FROM sessions WHERE ended_at IS NULL").fetchone()[0]) print("pricing_version null:", c.execute("SELECT COUNT(*) FROM sessions WHERE
pricing_version IS NULL").fetchone()[0]) print("actual_cost_usd null:", c.execute("SELECT COUNT(*) FROM sessions WHERE actual_cost_usd IS NULL").fetchone()[0]) print("estimated_cost_usd=0:",
c.execute("SELECT COUNT(*) FROM sessions WHERE estimated_cost_usd = 0 OR estimated_cost_usd IS NULL").fetchone()[0]) print("nonzero estimated:", list(c.execute("SELECT id, model,
billing_provider, estimated_cost_usd FROM sessions WHERE estimated_cost_usd IS NOT NULL AND estimated_cost_usd != 0"))) print("\n=== sources ===") for r in c.execute("SELECT source, COUNT(*)
FROM sessions GROUP BY source ORDER BY 2 DESC"): print(r) print("\n=== models ===") for r in c.execute("SELECT model, COUNT(*) FROM sessions GROUP BY model ORDER BY 2 DESC"): print(r)
print("\n=== billing_provider ===") for r in c.execute("SELECT billing_provider, COUNT(*) FROM sessions GROUP BY billing_provider ORDER BY 2 DESC"): print(r) print("\n=== billing_mode ===") for
r in c.execute("SELECT billing_mode, COUNT(*) FROM sessions GROUP BY billing_mode ORDER BY 2 DESC"): print(r) print("\n=== cost_status ===") for r in c.execute("SELECT cost_status, COUNT(*) FROM
sessions GROUP BY cost_status ORDER BY 2 DESC"): print(r) print("\n=== provider x model ===") for r in c.execute("SELECT billing_provider, model, COUNT(*) FROM sessions GROUP BY 1,2 ORDER BY 3
DESC"): print(r) print("\n=== provider x cost_status x billing_mode ===") for r in c.execute("SELECT billing_provider, cost_status, billing_mode, COUNT(*) FROM sessions GROUP BY 1,2,3 ORDER BY 4
DESC"): print(r) print("\n=== token totals ===") print(list(c.execute(""" SELECT SUM(COALESCE(input_tokens,0)), SUM(COALESCE(output_tokens,0)), SUM(COALESCE(cache_read_tokens,0)),
SUM(COALESCE(cache_write_tokens,0)), SUM(COALESCE(reasoning_tokens,0)), SUM(COALESCE(api_call_count,0)), SUM(COALESCE(message_count,0)), MIN(started_at), MAX(started_at) FROM sessions """)))
print("\n=== zero token / missing provider ===") print("zero tokens:", c.execute(""" SELECT COUNT(*) FROM sessions WHERE COALESCE(input_tokens,0)+COALESCE(output_tokens,
0)+COALESCE(cache_read_tokens,0)+COALESCE(cache_write_tokens,0)=0 """).fetchone()[0]) print("missing provider:", c.execute("SELECT COUNT(*) FROM sessions WHERE billing_provider IS
NULL").fetchone()[0]) print("missing model:", c.execute("SELECT COUNT(*) FROM sessions WHERE model IS NULL").fetchone()[0]) print("missing provider AND tokens:", list(c.execute(""" SELECT
COUNT(*), SUM(COALESCE(input_tokens,0)+COALESCE(output_tokens,0)+COALESCE(cache_read_tokens,0)+COALESCE(cache_write_tokens,0)) FROM sessions WHERE billing_provider IS NULL """))) print("\n===
missing provider details ===") for r in c.execute(""" SELECT source, model, billing_provider, cost_status, billing_mode, end_reason, COALESCE(input_tokens,0), COALESCE(output_tokens,0),
COALESCE(api_call_count,0), COALESCE(message_count,0), COUNT(*) FROM sessions WHERE billing_provider IS NULL GROUP BY 1,2,3,4,5,6,7,8,9,10 """): print(tuple(r)) print("\n=== date range local-ish
===") mn, mx = c.execute("SELECT MIN(started_at), MAX(started_at) FROM sessions").fetchone() print("min", mn, datetime.fromtimestamp(mn, tz=timezone.utc), datetime.fromtimestamp(mn))
print("max", mx, datetime.fromtimestamp(mx, tz=timezone.utc), datetime.fromtimestamp(mx)) print("\n=== contradictions ===") print("deepseek subscription_included:", list(c.execute(""" SELECT id,
model, billing_provider, billing_mode, cost_status, input_tokens, cache_read_tokens FROM sessions WHERE billing_provider='deepseek' AND billing_mode='subscription_included' """))) print("codex
unknown:", list(c.execute(""" SELECT id, model, billing_provider, billing_mode, cost_status, input_tokens FROM sessions WHERE billing_provider='openai-codex' AND (cost_status IS NULL OR
cost_status!='included') """))) print("\n=== cache write ===") print(c.execute("SELECT SUM(cache_write_tokens), MAX(cache_write_tokens) FROM sessions").fetchone()) print("\n=== columns with null
tokens ===") for col in ["input_tokens","output_tokens","cache_read_tokens","cache_write_tokens","reasoning_tokens","api_call_count"]: print(col, "null", c.execute(f"SELECT COUNT(*) FROM
sessions WHERE {col} IS NULL").fetchone()[0]) PY  0.1s

╭─ ⚕ Hermes ──────────────────────────────────────────────────────────────────╮
    The snapshot grew past the PRD's 301 sessions. I'll pull exact counts and provider/model mix so the report covers the real file.
╰──────────────────────────────────────────────────────────────────────────────╯
```

```
  ┊    $         python3 - <<'PY' import sqlite3 from datetime import datetime, timezone from zoneinfo import ZoneInfo from collections import Counter conn = sqlite3.connect("file:/tmp/hermes-
cost-snapshot.db?mode=ro", uri=True) c = conn.cursor() print("schema_version:", c.execute("SELECT * FROM schema_version").fetchall()) print("state_meta:", c.execute("SELECT * FROM
state_meta").fetchall()) print("sessions:", c.execute("SELECT COUNT(*) FROM sessions").fetchone()[0]) print("archived:", c.execute("SELECT archived, COUNT(*) FROM sessions GROUP BY
archived").fetchall()) print("ended_at null:", c.execute("SELECT COUNT(*) FROM sessions WHERE ended_at IS NULL").fetchone()[0]) print("\n=== sources ===") for r in c.execute("SELECT source,
COUNT(*) FROM sessions GROUP BY source ORDER BY 2 DESC"): print(r) print("\n=== models ===") for r in c.execute("SELECT model, COUNT(*) FROM sessions GROUP BY model ORDER BY 2 DESC"): print(r)
print("\n=== billing_provider ===") for r in c.execute("SELECT billing_provider, COUNT(*) FROM sessions GROUP BY billing_provider ORDER BY 2 DESC"): print(r) print("\n=== billing_mode ===") for
r in c.execute("SELECT billing_mode, COUNT(*) FROM sessions GROUP BY billing_mode ORDER BY 2 DESC"): print(r) print("\n=== cost_status ===") for r in c.execute("SELECT cost_status, COUNT(*) FROM
sessions GROUP BY cost_status ORDER BY 2 DESC"): print(r) print("\n=== provider x model ===") for r in c.execute("SELECT billing_provider, model, COUNT(*) FROM sessions GROUP BY 1,2 ORDER BY 3
DESC"): print(r) print("\n=== provider x cost_status x billing_mode ===") for r in c.execute("SELECT billing_provider, cost_status, billing_mode, COUNT(*) FROM sessions GROUP BY 1,2,3 ORDER BY 4
DESC"): print(r) print("\n=== token totals ===") print(c.execute(""" SELECT SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens), SUM(cache_write_tokens), SUM(reasoning_tokens),
SUM(api_call_count), SUM(message_count), MIN(started_at), MAX(started_at) FROM sessions """).fetchone()) print("nonzero estimated:", c.execute("SELECT id, model, billing_provider,
estimated_cost_usd FROM sessions WHERE estimated_cost_usd IS NOT NULL AND estimated_cost_usd != 0").fetchall()) print("deepseek subscription_included:", c.execute(""" SELECT id, model,
billing_provider, billing_mode, cost_status, input_tokens, cache_read_tokens FROM sessions WHERE billing_provider='deepseek' AND billing_mode='subscription_included' """).fetchall())
print("codex unknown:", c.execute(""" SELECT id, model, billing_provider, billing_mode, cost_status, input_tokens FROM sessions WHERE billing_provider='openai-codex' AND (cost_status IS NULL OR
cost_status!='included') """).fetchall()) print("cache write:", c.execute("SELECT SUM(cache_write_tokens), MAX(cache_write_tokens) FROM sessions").fetchone()) # window 2026-07-04 to 2026-08-13
America/Los_Angeles tz = ZoneInfo("America/Los_Angeles") from_ts = datetime(2026,7,4,0,0,0,tzinfo=tz).timestamp() to_excl = datetime(2026,8,14,0,0,0,tzinfo=tz).timestamp() print("\n=== window
2026-07-04 to 2026-08-13 PT ===") print("count:", c.execute("SELECT COUNT(*) FROM sessions WHERE started_at >= ? AND started_at < ?", (from_ts, to_excl)).fetchone()[0]) print("zero token in
window:", c.execute(""" SELECT COUNT(*) FROM sessions WHERE started_at >= ? AND started_at < ? AND input_tokens+output_tokens+cache_read_tokens+cache_write_tokens=0 """, (from_ts,
to_excl)).fetchone()[0]) print("open in window:", c.execute("SELECT COUNT(*) FROM sessions WHERE started_at >= ? AND started_at < ? AND ended_at IS NULL", (from_ts, to_excl)).fetchone()[0]) #
full range print("\n=== dates of all sessions (PT date) ===") days = Counter() for (ts,) in c.execute("SELECT started_at FROM sessions"): days[datetime.fromtimestamp(ts,
tz=tz).date().isoformat()] += 1 for d, n in sorted(days.items()): print(d, n) print("\n=== full token totals by provider ===") for r in c.execute(""" SELECT billing_provider, COUNT(*),
SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens), SUM(cache_write_tokens), SUM(reasoning_tokens), SUM(api_call_count) FROM sessions GROUP BY billing_provider ORDER BY 2 DESC """):
print(r) print("\n=== models not in starter table? ===") known = { 'gpt-5.5','gpt-5.6-sol','deepseek-v4-flash','deepseek-chat','deepseek-v4-pro','deepseek-reasoner', 'kimi-k2.6','kimi-
k1.5','mimo-v2.5-pro','glm-4.5-flash','gpt-4o','gpt-4o-mini' } for r in c.execute("SELECT DISTINCT model FROM sessions"): print(r[0], 'KNOWN' if r[0] in known else 'UNKNOWN') print("\n===
providers not in map ===") known_p = {'openai-codex','nous','deepseek','kimi-coding','xiaomi','zai','openai'} for r in c.execute("SELECT DISTINCT billing_provider FROM sessions"): print(r[0],
'KNOWN' if r[0] in known_p else 'UNKNOWN') # acceptance window token sums print("\n=== window token sums ===") print(c.execute(""" SELECT COUNT(*), SUM(input_tokens), SUM(output_tokens),
SUM(cache_read_tokens), SUM(cache_write_tokens), SUM(reasoning_tokens) FROM sessions WHERE started_at >= ? AND started_at < ? """, (from_ts, to_excl)).fetchone()) # UTC date window vs PT
from_utc = datetime(2026,7,4,0,0,0,tzinfo=timezone.utc).timestamp() to_utc = datetime(2026,8,14,0,0,0,tzinfo=timezone.utc).timestamp() print("UTC window count:", c.execute("SELECT COUNT(*) FROM
sessions WHERE started_at >= ? AND started_at < ?", (from_utc, to_utc)).fetchone()[0]) # sessions after Aug 13 PT print("after Aug 13 PT:", c.execute("SELECT COUNT(*) FROM sessions WHERE
started_at >= ?", (to_excl,)).fetchone()[0]) # sample session ids for zero token print("\nzero token session sample:") for r in c.execute(""" SELECT id, source, model, billing_provider,
started_at, ended_at, end_reason, message_count, api_call_count FROM sessions WHERE input_tokens+output_tokens+cache_read_tokens+cache_write_tokens=0 LIMIT 5 """): print(r) PY  0.1s

┌─ ⚕ Hermes ──────────────────────────────────────────────────────────────────┐
    I'll check sibling implementations for report shape, then inspect a few more snapshot fields before writing the stdlib tracker.
└─────────────────────────────────────────────────────────────────────────────┘

  ┊    find      cost_tracker.py  0.0s
  ┊    find      cost_report.md  0.0s
  ┊    $         python3 - <<'PY' import sqlite3 from datetime import datetime from zoneinfo import ZoneInfo conn = sqlite3.connect("file:/tmp/hermes-cost-snapshot.db?mode=ro", uri=True) c =
conn.cursor() # mismatched identity details print("=== zai models ===") for r in c.execute("SELECT model, COUNT(*), SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens),
SUM(api_call_count) FROM sessions WHERE billing_provider='zai' GROUP BY model"): print(r) print("\n=== xai ===") for r in c.execute("SELECT model, billing_mode, cost_status, COUNT(*),
SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens) FROM sessions WHERE billing_provider='xai' GROUP BY 1,2,3"): print(r) print("\n=== kimi-for-coding provider ===") for r in
c.execute("SELECT * FROM sessions WHERE billing_provider='kimi-for-coding'"): cols = [d[0] for d in c.description] row = dict(zip(cols, r)) keep =
['id','source','model','billing_provider','billing_mode','cost_status','input_tokens','output_tokens','cache_read_tokens','api_call_count','started_at'] print({k: row[k] for k in keep})
print("\n=== deepseek mismatched models ===") for r in c.execute("SELECT model, COUNT(*), SUM(input_tokens), SUM(cache_read_tokens) FROM sessions WHERE billing_provider='deepseek' AND model NOT
LIKE 'deepseek%' GROUP BY model"): print(r) print("\n=== openai-codex models ===") for r in c.execute("SELECT model, cost_status, COUNT(*) FROM sessions WHERE billing_provider='openai-codex'
GROUP BY 1,2"): print(r) print("\n=== snapshot mtime ===") import os, stat st = os.stat('/tmp/hermes-cost-snapshot.db') print('mode', oct(st.st_mode), 'mtime', st.st_mtime, 'size', st.st_size)
print("\n=== first/last session titles sample ===") tz = ZoneInfo('America/Los_Angeles') for r in c.execute("SELECT id, title, source, model, billing_provider, started_at FROM sessions ORDER BY
started_at LIMIT 3"): print(r[0], r[2], r[3], r[4], datetime.fromtimestamp(r[5], tz), r[1]) print('...') for r in c.execute("SELECT id, title, source, model, billing_provider, started_at FROM
sessions ORDER BY started_at DESC LIMIT 3"): print(r[0], r[2], r[3], r[4], datetime.fromtimestamp(r[5], tz), r[1]) PY  0.1s
  ┊    read      cost_report.md  0.1s
  ┊    read      cost_tracker.py  0.1s

┌─ ⚕ Hermes ──────────────────────────────────────────────────────────────────┐
    I have the data model and PRD rules. Next I'll replace the PyYAML tracker with a stdlib implementation and generate the report from the snapshot.I have the snapshot shape and the PRD rules.
Next I'll replace the PyYAML tracker with a stdlib implementation and generate the report from the snapshot.I'll precompute expected cash from the snapshot so the report totals can be checked
after the rewrite.
└─────────────────────────────────────────────────────────────────────────────┘

  ┊    $         python3 - <<'PY' import sqlite3 from decimal import Decimal conn = sqlite3.connect("file:/tmp/hermes-cost-snapshot.db?mode=ro", uri=True) c = conn.cursor() def cost(inp, out,
cr, ir, orr, crr): return (Decimal(inp)*Decimal(str(ir)) + Decimal(out)*Decimal(str(orr)) + Decimal(cr)*Decimal(str(crr))) / Decimal(1_000_000) # workhorse priced combos combos = [ ('deepseek',
'deepseek-v4-flash', 0.14, 0.28, 0.0028, 'quoted'), ('deepseek', 'deepseek-chat', 0.14, 0.28, 0.0028, 'est'), ('kimi-coding', 'kimi-k2.6', 0.95, 4.00, 0.16, 'est'), ('xiaomi', 'mimo-v2.5-pro',
0.435, 0.87, 0.0036, 'quoted'), ('zai', 'glm-4.5-flash', 0.50, 1.00, 0.005, 'est'), ] print("=== priced combos ===") quoted = Decimal(0) est = Decimal(0) for prov, model, ir, orr, crr, kind in
combos: n, inp, out, cr, cw, api = c.execute(""" SELECT COUNT(*), SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens), SUM(cache_write_tokens), SUM(api_call_count) FROM sessions WHERE
billing_provider=? AND model=? AND input_tokens+output_tokens+cache_read_tokens+cache_write_tokens>0 """, (prov, model)).fetchone() usd = cost(inp or 0, out or 0, cr or 0, ir, orr, crr)
print(f"{kind:7} {prov:15} {model:22} n={n:3} in={inp} out={out} cr={cr} usd={usd:.4f}") if kind=='quoted': quoted += usd else: est += usd # kimi-for-coding alias -> kimi-coding / kimi-k2.6 EST
n, inp, out, cr, api = c.execute(""" SELECT COUNT(*), SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens), SUM(api_call_count) FROM sessions WHERE billing_provider='kimi-for-coding'
AND model='kimi-k2.6' """).fetchone() usd = cost(inp or 0, out or 0, cr or 0, 0.95, 4.00, 0.16) print(f"est kimi-for-coding→kimi-coding kimi-k2.6 n={n} in={inp} out={out} cr={cr} usd={usd:.4f}")
est += usd print(f"\nQUOTED {quoted:.4f}") print(f"EST {est:.4f}") print(f"CASH {(quoted+est):.4f}") print("\n=== unpriced nonzero (provider, model) ===") priced = {('deepseek','deepseek-v4-
flash'),('deepseek','deepseek-chat'),('kimi-coding','kimi-k2.6'), ('xiaomi','mimo-v2.5-pro'),('zai','glm-4.5-flash'),('kimi-for-coding','kimi-k2.6'), ('openai-codex','gpt-5.5'),('openai-
codex','gpt-5.6-sol')} for r in c.execute(""" SELECT billing_provider, model, COUNT(*), SUM(input_tokens+output_tokens+cache_read_tokens+cache_write_tokens) FROM sessions WHERE
input_tokens+output_tokens+cache_read_tokens+cache_write_tokens>0 GROUP BY 1,2 ORDER BY 3 DESC """): key = (r[0], r[1]) # alias if key == ('kimi-for-coding','kimi-k2.6'): status='EST via alias'
elif key in priced: status='PRICED/QUOTA' else: status='UNPRICED' print(f"{status:16} {r}") print("\n=== openai-codex tokens ===") print(c.execute(""" SELECT COUNT(*),
SUM(input_tokens+output_tokens+cache_read_tokens+cache_write_tokens), SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens) FROM sessions WHERE billing_provider='openai-codex'
""").fetchone()) print("\n=== full snapshot token sum ===") print(c.execute("SELECT SUM(input_tokens), SUM(output_tokens), SUM(cache_read_tokens), SUM(cache_write_tokens), SUM(reasoning_tokens)
FROM sessions").fetchone()) print("session count", c.execute("SELECT COUNT(*) FROM sessions").fetchone()[0]) print("zero", c.execute("SELECT COUNT(*) FROM sessions WHERE
input_tokens+output_tokens+cache_read_tokens+cache_write_tokens=0").fetchone()[0]) print("open", c.execute("SELECT COUNT(*) FROM sessions WHERE ended_at IS NULL").fetchone()[0]) PY  0.2s

┌─ ⚕ Hermes ──────────────────────────────────────────────────────────────────┐
```

```
No response from provider for 150s (non-streaming, model: grok-4.6). Aborting call.
 API call failed (attempt 1/3): ReadError
   Provider: xai  Model: grok-4.6
   Endpoint: https://api.x.ai/v1
   Error: [Errno 32] Broken pipe
    Elapsed: 150.17s  Context: 22 msgs, ~52,877 tokens
Retrying in 2.3s (attempt 1/3)...
No response from provider for 150s (non-streaming, model: grok-4.6). Aborting call.
 API call failed (attempt 2/3): ReadError
   Provider: xai  Model: grok-4.6
   Endpoint: https://api.x.ai/v1
   Error: [Errno 32] Broken pipe
    Elapsed: 302.77s  Context: 22 msgs, ~52,877 tokens
Retrying in 5.9s (attempt 2/3)...
No response from provider for 150s (non-streaming, model: grok-4.6). Aborting call.
 API call failed (attempt 1/3): ReadError
   Provider: xai  Model: grok-4.6
   Endpoint: https://api.x.ai/v1
   Error: [Errno 32] Broken pipe
    Elapsed: 150.17s  Context: 24 msgs, ~56,411 tokens
Retrying in 2.6s (attempt 1/3)...
No response from provider for 150s (non-streaming, model: grok-4.6). Aborting call.
 API call failed (attempt 2/3): ReadError
   Provider: xai  Model: grok-4.6
   Endpoint: https://api.x.ai/v1
   Error: [Errno 32] Broken pipe
    Elapsed: 303.14s  Context: 24 msgs, ~56,411 tokens
Retrying in 4.6s (attempt 2/3)...
No response from provider for 150s (non-streaming, model: grok-4.6). Aborting call.
 API call failed (attempt 3/3): ReadError
   Provider: xai  Model: grok-4.6
   Endpoint: https://api.x.ai/v1
   Error: [Errno 32] Broken pipe
    Elapsed: 457.94s  Context: 24 msgs, ~56,411 tokens
API failed after 3 retries — [Errno 32] Broken pipe
   Final error: [Errno 32] Broken pipe
 I have the data model and expected totals. Next I'll replace the PyYAML tracker with a stdlib implementation that follows the PRD, then run it on the snapshot.I have the snapshot shape and
the PRD rules. Next I'll replace the PyYAML tracker with a stdlib implementation and generate the report from the snapshot.I have the data model and expected totals. Next I'll replace the PyYAML
tracker with a stdlib implementation and generate the report from the snapshot.
╰──────────────────────────────────────────────────────────────────────────────╯
 ─  ⚕ Hermes  ──────────────────────────────────────────────────────────────────

     API call failed after 3 retries: [Errno 32] Broken pipe

 ──────────────────────────────────────────────────────────────────────────────


Resume this session with:
  hermes --resume 20260817_071243_ddd397 -p felix

Session:        20260817_071243_ddd397
Duration:       17m 0s
Messages:       23 (1 user, 22 tool calls)
TASK3_RERUN EXIT=0 wall=1023s
[rerun] app present: /home/matthew/wiki/drafts/hermes-kit/taste/results/grok-4.6/costtest/cost_tracker.py
[rerun] bare invocation needs subcommand — retrying with 'report --db SNAP'
usage: cost_tracker.py [-h] [--from FROM_DATE] [--to TO_DATE] [--last LAST]
                       [--db DB_PATH] [--include-archived] [--out OUTPUT_DIR]
                       [--format {both,md,csv}] [--no-console]
                       [--rates RATES_FILE] [--timezone TIMEZONE]
cost_tracker.py: error: unrecognized arguments: report
RERUN_DONE grok-4.6
```

test5_vision/mimo.md

Visual critique of diagram-v1.png

1. Layout and hierarchy

The reading order is clear: title, then three bars, then the gap annotation, then the source. The bars dominate visually, which is appropriate. The title sits above and is legible.

However, the curved arrow connecting the Believed and Measured bars is a bit subtle — it competes with the bars for attention, and the curvature makes the "39 points" label feel like an afterthought rather than the focal point of the chart. The source line at the bottom right is unobtrusive but legible.

2. Readability

All text is crisp at 150 dpi. The percentage labels on the bars (+24%, +20%, -19%) are bold and well-placed. The axis labels and title use appropriate sizes. The white background provides good contrast.

One issue: the y-axis label reads "Perceived speed change (%)" — but the Measured bar is not perceived; it is the actual measured result. This is misleading.

3. Data integrity

- The negative bar is correctly encoded below the zero line, so the visual direction is honest.
- The gap arrow connects the correct bars (Believed → Measured) and is labeled "39 points," which matches the spec.
- However, the arrow's curvature and its placement inside the bar area may cause viewers to read it as part of the data rather than an annotation. A straight, outward-positioned arrow would be clearer.

4. Top 3 concrete fixes (priority order)

1. **Fix the y-axis label** — change "Perceived speed change (%)" to "Impact on completion time (%)" or similar, so the measured bar is not mislabeled as perceived.
2. **Make the gap annotation more prominent** — use a straight, thicker arrow positioned outside the bars (e.g., to the right of the Measured bar) so it reads as a clear callout rather than a decorative element.
3. **Add a subtle horizontal grid** — a light gray line at 0% would reinforce the positive/negative divide and make the negative bar's position more immediately obvious.

test5_vision/grok.md

Visual review: diagram-v1.png

Reviewed from the rendered PNG, not from the HTML. The file was loaded with native vision (full frame) and checked against PIL pixel samples on the 2× export.

Canvas: **3200 × 1520** (1600 × 760 CSS at 2×). Paper `#faf8f5` / (250, 248, 245).

Verdict: **CHANGES REQUIRED**

---

1. Layout and hierarchy

Reading order as the eye actually travels. The Newsreader headline — *They felt faster. The timer said slower.* — is the first hit: largest, darkest, top-left. The rust kicker PERCEPTION VERSUS MEASUREMENT only reads if you look up. The gray deck sits immediately under the head and is fully legible in one pass.

Then the eye drops into the chart and lands on the **solid black +20% bar**. It is the heaviest mark on the page. The hollow +24% bar is second (high-contrast outline, and it is the only mark that breaks out of the peach rectangle). The rust −19% bar is third — saturated, but shorter and parked on the left. The peach panel is a large mid-tone shape; it registers as *background*, not as the story. The footer is a separate island after a wide empty field.

Whitespace (measured on the 2× PNG).

Region	y range (2×)	Empty height
Last chart ink / axis unit	y ≈ 1081	—
Footer rule	y ≈ 1384	**303 px (151 CSS px)** of empty paper
Footer type	y ≈ 1420–1445	—

The chart occupies the upper-middle. The data does not sit in the optical center. Below the unit line the figure is unused cream.

What competes. Three statements of the same claim: peach panel + 39-POINT GAP tag, italic footer caption. The headline wins by contrast. The black bar wins among the marks. The 39-point gap does **not** read as the structural story. It reads as a shaded plot background that one bar happens to overflow.

---

2. Readability

Transcribed from the PNG:

- Kicker: PERCEPTION VERSUS MEASUREMENT
- Headline: They felt faster. The timer said slower.
- Deck: Change in task-completion time among experienced developers using AI. They predicted a speedup, then believed they had one. Everyone was wrong in the same direction.

- Direction labels: SLOWER · FASTER (small, wide-tracked, #8a847c-range gray (138, 132, 124) — present, easy to skip)
- Rows: Predicted / BEFORE TRYING AI · Believed / AFTER USING AI · Measured / TIMED EXPERIMENT
- Values: +24% · +20% · −19% (U+2212 minus is present and readable)
- Band tag: 39-POINT GAP (small rust, sitting on the top edge of the peach rectangle, near the zero line — not spanning the band)
- Ticks: −30 −20 −10 0 10 20 30 — all clean, none smashed
- Unit: Change in completion time, percent
- Footer: 39 points between what developers felt and what the timer recorded. · Source: METR (2025), arXiv:2507.09089 — complete, no collision

Collision / knockout. The predicted bar is a 2 px outline with an **opaque white fill**. Where it crosses the peach band it punches a white rectangle through the supposed 39-point span. The band is no longer a continuous interval; it is a panel with a hole in it.

Contrast. Peach band samples (240, 226, 212) against paper (250, 248, 245) — delta (10, 22, 33). Visible at a glance (unlike a 4-point wash), but still reads as paper tint, not as a measured interval. SLOWER/FASTER and the row sub-labels share the same ghost weight.

Bar weight. Each bar is ~34 CSS px tall inside a peach field ~286 CSS px tall. The field is mostly empty peach between three thin strips. Ink is not the problem. The empty interior of the band is.

---

3. Data integrity

The encoding of the *values* is honest. Linear scale is 19 CSS px per point, zero at plot-x = 570:

Series	Direction	Visual length vs neighbors
Predicted +24%	right of zero	longest rightward bar; extends past the +20 tick toward 30
Believed +20%	right of zero	ends on the +20 tick / right edge of the peach band
Measured −19%	left of zero	slightly shorter than the +20 bar; left tip defines the band's left edge

Rust + left + −19% + the SLOWER side all agree. The negative bar reads negative.

The peach rectangle's x-span is the +20 bar plus the −19 bar. Geometry of the 39-point interval is correct.

What is not honest in the *reading*:

1. The peach field looks like a plot background, not a dimension. It has no end-caps, no bracket, no verticals at −19 and +20. The only label (39-POINT GAP) sits on the top edge near zero — the one place the gap is *not*.
2. The hollow +24% bar overflows the right edge of the band. Overflow-of-a-box is how a viewer reads "this bar didn't fit the plot," not "the gap is belief-to-measured, and prediction is outside it."
3. The white fill of that outline bar cuts the interval. A gap you can see through is not a gap.
4. Zero is a 1 px warm-gray hairline inside the band; findable once you look, easy to miss on the way in. (Correctly *not* a full-height black rule.)

The 39-point gap is *computed* correctly and *drawn* at the right x-span. It is not *visible* as the story.

---

4. Top 3 concrete fixes

1. **Close the dead band under the axis.** 151 CSS px of unused paper separates the unit line from the footer. Pull the footer up until it sits a short measure under the axis — or spend the recovered height on thicker bars and a real dimension, then crop the canvas. The figure currently reads as a header + a floating chart + a caption stranded at the bottom.
1. **Encode the 39-point span as a dimension, not a wash.** Drop the outline-bar-with-white-fill (it knocks a hole in the interval). Run visible end-marks at the rust tip (−19) and the black tip (+20) — a bracket, or a pair of full-row-height guides — and put 39-point gap *on that span*, not on the top edge near zero. A light wash may stay, but it cannot be the only encoding.
1. **Tie polarity to the axis.** SLOWER and FASTER currently float above the plot in ghost type, far from the ticks. Move them onto the axis (under the −30…0 and 0…30 runs, or as ← slower / faster → on the axis line) so the leftward rust bar is named as slower without a second look.

What works (preserve in v2)

- Diverging form, linear scale, rust-left / ink-right polarity.
- Paper / ink / rust palette (#faf8f5 / #1a1814 / #c2410c).
- Category labels (Before trying AI, not "before AI").
- Clean axis: every tick from −30 to 30 is readable; unit is centered, not colliding with 30.
- Footer source line, complete and unmangled.
- Headline sentence, which is the right claim.
- Required caption used once, in the footer, not shouted four times.

test5_vision/kimi.md

Vision Critique: diagram-v1.png

1. Layout and Hierarchy

The reading order is broadly correct — title → subtitle → bars top-to-bottom → source. The title "The 39-Point Gap" anchors the page well. However, the **39-point gap annotation competes with and overlaps the Believed bar**, creating a visual traffic jam in the center of the chart. The diagonal arrow draws the eye but lands its label directly on top of the +20% bar, forcing the reader to decode layered information. The Predicted bar (light gray-blue) recedes too much — it looks like background rather than data.

2. Readability

- **Text-bar overlap**: The brown "39-point gap / between perception and measurement" text sits directly on the teal Believed bar. At this size and color combination, readability is poor.
- **Source collision**: The source line "Source: METR (2025), arXiv:2507.09089" overlaps or nearly overlaps the x-axis label "Impact on task completion time" at the bottom. They occupy the same vertical band.
- **Cramped negative label**: The "-19%" label on the Measured bar is squeezed between the bar end and the left-side category text, leaving almost no breathing room.
- **Left labels**: The category labels ("Predicted what developers expected", etc.) are small (10pt) and low-contrast against the light background.

3. Data Integrity

The encoding is **honest** — the negative bar extends left of the zero line, and the positive bars extend right. The 39-point gap is **geometrically correct** (arrow from -19 to +20 spans 39 points). However, the gap annotation is **visually muddled** because the text sits on top of data rather than in clear space. The color choice for Predicted (muted gray-blue) unintentionally signals "less important" rather than "expectation."

Top 3 Concrete Fixes (Priority Order)

1. **Relocate the 39-point gap annotation to clean white space**. Use a vertical bracket or callout above/between the Believed and Measured bars with the label positioned entirely outside the data ink. The current diagonal arrow + on-bar text is the single biggest readability failure.
2. **Separate the source line from the axis label vertically**. Move the source line to the bottom-right corner below the plot area, or reduce bottom margin clutter so the x-axis label and source do not collide.
3. **Fix the -19% label collision and strengthen left-side labels**. Move the negative value label slightly left or increase left margin padding. Increase the size or weight of the category labels so they don't fade into the background.

test5_vision/gpt-5.5.md

Vision critique of diagram-v1.png

Reviewed by inspecting the rendered PNG pixels at `/home/matthew/wiki/drafts/hermes-kit/taste/results/gpt-5.5/vision-native/diagram-v1.png`.

1. Layout and hierarchy

The main title is clear and dominant, and the chart reads generally left-to-right from the row labels to the values. The central “39-point gap” callout is the strongest object inside the plotting area, which is appropriate for the anchor message.

However, the callout competes with the data marks because it sits directly across the middle of the plot and visually interrupts the believed and measured rows. The bracket crosses the chart and the label box floats over the data rather than feeling attached to the two values it is supposed to compare. The row labels also sit very close to the plotting area; the measured row in particular collides with its value label.

2. Readability

Most large text is readable: title, subtitle, +24%, +20%, and the central 39-point callout have enough size and contrast. The bottom caption and source are also legible.

The key readability defect is on the measured row: “Measured by timer” overlaps with the red “−19%” label. The red value is partially sitting on top of the black row label, making both harder to parse. The x-axis label is clear but slightly small relative to the rest of the figure. The axis title says “completion speed/time,” which is semantically muddy; the study measured completion time, while positive values are being translated into “faster.”

3. Data integrity

The encoding is mostly honest: +24 and +20 extend to the right of zero, while −19 extends left of zero. The zero line is visible, and the negative measured result does read as negative because the red bar clearly goes left.

The 39-point gap is mathematically represented by a bracket spanning from −19 to +20, but because the bracket is drawn across the measured row and the label box covers the center, the viewer may read it as a decorative annotation rather than as a precise distance between the green +20 and red −19 points. The “predicted before study” value is useful context, but because it is another long positive bar, it slightly distracts from the intended perception-vs-measurement contrast.

Top 3 concrete fixes, in priority order

1. Fix the measured-row collision: increase the left label margin and place the red “−19%” value label in open space near the red point, far enough away from “Measured by timer.”

2. Make the 39-point comparison more explicit: connect the bracket to the +20 belief and −19 measured positions with clearer guide lines, and label it “felt +20 vs recorded −19” so the gap is visibly about perception versus measurement.
3. Clarify the scale and hierarchy: label the left side as “SLOWER” and the right side as “FASTER,” use subtle background tinting for negative versus positive territory, and demote the +24 prediction so it does not compete with the primary +20/−19 story.

test5_vision/gpt-5.6-luna.md

Vision critique — diagram-v1.png

1. Layout and hierarchy

The page has a clear editorial top-to-bottom structure: headline, short explanatory deck, horizontal bar chart, then caption/source. The title is strong and the three colored bars are easy to find. The main competition is between the orange “39 points” bracket and the bar values: the bracket sits above the chart and looks like a measurement, so its endpoints need to be exact. The long middle-row label is also visually unstable and pulls attention toward the left edge.

2. Readability

Most type is comfortably readable at the rendered 1600×1000 size, with good dark-on-warm-white contrast. The “Measured” row is especially legible because its red value sits left of the zero line. However, “Believed after study” is clipped on the left edge: only the latter part of the label is visible. That is an objective rendering defect, not a stylistic preference. The chart has generous vertical spacing, but the large empty area to the right makes the central chart feel slightly underweighted.

3. Data integrity

The negative bar is encoded honestly: it extends left from a clearly marked zero line, is labeled −19%, and the axis includes negative ticks. The positive bars likewise extend right from zero. However, the 39-point annotation is not honest in v1: its bracket runs from approximately +2% to +20%, rather than from the measured −19% endpoint to the believed +20% endpoint. The caption states the correct 39-point relationship, but the dominant visual annotation contradicts it. The “predicted” +24% bar is also present as a useful context point, though the chart does not explicitly distinguish prediction from post-study belief beyond the row labels.

Top 3 fixes, in priority order

1. **Correct the 39-point bracket.** Anchor it to the −19% measured endpoint and the +20% believed endpoint, and label it “39 percentage points” so the unit is unambiguous.
2. **Prevent label clipping.** Move the row-label column inward and/or reduce the label width so “Believed after study” is fully visible within the canvas.
3. **Strengthen the perception-versus-measurement story.** Add a restrained visual grouping or short explicit labels that separate the two perception rows from the measured timer result, without adding decorative clutter. Keep the zero line and negative direction prominent.

test5_vision/gpt-5.6-terra.md

Visual critique — diagram v1

Reviewed the rendered PNG at 1800 × 1125.

1. Layout and hierarchy

The reading order starts well: the dark, large headline is the first thing seen, then the two pale cards and the oversized “39” callout compete for second. The chart carrying the actual measured result sits lower and reads as a separate section. This makes the statement of the 39-point gap visually more prominent than the evidence for it. The generous whitespace is calm and editorial, but the right callout has enough visual mass that it pulls attention away from the timer result.

2. Readability

Typography is crisp and high-contrast at this canvas size. The headline, +24%, +20%, and −19% are all easily legible. The light gray scale/grid and small source text are appropriately subordinate. In the measured section, the orange bar ending at the zero line, the “−19%” label, and “slower” together make the negative direction unambiguous. There is no visible overlap or crowding.

3. Data integrity

The negative encoding is honest: the measured bar extends left from a clearly marked 0%, and is explicitly labeled “−19% slower.” The perception values are stated correctly. However, the diagram does not *show* the 39-point comparison: +20% lives in a top card and −19% lives on a different, lower axis; the 39-point tile is an assertion with no visual connector, shared scale, or endpoint-to-endpoint span. A reader has to calculate/accept the relationship rather than see it.

Top three fixes, in priority order

1. Put believed +20% and measured −19% on one shared diverging scale, then draw a labeled 39-point span directly between their endpoints. This makes the anchor claim an observed relationship rather than a detached callout.
2. Make the three observations a single ordered comparison (predicted, believed, measured). Preserve predicted +24% as context, but visually prioritize the belief-versus-timer pair because that is the caption’s comparison.

3. Reduce the visual dominance and isolation of the large pale 39-point card. Reallocate that space to the quantitative graphic and use a compact, directly attached annotation.

test5_vision/gpt-5.6-sol.md

Vision critique — diagram v1

1. Layout and hierarchy

The reading order is clear: kicker → two-line headline → explanatory deck → three-row diverging bar chart → gap annotation → caption/source. The large serif headline establishes the editorial tone effectively, and the left-hand row labels scan cleanly.

The main hierarchy problem is that the chart gives **predicted +24%** the longest and therefore most prominent bar, even though the story and caption are specifically about the **belief +20% versus measured −19%** comparison. The 39-point annotation is also detached below the plotting area, so the eye first reads three equal-status rows and only later discovers which two endpoints define the claimed gap. The exact caption is then repeated in the footer, creating two competing summaries.

2. Readability

Type is crisp and contrast is strong on the warm off-white ground. The muted gray deck, row notes, tick labels, and source remain legible. Grid lines are restrained, and the zero line is dark enough to orient the diverging bars without becoming a heavy black divider.

The weakest local detail is the measured value: **−19%** is squeezed into the narrow space between the left label block and the red bar. It sits much closer to “MEASURED” than the positive values sit to their rows and risks reading as part of the row label rather than as the bar endpoint. The bottom bracket is readable, but its center is erased by a large paper-colored label box; this makes the span look like two disconnected corner marks rather than one continuous measured interval.

3. Data integrity

The diverging zero-based scale is honest and effective. The negative result unmistakably extends left into the region labeled “SLOWER,” while both perceptions extend right into “FASTER.” Tick spacing is consistent, and the three bar lengths correctly encode +24, +20, and −19.

The **39-point** claim is numerically correct and its bracket endpoints align with −19 and +20. However, because the bracket is physically separated from the believed and measured bars—and interrupted across much of its middle—the defining comparison is not immediately visible. A reader has to infer that the bracket excludes the +24 prediction rather than seeing that relationship directly.

Top 3 fixes, in priority order

1. **Make the +20 belief versus −19 measured pair the dominant visual comparison.** De-emphasize the +24 prediction as context, and connect or directly span the two focal endpoints so the 39-point gap is visible at first glance.
2. **Move endpoint values into or immediately adjacent to their bars with one consistent treatment.** In particular, remove the cramped −19% label from the left label column.
3. **Integrate the gap annotation with the chart and remove summary duplication.** Use a continuous arrow/span aligned to −19 and +20, then keep the exact caption once near that span; retain the source as a quiet footer.

test5_vision/glm-5v-turbo.md

Vision Critique: diagram-v1.png (GLM-5V-Turbo Self-Review)

What I See (Pixel-Level Description)

The rendered figure is a horizontal bar chart (~1000×600px at 150dpi) on an off-white (#FAFAFA) background. From top to bottom:

1. **Title** (bold 17px, near-black): "AI's Impact on Developer Speed: Perception vs. Measurement"
2. **Y-axis** (left, rotated 90°): "Change in completion time (%)" with tick labels −20, −10, 0, 10, 20, 30
3. **Three bars**, left to right:

- **Blue bar** (#4A90D9): height reaches +24, label "+24%" centered above it in bold 16px dark text. X-label: "Predicted\n(before study)" - **Green bar** (#2E7D32): height reaches +20, label "+20%" above. X-label: "Believed\n(after study)" - **Red bar** (#C62828): extends downward from zero to −19, label "−19%" centered below it. X-label: "Measured\n(actual)"

1. **Zero line**: solid dark-gray horizontal rule at y=0, spanning full width
2. **39-point gap annotation**: an orange-red (#D84315) double-headed horizontal arrow between the right edge of the green bar and left edge of the red bar, positioned at approximately y=+2. Above it: "39-point\ngap" in bold 13px orange-red text
3. **Caption** (bottom-center, italic 11px gray): "39 points between what developers felt and what the timer recorded."
4. **Source line** (bottom-right, 9px light gray): "Source: METR (2025), arXiv:2507.09089"
5. **Grid**: light dashed horizontal lines at each major tick

1. Layout & Hierarchy

Reading order is adequate but not optimized. The natural scan path is: title → blue bar → green bar → red bar → annotation → caption. This works because the temporal sequence (predict → believe → measure) maps cleanly to left-to-right.

What competes for attention: The orange "39-point gap" annotation fights with the red bar for being the visual climax of the story. The annotation sits in dead whitespace between bars 2 and 3, disconnected from any structural element. It reads as an afterthought pasted onto the chart rather than the central thesis.

Verdict: Functional hierarchy, weak dramatic structure. The most important insight (the 39-point gap) is visually subordinate to the individual bars.

---

2. Readability

Element	Assessment
Title	Clear, good weight, legible at all sizes
Value labels (+24%, +20%, −19%)	Bold, high contrast, well-positioned (above positive, below negative)
X-axis labels	Two-line labels are readable but slightly small; "(before study)" / "(after study)" parentheticals are faint
Y-axis label	Standard rotated orientation; no issues
"39-point gap" annotation	Text is clear but the arrow length is ambiguous — it spans the gap between bars, not a distance of 39 units
Caption	Appropriately de-emphasized; italic styling is conventional
Source	Correctly minimal

No overlapping or cramped elements detected. Spacing is generally generous. The one tight spot: the −19% label sits close to the bottom of the red bar but does not touch it — acceptable.

Contrast issue: The green bar (#2E7D32) against the off-white background has a contrast ratio of approximately 4.8:1 — passes WCAG AA for large text but is the lowest-contrast element carrying data. Not a failure, but notable.

---

3. Data Integrity

What's honest:

- Negative value encoded as downward-extending bar — **correct and unambiguous**
- Bar heights are proportional to values (24 > 20 in height; 19-unit depth for negative bar)
- Zero line is present and visibly weighted — essential reference
- Axis range (−28 to +34) doesn't truncate any data

What's problematic:

CRITICAL: The 39-point gap annotation is symbolically dishonest. The orange arrow is drawn horizontally between two bars. Its length corresponds to the inter-bar spacing (approximately 0.8 axis units), NOT to 39 percentage points. A reader who tries to verify "39" against the visual will find no correspondence. The annotation functions as a label, not as a visual proof. In editorial visualization, this is a significant weakness — the central claim of the figure ("39 points") cannot be *seen*, only *read*.

Secondary concern: The three bars use three unrelated hues (blue, green, red). While red-for-negative is intuitive, the blue/green distinction between "predicted" and "believed" is arbitrary. There's no visual cue that the first two bars belong together as "perception" while the third is "measurement." A reader scanning quickly sees three equally-weighted categories, missing the perception-vs-measurement dichotomy that is the entire point.

Minor: The caption restates the annotation almost verbatim ("39 points between what developers felt and what the timer recorded"), adding no new information. Redundancy without purpose wastes cognitive load.

---

4. Top 3 Concrete Fixes (Priority Order)

Fix #1 (Priority: CRITICAL) — Make the 39-point gap VISIBLE as a measured distance

Current state: Horizontal symbolic arrow that measures nothing. **Fix:** Replace with a **vertical dimension bracket/arrow** that spans from the top of the green bar (+20) down through zero to the bottom of the red bar (−19). This bracket should literally measure 39 units on the chart's own y-axis scale. Label it "39-point gap" with leader lines. This transforms the gap from a textual claim into a geometric fact that the reader can verify visually.

Rationale: The entire point of this figure is the 39-point discrepancy. If you cannot SEE 39 units of distance somewhere on the chart, the figure fails its primary communicative job.

Fix #2 (Priority: HIGH) — Group perception vs. measurement visually

Current state: Three equal-weight bars in three different colors. **Fix:** Use a **divergent or grouped visual treatment**:

- Option A: Color-code "perception" bars (predicted + believed) in shades of one hue (e.g., blues), and "measurement" in a contrasting hue (e.g., red/coral). Add a subtle background band or bracket grouping bars 1–2 as "what developers said" and bar 3 as "what the timer showed."
- Option B: Use a lollipop or dot-plot style where the three values are points on a single axis, with a shaded region highlighting the perception-measurement gap.

Rationale: The narrative structure is 2-vs-1 (two perception values vs. one measurement), not 1-vs-1-vs-1. The current layout gives all three bars equal visual weight, hiding the story.

Fix #3 (Priority: MEDIUM) — Strengthen the caption and remove redundancy

Current state: Caption repeats the annotation text nearly verbatim. **Fix:** Rewrite the caption to add interpretive context rather than restating the number. For example: "All perceptions erred in the same direction — and the timer disagreed." Move the source line to a more conventional position (below the x-axis, right-aligned). Remove the redundant "39 points" from the caption since the visual bracket now shows it.

Rationale: Every element should earn its place. If the annotation shows the 39-point gap visually, the caption should explain why it matters, not repeat the number.

Summary Scorecard

Dimension	Grade	Notes
Layout/Hierarchy	B+	Clear reading order; annotation placement weakens climax
Readability	A-	No overlaps; green bar contrast marginal
Data Integrity	C+	Bars honest; 39-point gap NOT visually encoded — critical flaw
Overall Editorial Quality	B	Competent but misses the story's visual punch

Single most important finding: The 39-point gap — the entire raison d'être of this figure — exists only as text, not as geometry. You can read "39-point gap" but you cannot see 39 units of distance anywhere on the chart. This is the fix that matters most.

test5_vision/glm-5v-turbo/critique.md

Vision Critique: diagram-v1.png (GLM-5V-Turbo Self-Review)

What I See (Pixel-Level Description)

The rendered figure is a horizontal bar chart (~1000×600px at 150dpi) on an off-white (#FAFAFA) background. From top to bottom:

1. **Title** (bold 17px, near-black): "AI's Impact on Developer Speed: Perception vs. Measurement"
2. **Y-axis** (left, rotated 90°): "Change in completion time (%)" with tick labels −20, −10, 0, 10, 20, 30
3. **Three bars**, left to right:

- **Blue bar** (#4A90D9): height reaches +24, label "+24%" centered above it in bold 16px dark text. X-label: "Predicted\n(before study)" - **Green bar** (#2E7D32): height reaches +20, label "+20%" above. X-label: "Believed\n(after study)" - **Red bar** (#C62828): extends downward from zero to −19, label "−19%" centered below it. X-label: "Measured\n(actual)"

1. **Zero line**: solid dark-gray horizontal rule at y=0, spanning full width
2. **39-point gap annotation**: an orange-red (#D84315) double-headed horizontal arrow between the right edge of the green bar and left edge of the red bar, positioned at approximately y=+2. Above it: "39-point\ngap" in bold 13px orange-red text
3. **Caption** (bottom-center, italic 11px gray): "39 points between what developers felt and what the timer recorded."
4. **Source line** (bottom-right, 9px light gray): "Source: METR (2025), arXiv:2507.09089"
5. **Grid**: light dashed horizontal lines at each major tick

---

1. Layout & Hierarchy

Reading order is adequate but not optimized. The natural scan path is: title → blue bar → green bar → red bar → annotation → caption. This works because the temporal sequence (predict → believe → measure) maps cleanly to left-to-right.

What competes for attention: The orange "39-point gap" annotation fights with the red bar for being the visual climax of the story. The annotation sits in dead whitespace between bars 2 and 3, disconnected from any structural element. It reads as an afterthought pasted onto the chart rather than the central thesis.

Verdict: Functional hierarchy, weak dramatic structure. The most important insight (the 39-point gap) is visually subordinate to the individual bars.

---

2. Readability

Element	Assessment
Title	Clear, good weight, legible at all sizes
Value labels (+24%, +20%, −19%)	Bold, high contrast, well-positioned (above positive, below negative)
X-axis labels	Two-line labels are readable but slightly small; "(before study)" / "(after study)" parentheticals are faint
Y-axis label	Standard rotated orientation; no issues
"39-point gap" annotation	Text is clear but the arrow length is ambiguous — it spans the gap between bars, not a distance of 39 units
Caption	Appropriately de-emphasized; italic styling is conventional
Source	Correctly minimal

No overlapping or cramped elements detected. Spacing is generally generous. The one tight spot: the −19% label sits close to the bottom of the red bar but does not touch it — acceptable.

Contrast issue: The green bar (#2E7D32) against the off-white background has a contrast ratio of approximately 4.8:1 — passes WCAG AA for large text but is the lowest-contrast element carrying data. Not a failure, but notable.

3. Data Integrity

What's honest:

- Negative value encoded as downward-extending bar — **correct and unambiguous**
- Bar heights are proportional to values (24 > 20 in height; 19-unit depth for negative bar)
- Zero line is present and visibly weighted — essential reference
- Axis range (−28 to +34) doesn't truncate any data

What's problematic:

CRITICAL: The 39-point gap annotation is symbolically dishonest. The orange arrow is drawn horizontally between two bars. Its length corresponds to the inter-bar spacing (approximately 0.8 axis units), NOT to 39 percentage points. A reader who tries to verify "39" against the visual will find no correspondence. The annotation functions as a label, not as a visual proof. In editorial visualization, this is a significant weakness — the central claim of the figure ("39 points") cannot be *seen*, only *read*.

Secondary concern: The three bars use three unrelated hues (blue, green, red). While red-for-negative is intuitive, the blue/green distinction between "predicted" and "believed" is arbitrary. There's no visual cue that the first two bars belong together as "perception" while the third is "measurement." A reader scanning quickly sees three equally-weighted categories, missing the perception-vs-measurement dichotomy that is the entire point.

Minor: The caption restates the annotation almost verbatim ("39 points between what developers felt and what the timer recorded"), adding no new information. Redundancy without purpose wastes cognitive load.

---

4. Top 3 Concrete Fixes (Priority Order)

Fix #1 (Priority: CRITICAL) — Make the 39-point gap VISIBLE as a measured distance

Current state: Horizontal symbolic arrow that measures nothing. **Fix:** Replace with a **vertical dimension bracket/arrow** that spans from the top of the green bar (+20) down through zero to the bottom of the red bar (−19). This bracket should literally measure 39 units on the chart's own y-axis scale. Label it "39-point gap" with leader lines. This transforms the gap from a textual claim into a geometric fact that the reader can verify visually.

Rationale: The entire point of this figure is the 39-point discrepancy. If you cannot SEE 39 units of distance somewhere on the chart, the figure fails its primary communicative job.

Fix #2 (Priority: HIGH) — Group perception vs. measurement visually

Current state: Three equal-weight bars in three different colors. **Fix:** Use a **divergent or grouped visual treatment**:

- Option A: Color-code "perception" bars (predicted + believed) in shades of one hue (e.g., blues), and "measurement" in a contrasting hue (e.g., red/coral). Add a subtle background band or bracket grouping bars 1–2 as "what developers said" and bar 3 as "what the timer showed."
- Option B: Use a lollipop or dot-plot style where the three values are points on a single axis, with a shaded region highlighting the perception-measurement gap.

Rationale: The narrative structure is 2-vs-1 (two perception values vs. one measurement), not 1-vs-1-vs-1. The current layout gives all three bars equal visual weight, hiding the story.

Fix #3 (Priority: MEDIUM) — Strengthen the caption and remove redundancy

Current state: Caption repeats the annotation text nearly verbatim. **Fix:** Rewrite the caption to add interpretive context rather than restating the number. For example: "All perceptions erred in the same direction — and the timer disagreed." Move the source line to a more conventional position (below the x-axis, right-aligned). Remove the redundant "39 points" from the caption since the visual bracket now shows it.

Rationale: Every element should earn its place. If the annotation shows the 39-point gap visually, the caption should explain why it matters, not repeat the number.

Summary Scorecard

Dimension	Grade	Notes
Layout/Hierarchy	B+	Clear reading order; annotation placement weakens climax
Readability	A-	No overlaps; green bar contrast marginal
Data Integrity	C+	Bars honest; 39-point gap NOT visually encoded — critical flaw
Overall Editorial Quality	B	Competent but misses the story's visual punch

Single most important finding: The 39-point gap — the entire raison d'être of this figure — exists only as text, not as geometry. You can read "39-point gap" but you cannot see 39 units of distance anywhere on the chart. This is the fix that matters most.