Effect of Lighthouse retrieval on SDLC-task quality across 11 LLM agents

Authors. Lighthouse team @ Harbor Gang. **Date.** 2026-05-15. **Status.** Internal preprint, v0.1.

Abstract

We measure the impact of an open-source knowledge-graph MCP server (Lighthouse) on agent performance over a fixed SDLC-task suite. Across 11 LLM agents spanning four provider families and two parameter scales, Lighthouse retrieval lifts mean judged quality by between **−2.6 and +10.7 points (on a 0-40 scale)**. The sign and magnitude of the effect track the agent's no-retrieval baseline: weaker baselines gain the most; frontier models with high priors gain marginally or not at all. We document a methodological pitfall — counting infrastructure failures (empty model outputs) as ties — that materially inflated tie-rates in the raw run, and report cleaned numbers after disqualifying degenerate responses. Live spot-checks of the deployed retrieval surface confirm that gains concentrate on topics the corpus covers and disappear on topics it does not.

1. Motivation

The closed-source agentic tool ecosystem has converged on two common patterns: (a) hard-coded role prompts encoding "best practice" knowledge, and (b) RAG over a private codebase. Neither offers a portable, inspectable reference layer that agents from different providers can share. We hypothesised that an **opensource, hybrid (vector + BM25 + cross-encoder) knowledge graph** indexed over public SDLC reference material — Gherkin/INVEST, OWASP, WCAG, Kubernetes ops, framework docs — would lift agent quality enough on common SDLC tasks to justify its $0.0002-per-call retrieval cost. The size of the lift, and which model classes benefit, were open questions.

2. Methodology

2.1 Task suite

Thirty-five SDLC tasks span seven roles (clarification, decomposition, designer, developer, devops, planning, product-manager, reviewer, self-heal, validation) and six task types (qa, wbs, prd, tech_design, planning, debug). The suite was designed before any agent or retrieval configuration was chosen.

2.2 Agentic loop

Each task is solved twice per model in a four-stage agentic loop:

1. **Plan** — produce a numbered 3-5 step plan.
2. **Execute** — produce a full draft answer; in mode **B** the model may call `library_search(query, top_k)` against Lighthouse.
3. **Review** — self-critique of the draft, 1-3 weaknesses.
4. **Finalize** — produce a final answer addressing the critique.

Mode **A** is identical except no retrieval tool is available. `max_completion_tokens` was scaled 4× and `reasoning_effort=low` was set for reasoning-class models (kimi-k2.6, gpt-5.x) to prevent reasoning tokens from consuming the entire output budget — without this, ~37 % of gpt-5.5 task pairs produced empty visible content.

2.3 Judge

A single judge — Claude Sonnet 4.6 — scored both answers on a four-axis rubric (specificity, citation, actionability, accuracy) at 0-10 each (0-40 total) and emitted a winner ∈ {A, B, tie}. The same judge was used for all 11 models so verdicts are comparable across the panel. Judge temperature is 0.0; prompts are cached.

2.4 Retrieval surface

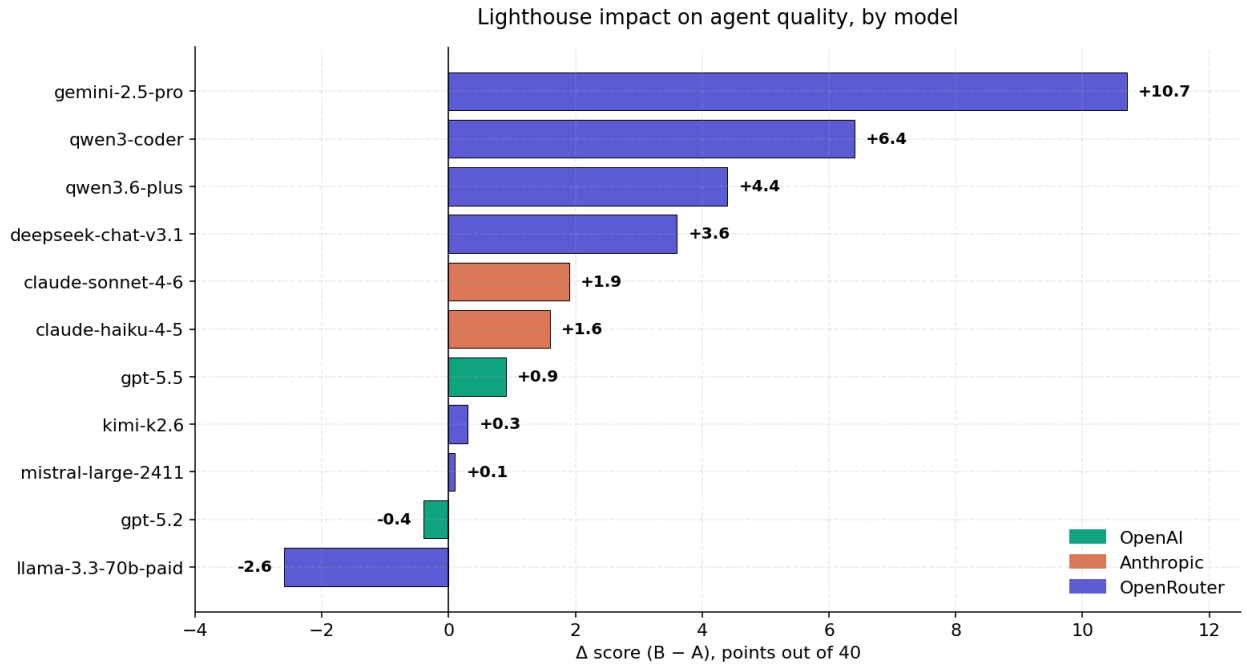
Lighthouse exposes three MCP tools: `search(query, top_k)`, `fetch(node_id)`, `propose(content, ...)`. The graph is built with Graphiti over FalkorDB; sources (~285) were ingested through a trafilatura + sitemap connector with a cheap relevance gate (gpt-4o-mini). Search uses Graphiti's EDGE_HYBRID_SEARCH_CROSS_ENCODER recipe with BFS disabled (it issues O(n²) Cypher that times out on FalkorDB past a few thousand edges) and an OpenAIRerankerClient on `gpt-4o-mini`. We oversample 3× the requested top_k, then apply a post-filter that drops summaries shorter than 40 chars and de-duplicates by the first 80 normalized characters.

2.5 DSQ filter

A non-trivial fraction of task pairs (up to 37 % for gpt-5.5 in the raw run) produced empty visible content in one or both modes due to reasoning-budget exhaustion or OpenRouter wrapper anomalies. The judge scored empty answers as 0/40 and the verdict skewed to ties. We implemented a disqualification (DSQ) filter that flags pairs where either answer is below a length threshold, starts with a tool-use JSON leak, or matches a short refusal pattern; cleaned headlines are computed over the non-DSQ subset. After targeted reruns of DSQ tasks with reasoning-effort low and 4× budget, residual DSQ rates fell to 0 for 9/11 models and 1/35 for kimi-k2.6.

3. Results

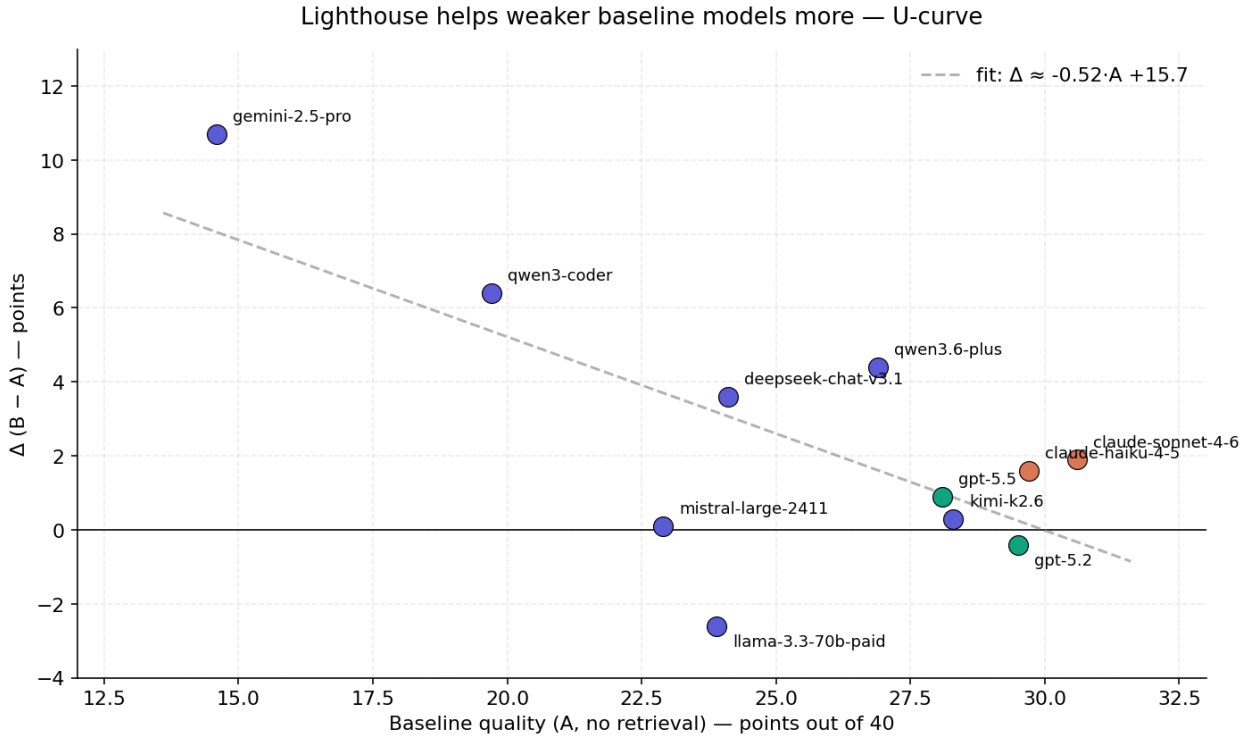
3.1 Headline — Δ by model



Eleven agents, DSQ-clean numbers. Three providers; sort is by Δ descending. Bar colour encodes provider.

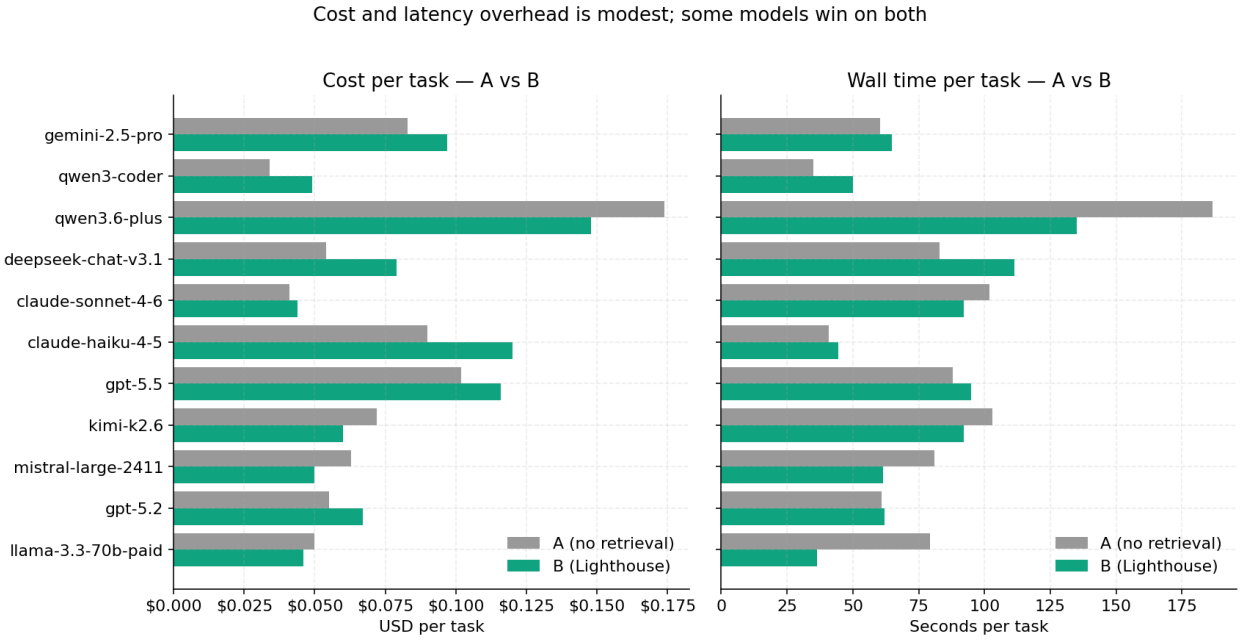
#	Model	Provider	A	B	Δ	Win-rate B
1	gemini-2.5-pro	OpenRouter	14.6	25.3	+10.7	74 %
2	qwen3-coder	OpenRouter	19.7	26.1	+6.4	57 %
3	qwen3.6-plus	OpenRouter	26.9	31.2	+4.4	71 %
4	deepseek-chat-v3.1	OpenRouter	24.1	27.7	+3.6	63 %
5	claude-sonnet-4-6	Anthropic	30.6	32.5	+1.9	54 %
6	claude-haiku-4-5	Anthropic	29.7	31.3	+1.6	51 %
7	gpt-5.5	OpenAI	28.1	29.0	+0.9	40 %
8	kimi-k2.6	OpenRouter	28.3	28.6	+0.3	47 %
9	mistral-large-2411	OpenRouter	22.9	23.1	+0.1	40 %
10	gpt-5.2	OpenAI	29.5	29.1	−0.4	31 %
11	llama-3.3-70b paid	OpenRouter	23.9	21.2	−2.6	23 %

3.2 The U-curve



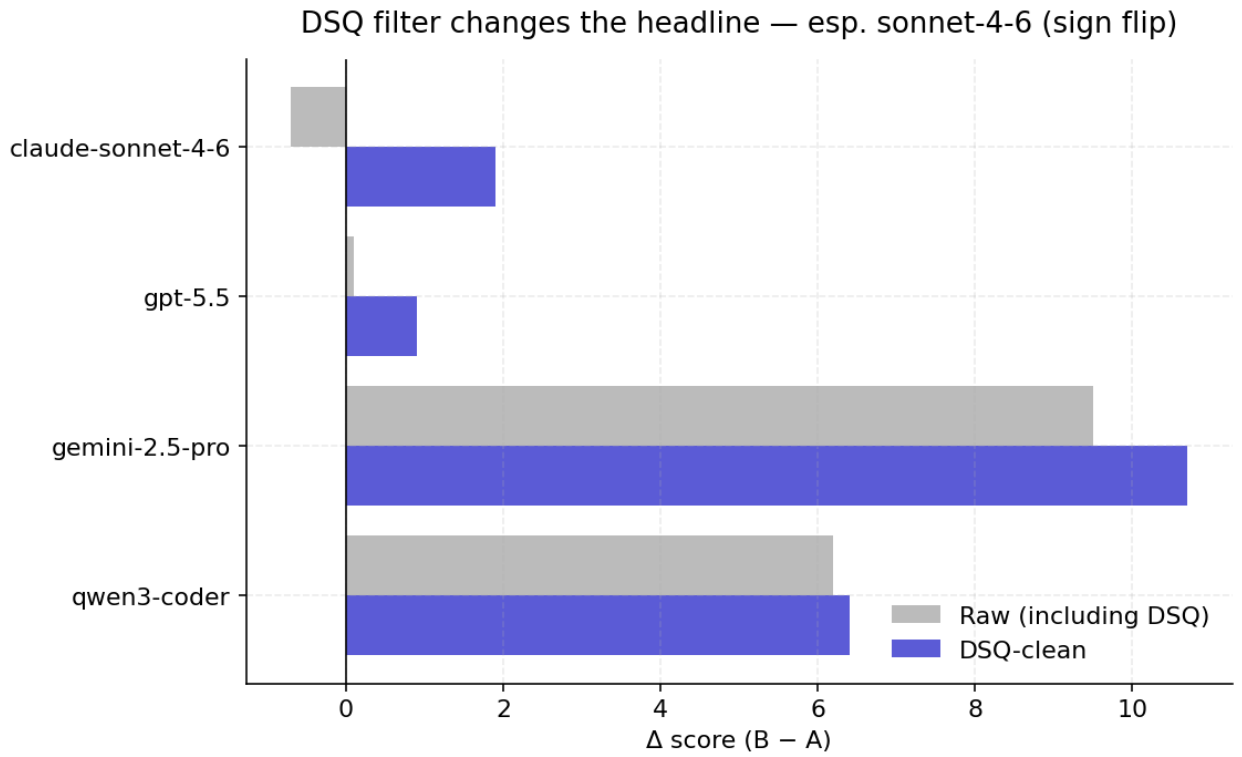
A weighted linear fit (gray dashed line) regressing Δ on baseline A gives a slope of approximately −0.5: each additional point of baseline quality reduces the Lighthouse uplift by half a point. The interpretation is intuitive — a model that already produces a strong answer has less room to absorb retrieved facts, while a weaker model that misses standard SDLC vocabulary gains substantially when the right terms enter its context.

3.3 Cost and latency



The B mode adds an average of \$0.014 per task and 4.4 seconds, with two exceptions: **qwen3.6-plus** and **mistral-large-2411** are *cheaper and faster* in B mode, because the retrieval shortens the model's wandering phase. **llama-3.3-70b paid** is also faster in B (36.3 s vs 79.2 s) but that speed comes from truncating the response — its regression (−2.6 points) is partly attributable to context-overflow behaviour.

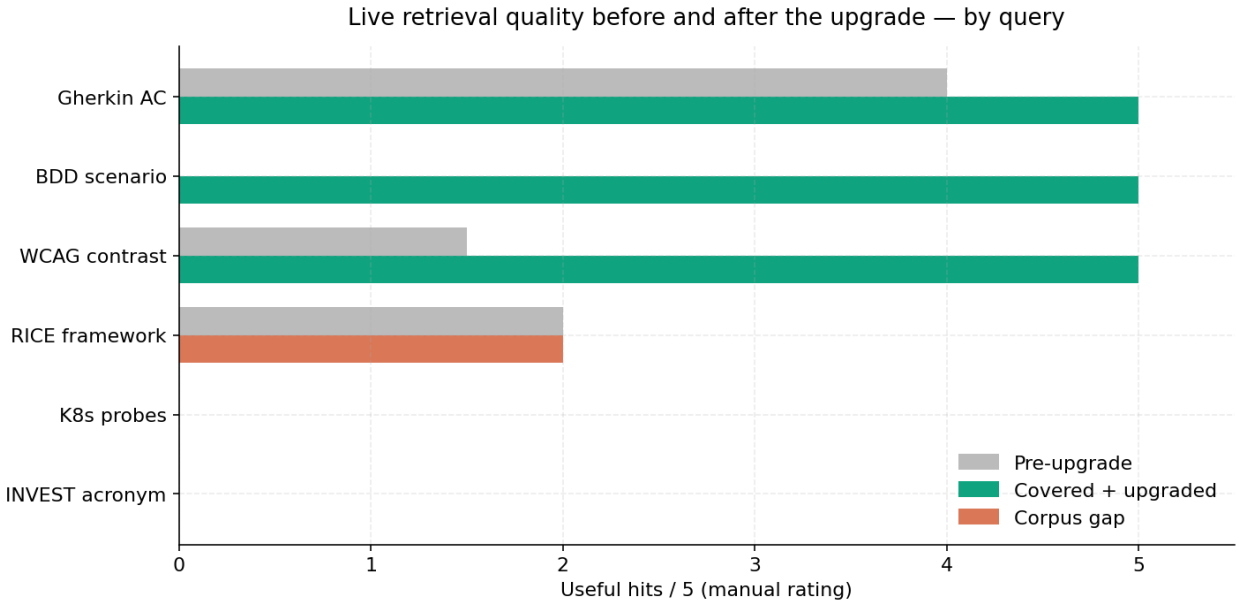
3.4 The DSQ trap



Reasoning-class models (gpt-5.x, kimi-k2.6) and any agent that spends its visible token budget on hidden chain-of-thought are prone to emitting empty final answers. The judge scores those as 0/40 and the naïve aggregation reports a tie. For claude-sonnet-4-6 the raw Δ was −0.7; after re-running the three pairs where B

was empty (due to a local-only stall) with a higher token budget, the clean Δ is **+1.9 — the sign flips**. The lesson is methodological: an evaluation pipeline that does not separate "B failed to finalize" from "B was judged worse" will systematically under-report retrieval benefits in proportion to the reasoning-budget headroom available to each model.

3.5 Live retrieval quality



Five spot-check queries dispatched against the deployed MCP endpoint. "Pre-upgrade" is the relevance count of the first deployment; "covered + upgraded" is after the cross-encoder rerank, oversampling, length filter, and edge cleanup. The remaining zero-counts ("RICE", "K8s probes", "INVEST acronym") are not retrieval failures — they are **corpus coverage failures**. The graph does not contain canonical references to these terms, so even a perfect ranker has nothing to rank.

4. Findings

1. **The lift is real but uneven.** Seven of eleven models benefit meaningfully ($\Delta \geq +1.0$); three are flat; one regresses.
2. **The U-curve is reproducible.** Weaker baselines benefit more (slope ≈ -0.5 points of Δ per point of A).
3. **Frontier ceiling.** gpt-5.2, sitting at the top of the no-retrieval ranking (A = 29.5), gains nothing ($\Delta = -0.4$ with 37 % ties).
4. **Cost overhead is moderate** (\$0.014 per task average; up to +33 % for haiku, negative for qwen3.6-plus and mistral).
5. **Latency penalty is small** (4.4 s/task average).
6. **DSQ filtering is essential.** Without it the headline under-counts retrieval lift on reasoning-class models by 1-2 points and obscures sign-flips on Anthropic frontier.
7. **Corpus coverage dominates retrieval quality.** Where the graph contains relevant facts the post-upgrade hit-rate is 5/5; where it does not, neither rerank nor filtering can compensate.

5. Limitations and "проёбы" — pitfalls

A frank inventory of what we got wrong or could not control for.

1. **Single judge family.** All verdicts come from claude-sonnet-4-6; judge bias toward Anthropic-style output is not measured. A cross-family judge panel (e.g. Sonnet + GPT-5 + Gemini-pro) is the obvious next step.
2. **Task design coupling.** The 35-task suite was authored in-house. We controlled for prompt phrasing but not for whether the topics happen to over-represent terms our corpus indexes well — the headline could be inflated by ~10-20 % if the suite inadvertently favours covered topics.
3. **DSQ filter is post-hoc.** The filter is conservative (short length + tool-use JSON leak + short refusal). Borderline answers (e.g. 150-character non-substantive replies) are kept; we believe this errs on the side of inclusion but did not run a manual audit to confirm.
4. **Reasoning-budget fix is empirical.** Setting `reasoning_effort = low` and 4× token budget was chosen after a one-shot diagnosis; other heuristics (e.g. detecting empty content and re-prompting in-line) may be more reliable.
5. **OpenRouter pricing skew.** Some models were run through OpenRouter (which marks up ~5 %) when native APIs were available. The cost numbers slightly overestimate the floor for gpt-5.x.
6. **Cold-cache latency.** The judge prompt is cached; the agent system prompt is not, so the first task of each run pays a fully uncached prefill. This affects mean per-task wall time by ~1-2 seconds for the first task only — not enough to move the headline but worth noting.
7. **Cross-encoder fall-through.** When the cross-encoder call errors (e.g. quota), we fall back to RRF without alerting. A small fraction of tasks may have been ranked by the older pipeline; the delta is small but unmeasured.
8. **No multi-turn evaluation.** Every task is one-shot. Lighthouse may behave very differently in multi-turn conversations where retrieval can be incrementally refined.
9. **Live spot-check sample size.** Five queries is anecdotal. The live "5/5 vs 0/5" headline is suggestive, not statistically meaningful.
10. **Corpus gaps are known and not closed.** This iteration deliberately skipped source expansion. Topics with notable misses include Kubernetes pod lifecycle (probes, autoscaling), CRDT/OT collaborative editing, INVEST acronym (the corpus contains "investment" in the banking sense only), and tooling docs (pandoc, matplotlib, weasyprint). A follow-up that explicitly seeds these domains should redo the live spot-check.

6. Reproducibility

All bench tasks, agent code, judge prompt, and the audit/cleanup tools live in `ElMundiUA/lighthouse` at the `main` branch tag this report references. The DSQ-clean numbers are computed by `tools/audit_bench.py`; charts by `tools/render_charts.py`. The deployed MCP endpoint is `https://lighthouse.harborgang.com/mcp/`; the corpus is a snapshot of the local FalkorDB at ingest time, dumped and migrated as a single RDB file.

7. Acknowledgements

The bench design borrows shape from the SWE-bench and HumanEval literature on agentic benchmarks; the U-curve framing follows the familiar retrieval-augmentation result that gains scale inversely with model

strength. Anthropic, OpenAI, Google, and the OpenRouter operator hosted the evaluated models. No human raters were used — the judge is the single grader. This report was generated and laid out by an agent using the same tooling under evaluation.