CLAWMARK: A Living-World Benchmark for Multi-Turn, Multi-Day, Multimodal Coworker Agents

Anonymous ACL submission

Abstract

Language-model agents increasingly act as persistent *coworkers* that work alongside users across many days, during which the surrounding environment keeps changing on its own and key evidence arrives in many modalities. Existing benchmarks miss this regime, scoring agents within a single static, largely text-centric episode. We introduce CLAWMARK, a benchmark for coworker agents built around multi-turn, multi-day tasks whose sandboxed service state evolves between turns, with fully rule-based verification and no LLM-as-judge. CLAWMARK comprises 100 tasks across 13 professional scenarios, scored by 1,537 deterministic checkers over post-execution service state. Benchmarking seven frontier systems shows the setting is far from solved: the strongest reaches 75.8 weighted score but only 20.0% strict Task Success, and performance drops sharply after the first exogenous environment change. CLAWMARK thus isolates adaptation to changing state as a central open challenge; we release the benchmark, evaluation harness, and construction pipeline to support reproducible coworker-agent evaluation.

1 Introduction

Language-model agents are moving from one-shot task solvers toward persistent *coworkers* that stay with a human across many working days. The surrounding environment keeps evolving while the agent works: new emails arrive, schedules shift, knowledge-base records are updated, and key evidence is distributed across images, scanned PDFs, audio, video, and spreadsheets. Frameworks such as OpenClaw and Claude Code already make this coworker-agent usage pattern increasingly operational. What remains under-measured is a benchmark setting that jointly evaluates agents on these properties, rather than under a single static session.

Existing agent benchmarks still fall short of this setting in three important ways, summarised in Table 1. First, *they evaluate at a time instant rather than across a time interval*: most existing benchmarks score an agent within a single session in which the environment is implicitly assumed not to change between steps. A coworker, by contrast, works across an interval where the environment can evolve between turns: a file read in step 1 may be different in step 2. Second, this assumption is reinforced by a *static-environment design* (Yao et al., 2024; Drouin et al., 2024; Xu et al., 2024): later state changes, when present at all, typically follow from the agent's own actions, rather than from exogenous updates coming from outside the interaction loop. Third, *inputs are text-centric*: although some benchmarks have added images (Koh et al., 2024; Xie et al., 2024), real office work often depends on raw multimodal evidence. The capabilities that distinguish a coworker (persistent state tracking, adaptation to external change, and multimodal evidence integration) therefore remain insufficiently measured.

Table 1: Structural comparison of agent benchmarks along the three axes of §1 plus the scoring choice. *MM* (multimodal): None / Part. (images only) / Full (audio, video, scanned PDFs, images, spreadsheets). *MD*: multi-day task (✓/✗). *Vf.*: verification, Rule-based / LLM-as-judge / Both. *Env.*: between-turn environment, Static or Dynamic. CLAWMARK is the only benchmark that is simultaneously multi-day, dynamic, and full-multimodal under rule-based scoring.

Benchmark	Tasks	Scen.	MM	MD	Vf.	Env.
WebArena (Zhou et al., 2023)	812	5	None	✗	Rule	Static
OSWorld (Xie et al., 2024)	369	9	Part.	✗	Rule	Static
Terminal-Bench (Merrill et al., 2026)	89	∼6	None	✗	LLM	Static
MCPMark (Wu et al., 2025)	127	5	None	✗	Rule	Static
Claw-Eval (Ye et al., 2026)	300	9	Full	✗	Both	Static
ClawsBench (Li et al., 2026)	153	15	Part.	✗	LLM	Static
CLAWMARK (Ours)	**100**	**13**	**Full**	✓	**Rule**	**Dyn.**

CLAWMARK is a benchmark for coworker agents targeting this regime. Each task is a multi-turn workflow spanning multiple in-universe work-

days (one turn per working day), executed against five stateful sandboxed services: filesystem, email, calendar, knowledge base, and spreadsheet. By "stateful sandboxed services" we mean runtime services inside the benchmark sandbox (Docker-mounted filesystem, GreenMail SMTP/IMAP, a Notion-compatible knowledge base, a Google-Sheets-compatible spreadsheet, and a Radicale CalDAV server), rather than static logs, cached snapshots, or third-party production endpoints. Between turns the environment changes independently of the agent, through announced events (which we call *loud events*) and unannounced *silent mutations*, and evidence is delivered untranscribed. Scoring is fully rule-based (§3.2) rather than LLM-as-judge over the agent's prose. CLAWMARK contains 100 tasks across 13 professional scenarios. Our contributions are threefold:

- We present CLAWMARK, a benchmark for coworker-agent evaluation that combines multi-turn multi-day tasks, a stateful sandboxed service environment, exogenous between-turn environment changes, and deterministic rule-based scoring in a single executable setting.

- We operationalise a *no-LLM-as-judge* scoring protocol: 1,537 deterministic Python checkers inspect post-execution service state, and each task is admitted to the released corpus only after two independent re-runs produce bit-identical checker verdicts and diagnostic messages, giving a deterministic alternative to model-judged scoring.

- We benchmark seven frontier agent systems on this setting. The top weighted score is 75.8 and the top strict Task Success is 20.0 (these are two different metrics on a 0–100 scale; see §3.2), and both metrics leave room to improve. The per-scenario picture is in Figure 1.

Because tasks unfold across multiple turns and external state changes between them, models with similar aggregate scores can follow meaningfully different adaptation trajectories over time. On the 73 tasks with exactly three turns, six of the seven evaluated models drop on Day 2, while the Claude Sonnet 4.6–GPT-5.4 gap narrows from $+6.5$ percentage points on Day 1 to $+4.0$ percentage points on Day 3 (§6.1). We will release the benchmark, the evaluation harness, and the construction pipeline to support reproducible evaluation of the next generation of coworker agents.

2 Related work

2.1 Agent benchmarks

Table 1 positions CLAWMARK against representative agent benchmarks. Realistic web and computer-use benchmarks such as WebArena (Zhou et al., 2023), Mind2Web (Deng et al., 2023), VisualWebArena (Koh et al., 2024), and OSWorld (Xie et al., 2024) establish strong single-episode evaluation settings. Related benchmarks extend tool coverage or execution domains, *e.g.,* MCP-Mark (Wu et al., 2025), MCP-Bench (Wang et al., 2025), SWE-bench (Jimenez et al., 2023), AgentBench (Liu et al., 2023), GAIA (Mialon et al., 2023), and Terminal-Bench (Merrill et al., 2026), but still evaluate progress within a fixed episode.

Multi-turn benchmarks such as tau-bench (Yao et al., 2024), WorkArena (Drouin et al., 2024), and TheAgentCompany (Xu et al., 2024) move beyond single-turn execution, yet later state changes still arise primarily from the interaction itself rather than from exogenous changes between workdays. Concurrent Claw-$*$ benchmarks target adjacent gaps: Claw-Eval (Ye et al., 2026) adds trajectory-aware scoring, ClawsBench (Li et al., 2026) trades reproducibility for ecological validity on live websites, and ClawArena (Ji et al., 2026) studies evolving information streams through perception-level questioning. CLAWMARK differs by combining multi-day tasks, exogenous between-turn mutation, raw multimodal evidence, and fully rule-based verification under deterministic checker re-runs (§4.2).

2.2 LLM agents

Rapid progress in LLM-based agent systems has enabled reliable multi-step tool use, code execution, and cross-service orchestration. Research frameworks such as SWE-agent (Yang et al., 2024), AutoGen (Wu et al., 2024), MetaGPT (Hong et al., 2023), and CAMEL (Li et al., 2023) study how language agents can act through tools, roles, and multi-agent interaction. Product and open-source scaffolds such as OpenClaw, Claude Code, Cursor, AutoGPT, and AgentGPT make similar capabilities operational against filesystems, shells, browsers, and external APIs.

Most of these systems are evaluated in episodic settings that reset between tasks; CLAWMARK instead targets the complementary coworker-agent

Figure 1: CLAWMARK results overview. Left: main leaderboard across seven frontier models under the single-run protocol (§5.1); Claude Sonnet 4.6 leads at 75.8 weighted score and the top strict Task Success is 20.0; both metrics leave room to improve. **Right:** distribution of the 100 tasks across the 13 professional scenarios; the benchmark covers specialised domains including legal assistance, investment analysis, and EDA that prior agent benchmarks have not reached.

regime, where the agent persists over an independently mutating external state. CLAWMARK is *framework-compatible*: its five-service tool schema is harness-agnostic, so any agent framework implementing it can be scored, with no claim of empirical equivalence across frameworks. We report all seven models under a single harness (OpenClaw, §5.1) to isolate model-side differences.

3 CLAWMARK

3.1 Overview

Each CLAWMARK task simulates a realistic office workflow that a coworker agent must carry out alongside its human user across multiple working days. A task is composed of two to six *turns*, where one turn corresponds to one in-universe working day, delivered as a wake-up message and executed against five stateful sandboxed services: filesystem, email, calendar, knowledge base, and spreadsheet. Between turns the framework mutates external state independently of the agent, as announced events (which we call *loud events*) delivered in the wake-up message, or as *silent mutations* that appear in the services without notification, so a competent coworker must refresh external state at each turn rather than act on cached assumptions from the previous day. Raw multimodal artifacts (photos, audio, scanned PDFs, video, spreadsheets) are first-class evidence, delivered without pre-transcription.

Scoring is fully rule-based: every task ships with 6–29 weighted Python checkers that query the post-turn state of the sandboxed services, and the task score is their weight-normalised pass rate on $[0, 1]$. The complete rule-based-scoring commitment, including the no-LLM-as-judge guarantee, is defined in §3.2. Figure 2 illustrates these elements on `insurance_task5`, a six-turn enterprise fire-claim adjudication.

CLAWMARK contains 100 tasks across 13 professional scenarios and 87 distinct in-task roles, running against the five sandboxed services with 1,072 raw multimodal artifacts (PDFs, images, audio, video, spreadsheets) and scored by 1,537 deterministic Python checkers of which 55 are red-line constraints. Tasks range from two to six turns (mean 3.6) and 6 to 29 checkers (mean 15.4). The corpus was produced by a task-first construction pipeline with multi-round human and agent-assisted review, described in detail in Section 4.

3.2 Evaluation

At evaluation time the framework runs each checker against the post-turn state of the sandboxed services (not a cached snapshot), and reports both the weight-normalised task score and a stricter aggregate *Task Success* metric. Throughout the paper, both are reported on a 0–100 percentage scale for readability.

$$\mathrm{score}(m, \tau) = \frac{\sum_{c \in \mathcal{C}_\tau} w_c \cdot p_c(m, \tau)}{\sum_{c \in \mathcal{C}_\tau} w_c} \in [0, 1], \quad (1)$$

$$\mathrm{Succ}(m) = \frac{100}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \mathbf{1}[\forall c \in \mathcal{C}_\tau,\ p_c(m, \tau) = 1] \in [0, 100], \quad (2)$$

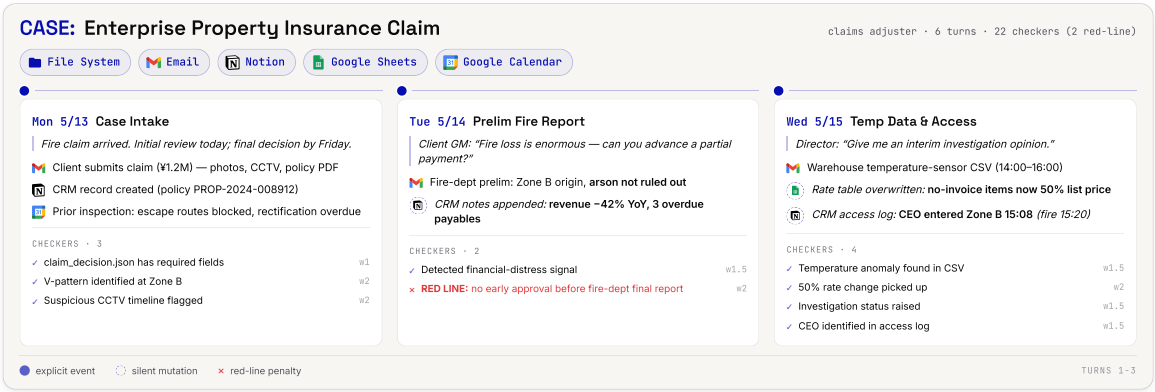

Figure 2: Anatomy of a CLAWMARK task. Example: `insurance_task5`, a six-turn enterprise property-insurance claim with 22 weighted checkers across five backends; turns 1–3 are shown. Each card is one in-universe working day, with wake-up prompts, loud/silent events, and per-turn checkers. Solid dots denote explicit events delivered in the prompt; dashed dots denote silent mutations injected between turns without notification. Red checker rows are high-weight red-line constraints implemented as deterministic checker failures inside the same weighted rubric.

where $\mathcal{C}_\tau$ denotes the checker set of task τ with per-checker weights w_c, $\mathcal{T}$ is the full 100-task corpus, and $p_c(m, \tau) \in \{0, 1\}$ is the deterministic pass/fail verdict that checker c produces against the post-turn service state left by model m on task τ. Each of the 1,537 checkers falls into one of four categories: filesystem / artifact inspection, external-backend state queries, email state queries, and numeric-tolerance or semantic-equivalence checks. **No LLM judge is invoked at any point during scoring**, whether during pass/fail decisions, tolerance checks, or aggregation; every verdict is a deterministic function of post-turn service state. A subset of 55 checkers (3.6%) are designated *red-line* constraints capturing compliance-sensitive actions a coworker must never take, and fall into four classes: premature-decision, compliance-bypass, data-exfiltration, and irreversible-write. In implementation, these red-line constraints are ordinary checker entries with fixed high weights w_{red} inside the same rubric, so a task with every non-red checker passing can still score substantially below 1.0 if a red-line checker fails. Per-scenario red-line counts are reported in Table 2.

Why report both weighted Score and strict Task Success. The two metrics answer different questions. Eq. (1) is a continuous signal that rewards partial progress and is appropriate for leaderboard ordering when rubrics vary in length (tasks in CLAWMARK have 6–29 checkers). Eq. (2) is a binary all-or-nothing signal that asks whether an agent completed the *entire* workflow a coworker was asked to perform, which is the deployment-relevant question in a professional setting. Because Task Success requires every checker to pass, it is sensitive to single-item rubric brittleness, long-tail checker dependencies, and the presence of red-line constraints (red-lines count as ordinary pass/fail items inside this metric, since the all-or-nothing aggregation does not reference per-checker weights). We therefore always report Task Success alongside weighted score rather than instead of it.

3.3 Design principles

Relative to prior agent benchmarks, CLAWMARK makes three design commitments. **Multi-turn timelines**: each task spans two to six in-universe working days (one day per turn) with clock advancement between turns, so the agent must sustain progress across day boundaries rather than emit a single-shot trajectory. **Dynamic environment**: between-turn mutation is injected in two layers. An `inject/stage{N}/` directory (legacy field name; one entry per turn) drops new files into the workspace, and turn-entry service-side Python appends email, rewrites spreadsheet rows, edits knowledge-base entries, and shifts calendar events. The agent therefore must refresh external state at the start of each turn rather than act on a day-1 mental model. **Full multimodal evidence**: raw artifacts are delivered without pre-transcription, and models must parse them with their own tools (`whisper`, `ffmpeg`, `PyMuPDF`, etc.).

At the implementation level, every task is fully specified by a single `task.py` together with per-turn inject layers and supporting artifacts, and runs

against the same five services (a Docker-mounted filesystem, GreenMail for SMTP/IMAP, a Notion-compatible knowledge base, a Google-Sheets-compatible spreadsheet, and a Radicale CalDAV server) inside an isolated `docker-compose` group. Appendix B provides a compact implementation-level view of how these task files are parsed into runtime objects and checked by the rule-based evaluation pipeline. Section 4 details how this corpus is produced: task distribution (§4.1) and the task-first construction pipeline and release gate (§4.2).

4 Benchmark construction

4.1 Task distribution

The 13 scenarios cover both general office roles (executive assistant, HR, content operation, e-commerce, journalist, project management, real estate, research assistant) and specialised professional domains that most existing agent benchmarks have not reached (clinical assistant, insurance, legal assistant, investment analyst, electronic design automation). The 87 in-task roles are substantive rather than cosmetic: the clinical assistant scenario alone includes a pharmacist assistant, a surgical scheduler, a charge nurse, and a chronic-disease clinic assistant, each with its own rubric. Per-scenario task, role, turn, and checker counts, together with red-line counts and dynamic-environment composition statistics (mean per-task shares of silent vs. loud between-turn changes), are reported in Table 2.

4.2 Construction pipeline

Producing 100 tasks across 13 heterogeneous professional scenarios while preserving multimodal authenticity and deterministic checker correctness presents a nontrivial authoring challenge. Text-only generation pipelines are insufficient on their own in this setting: the evaluation signal depends on artifacts that must look, sound, and parse like their real-world counterparts, and every checker must run against a stateful sandboxed service with bit-identical output across re-runs. We therefore adopt a *task-first* pipeline with four phases (task authoring, task-driven evidence sourcing, a review loop that repeats 3–5 rounds per task, and a release gate), summarised in Figure 3. Pipeline phases are distinct from in-task turns: each phase below is an authoring step, while a turn is one in-universe working day inside an executed task.

Table 2: Scenario composition of CLAWMARK (100 tasks, 87 distinct in-task roles, 55 red-line checkers). *# Roles* counts distinct `METADATA["role"]` strings; *Turns* and *Checkers* are per-task means; *Red-line* is the scenario red-line count (§3.3). *Silent % / Loud %* give the mean per-task share of between-turn injections that are un-announced vs. announced. These shares are descriptive metadata only, not used in scoring.

Scenario	# Tasks	# Roles	Turns	Checkers	Red-line	Silent %	Loud %
Clinical Assistant	4	4	3.5	17.5	15	58.5	41.5
Content Operation	12	11	3.3	13.5	3	49.9	50.1
E-commerce	9	5	3.1	15.0	0	52.1	47.9
EDA	1	1	2.0	18.0	0	50.0	50.0
Executive Assistant	7	7	3.7	17.7	0	47.8	52.2
HR	11	11	3.7	20.6	3	30.2	69.8
Insurance	7	7	5.1	13.0	14	66.9	33.1
Investment Analyst	6	6	4.0	17.7	0	50.0	50.0
Journalist	8	8	2.9	12.5	1	28.2	71.8
Legal Assistant	6	1	3.0	13.0	0	46.0	54.0
Project Management	8	7	3.1	13.1	6	56.2	43.8
Real Estate	6	6	3.5	18.5	2	53.7	46.3
Research Assistant	15	13	3.9	14.0	11	40.9	59.1
Total	**100**	**87**	**3.6**	**15.4**	**55**	**46.7**	**53.3**

Phase 1: Task authoring. The pipeline starts with task design, because the specification determines the multimodal inventory rather than the other way round. Each author writes a single `task.py` containing turn definitions, service seed hooks, between-turn injections (loud events and silent mutations), and a weighted checker rubric. Three invariants guide this step: every silent mutation is tied to at least one checker; every cross-modal contradiction spans at least two modalities; and every red-line is expressed as a deterministic state check rather than prose matching. The output is a concrete artifact list for the scenario, handed off to Phase 2.

Phase 2: Evidence sourcing. Each artifact is produced through one of three channels with a provenance tag: *web collection* of domain-realistic public documents (policy PDFs, government notices, corporate reports); *original recording* of audio / video / photographs (voice memos, walkthrough videos, whiteboard photos); and *targeted AI synthesis* (*e.g.,* Nano-Banana for photographs, procedural generators for forms and spreadsheets). The authoring-first order matters: the reverse (collect a corpus first, then craft tasks around whatever is available) produces information-dense but purpose-ambiguous artifacts whose checker coverage is incidental, not by design.

Phase 3: Review loop (3–5 rounds). Every task alternates between *task review* and *trajectory re-*

Figure 3: CLAWMARK construction pipeline. A task enters the release corpus only when task authoring, task-driven evidence sourcing, iterative review, and release-gate checks all pass.

view. Task review combines human artifact inspection with three AI audits: *multimodal integrity*, *checker-hacking*, and *task–checker correspondence*. Trajectory review runs two reference models end-to-end and asks an independent Codex-class reviewer agent to flag runtime-only design flaws such as ambiguous turn prompts, inject–checker races, under-specified deliverable schemas, and brittle string matches. Findings return to the author for revision, and the loop iterates 3 to 5 times per task.

Phase 4: Release gate. A task enters the released corpus only when four conditions hold simultaneously: (i) human sign-off on every multimodal artifact; (ii) clean results from all three task-review audits; (iii) no design-flaw finding from the reviewer agent on trajectories from two distinct reference models; and (iv) *bit-identical* checker verdicts and detail messages across two independent re-runs against the same frozen service state. Condition (iv) is the operational guarantee behind the no-LLM-as-judge claim; tasks that fail it more than twice are redesigned or dropped.

5 Experiments

We evaluate seven frontier models end-to-end on CLAWMARK, covering five proprietary models and two open-source models. This section describes the experimental setup (§5.1) and presents the main leaderboard together with the per-scenario breakdown (§5.2). Turn-level trajectory analysis and failure taxonomy follow in §6; case studies are deferred to Appendix G.

5.1 Experimental Setup

Models. We evaluate five proprietary models and two open-source models. The proprietary models are **Claude Sonnet 4.6**, **Claude Opus 4.6**, **GPT-5.4 (high)**, **Gemini 3.1 Pro Preview**, and **Qwen 3.6 Plus**. The open-source models are **Kimi K2.5** and **Kimi K2.6**. The Kimi K2.5 result reported here is from the public Infinigence-hosted endpoint; an internal instance-tuned Kimi K2.5 variant exists but is not reported in the main table to avoid mixing a public and a private baseline.

Framework and infrastructure. Every model runs under a single agent framework, **OpenClaw**, with identical tool schemas across models; no per-model prompt engineering is performed. For the Kimi-series models, we apply the upstream fix for incorrect tool-call identifier sanitisation in OpenClaw[1]. Each task executes inside an isolated `docker-compose` group comprising the agent container, GreenMail for SMTP/IMAP, a Notion-compatible knowledge base, a Google-Sheets-compatible spreadsheet, and a Radicale CalDAV server. Containers are torn down between tasks, so runs do not share state.

Evaluation settings and metric. All models use provider-default inference parameters, with *extended thinking* enabled where supported (Claude, GPT-5.4, Gemini) and prompt caching enabled where providers offer it. Weighted score and Task Success follow §3.2; both are reported on a 0–100 scale throughout this section. We additionally report wall-clock time for a full benchmark sweep and total input/output tokens. **Main-table results are from a single full sweep per model.** Appendix D reports a limited run-to-run stability

[1] https://github.com/openclaw/openclaw/issues/62319

Table 3: Main results on CLAWMARK (single full sweep per model). *Score* is the mean of Eq. (1) and *Task Success* follows Eq. (2), both on a 0–100 scale. *Red-line fail* is the fraction of the 55 red-line checker evaluations the model fails (Table 2). *Wall time* is total wall-clock for one sweep; *Input* merges `input+cacheRead+cacheWrite` tokens; *Tool calls* totals across 100 tasks (mean 48–71 per task). Rankings among models within a narrow score band should be read as tentative (Appendix D).

Model	Score	Task Success	Red-line fail	Wall time	Input tok.	Output tok.	Tool calls
Claude Sonnet 4.6	**75.8**	14.0	3.6%	22.3 h	257.8 M	2.57 M	5,736
Claude Opus 4.6	74.6	**20.0**	5.5%	22.6 h	266.7 M	2.02 M	6,112
GPT-5.4 (high)	72.0	9.0	3.6%	26.1 h	231.5 M	2.93 M	7,052
Kimi K2.6	68.4	7.0	7.3%	22.6 h	226.3 M	2.30 M	6,026
Gemini 3.1 Pro Preview	68.2	8.0	3.6%	18.9 h	338.8 M	1.77 M	5,877
Qwen 3.6 Plus	57.2	5.0	14.5%	33.3 h	315.1 M	4.56 M	6,119
Kimi K2.5	56.0	0.0	9.1%	22.8 h	214.0 M	1.47 M	4,776

check for two models; rankings among models that fall within a small weighted-score band should still be read with that caveat in mind.

5.2 Main Results

Table 3 reports the overall leaderboard across seven models and 100 tasks. By weighted score, Claude Sonnet 4.6 (75.8), Claude Opus 4.6 (74.6), and GPT-5.4 (72.0) cluster within a 3.8 pp band; because each main-table entry is a single full sweep, rankings among models within this narrow band should be interpreted cautiously. Under the stricter Task Success metric of Eq. (2), the ordering changes: Claude Opus 4.6 leads at 20.0, followed by Sonnet 4.6 at 14.0 and GPT-5.4 at 9.0, and fully correct end-to-end completion is much rarer than partial progress.

Both metrics leave room for improvement. No model exceeds 75.8 weighted score overall. Excluding the single-task EDA case, the highest per-scenario score is Claude Opus 4.6 at 92.6 on real estate, leaving headroom. The stricter Task Success metric makes the gap clearer: even the strongest model fully solves only 20.0% of tasks, Sonnet 4.6 fully solves 14.0%, GPT-5.4 fully solves 9.0%, and Kimi K2.5 fully solves 0.0%. On the hardest scenario, project management, every model scores below 44.0, and the mean weighted score across seven models is about 35.1. A partially correct trajectory that misses a single silent mutation, an incomplete backend writeback, or a turn-specific rubric item still forfeits Task Success credit, which is consistent with CLAWMARK's task design.

Per-scenario best is distributed across models. Table 4 shows that outright-best scenarios are split across Claude Sonnet 4.6, Claude Opus 4.6, GPT-5.4, and Gemini 3.1 Pro Preview. Coworker-agent evaluation therefore does not collapse to a single frontier-model ordering: project management is uniformly difficult, EDA separates Gemini from the Kimi family (with the caveat that EDA contains a single task), and red-line-heavy clinical / insurance / research-assistant scenarios reward compliance-aware trajectories. Additional per-scenario and efficiency observations are in Appendix E.

Table 4: Per-scenario weighted score (0–100), shaded on a red (low)– yellow – green (high) diverging scale. Bold marks the per-scenario best, which is split across four models; the two Anthropic models tie on EDA at 100.0. EDA contains a single task (Table 2), so its row is a case-level result, not a stable scenario-level trend.

Scenario	Sonnet 4.6	Opus 4.6	GPT-5.4	Kimi K2.6	Gemini 3.1	Qwen 3.6	Kimi K2.5
Clinical Assistant	**92.3**	88.3	86.9	87.4	73.2	77.9	71.0
Content Operation	66.2	**77.2**	71.2	73.2	72.1	66.9	54.9
E-commerce	**61.8**	57.4	50.3	50.4	55.5	45.6	46.3
EDA	**100.0**	**100.0**	95.7	8.7	91.3	43.5	43.5
Executive Assistant	73.9	75.6	**76.7**	67.8	49.2	62.9	56.0
HR	**82.5**	75.1	81.0	69.7	77.5	59.0	66.1
Insurance	81.9	**90.3**	87.0	75.9	74.8	62.5	61.5
Investment Analyst	80.8	68.9	82.0	82.1	**82.9**	30.9	61.7
Journalist	85.6	**86.2**	79.8	79.2	74.7	63.4	57.1
Legal Assistant	**84.1**	69.4	59.5	62.3	72.2	46.3	37.0
Project Management	41.7	**43.6**	37.9	26.9	39.3	22.6	33.6
Real Estate	89.1	**92.6**	87.0	84.5	82.6	87.9	72.6
Research Assistant	**81.4**	77.2	73.1	76.9	67.9	62.8	59.2

6 Analysis

The leaderboard reports overall performance, but it hides two properties that matter in CLAWMARK: how well models adapt after an exogenous state change, and which checker types drive most failures. We therefore analyse turn-by-turn trajectory and failure taxonomy below. Two illustrative case studies are deferred to Appendix G.

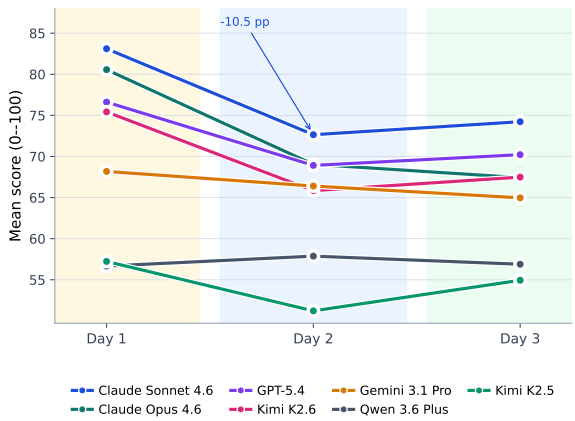

Figure 4: Day-by-day trajectory on the 73 tasks with exactly three turns. Day 2 is where the first external mutation lands: six of seven models drop there, while Qwen 3.6 Plus is the only model with a small Day-2 gain. By Day 3 recovery is partial, with most models still below their Day-1 baseline.

6.1 Turn-by-turn trajectory

Aggregate score masks meaningful differences in how models recover after the environment changes. We therefore focus on the 73 tasks with exactly three turns (one in-universe working day per turn) and plot mean score on Day 1, Day 2, and Day 3 in Figure 4. These trajectories reveal a broadly consistent pattern. Day 2 is where the first external mutation lands, and six of the seven models drop there. The largest Day-1 to Day-2 declines are Claude Opus 4.6 (80.6 to 69.0), Claude Sonnet 4.6 (83.1 to 72.6), and Kimi K2.6 (75.4 to 65.8); GPT-5.4 (76.6 to 68.9) and Kimi K2.5 (57.2 to 51.2) also fall meaningfully, while Gemini 3.1 Pro dips only slightly (68.2 to 66.4). Qwen 3.6 Plus is the lone exception, ticking up from 56.7 to 57.9. The first exogenous mutation is therefore a broad stressor, but not a perfectly uniform one.

By Day 3, recovery is partial and uneven: six of the seven models remain below their Day-1 baseline, and the Sonnet 4.6–GPT-5.4 gap narrows from 6.5 points on Day 1 to 4.0 on Day 3. Appendix F gives the full trajectory breakdown.

6.2 Failure-mode taxonomy

Failures concentrate in the two stressors that CLAWMARK is designed to test. Pooling 10,759 checker evaluations across seven models and 100 tasks yields 3,404 failures overall, or a benchmark-wide 31.6% per-evaluation failure rate (Table 5).

Silent-change detection fails at 56.5%, and **backend writeback** fails at 53.6%; both are nearly twice the benchmark-wide average. Models more often miss an exogenous update or fail to commit a backend action than they fail at ordinary extraction or deliverable checks. Backend writeback is also the single largest absolute failure bucket, contributing 567 failures, or 16.7% of all failures.

Red-line failures are rarer (7.1%) but concentrated: 26 red-line trips fall into 23 distinct (task, model) pairs, and on `pm_task2` every evaluated model trips at least one red-line. Appendix F details the remaining failure categories and red-line subclass breakdown, and Appendix G grounds the aggregate findings in two end-to-end trajectories.

Table 5: Failure-mode taxonomy, pooled across 7 models × 100 tasks (10,759 checker evaluations, 3,404 failures, 31.6% benchmark-wide failure rate). Evaluations are grouped by checker-ID pattern; *scenario-specific* covers task-bespoke rubric items. The two structural axes CLAWMARK targets, *silent-change detection* and *backend writeback*, fail at almost double the benchmark-wide rate.

Failure mode	Evals	Failures	Fail rate	Share of fails
Silent-change detection	315	178	**56.5%**	5.2%
Backend writeback	1,057	567	**53.6%**	16.7%
Cross-source consistency	203	69	34.0%	2.0%
Deliverable correctness	427	134	31.4%	3.9%
Evidence extraction	259	61	23.6%	1.8%
Compliance guardrail (should-not-do)	413	89	21.5%	2.6%
Red-line violation	364	26	7.1%	0.8%
Scenario-specific (task-bespoke)	7,721	2,280	29.5%	67.0%
All evaluations	**10,759**	**3,404**	**31.6%**	**100.0%**

7 Conclusion

CLAWMARK evaluates coworker agents in persistent, multi-day workflows with changing external state and raw multimodal evidence. Deterministic checkers over stateful sandboxed services make adaptation, writeback, and compliance-sensitive behaviour reproducible rather than judge-dependent. The results show substantial headroom: the best model reaches 75.8 weighted score, strict Task Success is 20.0%, and the dominant failures are silent-change detection and backend writeback. Coworker-agent correctness therefore requires not only reasoning over evidence, but also tracking changed state and writing correct actions back to the right service.

Limitations

Controlled services. CLAWMARK prioritises reproducibility over production realism: email, calendar, and filesystem services run locally, while Notion and Google Sheets use fresh per-task workspaces. This misses edge behaviour of fully shared production systems such as real rate limits, spam filters, and partial writes.

Coverage. The 100 tasks are not a completeness claim. The scenarios skew toward knowledge work, checker rubrics cover behaviours the authors anticipated, and silent-change annotations are not yet a release-time schema.

References

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, and 1 others. 2024. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Haonian Ji, Kaiwen Xiong, Siwei Han, Peng Xia, Shi Qiu, Yiyang Zhou, Jiaqi Liu, Jinlong Li, Bingzhou Li, Zeyu Zheng, and 1 others. 2026. Clawarena: Benchmarking ai agents in evolving information environments. *arXiv preprint arXiv:2604.04202*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008.

Xiangyi Li, Kyoung Whan Choe, Yimin Liu, Xiaokun Chen, Chujun Tao, Bingran You, Wenbo Chen, Zonglin Di, Jiankai Sun, Shenghan Zheng, and 1 others. 2026. Clawsbench: Evaluating capability and safety of llm productivity agents in simulated workspaces. *arXiv preprint arXiv:2604.05172*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023. Agentbench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, and 1 others. 2026. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and 1 others. 2025. Mcp-bench: Benchmarking tool-using llm agents with complex real-world tasks via mcp servers. *arXiv preprint arXiv:2508.20453*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*.

Zijian Wu, Xiangyan Liu, Xinyuan Zhang, Lingjun Chen, Fanqing Meng, Lingxiao Du, Yiran Zhao, Fanshi Zhang, Yaoqi Ye, Jiawei Wang, and 1 others. 2025. Mcpmark: A benchmark for stress-testing realistic and comprehensive mcp use. *arXiv preprint arXiv:2509.24002*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, and 1 others. 2024. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, and 1 others. 2026. Claw-eval: Toward trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Multi-turn evaluation: terminology and conventions

Throughout the paper we use a small terminology set (*turn*, *day*, and the legacy term *stage*) with the following relationships.

Multi-turn refers to tasks that contain *multiple independent interaction episodes*; each episode is itself a multi-step interaction (the agent issues many tool calls to advance the workflow). Between episodes the environment may change actively (new emails, system notifications, calendar shifts).

Turn = Day in CLAWMARK. In this paper one *turn* is exactly one in-universe working day. A two- to six-turn task therefore spans two to six in-universe working days, and the agent receives one wake-up message at the start of each turn. The vocabulary is unified accordingly: "Day 1 / Day 2 / Day 3" on plots and tables refers to the first, second, and third turns of a three-turn task; §6.1 reads turn-by-turn behaviour as day-by-day behaviour.

Stage appears only in two narrow contexts. (i) As a legacy field name in our task source: `inject/stage{N}/` directories and `stage0 / stage1 / ...` keys in `result.json` were named before we settled on the *turn* vocabulary, and the field names are preserved for code compatibility. They denote the same per-turn structures described above. (ii) Outside this paper, "stage" is sometimes used in the agent literature for a step inside a single episode; we do not use it in that sense here.

Phase (used in §4.2) refers to the four *authoring-pipeline* phases (task authoring, evidence sourcing, review loop, release gate) and is intentionally distinct from *turn*. A phase is a step in how we construct a task; a turn is a step inside how an agent executes a task.

B Task definition, parsing, and checking

Figure 5 shows the file bundle and runtime view at a glance. A typical `task.py` contains four kinds of declarations: (i) **turn entries**, one async function per turn (`turn0`, `turn1`, ...) defining the wake-up prompt, allowed tools, and any service-side mutation hooks; (ii) **inject layers**, one per turn, providing the new evidence files that appear at the start of that turn; (iii) **checker functions**, one per rubric item, returning a deterministic pass/fail by inspecting post-turn service state; and (iv) a **rubric** mapping checker IDs to weights and turn assignments. Red-line checkers are ordinary entries in the same rubric, distinguished by ID convention (`S*_redline_*`) and a fixed high weight.

The orchestrator runs each turn end-to-end inside an isolated `docker-compose` stack containing the agent container plus the five services (Docker-mounted filesystem, GreenMail SMTP/IMAP, the Notion API against a per-task workspace, the Google Sheets API against a per-task spreadsheet, and a Radicale CalDAV server). At end-of-turn, every checker for that turn is invoked against the post-turn sandboxed-service state; outcomes are recorded but the next turn proceeds regardless of failure. After the final turn, all rubric items are aggregated under Eqs. 1 and 2.

CLAWMARK's framework forms a tight correspondence between the natural-language description of a task and its executable form: the wake-up messages, the loud and silent updates, and the rubric items are each described once and executed in the same place. This keeps the cost of translating between intent and execution low, which is what makes 3–5 rounds of joint author–reviewer iteration affordable across the 100-task, 13-scenario corpus (§4.2).

Recent agent benchmarks are increasingly multi-service and multi-turn (§2.1). CLAWMARK's framework is shaped for this regime by construction, not retrofitted onto a single-service or single-turn substrate: services compose freely, and turns are first-class evaluation episodes with their own prompts and rubric items.

C Reproducibility and framework patches

Patches to OpenClaw applied to all sweeps. The seven-model sweep used a single OpenClaw build with model-specific routing patches: (i) disable tool-call-id sanitisation for OpenAI-compat models that reject rewritten ids (Kimi family); (ii) replace null tool-call `arguments` with `{}` for MiniMax-style outputs; (iii) declare `input: ["text"]` for text-only models when invoked through the multimodal harness; (iv) auto-route GPT-5 series to the `openai-responses` API with high thinking effort; (v) auto-route Gemini through the native `generateContent` endpoint to preserve `thoughtSignature` across tool-call round-trips; (vi) strip Gemini-unsupported JSON-Schema key-

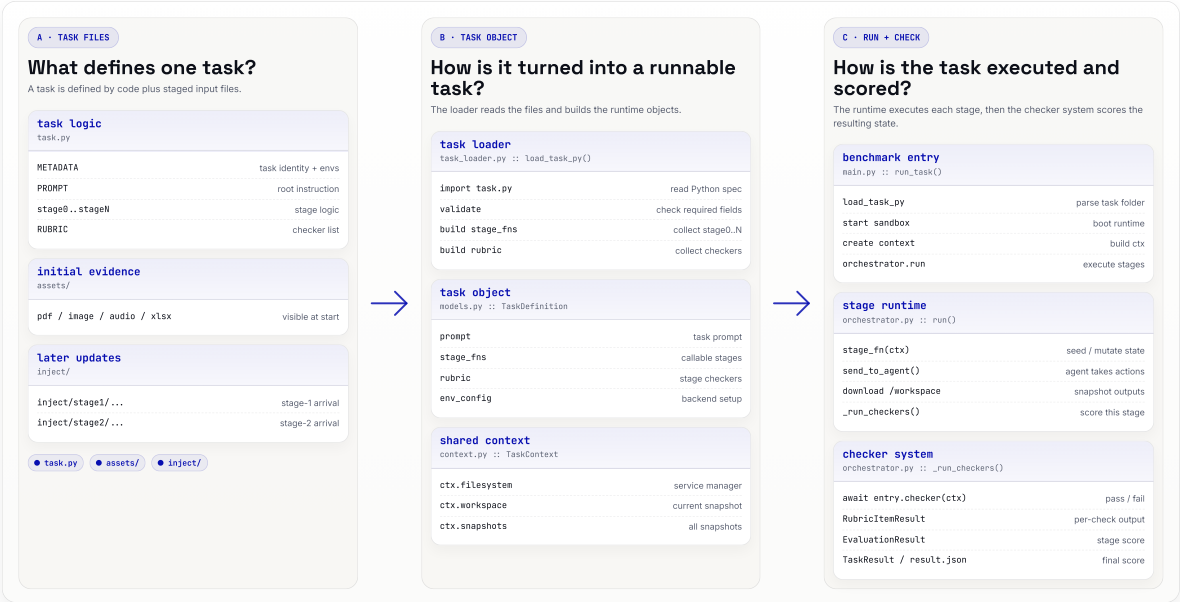

Figure 5: Implementation-level view of a CLAWMARK task. A task is defined by a compact file bundle: `task.py` specifies per-turn prompts, service seed hooks, and the checker rubric, while `assets/` and `inject/stage{k}/` (legacy field name; one entry per turn) provide static evidence and between-turn updates. The loader parses these files into runtime task objects, after which the orchestrator executes turns against the sandboxed services and runs deterministic Python checkers over post-turn state. Per-turn outcomes are aggregated into the per-task result record and final score.

words from tool definitions. These patches are applied uniformly and do not advantage any particular model.

Container limits and timeouts. Per-turn agent timeout is two hours (forced, regardless of task `METADATA`); the LLM idle timeout inside that window is 30 minutes. Default parallelism is 4–8 concurrent compose stacks. Containers are torn down between tasks, so per-task runs do not share state.

Inference settings. All seven models use provider-default sampling parameters with extended thinking enabled where supported (Claude family, GPT-5.4, Gemini 3.1 Pro) and prompt caching enabled where supported. We do not perform per-model prompt engineering. Wall-clock, token, and tool-call totals reported in Table 3 reflect a single full sweep per model.

D Run-to-run stability

To bound the run-to-run noise behind Table 3, we ran three independent full sweeps of the 100-task corpus for two models chosen from opposite sides of the open/proprietary divide: Kimi K2.6 (open-source) and GPT-5.4 (proprietary). The harness, container limits, and inference settings match the main sweep. The three per-model weighted scores span a 2.8 pp range for Kimi K2.6 (68.4, 70.8, 71.2) and a 1.0 pp range for GPT-5.4 (72.0, 72.5, 73.0). Both ranges are small relative to the 19.8 pp cross-model spread of Table 3, indicating that the single-sweep results are stable.

E Additional experimental observations

Cost normalisation. For the efficiency view in §5.2, we treat total tool calls and total (input + output) tokens as compute-side proxies rather than as direct dollar-cost signals: API unit pricing differs by provider and changes month to month, so a frozen cost-per-token table would misrepresent at least one model by the time of camera-ready. Readers who need a monetary bound can recombine the per-model token counts in Table 3 with the provider's published rate at read time.

Scenario-level ordering. Table 4 shows outright-best-by-scenario splits among four models: Claude Sonnet 4.6 on clinical assistant, e-commerce, HR, legal, and research assistant (five); Claude Opus 4.6 on content operation, insurance, journalist, project management, and real estate (five); GPT-5.4 on executive assistant; and Gemini 3.1 Pro Preview on

investment analyst. The two Anthropic models tie on the single EDA task (100.0 each). Below third place the ranking reshuffles more sharply: Kimi K2.6 is competitive with Gemini on investment analyst (82.1 vs. 82.9), but trails it sharply on EDA (8.7 vs. 91.3) due to a single vision-dependent task it routed incorrectly. Specialisation-driven scenarios are where models most separate: EDA separates Gemini from the Kimi family, project management is uniformly difficult for all models, and clinical / insurance / research-assistant are where red-line-heavy rubrics (Table 2) reward compliance-aware trajectories.

Score and compute proxies. There is no monotone relationship between score and tool or token consumption. Three per-model point comparisons illustrate: +23% tool calls at -3.8 pp score (GPT-5.4 vs. Sonnet), +31% input tokens at -7.6 pp (Gemini vs. Sonnet), and $1.8\times$ output tokens at -18.6 pp (Qwen vs. Sonnet). On *score per thousand tool calls*, Sonnet 4.6 leads at 13.2, followed by Opus 4.6 (12.2), Kimi K2.5 (11.7), Gemini (11.6), Kimi K2.6 (11.3), GPT-5.4 (10.2), and Qwen (9.3). The two top-scoring models are also the two most tool-efficient, so under this proxy score and efficiency move together rather than trade off.

F Additional analysis details

Turn-level recovery. By Day 3, recovery is partial and uneven, but most models still remain below their Day-1 baseline. Sonnet 4.6, GPT-5.4, and Kimi K2.6 all rebound modestly relative to Day 2 ($+1.6$, $+1.3$, and $+1.6$ pp respectively), while Kimi K2.5 posts the largest Day-2 $\rightarrow$ Day-3 rebound at $+3.7$ pp. Even so, six of the seven models finish Day 3 below Day 1; only Qwen 3.6 Plus returns to essentially parity with Day 1 ($+0.2$ pp). The clearest comparison is Sonnet 4.6 versus GPT-5.4: their gap narrows from $+6.5$ pp on Day 1 to $+3.7$ pp on Day 2 and $+4.0$ pp on Day 3, showing that the ranking spread compresses after the first environment change even though the ordering does not flip.

Failure categories beyond the two main stressors. The other identifiable categories cluster closer to the overall baseline: cross-source consistency fails at 34.0%, deliverable correctness at 31.4%, evidence extraction at 23.6%, and compliance guardrails at 21.5%. Scenario-specific rubric items account for 67.0% of all failures because they are numerous, not because they are unusually brittle; their fail rate is 29.5%, close to the benchmark-wide average.

Red-line incidents. Red-line checkers fail only 7.1% of the time (26 failed evaluations over the 364 red-line evaluations identified by the failure-mode taxonomy script across seven models), but the incidents concentrate in 13 tasks and 23 distinct (task, model) pairs. *Per-model* fail rates are reported in the *Red-line fail* column of Table 3: the three frontier systems cluster at the low end (Claude Sonnet 4.6 3.6%, GPT-5.4 3.6%, Gemini 3.1 Pro 3.6%), Claude Opus 4.6 and Kimi K2.6 sit in the middle (5.5% / 7.3%), and Qwen 3.6 Plus is the outlier at 14.5%, roughly $4\times$ the frontier top-3. Kimi K2.5 is intermediate at 9.1%. *Per-subclass*, compliance-bypass is the hardest red-line family at 10.4% (8 / 77), followed by data-exfiltration at 8.6% (6 / 70), premature-decision at 6.1% (9 / 147), and irreversible-write at 3.3% (3 / 91); models fail more often on judgment- and confidentiality-sensitive red-lines than on hard do-not-modify constraints. *Per-scenario*, red-lines are not uniformly distributed (Table 2: 15 in clinical, 14 in insurance, 11 in research assistant, 6 in project management, 3 in content operation, 3 in HR, 2 in real estate, 1 in journalist, and 0 in the remaining five), so red-line fail rates should be read against scenario-specific denominators rather than the benchmark-wide 7.1%.

G Case studies

Case 1: cross-modal reasoning chain on `content_operation_task7`. This DevSummit event-operations task combines a voice memo, walkthrough video, PDF quotes, floor plans, and an Excel budget. GPT-5.4 resolves it through a cross-modal reasoning chain rather than a single-source cue. Table 6 shows the key trajectory from its highest-scoring run (80.0%). The important transition is from step 1 to step 2: the model first extracts an investigation lead from the audio ("capacity may be inflated"), then uses `ffmpeg` to convert the walkthrough video into image frames and searches those frames with that specific question in mind. Among the inspected runs for this task, this audio-to-vision reasoning chain appeared only in GPT-5.4's trajectory.

Table 6: Positive case study. GPT-5.4's highest-scoring trajectory on `content_operation_task7` (score 80.0). The causal transition from audio (step 1) to video-frame vision (step 2) was unique among the evaluated models.

#	Source	Tool	Discovery
1	Voice memo	`whisper`	Patricia: capacity claims may be inflated → sets lead
2	Walkthrough video	`ffmpeg` → vision	Fire marshal notice: 180 persons vs marketed 300
3	PDF quote p.3 cl.7	`PyMuPDF`	200-person minimum spend: $9,000 vs quoted $6,750
4	Excel budget	ZIP/XML parser	Hidden row 14: AV equipment $12,000 (invisible in UI)
5	Notion database	API query	Caterer safety certificate expired Jan 2025
6	Email + calendar	IMAP / CalDAV	Keynote cancellation → three replacement options

Case 2: red-line violation on `insurance_task1`. `insurance_task1` is a four-turn auto-insurance claim adjudication. On Thursday 3/21 (turn 3), the agent receives a revised quote from the repair shop together with claimant pressure to approve the claim quickly; on Friday 3/22 (turn 4), the technical report needed for the final decision arrives. The red-line checker `S3_redline_no_direct_approve` (weight 2.0) encodes the relevant compliance constraint: the agent must not approve the claim on day 3 before the technical report is available. Kimi K2.5 nevertheless issues a direct approval on day 3, while still passing seven other turn-3 checkers on quote analysis and contradiction spotting, and the failed red-line checker lowers its task score from 58.1% (counterfactual) to **48.8%** (actual), a -9.3 pp reduction driven entirely by the single violation. This illustrates the kind of surface-complete but compliance-violating behaviour that LLM-as-judge evaluation typically misses: the violation is defined by what the agent *did* to the sandboxed service state, not by the prose it emitted. Across the full benchmark, 26 red-line trips fall into 23 distinct (task, model) pairs, most dramatically on `pm_task2`, where *all seven models* trip at least one red-line.