MedicalHarness: A Controlled Evaluation of LLMs and Agent Harnesses on Medical Tasks

Anonymous authors
Paper under double-blind review

Abstract

LLM agents are increasingly built for medical work and scored on clinical benchmarks. Each such score, however, comes from a model running inside an agent harness, the system that controls the loop between the model and its environment. An agent's score is therefore a property of a model–harness pair. For medical agents, how much outcomes change with the harness has rarely been measured. Measuring this change, and explaining it, raises two challenges. First, a harness comparison must change nothing but the harness and be repeated across models and kinds of task. Second, comparing whole harnesses leaves their mechanisms bundled together, so it cannot show when an individual mechanism helps. To address these challenges, we present MedicalHarness, a controlled study of models and agent harnesses on medical tasks. We first build MedicalHarnessBench to evaluate agents on 107 tasks across four domains that each test a different harness capability. Using this benchmark, we run five open-weight models under five agent harnesses, changing only the harness within a comparison, and analyze both outcomes and execution traces. To study individual mechanisms, we build MH-Lab, a controlled harness that switches off context management, planning or tool exposure one at a time within a shared execution loop. We find that the harness and its interaction with the model account for about a quarter of the outcome variance, and that no single harness is best across models and tasks. Code and data are available at `https://anonymous.4open.science/r/MedicalHarness/`.

1 Introduction

LLM agents have drawn growing interest in medicine, with the promise of supporting clinical work and reducing clinicians' administrative burden (Wang et al., 2025a; Jiang et al., 2025). These applications require agents to act within clinical systems, from computing risk scores and finding facts in patient records to checking trial eligibility and resolving insurance denials across hospital portals. To assess how well agents complete such work, a growing set of benchmarks evaluates them in simulated clinical environments (Jiang et al., 2025; Schmidgall et al., 2026; Liu et al., 2026; Bedi et al., 2026; Chen et al., 2026). The resulting scores reflect both the model and the agent system that guides its interactions with the environment. Understanding these results therefore raises a basic question: *how much do outcomes change when the same model runs inside a different agent system?*

The system around the model is its agent harness. Harnesses such as Claude Code (Anthropic, 2026), Codex (OpenAI, 2026), OpenClaw (OpenClaw contributors, 2026) and Hermes (Nous Research, 2026) determine what the model sees, which tools it can call, how its actions are executed, how its history is managed and when execution ends (Li et al., 2026). History management affects which clinical evidence remains available, while stopping rules determine whether an unfinished workflow can continue. Such differences can lead the same model to different outcomes (Zhang et al., 2026b; Yao et al., 2026). An agent's score is therefore a property of a model–harness pair. In medicine, however, few studies compare harnesses under controlled conditions across models and kinds of task. Making such comparisons, and explaining the differences they reveal, raises two challenges.

Challenge 1: measuring and understanding harness differences. When agent comparisons change the model, serving conditions or execution budget along with the harness, the resulting score gaps mix these effects (Zhang et al., 2026b; Yao et al., 2026). A controlled comparison must therefore run the same model on the same tasks, with the serving setup and budgets held fixed. A result from

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

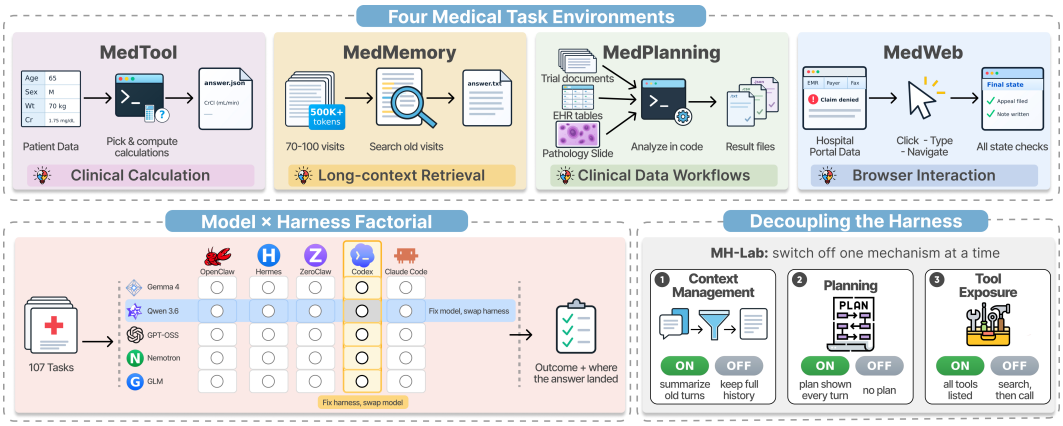


Figure 1: **Overview of `MedicalHarness`. Top:** `MedicalHarnessBench`'s four environments, each testing a different harness capability: `MedTool` (clinical calculation), `MedMemory` (long-context retrieval), `MedPlanning` (clinical data workflows) and `MedWeb` (browser interaction). **Bottom left:** every model runs under every harness on all 107 tasks, so each row varies only the harness and each column only the model. **Bottom right:** `MH-Lab` switches off one mechanism at a time, keeping the rest of its loop fixed: context management (summarizing old turns), planning (a plan shown every turn) or tool exposure (all tools listed instead of hidden behind search).

one model on one task type may not hold elsewhere, so this comparison must be repeated across models and different kinds of medical task. Even under these controls, the same low score can reflect different failures, such as a wrong answer, no answer at all or an unfinished workflow, so execution traces and task diagnostics are needed to tell them apart.

Challenge 2: understanding when individual mechanisms help. Whole-harness comparisons leave context management, planning and tool exposure bundled together, making their individual effects difficult to identify (Li et al., 2026). Whether a mechanism helps can also depend on the model and the task. Some mechanisms, such as summarization, operate only under particular conditions, so enabling them does not ensure that they are used. Studying their effects therefore requires changing one mechanism at a time within a shared execution loop, with the model, task, underlying tools and budget held fixed, and in a setting where the mechanism actually fires.

To address these challenges, we present `MedicalHarness`, a controlled study of models and agent harnesses on medical tasks (Figure 1). We first build `MedicalHarnessBench` to evaluate agents across four capability domains. Using this benchmark, we compare model–harness pairs under matched execution conditions and analyze their outcomes and execution traces. To study individual mechanisms, we build `MH-Lab`, which switches off context management, planning or tool exposure one at a time within a shared execution loop. We find that the harness and its interaction with the model account for a substantial share of outcome variance, and that no single harness is best across models and tasks. Harnesses also usually differ more in how often they answer than in their scores on commonly answered questions, and hiding tools behind a search interface, meant to save context, can instead cost tokens or accuracy. Our contributions are:

- **Benchmark.** We introduce `MedicalHarnessBench`, a benchmark of 107 executable medical tasks spanning clinical calculation, long-context retrieval, clinical data workflows and browser interaction, four domains that each test a different harness capability.
- **Controlled comparison study.** We evaluate five open-weight models under five harnesses with three seeds (8,025 episodes), changing only the harness within a comparison. We pair a model–harness variance analysis with answer-production and execution diagnostics.
- **Controlled mechanism study.** We build `MH-Lab` to test context management, planning and tool exposure through one-at-a-time interventions within a shared execution loop.

2 Related Work

Medical agent benchmarks. MedAgentBench evaluates task execution in a virtual EHR (Jiang et al., 2025), and AgentClinic simulates interactive clinical decision-making (Schmidgall et al., 2026). HealthAgentBench assembles executable environments across clinical data, imaging and

trial matching (Liu et al., 2026), HealthAdminBench and χ-Bench evaluate agents on healthcare administration and managed-care workflows (Bedi et al., 2026; Chen et al., 2026), MedMCP-Calc exposes medical calculators to agents through MCP (Zhu et al., 2026), and MedMemoryBench tests memory over long patient histories (Wang et al., 2026). These benchmarks measure what an agent can do in a medical environment. HealthAgentBench and χ-Bench also compare harnesses, but `MedicalHarness` is dedicated to this comparison and makes it more comprehensive and better controlled. It runs every model under every harness on more tasks, spanning four kinds that each test a different harness capability, serves all models locally under one setup so that only the harness changes, and adds controlled experiments on individual mechanisms.

Harnesses and their mechanisms. A harness, also called a scaffold, changes what a fixed model can do (Yang et al., 2024; Wang et al., 2025b), yet general agent benchmarks such as AgentBench and WebArena are built mainly to compare models (Liu et al., 2024; Zhou et al., 2024). Recent work evaluates the harness itself. Harness-Bench measures harness effects across models (Yao et al., 2026), Zhang et al. (2026b) model the harness as the controller of a closed loop around the model and propose disclosure standards with a variance decomposition, Li et al. (2026) organize harness components into seven families, and Fan et al. (2026) ablate components for coding agents. Controlled comparisons outside medicine find that harness effects vary with the task and the model, from large cost but small accuracy differences on terminal coding (Vats & Golev, 2026) to accuracy gaps of up to 28 points on GAIA (Starace, 2026), and Zhang et al. (2026a) show that a score reflects the model only if the harness leaves the decisions to it. `MedicalHarness` also separates how often agents answer from their scores on commonly answered questions, and uses `MH-Lab` to switch off three mechanisms that product harnesses share, context management (Packer et al., 2023), planning and tool exposure (Qin et al., 2024), one at a time within one execution loop.

3 Problem Formulation

We first define what a harness controls and what we hold fixed when comparing harnesses (Section 3.1). We then separate variation in benchmark scores into model, harness, interaction and repeated-run components (Section 3.2).

3.1 Agent harness

A harness decides what the model sees, how its actions are executed, and when the episode ends.

Definition 3.1 (Agent harness)**.** An agent harness is a triple of procedures $h = (C_h, U_h, T_h)$ that runs a model m on a task x. From the task's initial state s_0, while $T_h(s_t) = 0$ it repeats

$$c_t = C_h(s_t), \qquad a_t \sim \pi_m(\cdot \mid c_t), \qquad (o_t, s_{t+1}) \sim U_h(\cdot \mid s_t, a_t; x).$$

Here $\pi_m(\cdot \mid c_t)$ is the model's response distribution. C_h builds the context, U_h executes the action in the task environment, collects the observation o_t and updates the state, and T_h decides when to stop. The state contains the history, files, plan, environment state and remaining budget. An agent is the pair (m, h), and the task's oracle scores its trajectory τ as $Y = R_x(\tau) \in [0, 1]$.

C_h selects the instructions, tool descriptions and history shown to the model. State updates in U_h include tool routing, output checks and any auxiliary model calls, such as summarization, that prepare the next turn. In our study these calls use the same model π_m, are recorded in the trajectory and count toward the episode's budget.

This view guides both comparisons. When swapping harnesses, we hold the tasks and environments, model checkpoint, serving stack, context window, episode deadline and run seed set fixed. Each harness runs as released, so its prompts, tool formats, retry and stopping rules, output limits and sampling fields belong to the harness being compared. When removing a mechanism from `MH-Lab`, we change the procedures that implement it and keep the rest fixed (Section 6.4). Context management and planning change what C_h builds and U_h keeps. Tool exposure changes both the tool descriptions in C_h and the routing and feedback in U_h.

3.2 MODEL–HARNESS DECOMPOSITION

To measure how much the harness matters, we split variation in benchmark scores into four parts: the model, the harness, their interaction, and variation across repeated runs. The interaction captures whether a harness suits some models better than others. Repeated runs of the same model–harness pair can also produce different scores. We report this variation separately.

Definition 3.2 (Model–harness decomposition)**.** Within an environment, let every model $m \in \mathcal{M}$ run under every harness $h \in \mathcal{H}$ with every seed $\sigma \in \mathcal{S}$, and let $\bar{y}_{mh\sigma}$ be the mean outcome over the environment's tasks. We decompose

$$\bar{y}_{mh\sigma} = \mu + \alpha_m + \beta_h + \gamma_{mh} + \varepsilon_{mh\sigma}. \tag{1}$$

Here μ is the grand mean. The model effect α_m is the model's mean score across harnesses and seeds minus μ. The harness effect β_h is the harness's mean across models and seeds minus μ. The interaction γ_{mh} is the pair's mean score minus $(\mu + \alpha_m + \beta_h)$, and $\varepsilon_{mh\sigma}$ is a run's deviation from its pair's mean.

The total score variation is $\mathrm{SS}_T = \sum_{m,h,\sigma}(\bar{y}_{mh\sigma} - \mu)^2$. Summing each component's squared values over all (m, h, σ) gives SS_M, SS_H, SS_{MH} and SS_E, respectively. The cross terms cancel in this balanced design, giving

$$\mathrm{SS}_T = \mathrm{SS}_M + \mathrm{SS}_H + \mathrm{SS}_{MH} + \mathrm{SS}_E, \tag{2}$$

with the expanded formulas in Section E.

When $\mathrm{SS}_T > 0$, we report each component as a share of the total score variation. The *harness share*, $(\mathrm{SS}_H + \mathrm{SS}_{MH})/\mathrm{SS}_T$, combines the harness effect and the model–harness interaction. The share $\mathrm{SS}_E/\mathrm{SS}_T$ measures run-to-run variation.

We also ask whether scores vary more when changing the harness under a fixed model or changing the model under a fixed harness. We measure these as harness variance (HV) and model variance (MV) (Zhang et al., 2026b): $\mathrm{HV}(m) = \mathrm{Var}_h(\bar{y}_{mh\cdot})$ and $\mathrm{MV}(h) = \mathrm{Var}_m(\bar{y}_{mh\cdot})$, where the dot averages over seeds. We use population variances over the selected harnesses or models, dividing by $|\mathcal{H}|$ or $|\mathcal{M}|$, respectively.

Proposition 3.1 (Harness and model variance)**.** *In a balanced model × harness × seed design, let* $N = |\mathcal{M}||\mathcal{H}||\mathcal{S}|$*. Averaging* HV *over models and* MV *over harnesses gives*

$$\overline{\mathrm{HV}} = \frac{\mathrm{SS}_H + \mathrm{SS}_{MH}}{N}, \qquad \overline{\mathrm{MV}} = \frac{\mathrm{SS}_M + \mathrm{SS}_{MH}}{N}.$$

Consequently, $\overline{\mathrm{HV}} > \overline{\mathrm{MV}}$ *if and only if* $\mathrm{SS}_H > \mathrm{SS}_M$*.*

Proof. See Section E.

The interaction enters both averages. Which average is larger therefore depends only on the two main effects. The decomposition reports the interaction separately.

4 MedicalHarnessBench

4.1 ENVIRONMENTS

MedicalHarnessBench covers four capability domains: clinical calculation, long-context retrieval, clinical data workflows and browser interaction. Its 107 tasks span clinical information processing and hospital administration (Figure 2).

Clinical calculation (MedTool, 32 tasks). Each task provides a clinician's request and a patient's record stored in a SQLite database. The output is an answer file listing the required calculators and their results. The agent must identify which calculations apply, retrieve their inputs from records containing distractors, and follow dependencies or conditional branches between calculations to produce 2–10 required outputs.

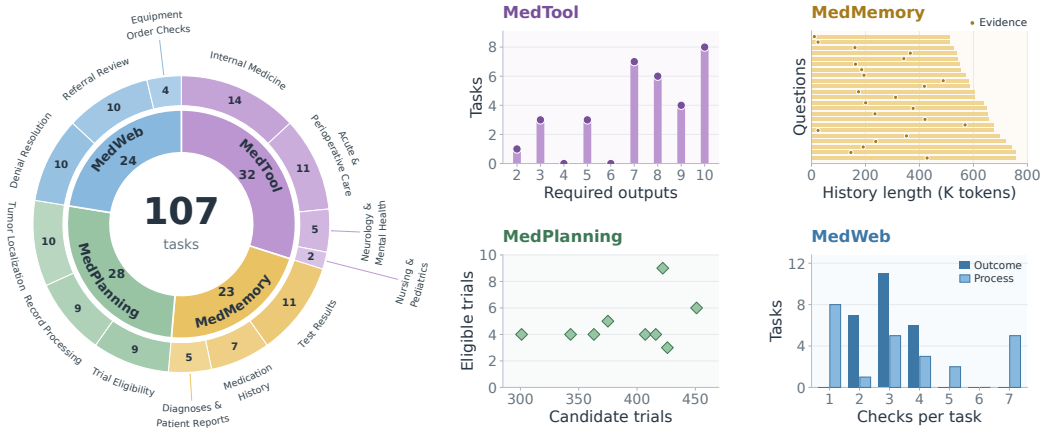


Figure 2: **Task composition and characteristics. Left:** the 107 task instances grouped by clinical domain or workflow (Table 4). **Right:** required `MedTool` outputs, one `MedMemory` history per bar with source-evidence session starts marked by dots (Qwen3.6 tokens), candidate and eligible trials for nine `MedPlanning` trial-eligibility tasks, and separate distributions of `MedWeb` outcome and process check counts.

Long-context retrieval (`MedMemory`, 23 tasks). The input is a question and a patient history stored as files, spanning 70–100 dialogue sessions. The output is a short factual answer, such as a dose, lab value or diagnosis. The history exceeds a single context window, and the relevant evidence can lie dozens of sessions before the question. The agent must locate that evidence in the files and bring it into context to answer correctly.

Clinical data workflows (`MedPlanning`, 28 tasks). Each task provides an instruction and a workspace of clinical data, and requires structured result files. Tasks include trial eligibility screening, EHR processing and tumor localization on whole-slide images. The challenge is to complete a workflow across many documents, records or image regions. Trial screening, for example, requires finding all eligible trials among hundreds of candidates.

Browser interaction (`MedWeb`, 24 tasks). The input is a workflow request, such as resolving an insurance denial, and a browser session across an EMR, a payer portal and a fax portal. The output is the updated application state. The agent must inspect records, navigate between applications and carry out the required updates, keeping track of which steps remain unfinished.

4.2 Scoring

Outcome. Each task receives a score in $[0, 1]$. We average scores over tasks within each environment. `MedTool` multiplies calculator-selection F1 by the fraction of all required outputs that are correct. The other three environments use binary outcomes. `MedMemory` matches normalized answers against accepted aliases. `MedPlanning` applies the task's own checks, and `MedWeb` requires every outcome check on the final application state to pass.

Task diagnostics (Proc.). Alongside outcome, Table 1 reports calculator-selection F1 for `MedTool`, reference-set recall for `MedPlanning` and the share of process checks passed for `MedWeb`. For `MedMemory`, which has no intermediate environment checks, an LLM judge scores a condensed interaction trace reconstructed from the proxy logs (Section 6.1). Its score averages three dimensions: appropriate tool use, coherent and evidence-grounded actions, and recovery from tool failures. Section C gives the scoring rules and trace extraction procedure.

5 MH-Lab

To study individual harness mechanisms, we build MH-Lab (MedicalHarness Lab), a minimal harness with a shared execution loop and three component switches. Implemented in LangGraph (LangChain, 2025), it follows a ReAct-style loop (Yao et al., 2023) that constructs the model's context, calls the model, executes its tool calls, and updates the state for the next turn (Figure 3).

We focus on three common mechanisms that govern how agents retain information, track unfinished work, and access tools. Medical tasks often accumulate evidence across records, require dependent steps, and involve repeated interactions with files or clinical applications. Context management determines which parts of a growing history remain available to the model. Planning maintains an explicit record of progress and remaining steps. Tool exposure determines how the model discovers and invokes available tools. Each mechanism can be varied within the same loop while preserving the task environment and underlying task tools.

Context management. As the context fills, the same model summarizes older history for subsequent requests while retaining recent interactions. Summary calls count toward the episode's budget. Switching it off retains the accumulated history without summarization. **Planning.** An `update_plan` tool and plan-related instructions support tracking progress. The current plan appears on every turn to make completed and unfinished steps explicit. Switching it off removes the tool, instructions, reminders and displayed plan. **Tool exposure.** Task tools are exposed with their names, descriptions and argument schemas. Switching off direct exposure keeps the same tools and executors but requires discovery before calls through `tool_call`.

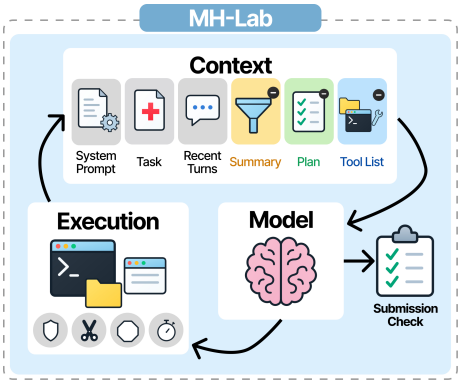


Figure 3: `MH-Lab`. A shared execution loop with three component switches.

For each comparison, we change one mechanism while keeping the model, task instructions, underlying task tools, stopping rules, and episode budget fixed. We record model inputs, tool calls, plan updates, summary events, and termination reasons to relate changes in outcomes to changes in execution. Implementation details are provided in Section H.

6 Experiments

Our experiments address three research questions. We first measure performance variation across models and harnesses (RQ1), then examine how harnesses differ in task execution (RQ2). Finally, we use `MH-Lab` to test the effects of individual mechanisms with two models on all four environments (RQ3).

6.1 Experimental setup

Models and harnesses. We evaluate five open-weight models spanning 26B to 120B parameters: `Gemma4-26B-A4B`, `Qwen3.6-35B-A3B`, `GPT-OSS-120B`, `Nemotron-3.5-30B-A3B` and `GLM-4.7-Flash`. The main table shows the first three, with the other two in Table 9. Each model is served locally by vLLM (Kwon et al., 2023) with one checkpoint and context window, shared by five harnesses through their custom-provider interfaces: OpenClaw (OpenClaw contributors, 2026), Hermes (Nous Research, 2026), ZeroClaw (ZeroClaw Labs, 2026), Codex (OpenAI, 2026) and Claude Code (Anthropic, 2026). Harnesses keep their shipped settings apart from the changes listed in Section D.

Execution. An episode is one run of a task by a model–harness pair under one seed, with a one-hour wall-clock limit, in a sandbox that hides graders, gold answers and other episodes. A proxy on every model call records requests, responses and token counts, forming the execution traces of Section 6.3.

Measures. Besides outcome and task diagnostics (Section 4.2), we report the answer rate on `MedTool` and `MedMemory`, the fraction of episodes with a scorable answer in the answer file or final reply, and token use, the input and output tokens of all model calls in millions per episode (MTok).

Protocol. Every model–harness pair runs all 107 tasks with seeds 42, 43 and 44, for 8,025 episodes. The mechanism study runs `Qwen3.6-35B-A3B` and `GPT-OSS-120B` in `MH-Lab` on the same tasks and seeds, with a 65,536-token input allowance per request. Tables report means and SDs over seeds, and paired comparisons bootstrap tasks after averaging over seeds, giving 95% intervals (Card et al., 2020; Miller, 2024; Madaan et al., 2024; Heineman et al., 2025).

Table 1: **Outcome and process under a fixed model.** Only the harness varies within a block. Mean ± SD over seeds. **Proc.**: calculator selection F1 (`MedTool`), recall of the reference set on the 17 `MedPlanning` tasks that have one, share of process checks passed (`MedWeb`). †`MedMemory` records only the answer, so its Proc. is the LLM judge's score (section C). **MTok**: million tokens per episode. *Rank*: mean outcome rank within a block. **Bold** and underlined: best and second best Outc. and Proc. per capability within a block. Rows are ordered by mean rank over all five models, worst first.

Harness	MedTool			MedMemory			MedPlanning			MedWeb			Rank
	Outc.↑	Proc.↑	MTok↓	Outc.↑	Proc.↑†	MTok↓	Outc.↑	Proc.↑	MTok↓	Outc.↑	Proc.↑	MTok↓	
Gemma4-26B-A4B													
ZeroClaw	0.08±0.01	0.37±0.01	0.11	0.49±0.03	0.32±0.01	0.48	0.01±0.02	0.20±0.02	1.79	0.03±0.02	0.20±0.05	6.67	3.75
OpenClaw	0.10±0.01	0.45±0.04	0.52	0.57±0.08	0.35±0.05	1.04	0.00±0.00	0.20±0.01	4.51	0.04±0.04	**0.38±0.06**	11.95	3.00
Hermes	**0.13±0.04**	0.46±0.05	0.11	0.57±0.12	0.23±0.07	0.78	0.00±0.00	**0.32±0.04**	1.04	0.01±0.02	0.15±0.13	4.02	3.75
Codex	0.09±0.01	**0.47±0.04**	0.06	0.77±0.05	**0.48±0.07**	0.73	0.01±0.02	0.31±0.04	2.30	**0.11±0.02**	0.29±0.05	20.04	2.25
Claude Code	0.06±0.02	0.28±0.01	0.29	**0.78±0.04**	0.36±0.14	1.23	**0.04±0.04**	0.29±0.07	4.75	0.10±0.10	0.30±0.02	9.66	2.25
Qwen3.6-35B-A3B													
ZeroClaw	0.08±0.03	0.25±0.06	0.41	0.67±0.07	0.30±0.07	0.48	0.02±0.02	0.19±0.05	4.03	0.32±0.09	0.45±0.02	6.73	5.00
OpenClaw	0.10±0.01	0.42±0.03	0.47	0.83±0.09	0.44±0.04	0.83	**0.10±0.02**	0.47±0.03	2.84	0.46±0.04	0.47±0.04	7.93	2.50
Hermes	**0.13±0.03**	0.45±0.05	0.24	0.78±0.08	0.55±0.04	0.41	0.08±0.05	**0.50±0.06**	1.69	0.33±0.07	0.42±0.07	7.83	3.00
Codex	0.12±0.01	**0.50±0.01**	0.23	**0.86±0.03**	**0.65±0.04**	0.62	0.06±0.05	0.45±0.01	2.13	0.42±0.04	0.47±0.03	5.08	2.50
Claude Code	0.10±0.02	0.46±0.03	0.25	0.86±0.10	0.42±0.03	0.62	0.10±0.04	0.45±0.03	2.68	**0.47±0.05**	**0.48±0.01**	8.11	2.00
GPT-OSS-120B													
ZeroClaw	0.12±0.02	0.46±0.03	0.10	0.36±0.03	0.26±0.07	0.29	0.01±0.02	0.11±0.06	0.63	0.00±0.00	0.27±0.02	1.24	4.25
OpenClaw	0.12±0.01	0.43±0.02	0.24	0.51±0.05	0.32±0.03	0.51	0.02±0.02	0.20±0.04	0.94	0.03±0.02	0.22±0.03	1.77	3.75
Hermes	0.12±0.01	0.49±0.03	0.10	0.68±0.07	0.22±0.01	0.37	0.04±0.04	0.38±0.03	0.61	0.00±0.00	0.06±0.03	0.22	2.75
Codex	0.13±0.02	0.44±0.06	0.22	0.68±0.07	**0.38±0.04**	0.51	0.04±0.00	**0.42±0.08**	0.63	0.00±0.00	0.00±0.00	0.10	3.25
Claude Code	**0.16±0.02**	**0.52±0.02**	0.30	**0.80±0.09**	0.35±0.06	0.47	**0.05±0.04**	0.28±0.03	1.55	**0.26±0.09**	**0.33±0.09**	4.09	1.00

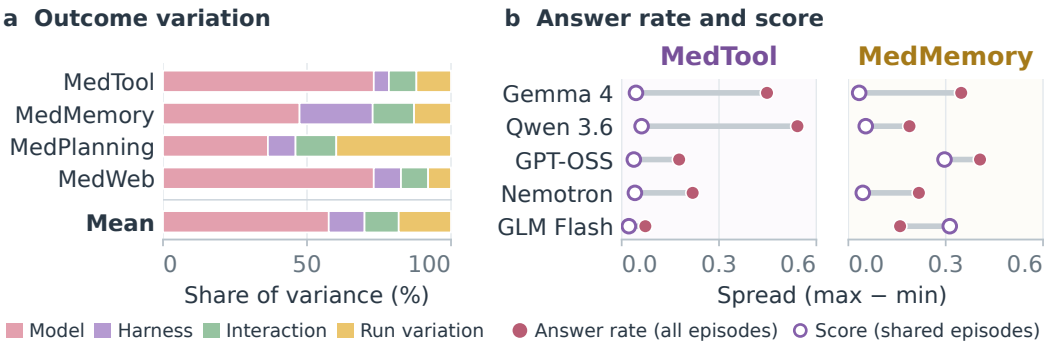


Figure 4: **Outcome variation and answer production.** Five models, five harnesses and three seeds. **(a)** Variance shares of per-seed mean outcomes (eq. (1)). Mean weights the four environments equally. **(b)** Per-model spreads (maximum minus minimum) across harnesses in answer rate over all episodes and mean score on matched task–seed pairs answered by every harness. Score is the task outcome (Section 4.2). Answers in either the file or final reply count.

6.2 RQ1: How much does the harness matter?

Table 1 compares harnesses under the three main models. Figure 4a splits the outcome variance of all five models into four shares, for the model, the harness, their interaction and run-to-run variation across seeds (Equation (1)). We ask how large the harness share is and whether the best harness stays the same across models and tasks. Two findings emerge.

The model explains the largest share, but the harness matters differently across tasks. Averaged over the four environments, the model accounts for 58% of the variance, the harness for 12%, their interaction for 12% and run-to-run variation across seeds for the remaining 18%. The harness and its interaction thus account for about a quarter, but the shares differ by environment. On `MedMemory` the harness main effect reaches 25%, and 40% with the interaction, while on `MedPlanning`, where success is rare, run-to-run variation accounts for 40%, comparable to the model's 36%. Proposition 3.1 compares the two directions directly: across all five models, HV/MV ranges from 0.18 on `MedTool` to 0.64 on `MedMemory`, so changing the model moves scores more than changing the harness everywhere. Among the three main models, HV/MV on `MedMemory` rises to 1.45, the only case where the harness main effect exceeds the model's (Section E).

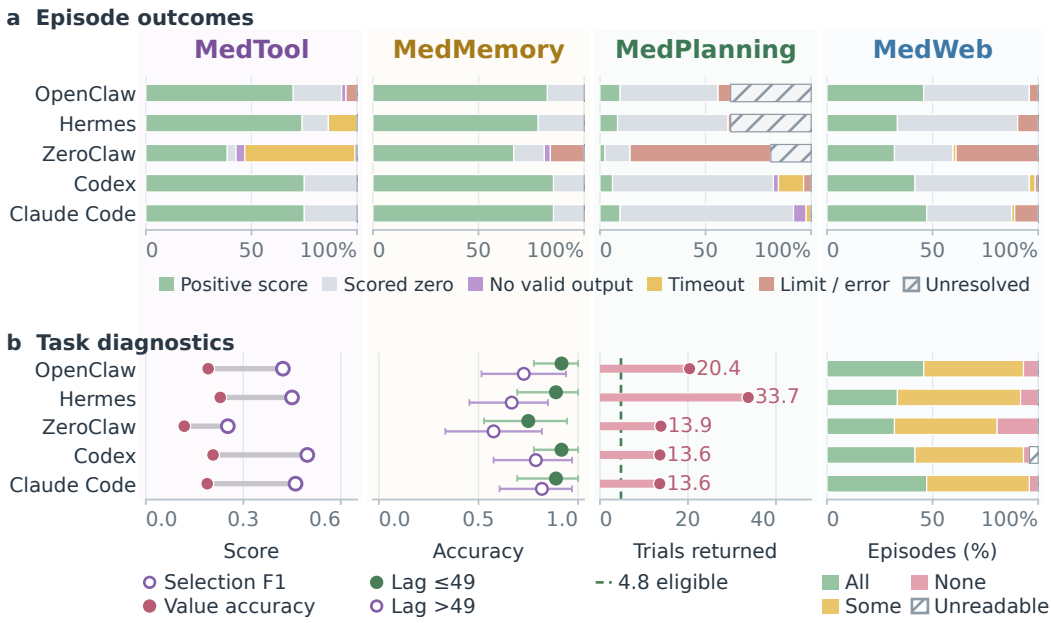


Figure 5: **Episode outcomes and task diagnostics under** `Qwen3.6-35B-A3B`, over three seeds. **(a)** Episodes grouped by score, ending and output evidence. Positive score includes partial credit. **(b)** `MedTool`: selection F1 and value accuracy. `MedMemory`: accuracy by evidence lag in sessions, with 95% task-bootstrap intervals. `MedPlanning`: mean trials returned versus eligible on nine trial tasks, while (a) covers all `MedPlanning` tasks. `MedWeb`: episodes passing all, some or no outcome checks, with unreadable states separate.

No harness is best across models and tasks. The leader differs by model even within one environment. On `MedTool`, Hermes leads three models, Claude Code leads `GPT-OSS-120B`, and Codex leads `Nemotron-3.5-30B-A3B`. Leads are also small relative to uncertainty from the sampled tasks. Only 2 of the 20 model–environment pairs have a 95% interval for the leader's gap over the runner-up that excludes zero, and both are Claude Code under `GPT-OSS-120B`, on `MedMemory` and `MedWeb`. The clearest leads thus hold for a single model. Table 10 gives all rankings and intervals.

6.3 RQ2: How do harnesses differ in task execution?

We examine whether episodes end with an answer and what makes each environment difficult.

Harnesses differ in whether an episode ends with an answer. ❶ On `MedTool` and `MedMemory`, Figure 4b compares the spread across harnesses in answer rate, over all episodes, with the spread in score, over the task–seed pairs every harness answered. The answer-rate spread is the larger in 9 of the 10 model–environment pairs. Scores on these shared questions differ by only 0.02–0.06 in eight pairs, while answer rates differ by up to 0.54. The exception is `GLM-4.7-Flash` on `MedMemory` (Section F.2). ❷ Unsuccessful episodes end in different ways. Under `Qwen3.6-35B-A3B` (Figure 5a), ZeroClaw reaches the wall-clock limit in 50 of 96 `MedTool` episodes, while Codex and Claude Code usually produce a scorable output. A single outcome score hides these differences, so we report answer rates alongside it (Section G decomposes the variance).

Each environment exposes a different difficulty. Figure 5b shows the task diagnostics. ❶ On `MedTool`, calculator-selection F1 is about twice value accuracy across harnesses, so choosing the right calculators does not ensure correct values. ❷ On `MedMemory`, questions with more distant evidence have lower accuracy under every harness. ❸ On the `MedPlanning` trial tasks, agents return 13.6 to 33.7 trials on average against 4.8 eligible, three to seven times too many. ❹ On `MedWeb`, fewer than half of episodes pass every outcome check under any harness, and about half pass only some. These diagnostics separate selecting tools, retrieving evidence, filtering candidates and completing workflows, which one score would compress.

Mechanisms inside the product harnesses. At the native windows, summarization fires in only 2–5% of episodes, which caps its share of these gaps at that fraction. On `MedTool` and `MedMemory`, the gaps come mainly from answer delivery, and switching the matching mechanism inside a product

Table 2: **Removing one mechanism from MH-Lab.** 65,536-token input budget, seeds 42/43/44. *full*: outcome and million tokens per episode, mean ± SD over seeds. Other rows: paired change in outcome (± SD of the per-seed change) and token ratio to *full*. Table 13 gives the intervals.

Condition	MedTool		MedMemory		MedPlanning		MedWeb	
	Outc.↑	MTok↓	Outc.↑	MTok↓	Outc.↑	MTok↓	Outc.↑	MTok↓
Qwen3.6-35B-A3B								
full	0.10±0.02	0.23±0.01	0.88±0.03	0.43±0.12	0.12±0.05	1.62±0.26	0.44±0.02	5.96±0.38
− context management	+0.01±0.02	×1.1±0.0	−0.12±0.03	×0.5±0.2	−0.04±0.06	×0.4±0.1	−0.39±0.09	×0.3±0.0
− planning	−0.02±0.04	×1.0±0.1	−0.04±0.04	×0.8±0.4	+0.01±0.08	×0.9±0.2	0.00±0.07	×0.7±0.2
− tool exposure	0.00±0.02	×3.0±1.6	−0.14±0.05	×2.4±0.7	+0.01±0.05	×2.6±0.5	0.00±0.07	×0.8±0.1
GPT-OSS-120B								
full	0.14±0.03	0.28±0.07	0.72±0.03	0.66±0.13	0.06±0.02	0.99±0.38	0.18±0.08	7.70±1.58
− context management	−0.01±0.04	×1.0±0.3	+0.06±0.05	×0.8±0.1	+0.02±0.02	×0.6±0.3	−0.17±0.14	×0.4±0.1
− planning	−0.02±0.04	×0.4±0.1	−0.01±0.03	×0.7±0.2	+0.02±0.02	×0.9±0.4	−0.18±0.09	×0.2±0.1
− tool exposure	−0.04±0.06	×3.2±1.8	−0.06±0.07	×1.3±0.4	−0.04±0.00	×1.9±0.7	−0.15±0.14	×1.0±0.0

harness moves outcomes accordingly: ZeroClaw's submission check raises Qwen3.6-35B-A3B's MedMemory score from 0.42 to 0.51, and turning off Hermes's tool search lifts Qwen3.6-35B-A3B from 0.17 to 0.50 on 12 MedWeb tasks (Section I).

6.4 RQ3: Which mechanisms matter, and when?

Table 2 compares the full MH-Lab with variants that disable one mechanism at a time, keeping task settings and other execution limits fixed (Section 5). Its input allowance is below the native windows, where summarization is rare (Section I), so that context management fires.

Context management keeps long trajectories running. Without summaries, accumulated history overflows the input budget in 67 of 72 MedWeb episodes for Qwen3.6-35B-A3B and 48 of 65 for GPT-OSS-120B (Table 14). Outcomes fall by 0.39 and 0.17, respectively. Lower token use reflects early termination. On MedPlanning, binary success stays rare, but reference-set recall falls by 0.29 for Qwen3.6-35B-A3B and 0.16 for GPT-OSS-120B (Table 13), so summaries preserve partial progress.

Planning helps some models persist. Without the plan tool and displayed plan, GPT-OSS-120B's median turns on the 60 paired MedWeb episodes fall from 176.5 to 63, and outcome and process scores drop by 0.18 and 0.19. Qwen3.6-35B-A3B shows no clear outcome change on MedWeb. On MedTool, however, its episodes without the required output rise from 24 to 47 of 96 and timeouts double. Similar scores can conceal changes in whether the agent finishes.

Tool interfaces must fit the model's calling behavior. With tools behind search and `tool_call`, models discover a tool but invoke it by name, causing rejected calls. This occurs in 95 of 96 MedTool episodes for Qwen3.6-35B-A3B, which uses three times as many tokens at the same outcome. GPT-OSS-120B loses outcome on all four environments, most on MedWeb (−0.15). These failures arise when discovery must be followed by `tool_call`, so a discovery interface should make the found tool directly callable, as Claude Code does.

7 Conclusion

We introduced MedicalHarnessBench and used it to measure how outcomes vary with the model and harness under controlled conditions. The model explains the largest share, but the harness and its interaction with the model account for about a quarter of the variance, and no single harness is best across models and tasks. On clinical calculation and long-context retrieval, harnesses usually differ more in answer rate than in scores on questions answered by every harness. A mechanism changes outcomes only when it fires and when its interface fits the model. Inside MH-Lab, context management decides whether long traces survive and planning sustains longer browser workflows for GPT-OSS-120B. Hiding tools behind a search interface can cost tokens or accuracy when models call the tools by name. Medical agent results should therefore report the model–harness pair and, when an answer is required, its answer rate alongside the score. Section A turns these findings into design directions for medical agent harnesses.

AI USE STATEMENT

We used Large Language Models (ChatGPT/Claude/Gemini) for grammatical correction and editing of this manuscript, and as coding assistants for parts of the evaluation infrastructure, the analysis scripts and MH-Lab. Separately, language models served as instruments of the study: `gpt-4o-mini` scored the process rubric (Section C), and non-target models reviewed candidate tasks during verification (Section B). The authors reviewed all code and verified every reported number against the archived episode records. All technical contributions, analyses and scientific conclusions are the authors' own responsibility.

ETHICS STATEMENT

MedicalHarness is a research study of LLM agents on medical tasks. The agents evaluated here are research systems, and their scores do not indicate fitness for clinical use, which would require clinical validation and expert review. All tasks derive from public benchmarks built on synthetic or de-identified data: MedMCP-Calc (MIT), HealthAgentBench (MIT, only tasks that need no credentialed data), HealthAdminBench (Apache-2.0, synthetic portal records) and MedMemoryBench (synthetic patient personas seeded from de-identified cases). No task uses credentialed clinical data. Agents ran on locally served models, and only the process judge received condensed episode traces through an external API, all from synthetic or public tasks. Because the MedMemoryBench data license is stated inconsistently upstream, we release item identifiers, hashes and the construction code rather than the chart text.

REPRODUCIBILITY STATEMENT

Every model is an open-weight checkpoint served locally with a fixed context window (Section 6.1), and every configuration was run with seeds 42, 43 and 44. Section B describes how each environment was built and verified, Section D lists the harness versions and the defaults we changed, Section C gives the grading rules, and Section H specifies MH-Lab. Code, harness configurations and task manifests are available at `https://anonymous.4open.science/r/MedicalHarness/`, with MedMemoryBench items released as identifiers and hashes together with their construction code.

REFERENCES

Anthropic. Claude Code. `https://github.com/anthropics/claude-code`, 2026.

Suhana Bedi, Ryan Welch, Ethan Steinberg, Michael Wornow, Taeil Matthew Kim, Haroun Ahmed, Peter Sterling, Bravim Purohit, Qurat Akram, Angelic Acosta, Esther Nubla, Pritika Sharma, Michael A. Pfeffer, Sanmi Koyejo, and Nigam H. Shah. HealthAdminBench: Evaluating computer-use agents on healthcare administration tasks, 2026. URL `https://arxiv.org/abs/2604.09937`.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Haolin Chen, Deon Metelski, Leon Qi, Tao Xia, Joonyul Lee, Steve Brown, Kevin Riley, Frank Wang, T. Y. Alvin Liu, Hank Capps, Zeyu Tang, Xiangchen Song, Lingjing Kong, Fan Feng, Tianyi Zeng, Zhiwei Liu, Zixian Ma, Hang Jiang, Fangli Geng, Yuan Yuan, Chenyu You, Qingsong Wen, Hua Wei, Yanjie Fu, Yue Zhao, Carl Yang, Biwei Huang, Kun Zhang, Caiming Xiong, Sanmi Koyejo, Eric P. Xing, Philip S. Yu, and Weiran Yao. CHI-Bench: Can AI agents automate end-to-end, long-horizon, policy-rich healthcare workflows?, 2026. URL `https://arxiv.org/abs/2605.16679`.

Run-Ze Fan, Zihao Zhang, Simin Ma, Yebowen Hu, Shouju Wang, Kaiqiang Song, Fei Liu, Hamed Zamani, and Xiaoyang Wang. An empirical study of harness design for coding agents, 2026. URL `https://arxiv.org/abs/2609.20804`.

David Heineman, Valentin Hofmann, Ian Magnusson, Yuling Gu, Noah A. Smith, Hanna Hajishirzi, Kyle Lo, and Jesse Dodge. Signal and noise: A framework for reducing uncertainty in language model evaluation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17073–17114. Curran Associates, Inc., 2025. doi: 10.52202/085713-0576. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/18b3b2947174ae9b1a8fc91ff89f4eb6-Paper-Conference.pdf.

Yixing Jiang, Kameron C. Black, Gloria Geng, Danny Park, James Zou, Andrew Y. Ng, and Jonathan H. Chen. MedAgentBench: A virtual EHR environment to benchmark medical LLM agents. *NEJM AI*, 2(9):AIdbp2500144, 2025. doi: 10.1056/AIdbp2500144. URL https://doi.org/10.1056/AIdbp2500144.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

LangChain. LangGraph. https://github.com/langchain-ai/langgraph, 2025.

Jinzhe Li, Yuan Wu, and Yi Chang. Harness engineering for LLM agents: A survey of harness component taxonomy, evaluation, and model–harness coevolution. Preprints.org, 2026. URL https://doi.org/10.20944/preprints202606.2203.v1. Preprint.

Qianchu Liu, Sheng Zhang, Guanghui Qin, Jeya Maria Jose Valanarasu, Maximilian Rokuss, Mingyu Lu, Timothy Ossowski, Juan Manuel Zambrano Chaves, Cliff Wong, Peniel Argaw, Yashna Hasija, Mu Wei, Wen-wai Yim, Qin Liu, Zilin Jing, Jason Entenmann, Naoto Usuyama, Tristan Naumann, and Hoifung Poon. HealthAgentBench: A unified benchmark suite of realistic agentic healthcare environments for challenging frontier AI agents, 2026. URL https://arxiv.org/abs/2606.31179.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks, 2024. URL https://arxiv.org/abs/2406.10229.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations, 2024. URL https://arxiv.org/abs/2411.00640.

Nous Research. Hermes Agent. https://github.com/NousResearch/hermes-agent, 2026.

OpenAI. Codex CLI. https://github.com/openai/codex, 2026.

OpenClaw contributors. OpenClaw. https://github.com/openclaw/openclaw, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Samuel Schmidgall, Rojin Ziaei, Carl Harris, Ji Woong Kim, Eduardo Pontes Reis, Jeffrey Jopling, and Michael Moor. AgentClinic: a multimodal benchmark for tool-using clinical AI agents. *npj Digital Medicine*, 9(1):499, 2026. doi: 10.1038/s41746-026-02674-7. URL https://www.nature.com/articles/s41746-026-02674-7.

Jason Starace. Scaffold effects on GAIA: A controlled comparison, 2026. URL https://arxiv.org/abs/2606.08529.

Naman Vats and Oleg Golev. The scaffold effect in coding agents: Harness choice as a hidden variable in coding-agent evaluation, 2026. URL https://arxiv.org/abs/2607.22585.

Wenxuan Wang, Zizhan Ma, Zheng Wang, Chenghan Wu, Jiaming Ji, Wenting Chen, Xiang Li, and Yixuan Yuan. A survey of LLM-based agents in medicine: How far are we from baymax? In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025a. URL https://aclanthology.org/2025.findings-acl.539/.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a4b6ad6b48850c0c331d1259fc66a69c-Paper-Conference.pdf.

Yihao Wang, Haoran Xu, Renjie Gu, Yixuan Ye, Xinyi Chen, Xinyu Mu, Yuan Gao, Chunxiao Guo, Peng Wei, Jinjie Gu, Huan Li, Ke Chen, and Lidan Shou. MedMemoryBench: Benchmarking agent memory in personalized healthcare, 2026. URL https://arxiv.org/abs/2605.11814.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. HarnessBench: Measuring harness effects across models in realistic agent workflows, 2026. URL https://arxiv.org/abs/2605.27922.

ZeroClaw Labs. ZeroClaw. https://github.com/zeroclaw-labs/zeroclaw, 2026.

Yonghong Zhang, Shadi Motaali, Vu Phong Dinh, Avin Piroutiniya, Jorge E. López de Vergara, Luis de Pedro, Ricardo Correia, Isabel M. Parra, and Yong Xie. The double measurement confound in agent benchmarks: De-scaffolding, ground-truth scoring, and reliability beyond the mean, 2026a. URL https://arxiv.org/abs/2609.09218.

Yunbei Zhang, Janet Wang, Yingqiang Ge, Weijie Xu, Jihun Hamm, and Chandan K. Reddy. Stop comparing LLM agents without disclosing the harness, 2026b. URL https://arxiv.org/abs/2605.23950.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

Yakun Zhu, Yutong Huang, Shengqian Qin, Zhongzhen Huang, Shaoting Zhang, and Xiaofan Zhang. MedMCP-Calc: Benchmarking LLMs for realistic medical calculator scenarios via MCP integration. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4845–4873, 2026.

A Design implications for medical agent harnesses

Our execution diagnostics and mechanism experiments (Sections 6.3 and 6.4) suggest three design directions for harnesses that carry out medical tasks.

Preserve clinical evidence. On `MedMemory`, accuracy falls as supporting evidence lies further back in the record. The context-management experiments also show that summaries can keep long trajectories running when history exceeds the input budget. These findings motivate context management that controls length while keeping relevant evidence accessible. A task-specific evidence record could retain key facts with their dates and source locations, allowing the agent to revisit the original record when needed. For example, after compression, a medication entry should still indicate when it was recorded and whether treatment was ongoing or had ended.

Ground plans in clinical criteria. On trial-screening tasks, agents frequently return ineligible candidates. The planning experiments also show that maintaining a plan helps some models sustain longer workflows, but does not consistently improve outcomes. A useful direction is to make each plan step resolve a specific clinical condition. For trial eligibility, the plan could track each candidate's inclusion and exclusion criteria, the supporting evidence and whether each condition is met, not met or unknown. Unresolved conditions would guide further retrieval before the agent decides whether to include the trial in its final result.

Verify task completion. On `MedTool`, selecting the right calculators does not ensure correct values, and on `MedWeb`, many episodes satisfy only some outcome checks. These observations motivate checks of observable task requirements after key actions and before stopping. A simple version already helps when the model has the answer but has not filed it: enabling ZeroClaw's submission check, which adds a reminder when the answer file is missing, raises `Qwen3.6-35B-A3B`'s `MedMemory` score from 0.42 to 0.51 (Section I). Calculator workflows could check that every requested quantity has a value and that its inputs match the patient record. Browser workflows could reread the relevant portal state after an action, such as updating an authorization request, to confirm that the intended change took effect. Such checks would identify incomplete steps and give the agent a concrete target for a retry.

B Benchmark construction

We build each environment from a public benchmark, selecting the tasks that exercise its capability and verifying every task we keep (Table 3). Of the $2{,}293$ upstream tasks, 107 are kept. Automatic checks and, on `MedTool` and `MedMemory`, models outside the study narrow the candidates. A human expert then reviews every kept task, checking that its gold answer is correct, that neither the prompt nor the recent record gives the answer away, that the task is unambiguous and that it is clinically sound. Table 4 expands the descriptive groups in Figure 2.

Table 3: **How each environment was built.** Each selection step lists the tasks it leaves. A human expert then reviews every kept task.

Env.	Source (license)	Pool	Selection (tasks left)	Kept	Verification
`MedTool`	MedMCP-Calc (MIT)	118	difficulty screen (107) → reliability contract and independent model audit (32)	32	scorer accepts the gold and rejects a perturbed answer
`MedMemory`	MedMemoryBench (Apache-2.0)	1,986	exact-match answers (349) → long history (241) → answer not restated in recent sessions (214) → entity and date questions whose gold the record supports (51) → two blind model reviews (14, plus 9 from an earlier round)	23	reviewers lock answers before the gold is compared
`MedPlanning`	HealthAgentBench (MIT)	54	runs without credentialed data (28)	28	upstream task tests
`MedWeb`	HealthAdminBench (Apache-2.0)	135	2–8 outcome checks and a process check, fixed quota per family (24)	24	upstream deterministic checks

`MedTool`**.** Each task gives a clinical request and a patient identifier. The agent reads a per-task SQLite record through read-only SQL and Python, and the needed values sit among distractors. The difficulty screen runs GPT-5.4-mini, which we do not evaluate. Tasks it solves from the given inputs or with plain SQL and Python are too easy, and tasks on which it makes no progress at all point to knowledge the task does not supply, so the screen removes both. Each remaining task must then pass a contract: sufficient inputs, a complete and consistent gold answer, a clinically coherent case, a real dependency or branch between calculators, verified formulas, a scorer that accepts the gold and rejects a perturbed answer, and an accept verdict from an independent model audit.

`MedMemory`**.** Each task asks one question about a history of 70–100 consultation sessions, with the evidence at least 10 sessions before the question. Upstream questions fail in two main ways. Some answers are restated in the recent sessions, so the question needs no long-range retrieval. Others have a gold answer that the visible record does not support, most often a date kept only in metadata. Automatic checks remove clear cases of both, and blind review catches the rest. Two reviewers from models outside the study read each remaining question with its history, one after the other. Each judges whether the question is valid and locks an answer before the gold is revealed, and a question is kept only when both judge it valid and their answers agree. Comparing the agreed answers with the gold after the lock then checks each gold answer. Where the reviewers give the same value in the record's own words, we add that wording as an accepted answer. Answering without the history, or searching only the recent sessions, scores $0/23$, and a search over the whole chart scores $4/23$.

`MedPlanning`**.** The 28 tasks are all HealthAgentBench tasks that run without credentialed data, so the environment can be rerun in full. They cover trial eligibility, EHR data quality, EHR-to-MEDS conversion and tumor-region selection on whole-slide images (Table 4), and upstream marks 23 of them hard and 5 as controls.

`MedWeb`**.** We keep tasks whose outcome rests on 2–8 final-state checks and that include at least one process check, which the Proc. diagnostic needs. The check range leaves out the one family whose outcome joins 9–17 checks. Within each family we take the first tasks by index.

Examples and difficulty. Table 5 gives one task per environment, paraphrased from its prompt, and the best mean outcome any of the 25 model–harness pairs reaches.

Table 4: **Composition of the 107 selected task instances.** The 13 display groups summarize source clinical areas and care settings (`MedTool`), requested information (`MedMemory`), and task objectives (`MedPlanning`, `MedWeb`).

Environment	Display group	n	Detailed content
`MedTool` (32 tasks)	Internal Medicine	14	Gastroenterology/hepatology (5), infectious diseases (3), respiratory medicine (2), hematology/oncology (2), cardiovascular medicine (1), and nephrology/electrolytes (1).
	Acute & Perioperative Care	11	Emergency medicine/trauma (5), intensive care (2), and anesthesia/perioperative care (4).
	Neurology & Mental Health	5	Neurosurgery/neurology (3) and mental/psychological health (2).
	Nursing & Pediatrics	2	Inpatient nursing-risk assessment (1) and pediatric-asthma assessment (1).
`MedMemory` (23 tasks)	Test Results	11	Retrieve recorded laboratory, imaging, or measured examination findings, including changes in measured indices.
	Medication History	7	Retrieve drug identities or classes, doses, dose changes, discontinuation, or the identity of an allergenic medication.
	Diagnoses & Patient Reports	5	Recorded diagnoses (3), the date of a reported numbness event (1), and a reported perceptual change (1).
`MedPlanning` (28 tasks)	Trial Eligibility	9	Identify all eligible candidate trials from a patient note and trial inclusion/exclusion criteria.
	Record Processing	9	Flag erroneous clinical-record rows (8), or configure and run MIMIC-IV-to-MEDS data-format conversion (1).
	Tumor Localization	10	Select tumor-containing grid tiles on ten whole-slide pathology images and submit their coordinates.
`MedWeb` (24 tasks)	Denial Resolution	10	Submit an appeal (7), escalate to a supervisor (2), or record a corrected-claim disposition (1), then document the resolution.
	Referral Review	10	Verify and clear a referral (4), or identify a problem, withhold authorization, and keep the referral open (6).
	Equipment Order Checks	4	Hold an equipment order for an expired evaluation, missing prescription, inactive insurance, or unsigned prescription (1 each), then withhold the fax and document the issue.

Counts refer to task instances, not unique patients or workflows. The eight record-quality tasks include four clue/no-clue pairs.

Table 5: **One example per environment and the best outcome over all model–harness pairs.**

Env.	Example task (paraphrased)	Best
`MedTool`	A patient with ulcerative colitis is admitted with frequent bloody stools and systemic symptoms. Decide which severity and risk calculators the question calls for, pull their inputs from the patient's record, compute them, and write every value to the answer file.	0.16
`MedMemory`	Given 80 consultation sessions in order, answer one question about a dose, lab value or diagnosis that was recorded once, dozens of sessions before the question.	0.86
`MedPlanning`	Given one admission note and a directory of trial documents, list every trial whose inclusion criteria the patient meets and none of whose exclusion criteria apply. Other tasks select the tumor region on a whole-slide image.	0.10
`MedWeb`	For a no-authorization claim denial, review the denial, the remittance image and the patient inquiry in the EMR, check eligibility on the payer portal, carry out the right resolution there, and document it in a triage note.	0.47

C SCORING AND TASK DIAGNOSTICS

Every outcome comes from a deterministic oracle (Section 4.2). For each environment, Table 6 gives the outcome rule, where the answer is read and the task diagnostic reported as Proc.

Table 6: **Scoring rules.**

Env.	Outcome	Answer read from	Diagnostic (Proc.)
`MedTool`	calculator-selection F1 $\times$ share of required outputs correct, with omitted outputs counted wrong. Numbers match within $\max(0.1, 0.02\|v\|)$ of the reference v, and categories after normalization	answer file, else final reply	calculator-selection F1
`MedMemory`	1 if the normalized answer matches an accepted alias	answer file, else final reply	LLM judge on the condensed trace
`MedPlanning`	1 if the task's own tests pass	output files	reference-set recall (17 tasks)
`MedWeb`	1 if every outcome check on the final state passes	application state	share of process checks passed

Answer recovery. When the answer file is missing or empty, we parse the final reply with the task's own scorer, which needs a parseable calculator submission on `MedTool` or an explicit answer line on `MedMemory`. A recovered answer counts as *stated*, and an episode with neither a file answer nor a recoverable reply counts as *absent*. Scoring the file alone would change the harness rankings (Section F.2).

LLM judge. The `gpt-4o-mini` judge scores the process on `MedMemory`, which has no intermediate checks. It reads a condensed trace rebuilt from the proxy logs: repeated history is removed, each round's new messages and tool calls are kept, tool calls are read from both the OpenAI and Anthropic request formats, and long message bodies are shortened. Its score averages three ratings in $[0, 1]$: appropriate tool use, coherent and evidence-grounded actions, and recovery from tool failures.

D HARNESS CONFIGURATION AND CONTROLS

All five harnesses reach the same locally served models through one proxy, which records requests and responses. Table 7 lists their versions and the few defaults we changed so that each harness can complete the tasks.

Table 7: **Harness versions and the defaults we changed.** Versions are those installed at the end of the study, except Claude Code, which updates itself and ran one version within each model block.

Harness	Version	Default changed, and why
OpenClaw	2026.9.2	output cap 16K $\rightarrow$ 65,536 (truncated tool calls on 15 of 32 `MedTool` tasks), clock 900 s $\rightarrow$ 3,600 s
Hermes	0.21.0	none
ZeroClaw	0.8.5	tool-iteration limit $\approx$ 11 $\rightarrow$ 200 (past it, the tool list is dropped)
Codex CLI	0.156.1	API-key variable supplied, MCP tools auto-approved (`MedWeb`)
Claude Code	2.1.267–2.1.270	none (the proxy translates its Anthropic-format traffic)

Serving settings. vLLM serves `Gemma4-26B-A4B`, `Qwen3.6-35B-A3B` and `Nemotron-3.5-30B-A3B` with a 262,144-token context window, and `GPT-OSS-120B` and `GLM-4.7-Flash` with 131,072 tokens. On supported request paths the proxy fills in sampling fields a harness leaves out, with temperature 0.8 for `Qwen3.6-35B-A3B` and 1.0 for the other models and top-p 0.95, and it keeps every value a harness sends. In the four model blocks after `Qwen3.6-35B-A3B`, the proxy also caps output at 65,536 tokens for `Gemma4-26B-A4B` and `Nemotron-3.5-30B-A3B` and 16,384 for `GPT-OSS-120B` and `GLM-4.7-Flash`, keeping any lower harness limit, and clients time out after 1,800 s (a stream-idle timeout for Codex) under a 3,600 s proxy timeout. In the `Qwen3.6-35B-A3B` block, output limits come from the harnesses alone.

Isolation and accounting. Each episode runs once per seed in its own namespace sandbox, with a private temporary directory, a memory cap and a one-hour deadline, and it cannot see graders,

gold answers or other episodes. Each isolated `MedWeb` episode starts from a fresh browser context, and no episode is rerun after its score is seen. Before a harness entered the study, we traced the network connections of one episode to confirm that all model traffic reached the local proxy. This check excluded one candidate harness, Moltis, which answered with an off-machine model through credentials it found on the host.

E Details of the model–harness decomposition

Within one environment, $\bar{y}_{mh\sigma}$ is the mean outcome over tasks for model m, harness h and seed σ. A dot denotes an average over that index. The terms in Equation (1) are

$$\begin{aligned} \mu &= \bar{y}_{\cdots}, \\ \alpha_m &= \bar{y}_{m\cdot\cdot} - \mu, \qquad \beta_h = \bar{y}_{\cdot h\cdot} - \mu, \\ \gamma_{mh} &= \bar{y}_{mh\cdot} - \bar{y}_{m\cdot\cdot} - \bar{y}_{\cdot h\cdot} + \mu, \\ \varepsilon_{mh\sigma} &= \bar{y}_{mh\sigma} - \bar{y}_{mh\cdot}. \end{aligned} \tag{3}$$

These definitions give $\sum_m \alpha_m = \sum_h \beta_h = 0$, $\sum_m \gamma_{mh} = 0$ for each h, $\sum_h \gamma_{mh} = 0$ for each m, and $\sum_\sigma \varepsilon_{mh\sigma} = 0$ for each pair (m, h).

Writing $M = |\mathcal{M}|$, $H = |\mathcal{H}|$ and $S = |\mathcal{S}|$, the sums of squares are

$$\mathrm{SS}_M = HS \sum_m \alpha_m^2, \qquad \mathrm{SS}_H = MS \sum_h \beta_h^2,$$

$$\mathrm{SS}_{MH} = S \sum_{m,h} \gamma_{mh}^2, \qquad \mathrm{SS}_E = \sum_{m,h,\sigma} \varepsilon_{mh\sigma}^2, \tag{4}$$

$$\mathrm{SS}_T = \sum_{m,h,\sigma} (\bar{y}_{mh\sigma} - \mu)^2. \tag{5}$$

Expanding $\bar{y}_{mh\sigma} - \mu = \alpha_m + \beta_h + \gamma_{mh} + \varepsilon_{mh\sigma}$, all cross terms vanish by the zero-sum identities above, for example $\sum_{m,h,\sigma} \alpha_m \beta_h = S\,(\sum_m \alpha_m)(\sum_h \beta_h) = 0$. This proves Equation (2). For $\mathrm{SS}_T > 0$, dividing by SS_T gives the four shares. We compute these shares separately within each environment and report their arithmetic mean across the four environments.

Proof of Proposition 3.1. Using population variances, a cell at a fixed model deviates from the model mean by $\bar{y}_{mh\cdot} - \bar{y}_{m\cdot\cdot} = \beta_h + \gamma_{mh}$. Hence

$$\begin{aligned} \overline{\mathrm{HV}} &= \frac{1}{M} \sum_m \frac{1}{H} \sum_h (\beta_h + \gamma_{mh})^2 \\ &= \frac{1}{H} \sum_h \beta_h^2 + \frac{1}{MH} \sum_{m,h} \gamma_{mh}^2 = \frac{\mathrm{SS}_H + \mathrm{SS}_{MH}}{N}. \end{aligned}$$

The cross term is zero because $\sum_m \gamma_{mh} = 0$ for every h. Exchanging models and harnesses gives $\overline{\mathrm{MV}} = (\mathrm{SS}_M + \mathrm{SS}_{MH})/N$. Subtracting the two expressions proves the comparison in Proposition 3.1. □

Values by environment. Table 8 gives the shares behind Figure 4a and the HV/MV ratio of Proposition 3.1. With all five models the ratio stays below one in every environment, so changing the model moves scores more than changing the harness. Among the three main models, `MedMemory` is the only environment where it exceeds one. For comparison, Zhang et al. (2026b) report 7.8 on a SWE-bench subset with three frontier models and three configurations of one harness, so the ratio depends on which models, harnesses and tasks are compared.

Table 8: **Variance shares and HV/MV by environment.** Shares of the outcome variance over all five models, and the HV/MV ratio over all five and over the three main models.

	MedTool	MedMemory	MedPlanning	MedWeb	**Mean**
Model	73.2%	47.4%	36.4%	73.2%	57.5%
Harness	5.3%	25.4%	9.6%	9.5%	12.4%
Interaction	9.5%	14.4%	14.1%	9.4%	11.9%
Run-to-run	12.0%	12.8%	39.8%	8.0%	18.2%
HV/MV, five models	0.18	0.64	0.47	0.23	
HV/MV, three main models	0.82	1.45	0.45	0.19	

F Additional results

F.1 Performance, rankings and token use

Two further models. Table 9 repeats the main comparison for `Nemotron-3.5-30B-A3B` and `GLM-4.7-Flash`. `Nemotron-3.5-30B-A3B` scores near zero on `MedPlanning` and `MedWeb` under every harness. `GLM-4.7-Flash` is the only model under which ZeroClaw is not last overall, and its `MedPlanning` outcomes stay near zero: in 92 of its 150 pathology episodes the agent computed a tissue statistic with a threshold that matched almost no pixel of the H&E stain and reported the slide as tumor-free.

Ranking stability. Table 10 gives the leader, its probability of ranking first and its gap to the runner-up for every model and environment.

Token use. Token use per episode, which counts the history resent with every call, differs by a median $2.6\times$ across harnesses within a model–environment pair. The highest-scoring harness also uses the fewest tokens in only 3 of 20 pairs. Under `Qwen3.6-35B-A3B`, Hermes reaches 0.78 on `MedMemory` with 0.41 million tokens per episode, while Codex reaches 0.86 with 0.62 million.

F.2 Execution outcomes and scoring analyses

Episode endings. Figure 5a groups the episodes of `Qwen3.6-35B-A3B` by how they end. Categories are decided in a fixed order: explicit termination evidence first, then the state of the answer or evaluated output, and an error earlier in a run that continues does not decide the category. Positive score includes partial credit on `MedTool`. Scored zero is a scorable output or application state that earns no credit. No valid output is a required output that is missing, unparseable or in the wrong format. Timeout is the one-hour wall-clock limit. Limit or error covers context and request limits and other terminal errors. Unresolved episodes lack the evidence to decide.

Answer rate and score. Table 11 gives the values behind Figure 4b, counting an answer wherever the agent gave it. Across the five models, 15% of `MedTool` episodes and 13% of `MedMemory` episodes end without an answer, while 96% of `MedPlanning` episodes produce output files. Because a harness that answers more may answer harder items, scores are compared on the n matched task–seed pairs that every harness answered. On these pairs, scores vary little across harnesses except for `GPT-OSS-120B` and `GLM-4.7-Flash` on `MedMemory`, with ranges of 0.56–0.85 and 0.44–0.75.

Answer time. The bottom rows of Table 11 recount answers retrospectively at earlier cutoffs, to show how the answer-rate gap depends on the one-hour limit. Each answer's time is the last write of the answer file in the saved trajectory, or the episode's end for reply-only answers and files without a usable timestamp. With more time, the `MedTool` answer-rate spread narrows from 0.49 to 0.29, while the outcome spreads stay at 0.04–0.05 on `MedTool` and 0.24–0.25 on `MedMemory`.

File-only scoring. On `MedTool` and `MedMemory`, 14% and 17% of answers appear only in the final reply (Figure 6). Scoring the answer file alone would lower the `MedTool` scores of `Gemma4-26B-A4B` from 0.06–0.13 to 0.00–0.05 and the `MedMemory` scores of `Nemotron-3.5-30B-A3B` from 0.55–0.71 to 0.20–0.43. It would also change the harness rankings, with Kendall τ as low as 0.40 and a different leader under `Gemma4-26B-A4B` on `MedMemory`.

Table 9: **Outcome and process under a fixed model.** The two further models. Only the harness varies within a block. Mean ± SD over seeds. **Proc.**: calculator selection F1 (`MedTool`), recall of the reference set on the 17 `MedPlanning` tasks that have one, share of process checks passed (`MedWeb`). †`MedMemory` records only the answer, so its Proc. is the LLM judge's score (section C). **MTok**: million tokens per episode. *Rank*: mean outcome rank within a block. **Bold** and underlined: best and second best Outc. and Proc. per capability within a block. Rows are ordered by mean rank over all five models, worst first.

Harness	MedTool			MedMemory			MedPlanning			MedWeb			Rank
	Outc.↑	Proc.↑	MTok↓	Outc.↑	Proc.↑†	MTok↓	Outc.↑	Proc.↑	MTok↓	Outc.↑	Proc.↑	MTok↓	
Nemotron-3.5-30B-A3B													
ZeroClaw	0.05±0.02	0.23±0.04	0.53	0.55±0.09	0.20±0.07	0.88	0.01±0.02	0.17±0.10	3.23	0.00±0.00	0.08±0.02	9.26	4.50
OpenClaw	0.06±0.01	0.28±0.07	0.65	0.59±0.05	0.24±0.05	1.35	0.00±0.00	0.15±0.01	5.00	0.00±0.00	0.08±0.02	12.22	3.75
Hermes	0.07±0.00	**0.36±0.02**	0.23	0.62±0.05	0.31±0.05	0.85	**0.02±0.04**	**0.40±0.08**	1.87	0.00±0.00	0.07±0.03	11.54	2.50
Codex	**0.08±0.01**	0.33±0.03	0.60	0.68±0.05	**0.41±0.05**	0.72	0.02±0.02	0.25±0.09	2.95	**0.04±0.04**	0.08±0.01	8.01	1.50
Claude Code	0.06±0.01	0.29±0.03	0.35	**0.71±0.05**	0.29±0.05	1.20	0.00±0.00	0.22±0.09	3.53	0.01±0.02	**0.12±0.05**	9.76	2.75
GLM-4.7-Flash													
ZeroClaw	0.03±0.01	0.31±0.00	0.41	0.48±0.12	0.44±0.05	0.23	0.00±0.00	0.10±0.06	1.55	0.15±0.10	0.31±0.06	2.65	3.25
OpenClaw	0.02±0.00	0.29±0.02	0.58	0.43±0.04	0.36±0.02	0.59	0.00±0.00	0.11±0.04	1.63	0.00±0.00	0.27±0.06	3.71	4.25
Hermes	**0.04±0.02**	**0.32±0.04**	0.41	0.35±0.08	**0.48±0.06**	0.28	0.01±0.02	**0.33±0.04**	1.03	0.06±0.05	0.14±0.03	1.10	3.00
Codex	0.04±0.00	0.28±0.03	0.61	**0.54±0.05**	0.38±0.09	1.55	0.01±0.02	0.28±0.06	2.64	0.21±0.15	**0.34±0.11**	8.30	2.00
Claude Code	0.02±0.00	0.28±0.03	0.87	0.48±0.08	0.34±0.08	1.28	**0.06±0.05**	0.23±0.14	2.80	**0.25±0.04**	0.32±0.04	4.40	2.50

Table 10: **Ranking stability.** Leader per model and environment, the bootstrap probability P that it ranks first and its gap to the runner-up with a 95% interval (paired bootstrap over tasks, 20,000 resamples). Bold marks an interval that excludes zero.

Model	MedTool			MedMemory		
	Leader	P	Gap [95% CI]	Leader	P	Gap [95% CI]
Gemma4-26B-A4B	Hermes	0.95	+0.04 [0.00, +0.09]	Claude Code	0.61	+0.01 [−0.07, +0.10]
Qwen3.6-35B-A3B	Hermes	0.91	+0.02 [−0.01, +0.04]	Codex	0.47	0.00 [−0.10, +0.10]
GPT-OSS-120B	Claude Code	0.94	+0.02 [0.00, +0.04]	Claude Code	0.97	**+0.12 [+0.03, +0.23]**
Nemotron-3.5-30B-A3B	Codex	0.80	+0.01 [−0.01, +0.03]	Claude Code	0.68	+0.03 [−0.06, +0.12]
GLM-4.7-Flash	Hermes	0.59	0.00 [−0.01, +0.02]	Codex	0.66	+0.06 [−0.09, +0.20]

Model	MedPlanning			MedWeb		
	Leader	P	Gap [95% CI]	Leader	P	Gap [95% CI]
Gemma4-26B-A4B	Claude Code	0.78	+0.02 [−0.02, +0.08]	Codex	0.61	+0.01 [−0.07, +0.11]
Qwen3.6-35B-A3B	OpenClaw	0.39	0.00 [−0.07, +0.07]	Claude Code	0.59	+0.01 [−0.10, +0.12]
GPT-OSS-120B	Claude Code	0.47	+0.01 [−0.06, +0.08]	Claude Code	1.00	**+0.24 [+0.11, +0.38]**
Nemotron-3.5-30B-A3B	Hermes	0.41	0.00 [−0.04, +0.04]	Codex	0.84	+0.03 [−0.03, +0.08]
GLM-4.7-Flash	Claude Code	0.93	+0.05 [−0.01, +0.12]	Claude Code	0.76	+0.04 [−0.06, +0.15]

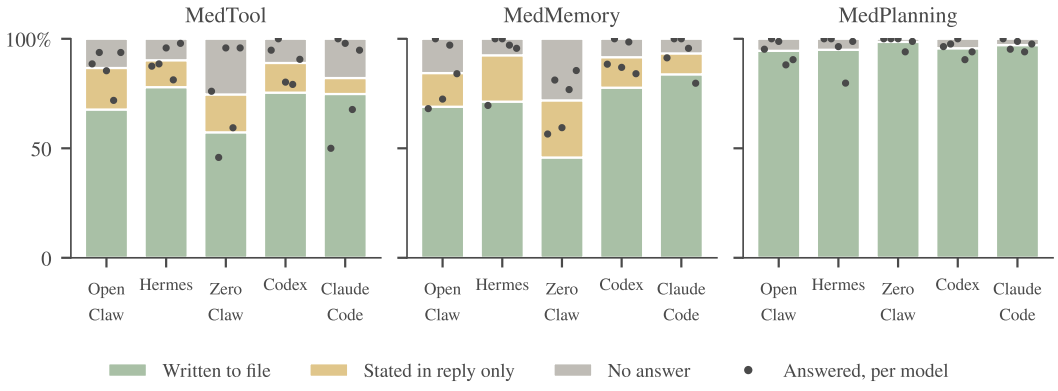


Figure 6: **Where the answer ends up.** Share of episodes whose answer was written to the named file, stated only in the final reply or absent, averaged over five models. Dots show each model's answer rate.

Table 11: **Answer rate and score.** Top: minimum–maximum across harnesses of the answer rate over all episodes and mean score on the n matched task–seed pairs every harness answered. Score is the task outcome (Section 4.2). Bottom: maximum-minus-minimum spreads over all episodes, averaged over five models, at retrospective answer-time cutoffs T with a 30-second clock tolerance. Recorded answers after the cutoff count as missing and score zero. Answers given only in the reply count.

	MedTool			MedMemory		
Model	Answer rate	Score (shared)	n	Answer rate	Score (shared)	n
Gemma4-26B-A4B	0.50–0.95	0.12–0.16	28	0.57–0.91	0.87–0.90	30
Qwen3.6-35B-A3B	0.46–1.00	0.10–0.16	39	0.81–1.00	0.82–0.88	56
GPT-OSS-120B	0.80–0.98	0.13–0.17	57	0.59–1.00	0.56–0.85	27
Nemotron-3.5-30B-A3B	0.59–0.81	0.08–0.12	28	0.77–0.99	0.69–0.73	45
GLM-4.7-Flash	0.91–0.98	0.02–0.04	72	0.80–0.96	0.44–0.75	32
Cutoff	Rate spread	Outcome spread		Rate spread	Outcome spread	
$T = 15$ min	0.49	0.04		0.28	0.25	
$T = 30$ min	0.42	0.05		0.27	0.24	
$T = 45$ min	0.37	0.05		0.27	0.25	
$T = 60$ min	0.29	0.04		0.26	0.25	

G SPLITTING THE HARNESS VARIANCE INTO ANSWER RATE AND ANSWER SCORE

When a task asks for an answer, an episode can end without one. For a harness h, with the model and environment fixed, let $A = 1$ if an episode ends with an answer, wherever the agent gives it, and $Y = 0$ if $A = 0$. The answer rate is $r_h = \Pr(A = 1 \mid h)$ and the mean score among answered episodes is $q_h = \mathbb{E}[Y \mid A = 1, h]$, so that $\bar{y}_h = r_h q_h$. With $a_h = r_h - \bar{r}$ and $b_h = q_h - \bar{q}$ the deviations from the means over harnesses, expanding $\bar{y}_h = \bar{r}\bar{q} + \bar{q}a_h + \bar{r}b_h + a_h b_h$ gives

$$\mathrm{Var}_h\, \bar{y}_h = V_r + V_q + R, \qquad V_r = \bar{q}^2\, \mathrm{Var}(a), \qquad V_q = \bar{r}^2\, \mathrm{Var}(b),$$
$$R = 2\bar{r}\bar{q}\, \mathrm{Cov}(a, b) + \mathrm{Var}(ab) + 2\bar{q}\, \mathrm{Cov}(a, ab) + 2\bar{r}\, \mathrm{Cov}(b, ab),$$

with moments taken over harnesses. V_r is the variance the answer rate alone would produce, and V_q the variance the answer score alone would produce. The first term of R vanishes when answer rate and mean answer score are uncorrelated across harnesses, and the others are of third and fourth order in the deviations. The covariance is negative when harnesses that answer more score lower on what they answer, as when they answer harder items.

Over the 10 model–environment pairs on `MedTool` and `MedMemory`, the first-order answer-rate share $V_r/(V_r + V_q)$ has a median of 61% and is the larger in 7 of the 10 pairs. The remainder $|R|$ is below a third of $\mathrm{Var}_h\, \bar{y}_h$ in 6 of the 10 pairs. It is larger where the outcome variance is small (`MedTool` under `Qwen3.6-35B-A3B` and `GPT-OSS-120B`) and where answer rate and mean answer score move in opposite directions (`Nemotron-3.5-30B-A3B` and `GLM-4.7-Flash` on `MedMemory`). For `Qwen3.6-35B-A3B` on `MedTool`, $V_r + V_q$ is 3.05 times $\mathrm{Var}_h\, \bar{y}_h$, so the negative remainder cancels about two thirds of them. Because q_h averages over the items h answered, Section 6.3 compares scores on matched task–seed pairs every harness answered.

H MH-Lab: IMPLEMENTATION AND ABLATION DETAILS

This section gives the configuration and comparison protocol behind Table 2, the intervals and task diagnostics of its outcome changes and the execution evidence that explains them.

H.1 IMPLEMENTATION AND COMPARISON PROTOCOL

Table 12 lists the configuration of the final matrix. Context management summarizes older turns, as the product harnesses in Section I do. The input budget sits below the native windows so that summarization fires on long trajectories. `Qwen3.6-35B-A3B` writes long reasoning, so it gets a larger window and output cap than `GPT-OSS-120B` while keeping the same input budget.

Table 12: MH-Lab **configuration of the final matrix.**

Setting	Value
Matrix	two models, four conditions, 107 tasks and three seeds, for 2,568 scheduled episodes
Input budget	65,536 tokens per request for both models, with 5% reserved for token-estimation error
Window and output cap	`Qwen3.6-35B-A3B`: 131K window, 65K output cap. `GPT-OSS-120B`: 80K window, 16K output cap
File tools	seven workspace tools for reading, writing, editing, listing and searching files and running shell commands, with workspace confinement and read-before-write checks
Browser tools	fourteen browser tools
Execution limits	tool results capped at 24K characters, loop detection, a 500-step limit and a one-hour deadline
Context management	summarization when estimated input reaches 80% of the input budget after the 5% reserve
Planning	an `update_plan` tool, planning instructions, reminders and the plan shown on every turn
Tool exposure off	task tools reachable only through `tool_search`, `tool_describe` and `tool_call`, and a tool must be discovered before it is called. The planning tool stays directly exposed

Pairing and exclusions. Each comparison pairs the same task and seed under *full* and the switched condition, and a pair is dropped when either episode is excluded. Excluded are provider failures (23 `MedWeb`, 2 `MedPlanning` and 1 `MedTool` episodes of `GPT-OSS-120B` whose responses vLLM could not parse, and 1 `MedPlanning` episode of `Qwen3.6-35B-A3B`) and two `Qwen3.6-35B-A3B` planning-off pathology episodes stopped by the 200 GiB memory cap when a shell script loaded a whole slide, which left no result. Since some of these failures stem from the model's own output or actions, we also count every excluded episode as a failure. All statistically significant outcome effects in Table 2 then retain their direction and significance. The largest shifts are on `MedWeb` for `GPT-OSS-120B`, where the three changes move from -0.17, -0.18 and -0.15 to -0.14, -0.17 and -0.12.

H.2 OUTCOME INTERVALS AND TASK DIAGNOSTICS

Table 13 gives the 95% intervals of the outcome changes in Table 2, next to `MedPlanning` recall and the share of `MedWeb` process checks passed. These two measures show one effect the pass rates hide. Without context management, `MedPlanning` recall falls by 0.29 for `Qwen3.6-35B-A3B` and 0.16 for `GPT-OSS-120B` although the outcome barely moves, because the pass threshold is rarely met in any condition. For the mechanism comparisons, we report environment-based task diagnostics.

H.3 EXECUTION EVIDENCE

Table 14 gives, for each mechanism, the execution measure behind its outcome change.

Context management. Without summaries, traces outgrow the input budget. Overflows rise from almost none to 67 of 72 `MedWeb` episodes for `Qwen3.6-35B-A3B` and 48 of 65 for `GPT-OSS-120B`, and to 57 of 84 `MedPlanning` episodes for `Qwen3.6-35B-A3B`.

Table 13: **Outcome and process changes behind Table 2.** *full*: mean ± SD over seeds. Other rows: paired change in outcome with a 95% bootstrap interval over tasks, and paired change (± SD of the per-seed change) in `MedPlanning` recall on the 17 tasks with a reference set and in the share of `MedWeb` process checks passed.

Condition	`MedTool` Outc. [95% CI]	`MedMemory` Outc. [95% CI]
	`Qwen3.6-35B-A3B`	
full	0.10±0.02	0.88±0.03
− context management	+0.01 [−0.01, +0.04]	−0.12 [−0.23, −0.01]
− planning	−0.02 [−0.04, 0.00]	−0.04 [−0.12, +0.01]
− tool exposure	0.00 [−0.02, +0.03]	−0.14 [−0.23, −0.07]
	`GPT-OSS-120B`	
full	0.14±0.03	0.72±0.03
− context management	−0.01 [−0.04, +0.02]	+0.06 [−0.09, +0.20]
− planning	−0.02 [−0.04, +0.01]	−0.01 [−0.10, +0.09]
− tool exposure	−0.04 [−0.07, −0.02]	−0.06 [−0.14, +0.03]

Condition	`MedPlanning` Outc. [95% CI]	`MedPlanning` Recall	`MedWeb` Outc. [95% CI]	`MedWeb` Checks
	`Qwen3.6-35B-A3B`			
full	0.12±0.05	0.54±0.03	0.44±0.02	0.44±0.07
− context management	−0.04 [−0.12, +0.02]	−0.29±0.01	−0.39 [−0.57, −0.22]	−0.17±0.11
− planning	+0.01 [−0.07, +0.10]	−0.04±0.11	0.00 [−0.08, +0.08]	−0.07±0.08
− tool exposure	+0.01 [−0.06, +0.07]	−0.12±0.14	0.00 [−0.06, +0.07]	−0.02±0.05
	`GPT-OSS-120B`			
full	0.06±0.02	0.40±0.03	0.18±0.08	0.33±0.04
− context management	+0.02 [−0.02, +0.07]	−0.16±0.06	−0.17 [−0.28, −0.07]	−0.18±0.17
− planning	+0.02 [−0.02, +0.07]	−0.08±0.03	−0.18 [−0.29, −0.08]	−0.19±0.05
− tool exposure	−0.04 [−0.10, 0.00]	−0.04±0.06	−0.15 [−0.26, −0.06]	−0.10±0.05

Table 14: **Execution evidence behind Table 2.** Each cell goes from *full* to the switched condition. Counts are over the valid episodes of each condition, and median turns over the task–seed pairs valid in both. Where a condition lost episodes to exclusion, counts show their denominators. `MedWeb` is scored on application state and has no file to submit. With tools hidden, the columns give the mean number of rejected direct calls per episode and the episodes with at least one.

Environment	− context management	− planning			− tool exposure	
	Overflows	Median turns	No valid submission	Timeouts	Rejected calls per episode	Episodes with a rejected call
`Qwen3.6-35B-A3B`						
`MedTool`	0 → 0	10 → 9	24 → 47	22 → 44	18	95/96
`MedMemory`	0 → 8	9 → 6	1 → 0	1 → 0	43	66/69
`MedPlanning`	2 → 57	37 → 38	3/84 → 3/81	8/84 → 16/81	79	84/84
`MedWeb`	0 → 67	99.5 → 78.5	–	3 → 1	31	47/72
`GPT-OSS-120B`						
`MedTool`	0 → 1	18 → 10	6/96 → 10/95	0/96 → 0/95	12	86/96
`MedMemory`	0 → 4	15 → 16	0 → 5	0 → 0	6	57/69
`MedPlanning`	0/84 → 22/83	31 → 29	1/84 → 0/83	1/84 → 0/83	34	82/84
`MedWeb`	0/66 → 48/65	176.5 → 63	–	6/66 → 1/66	64	68/68

Planning. Without a plan, `GPT-OSS-120B` ends `MedWeb` episodes much sooner, with median turns falling from 176.5 to 63 on the paired episodes. `Qwen3.6-35B-A3B` instead leaves more `MedTool` work unfinished, with missing outputs rising from 24 to 47 of 96 and timeouts from 22 to 44.

Tool exposure. With tools behind search, models still name hidden tools directly, `Qwen3.6-35B-A3B` in 95 of 96 `MedTool` episodes and `GPT-OSS-120B` in 86. Between 98% and 100% of these rejected calls follow a successful search, so the models find the tool but then call it through the wrong interface. The discovery gate itself (`tool_call` before a search) fires at most once per episode.

I MODULE ACTIVATION AND SWITCHES ON THE PRODUCT HARNESSES

Module activation. Before a mechanism inside a product harness can be ablated, it has to fire. We therefore counted how often each advertised module was used, in the archived episodes of the main study and of a pilot with Deep Agents, LangChain's agent harness. Hermes, OpenClaw and Claude Code expose skill tools in every episode, and none of the 321 episodes per model called one. Delegation was invoked in 0–5 episodes per model, for instance in 4 of 32 `MedTool` episodes by Deep Agents with `Qwen3.6-35B-A3B`. Hermes's loop guards intervened in no episode, and neither Hermes nor OpenClaw forwards its reasoning-effort setting to a custom vLLM provider. Summarization fires in 2–5% of episodes at the native windows.

Detectability. A mechanism that fires this rarely cannot be detected by switching it off. With shared sampling randomness, an episode in which it never fires ends the same way with or without it, so for a firing rate p the paired effect is at most p, and at most $\sqrt{p/(1-p)}$ standard deviations of the paired difference. With 69–96 paired episodes per environment, the expected z-statistic of switching summarization off is then at most 1.19 ($p = 0.02$, $n = 69$) to 2.25 ($p = 0.05$, $n = 96$), even if every firing turned a success into a failure. Our serving stack does not repeat a trajectory exactly under a fixed seed, which only makes such a mechanism harder to detect. The ablation in Section 6.4 therefore uses an input budget at which context management fires. The switches in Table 15 were run where each mechanism is reachable, as paired comparisons on the same tasks with every other setting held fixed.

Table 15: **Paired switches on the product harnesses.** Outcome and context overflows go from the default to the switched setting. *Pairs* counts paired episodes, over three seeds where it is 69 and over one seed otherwise. *Tool search* is Hermes's `tool_search`/`tool_call` layer. With it off, `GPT-OSS-120B` browses in 24/24 `MedWeb` episodes instead of 3/24. The *submission check* adds one reminder round when the answer file is missing.

Module	Harness × model	Environment, window	Outcome	Ovf.	Pairs
Summarization	OpenClaw × `Qwen3.6-35B-A3B`	`MedMemory`, 80K	$0.62 \to 0.44$	$8 \to 25$	69
Summarization	Deep Agents × `Qwen3.6-35B-A3B`	`MedMemory`, 80K	$0.52 \to 0.38$	$0 \to 10$	69
Summarization	Deep Agents × `Qwen3.6-35B-A3B`	`MedMemory`, 64K	$0.54 \to 0.45$	$0 \to 15$	69
Summarization	Hermes × `Qwen3.6-35B-A3B`	`MedMemory`, 80K	$0.83 \to 0.83$	$0 \to 0$	23
Summarization	Deep Agents × `GPT-OSS-120B`	`MedMemory`, 80K	$0.52 \to 0.48$	$0 \to 4$	69
Tool-output truncation	Hermes × `Qwen3.6-35B-A3B`	`MedMemory`, native	$0.78 \to 0.78$	–	23
Tool-output truncation	Hermes × `Qwen3.6-35B-A3B`	`MedPlanning`, native	$0.11 \to 0.06$	–	18
Tool-output truncation	Hermes × `Qwen3.6-35B-A3B`	`MedTool`, native	$0.16 \to 0.14$	–	32
Tool search (off)	Hermes × `Qwen3.6-35B-A3B`	`MedWeb`, native	$0.17 \to 0.50$	–	12
Tool search (off)	Hermes × `GPT-OSS-120B`	`MedWeb`, native	$0.00 \to 0.00$	–	24
Submission check (on)	ZeroClaw × `Qwen3.6-35B-A3B`	`MedMemory`, native	$0.42 \to 0.51$	–	69
Submission check (on)	Hermes × `GPT-OSS-120B`	`MedMemory`, native	$0.65 \to 0.52$	–	23

Switches on the product harnesses. The results match RQ3 (Section 6.4). Summarization matters where a trace would otherwise outgrow the window and the harness has no other fallback: OpenClaw and Deep Agents end such an episode on the provider's overflow error, Hermes trims the history and continues, and `GPT-OSS-120B` rarely reaches the window at all. Tool search decides whether the browser tools are visible. With it off, `GPT-OSS-120B` browses in every episode but still completes none, and `Qwen3.6-35B-A3B` rises from 0.17 to 0.50. The submission check turns a finished episode without a file into a filed answer only when the model already had an answer. `Qwen3.6-35B-A3B`'s stated answers are right, so its score rises. The four answers `GPT-OSS-120B` filed after a reminder are all wrong, so the check changes none of its scores, and its drop from 0.65 to 0.52 comes from episodes that already differed before any reminder. Truncating tool outputs, which Hermes does in nearly every episode, has no clear effect.