



0xClaw: Autonomous Open-ended Agents with Phase-aware Budget Allocation and Temporal Pressure

Yu Zhou^{1*}, Xinyuan Dou^{1*}, Yudong Cao², Junhao Song^{1†}

¹Imperial College London, ²Zapata Quantum

* Equal contribution. † Project lead: junhao.song23@imperial.ac.uk

🏆 Hackathon Champion: Oxclaw-ai.github.io 🔄 Oxclaw-ai/0xClaw 🤖 Oxclaw-ai/budgetrouter-qwen3.5-2b-lora

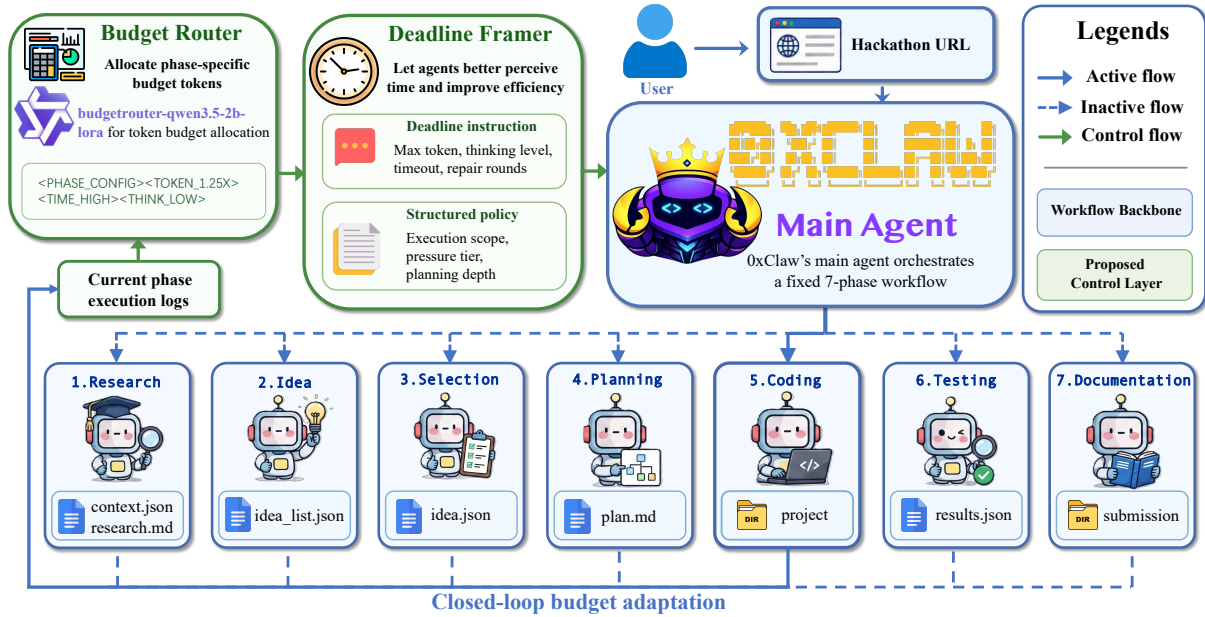


Figure 1: **Phase-aware budget control pipeline.** 0xClaw runs a hackathon as seven phases, from research to documentation. The **Budget Router** reads the workspace state and predicts a control token for each phase. The **Deadline Framer** combines the token with cumulative spend into a deadline instruction, injected at the boundary.

Abstract

Large language model agents handle structured coding tasks well, but open-ended projects are harder. A hackathon gives an agent an open goal, a hard deadline and several artifacts to deliver. The agent has to decide how to split its budget across phases and when to stop widening scope. Most agents do neither. 0xClaw is a multi-agent framework that splits a hackathon run into seven phases. Each phase hands a typed artifact to the next. The Budget Router is a small fine-tuned Qwen3.5-2B model. It reads the workspace state and predicts a per-phase control token for token budget, time, reasoning effort and repair rounds. The Deadline Framer turns each token, together with the cumulative execution log, into a plain deadline instruction given to the agent at the start of each phase. Our study covers 20 hackathon tasks, 10 baselines and more than 200 agent runs. 0xClaw matches Claude Code at the best pass@1 (1.00) and ex-

ceeds ARIS (0.95), the two strongest baselines. Against Claude Code, it achieves comparable quality and pass@1 while using 59% fewer tokens. The trade-off is wall-clock time, about 2.5 times that of ARIS.

1 Introduction

Large language models (LLMs) now run entire projects, not just single steps. Recent agent frameworks carry a task from start to finish, across research (Lu et al., 2024; Yamada et al., 2025), software development (Jiang et al., 2025; Yang et al., 2024), and multi-agent task decomposition (Baek et al., 2025; Zhang et al., 2025b; Yue et al., 2025). A hackathon is a hard case of this setting (Falk et al., 2025; Ege et al., 2024). Figure 2 positions hackathons against four adjacent agent settings. Only hackathons combine open goals, hard deadlines, multiple artifacts, and phase-aware scope trade-offs. The agent has to spread a limited bud-

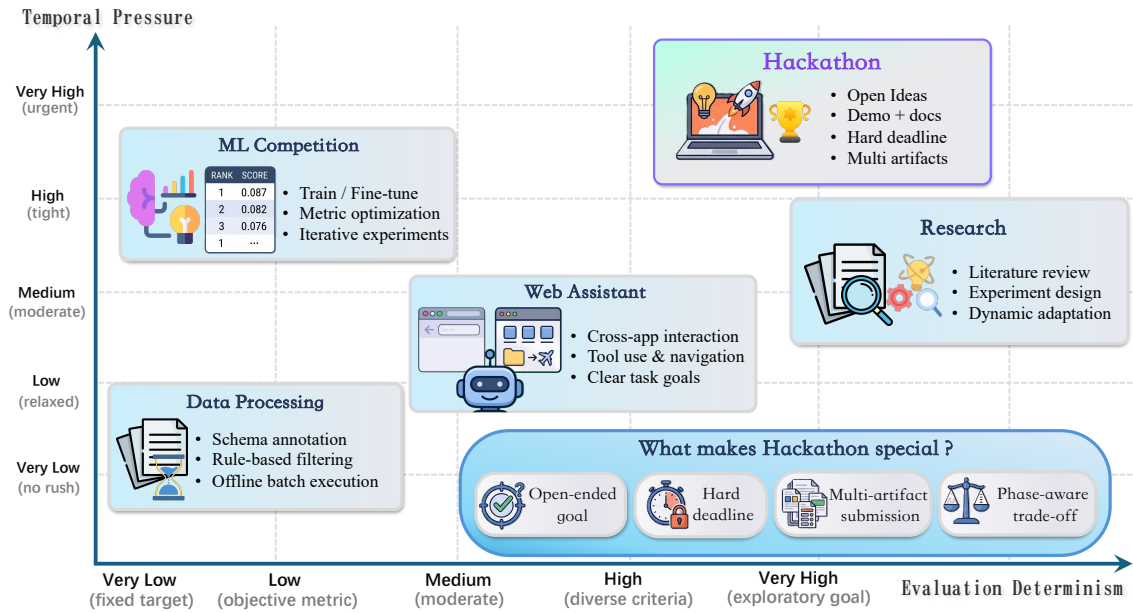


Figure 2: **Hackathon tasks in the landscape of agent settings.** Five settings on two axes: evaluation determinism and temporal pressure. Hackathons sit at the high-pressure, open-criteria corner and combine open goals, hard deadlines, multi-artifact submissions, and phase-aware trade-offs. No other setting combines all four.

get across phases and stop widening scope as the deadline nears. Most agents control neither. They have no way to plan spending across phases and no way to turn the deadline into a change in behavior.

Prior work mostly improves how tasks are split and assigned. Multi-agent systems build roles and execution graphs (Qian et al., 2024; Zhang et al., 2025a; Yue et al., 2025). Budget-aware methods cap tokens or tool calls for one response or reasoning trace (Han et al., 2025; Li et al., 2025; Liu et al., 2025). Neither line plans a budget across the phases of a whole project. Neither turns a deadline into per-phase behavior.

A phase-aware agent is hard to build for three reasons. (i) Long context: the agent carries requirements, ideas, failed attempts and test results in one growing window. Recall drops as it grows (Liu et al., 2024; Li et al., 2026). (ii) Local budget: existing methods are bound to one call at a time and do not plan spending across phases (Han et al., 2025; Liu et al., 2025). (iii) No time sense: agents need an external deadline signal (Sehgal et al., 2026), but converting it into phase-specific instructions remains unstudied.

We present **0xClaw**, a multi-agent framework for hackathon tasks. It runs as seven fixed phases, from research to documentation. Each phase passes only a typed handoff artifact to the next, not the full

history, keeping context small. The **Budget Router** reads the workspace state and predicts a per-phase control token covering token budget, time, reasoning effort, and repair rounds. The **Deadline Framer** combines the token with the cumulative execution log into a plain deadline instruction at the phase boundary, with no extra training. Figure 1 shows the pipeline. Our contributions are as follows:

- (1) We present 0xClaw, a seven-phase framework that passes a typed handoff artifact between phases and forwards only what the next phase needs, from research through documentation.
- (2) We introduce two budget controllers. The **Budget Router** predicts a per-phase control token from the workspace state, covering token budget, time, reasoning effort, and repair rounds. The **Deadline Framer** turns each token into natural-language deadline guidance without training, using the cumulative execution log at each phase boundary.
- (3) Across 20 hackathon tasks and more than 200 agent runs, 0xClaw achieves the highest pass@1 (1.00) and the highest Novelty score across all baselines, while delivering comparable quality to the strongest baseline with 59% fewer tokens.

2 Related Work

Open-ended agents. Early work on LLM agents automated individual stages such as code generation or literature summarization (Yang et al., 2024). Open-ended generation has also reached multi-modal conversational output (Song et al., 2026b). The AI Scientist (Lu et al., 2024; Yamada et al., 2025) drives end-to-end research from idea to write-up. AIDE (Jiang et al., 2025) and SWE-agent (Yang et al., 2024) traverse repositories and iteratively refine code. General-purpose agent platforms include MetaGPT (Hong et al., 2024), ResearchAgent (Baek et al., 2025), OpenHands (Wang et al., 2025), OpenHands-Versa (Soni et al., 2026), AI-Researcher (Tang et al., 2026), ARIS (Yang et al., 2026), Hermes (Nous Research, 2026), OpenClaw (OpenClaw Contributors, 2026), and NanoBot (Ren and HKU Data Intelligence Lab, 2026). Adaptive workflow methods further refine how agents are composed: AFlow (Zhang et al., 2025b), agentic supernets (Zhang et al., 2025a), and MasRouter (Yue et al., 2025); such composition by search echoes older bio-inspired optimization (Song et al., 2024). None of these systems controls per-phase resource budgets, and error cascades remain a known failure mode of long agent pipelines (Xie et al., 2026). 0xClaw treats hackathon execution as a phase-structured pipeline with typed handoffs and adds explicit per-phase budget and deadline control.

Budget-aware reasoning and tool use. Token-budget methods compress reasoning at the response level. TALE (Han et al., 2025) embeds an estimated token budget directly in the prompt. SelfBudgeter (Li et al., 2025) adapts token allocation per problem. BATS (Liu et al., 2025) extends budget control to tool-using agents. Constraint-driven schedulers (Wang et al., 2026) jointly bound time and tokens for online agentic workflows. These methods bound a single response, reasoning trace, or tool call. None allocates budget across the distinct phases of a project. The **Budget Router** learns a per-phase budget policy from hackathon execution logs and emits a compact control token for each phase.

Deadline pressure on LLM agents. Stress and pressure shape agent behavior. StressPrompt (Shen et al., 2025) shows that moderate prompt-level stress improves LLM performance on closed NLP tasks. Pasichnyk (Pasichnyk, 2026) reproduces the inverted-U pattern in multi-agent grid-world sim-

ulations. Both match the Yerkes-Dodson inverted-U: moderate arousal helps, extremes hurt. Real-time deadlines (Sehgal et al., 2026) show that LLM agents fail to track elapsed time in negotiation dialogue and need explicit deadline signals. Latency-sensitive agents (Kang et al., 2025) adapt model size and quantization to wall-clock pressure. These study stress as a prompt, a constraint, or a temporal signal. None converts a deadline into per-phase behavioral instructions for an open project workflow. The **Deadline Framer** maps each control token into a qualitative deadline instruction injected at the phase boundary, with no numerical clocks or reinforcement learning.

3 Task Definition

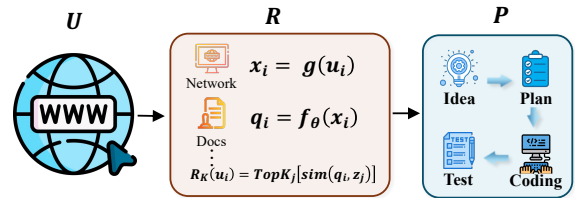


Figure 3: **Overview of the 0xClaw task.** The agent ingests problem-specification URLs, retrieves relevant context from a knowledge bank, and runs a seven-phase workflow from research to documentation under a shared token budget B and time budget T .

As shown in Figure 3, we formulate the hackathon task as a resource-constrained generation problem from web inputs U to an executable product P . Given a set of N problem-specification URLs $U = \{u_i\}_{i=1}^N$, a retrieval module R parses each URL into a structured task context $x_i = g(u_i)$ that records the problem statement, deliverables, and evaluation criteria, then encodes it as a query $q_i = f_\theta(x_i)$. Given a corpus of M hackathon examples, documents, and code snippets $\{z_j\}_{j=1}^M$, the module retrieves the top- K most relevant items per URL:

$$R_K(u_i) = \text{TopK}_j[\text{sim}(q_i, z_j)]. \quad (1)$$

The retrieved context $R_K(U)$ then drives the 0xClaw workflow $\mathcal{A}$, which executes seven ordered phases from research to documentation to produce a final product:

$$P = \mathcal{A}(U, R_K(U); B, T), \quad (2)$$

where B and T denote the run-level token and time budgets. Beyond functional code, the task evaluates how well the agent retrieves useful context,

allocates the shared budget across phases, controls project scope, and produces a feasible submission under explicit budget and deadline constraints.

4 Method

4.1 0xClaw Framework

We instantiate the workflow $\mathcal{A}$ from Section 3 by decomposing agent control into two complementary layers (Figure 4). Let $\mathcal{P} = (p_1, \dots, p_7)$ be the fixed phase order from research to documentation. The initial workspace is $\mathcal{W}_1 = \{U, R_K(U)\}$, the URLs and retrieved context from Section 3. The cumulative log is empty, $L_{<1} = \emptyset$. Each phase p_k reads workspace $\mathcal{W}_k$. It then writes a handoff artifact $\mathcal{H}_k$ (e.g., ideas.json for the idea phase, project/ for coding) and an execution log entry ℓ_k that records realized token use. The two streams evolve as

$$\mathcal{W}_{k+1} = \mathcal{U}(\mathcal{W}_k, \mathcal{H}_k), \quad L_{<k+1} = L_{<k} \oplus \ell_k, \quad (3)$$

where $\mathcal{U}$ is a typed update that merges $\mathcal{H}_k$ into the workspace and $\oplus$ appends a log entry. A run terminates after p_7 or earlier if budget B or T is exhausted. The phase-level composite policy is

$$\Pi_k = \mathcal{M} \circ \mathcal{F} \circ \mathcal{R}_\theta, \quad (4)$$

where $\mathcal{R}_\theta$ and $\mathcal{F}$ are invoked once at phase entry. The frozen main agent $\mathcal{M}$ is then sampled repeatedly within the phase under the conditioning (c_k, m_k) . The two controllers see disjoint information. $\mathcal{R}_\theta$ depends only on the current workspace, while $\mathcal{F}$ also reads the cumulative log $L_{<k}$. This split keeps the router Markovian and trainable from static traces. The framer then injects the cumulative-spend signal at the prompt boundary where the agent can actually consume it. Algorithm 1 gives the full procedure.

4.2 Budget Router

The Budget Router $\mathcal{R}_\theta$ predicts a discrete five-field control tuple:

$$c_k = \langle \text{PHASE}_k, \text{TOKEN}_k, \text{TIME}_k, \text{THINK}_k, \text{REPAIR}_k \rangle, \quad (5)$$

where $\text{TOKEN}_k \in \{0.25, 0.5, 0.75, 1.0, 1.25\}$ scales the per-call max-tokens cap, $\text{TIME}_k \in \{\text{LOW}, \text{MEDIUM}, \text{HIGH}\}$ scales the timeout, $\text{THINK}_k \in \{\text{OFF}, \text{LOW}, \text{HIGH}\}$ sets reasoning effort, and $\text{REPAIR}_k \in \{0, 1, 2\}$ caps repair rounds. The grammar admits

$7 \times 5 \times 3 \times 3 \times 3 = 945$ tuples. We refer to c_k as the control token of phase k .

Project-relative supervision. $\mathcal{R}_\theta$ is a Low-Rank Adaptation (LoRA) adapter (Hu et al., 2022) over Qwen3.5-2B. We use θ for the adapter parameters, distinct from the query encoder of Section 3. Training data comes from the HackRep corpus of public hackathon repositories (Halmans et al., 2026). Each project is expanded into seven phase-level instances and paired with a workspace feature vector (language, topics, repository size, README length, keyword group counts, scientific-content probability). Targets are produced by weakly supervised labeling functions (Ratner et al., 2017) over signals including code size, project duration, README structure, API and deployment complexity, test signals, commit churn, and team complexity. Phase scores are centered within each project before label assignment. This step rules out the trivial all-high-budget solution. A large project no longer maps to high budgets across every phase, but to a specific allocation pattern (e.g., coding-heavy versus research-heavy). The router therefore learns an allocation policy over phases, not a regression on absolute project size. At inference time the router operates on prompt-visible workspace features alone. The post-hoc signals that drive the labeling functions, such as final code size, commit churn, and deployment outcome, are by construction unavailable. The router learns a predictive mapping from prompt-time features to the labeling targets, not a recapitulation of training labels. Section 5 confirms this gap empirically.

Algorithm 1 0xClaw($U, \mathcal{P}; B, T$): phase-aware budget routing and deadline framing

Require: Problem URLs U , phase order $\mathcal{P} = (p_1, \dots, p_7)$, run budgets B and T

- 1: $\mathcal{W}_1 \leftarrow \{U, R_K(U)\}; \quad L_{<1} \leftarrow \emptyset$
- 2: **for** $k = 1, \dots, 7$ **do**
- 3: $c_k \leftarrow \mathcal{R}_\theta(p_k, \mathcal{W}_k) \triangleright$ control tuple, Eq. 5
- 4: $m_k \leftarrow \mathcal{F}(c_k, L_{<k}) \triangleright$ deadline instruction
- 5: $(\mathcal{H}_k, \ell_k) \leftarrow \mathcal{M}(p_k, \mathcal{W}_k, m_k; c_k) \triangleright$ run main agent under c_k, m_k
- 6: $\mathcal{W}_{k+1} \leftarrow \mathcal{U}(\mathcal{W}_k, \mathcal{H}_k); \quad L_{<k+1} \leftarrow L_{<k} \oplus \ell_k$
- 7: **end for**
- 8: **return** submission P extracted from $\mathcal{W}_8$

Mapping to runtime. The tuple c_k maps deterministically to the agent’s effective `ModelProfile` for phase k . The fields `max_tokens`, `timeout`,

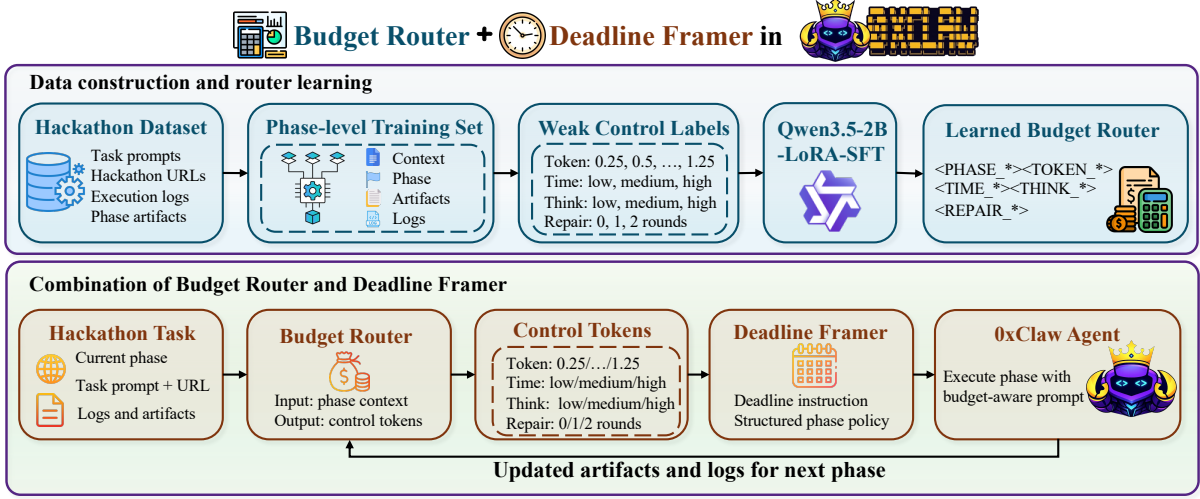


Figure 4: **The framework of 0xClaw.** The **Budget Router** learns per-phase control tokens from hackathon traces. The **Deadline Framer** converts each token into execution guidance for the seven-phase agent pipeline.

reasoning_effort, and repair_rounds follow floor-clipped multiplication or fixed lookup against per-phase base values. A run-wide budget clamp then trims the phase ceiling to fit the remaining run budget. A safe default (TOKEN=1.0, TIME=MEDIUM, THINK=LOW, REPAIR=0) is applied on parser failure.

4.3 Deadline Framer

$\mathcal{R}_\theta$ is Markovian on $(p_k, \mathcal{W}_k)$ and cannot react to budget overruns in prior phases. $\mathcal{F}$ adds that signal at the prompt layer. Given c_k and the cumulative log $L_{<k}$, it produces a deadline instruction m_k that conditions the main agent for phase p_k . $\mathcal{F}$ has no learned parameters and issues no LLM call.

Pipeline state and pressure tier. $\mathcal{F}$ summarizes $L_{<k}$ as a typed state

$$\sigma_k = \{(p_j, B_j^{\text{pred}}, B_j^{\text{used}}) : j < k\}, \quad (6)$$

where B_j^{pred} is the per-call max-tokens cap applied for phase j and B_j^{used} is its realized completion-token spend. To prevent short prior completions from appearing to free budget for later phases, $\mathcal{F}$ clips $B_j^{\text{used}} \leftarrow \max(B_j^{\text{used}}, B_j^{\text{pred}})$ before computing pressure. The pair (c_k, σ_k) is then mapped to a discrete pressure tier that captures how tight cumulative spend is at phase k relative to the planned schedule. The tier modulates the reading of TIME_k . The same $\text{TIME}_k = \text{HIGH}$ tag is read more strictly under historical overspending than under nominal spending.

Tier-based satisficing. From the tier together with TOKEN_k , $\mathcal{F}$ derives a structured policy

that names the intended execution scope and the stop policy. Looser tiers permit full scope and quality-first behavior. Tighter tiers progressively narrow the scope and stop earlier (e.g., *stop after first runnable artifact* under tight pressure), in the spirit of satisficing under resource pressure (Simon, 1955). $\mathcal{F}$ renders this policy into m_k as imperative prose. At phase entry m_k is concatenated with the phase command into the main agent’s user message. Numerical budgets are omitted from m_k by design. Frozen agents track numerical resource state poorly (Sehgal et al., 2026). Qualitative tiers instead act as a satisficing handle the agent can actually use.

5 Experiments

5.1 Experimental Setup

We evaluate 0xClaw in two layers. Layer 1 trains the Budget Router on hackathon traces and measures offline policy accuracy. Layer 2 runs the full controller stack on 20 unseen hackathons against ten agent baselines.

Budget Router. The router is a LoRA (Hu et al., 2022) adapter over Qwen3.5-2B, trained on 50,669 HackRep (Halmans et al., 2026) projects (2,819/2,823 val/test) with weakly supervised control tuples following Section 4.2. It predicts the five-field control tuple in Eq. 5. We compare against four non-neural baselines: **static default**, **phase majority**, **feature heuristic** and a **lightweight ML decision-table classifier**, together with Qwen3.5-2B zero-shot and few-shot prompting and cross-model prompting checks on three foundation models (Appendix Table 6). HackRep

filtering, label construction, training hyperparameters, and baseline mechanics are in Appendix A.1. **Hackathon benchmark.** Twenty completed public hackathons disjoint from HackRep span web apps, data tools, generative interfaces, and developer tooling. Agents receive only the competition URL and task description; winner entries are withheld at runtime and used solely for contamination checks. We compare ten agent baselines: OpenClaw (OpenClaw Contributors, 2026), OpenHands (Wang et al., 2025), OpenHands-Versa (Soni et al., 2026), AI-Researcher (Tang et al., 2026), ARIS (Yang et al., 2026)¹, Hermes-agent (Nous Research, 2026), OpenCode (OpenCode Contributors, 2026), Copilot CLI (GitHub Next, 2026), Claude Code (Anthropic, 2026), and Codex (OpenAI, 2026). All share the DeepSeek V4 Flash backbone, isolating the controller contribution from base-model strength. Run protocol and contamination control are in Appendix A.2.

Evaluation. Outputs are scored by Claude Opus 4.7 (temperature 0, thinking disabled) on anonymized, order-randomized inputs (Zheng et al., 2023; Wang et al., 2023). **Quality** is a six-component composite (Relative Improvement to Human, Logical Lines of Code (LLoC), Novelty, Usefulness, Semantic Alignment, Distance); **Gate** ($n/20$) requires all seven phase artifacts together with a smoke-test pass (Song et al., 2026a); **pass@1** (Chen et al., 2021) is the fraction whose first end-to-end run is Gate-passing. We score the first run only, prioritizing validity over probabilistic scaling (Song et al., 2026c). The deterministic Gate and pass@1 metrics triangulate the judge-driven quality result. Component definitions and judge protocol are in Appendix A.2.

5.2 Budget Router Policy Evaluation

We evaluate the router on a fixed 1,000-instance test subset, averaged over five complete runs. Training dynamics are in Figure 7, and prompting baselines for additional foundation models are in Appendix Table 6. Every method in Table 1 reaches a parse rate of 1.000 except Qwen3.5-2B zero-shot (0.9931), so we omit the parse column and report the structured-prediction metrics directly.

The LoRA router dominates every non-neural baseline (Table 1). Average accuracy rises from 0.8159 (Lightweight ML) to 0.9618, a relative gain

of 17.88%. Exact Match jumps from 0.6656 to 0.9301, Token Accuracy from 0.7531 to 0.9530, and Token MAE drops from 0.2527 to 0.0470, an 81.40% reduction. The router does not collapse to a single majority class. It learns the mapping from phase and workspace context to a discrete control tuple.

Figure 5 shows the resulting label distribution. The TOKEN tier shifts from $0.75\times$ dominant in research (86%) to $1.25\times$ dominant in coding (78%), with planning sitting at 53% at $1.25\times$. The TIME tier is mostly medium, except for coding (39% high) and planning (18% high). These phase-conditioned distributions are the pattern the Budget Router needs to learn.

Prompting alone does not solve budget routing. Qwen3.5-2B zero-shot and few-shot both underperform every non-neural baseline on the structured fields. Larger foundation models behave the same: GPT-5.4-mini, DeepSeek V4 Flash, and Z.ai GLM-5.1 all stay below 0.55 average accuracy across prompting setups (Appendix Table 6). Structured control prediction needs supervised fine-tuning (SFT).

Table 2 reports per-phase Exact Match. The router clears 0.91 on every phase and gains 20.8–58.5% relative to the strongest non-SFT baseline per phase, or 16–34 absolute points. The largest gain is on research (+58.52%), where static and phase-majority baselines hover near 0.14–0.48 because the research phase has high label entropy. Per-phase uniformity rules out the explanation that the router fits a few easy phases.

5.3 Hackathon Benchmark

As shown in Table 3, under the Claude Opus 4.7 judge, 0xClaw reaches the highest pass@1 and near-best Quality across the 20 tasks. Quality is 77.0, only 0.1 lower than Claude Code (77.1), the strongest baseline, and +2.4 over ARIS, the second best baseline. pass@1 reaches 1.00, matching Claude Code, with ARIS at 0.95. Gate pass rate is 20/20, one above ARIS.

Quality gains come without proportional token spend. 0xClaw uses 3.72M tokens on average, 59.4% fewer than Claude Code (9.17M) and 14% fewer than ARIS (4.32M). Lighter baselines such as OpenCode (0.89M) and Copilot CLI (1.04M) use fewer tokens but stop earlier, giving Quality scores 5.6–8.9 points below 0xClaw. Quality per token, the ratio that captures this trade-off, is 20% higher for 0xClaw than for ARIS.

¹We adapt ARIS to engineering tasks; the supplementary material refers to this adaptation as Auto-claude-code-research-in-sleep.

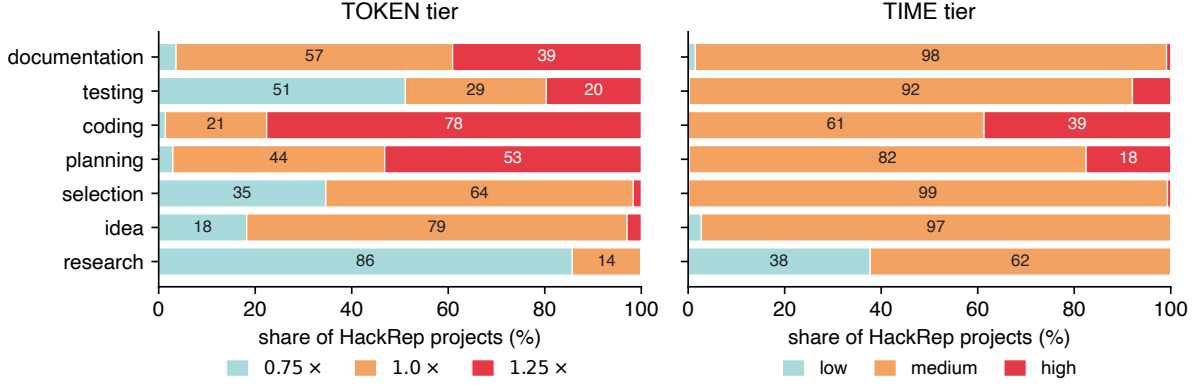


Figure 5: **HackRep per-phase budget distribution.** Share of HackRep projects falling in each TOKEN tier (left) and TIME tier (right), broken down by phase. TOKEN demand shifts from $0.75\times$ dominant in research to $1.25\times$ dominant in coding; TIME pressure peaks at coding (39% high) and planning (18% high).

Method	Exact $\uparrow$	TOKEN $\uparrow$	TIME $\uparrow$	THINK $\uparrow$	REPAIR $\uparrow$	Token MAE $\downarrow$	Average $\uparrow$
Static default	0.3617	0.4380	0.8424	0.6844	0.7104	0.5620	0.6074
Phase majority	0.5260	0.6640	0.8424	0.8173	0.8283	0.3723	0.7356
Feature heuristic	0.4549	0.5684	0.8290	0.7711	0.7930	0.4921	0.6833
Lightweight ML	<u>0.6656</u>	<u>0.7531</u>	<u>0.9067</u>	<u>0.8629</u>	<u>0.8911</u>	<u>0.2527</u>	<u>0.8159</u>
Qwen3.5-2B zero-shot	0.0029	0.0640	0.7066	0.2490	0.6922	1.7503	0.3429
Qwen3.5-2B few-shot	0.3486	0.4393	0.7620	0.6844	0.6927	0.5607	0.5854
Qwen3.5-2B LoRA SFT (Ours)	0.9301	0.9530	0.9743	0.9710	0.9806	0.0470	0.9618
	+39.74%	+26.54%	+7.46%	+12.53%	+10.04%	-81.40%	+17.88%

Table 1: **Budget Router policy evaluation.** Offline accuracy on the held-out test set. Values are on a 0–1 scale; Token MAE (mean absolute error) is lower-is-better. Red annotations on the final row mark relative gains over Lightweight ML, the strongest baseline without supervised fine-tuning (SFT).

OxClaw scores 80.9 on Usefulness, only 0.3 below Claude Code and 2.7 above ARIS (78.2). Novelty is 51.9, the highest in the column, 0.9 above OpenHands (51.0). Semantic Alignment and Distance stay within 2 points of the leading baselines, indicating outputs remain aligned with the task while staying distant from the winning reference. The wall-clock cost is about $2.5\times$ ARIS, driven by per-run router inference and longer controlled execution. The full breakdown is in Appendix Table 8. Results under a second judge, GPT-5.5, are in Appendix Table 7.

Statistical analysis. Wilson 95% confidence intervals (Wilson, 1927) for pass@1 give [0.84, 1.00] for OxClaw, [0.76, 0.99] for ARIS, and [0.53, 0.89] for Hermes-agent. The OxClaw lower bound (0.84) sits above the point estimates of seven of the ten baselines. The remaining systems cluster near the binary ceiling, which reflects pass@1 saturation rather than a non-significant gap, so we

base the discriminative comparison on Quality. A paired Wilcoxon signed-rank test (Wilcoxon, 1945; Demšar, 2006) on per-task Quality against ARIS over the 20 tasks yields $W = 121.5$, $p = 0.002$ (one-sided, exact), with paired Cohen’s $d_z = 0.70$. OxClaw beats ARIS on 13 of the 20 tasks, ties on 4, and loses on 3. Per-task scores are released with the code so the test can be reproduced exactly.

5.4 External Coding Benchmarks

To test generalization beyond hackathons, we sample 20 tasks from each of three external coding benchmarks: Aider (Aider-AI, 2024), CanItEdit (Cassano et al., 2023), and SWE-bench (Jimenez et al., 2024) (Table 4). All systems use the same DeepSeek V4 Flash backbone as the main benchmark; the only difference is the agent platform: OpenCode, Codex, and OxClaw. OxClaw achieves pass@1 of 1.00, 0.80, and 0.90 on the three subsets. On these benchmarks, which lack the open goals

Method	Research ↑	Idea ↑	Selection ↑	Planning ↑	Coding ↑	Testing ↑	Documentation ↑
Static default	0.1440	<u>0.7830</u>	0.6250	0.4380	0.0000	0.0000	0.5420
Phase majority	0.4800	<u>0.7830</u>	0.6250	0.3610	0.3890	0.5020	0.5420
Feature heuristic	0.4180	0.6450	0.5500	0.4130	0.4980	0.2480	0.4120
Lightweight ML	<u>0.5810</u>	0.7810	<u>0.6900</u>	<u>0.5990</u>	<u>0.6960</u>	<u>0.6590</u>	<u>0.6530</u>
Qwen3.5-2B zero-shot	0.0000	0.0000	0.0180	0.0020	0.0000	0.0000	0.0000
Qwen3.5-2B few-shot	0.1270	0.7220	0.6200	0.4350	0.0000	0.0110	0.5250
Qwen3.5-2B LoRA SFT (Ours)	0.9210	0.9460	0.9390	0.9140	0.9390	0.9320	0.9200
	+58.52%	+20.82%	+36.09%	+52.59%	+34.91%	+41.43%	+40.89%

Table 2: **Per-phase exact match of Budget Router.** Exact Match for each of the seven phases on the held-out test set. Red subscripts on the final row mark relative gains over the strongest non-SFT baseline per phase.

Agent	Complete↑	Gate↑	Quality↑	Avg. Tokens↓	LLoC	Novelty↑	Usefulness↑	Align↑	Distance↑	pass@1↑
OpenClaw	20/20	16/20	72.6	1.68M	757.9	49.7	76.9	77.4	97.0	0.80
OpenHands	20/20	16/20	72.2	2.19M	827.5	<u>51.0</u>	76.7	77.3	96.4	0.80
OpenHands-Versa	18/20	11/20	69.5	2.24M	632.2	50.0	74.3	74.7	97.2	0.55
AI-Researcher	17/20	13/20	65.5	1.15M	1478.6	42.2	70.1	71.4	97.9	0.65
ARIS	20/20	<u>19/20</u>	74.6	4.32M	1604.7	48.1	78.2	<u>77.7</u>	97.1	<u>0.95</u>
Hermes-agent	<u>19/20</u>	15/20	72.7	3.91M	1244.7	48.7	76.7	76.9	96.5	0.75
OpenCode	20/20	9/20	68.1	0.89M	442.5	47.7	73.5	73.5	<u>97.6</u>	0.45
Copilot CLI	20/20	18/20	71.4	<u>1.04M</u>	638.0	49.8	76.5	76.1	96.9	0.90
Claude Code	20/20	20/20	77.1	9.17M	1643.2	47.4	81.2	78.4	97.1	1.00
Codex	20/20	10/20	67.9	5.00M	651.1	49.9	74.0	74.9	96.7	0.50
0xClaw-BR+DF (Ours)	20/20	20/20 ₊₁	<u>77.0</u> _{+2.4}	3.72M _{-14%}	1365.9	51.9 _{+3.8}	<u>80.9</u> _{+2.7}	76.7	96.7	1.00 _{+0.05}

Table 3: **Hackathon benchmark.** Agent results across 20 tasks. Red subscripts on the final row mark gains over ARIS, the second-strongest baseline, in absolute units except Avg. Tokens, which is a relative reduction. BR and DF denote Budget Router and Deadline Framer. Runtime metrics are in Appendix Table 8; ablations in Table 5.

and hard deadlines that 0xClaw’s controllers target, 0xClaw transfers without degradation, matching or slightly exceeding both CLI baselines under the same backbone.

Agent	Aider↑	CanItEdit↑	SWE-bench↑
OpenCode	1.00	<u>0.75</u>	<u>0.80</u>
Codex	1.00	<u>0.75</u>	<u>0.80</u>
0xClaw (Ours)	1.00	0.80	0.90

Table 4: **External coding benchmarks.** Pass@1 results on Aider, CanItEdit, and SWE-bench. The best result is bolded and the second-best result is underlined.

5.5 Ablation Study

We compare four variants: **Native** (no controller), **DF-only** (Deadline Framer only), **BR-only** (Budget Router only), and **BR+DF** (both).

The Budget Router and the Deadline Framer are complementary. BR+DF reaches Quality 77.0 with 3.72M tokens. DF-only cuts tokens by 21% over

Native (4.13M→3.25M) at near-identical Quality (75.0→75.5), a direct empirical signature of the satisficing mechanism in Section 4: under deadline framing the agent commits to a smaller scope without quality collapse. BR-only matches BR+DF on Quality (76.9 vs. 77.0) but uses more tokens (3.82M) because the agent keeps exploring when no deadline frame is given. Combining the two cuts token use by 10% over Native, raises Gate pass to 20/20, and lifts pass@1 from 0.90 to 1.00.

5.6 Discussion

In open hackathons, agents can trade arbitrarily many tokens and seconds for marginal Quality gains. 0xClaw makes that trade explicit by separating phase budget from scope. The **Budget Router** decides token spend per phase. The **Deadline Framer** decides scope through qualitative deadline cues. The two are complementary: Budget Router alone keeps Quality but spends more tokens, Deadline Framer alone saves tokens at unchanged Quality, and together they reach the highest Quality,

Variant	BR	DF	Complete $\uparrow$	Gate $\uparrow$	Quality $\uparrow$	Avg. Tokens $\downarrow$	LLoC	Novelty $\uparrow$	Usefulness $\uparrow$	Align $\uparrow$	Distance $\uparrow$	pass@1 $\uparrow$
Native	–	–	20/20	18/20	75.0	4.13M	1245.5	51.4	79.2	78.0	<u>95.8</u>	0.90
DF-only	–	✓	20/20	<u>19/20</u>	75.5	3.25M	1400.5	50.2	79.5	76.5	95.0	<u>0.95</u>
BR-only	✓	–	20/20	20/20	<u>76.9</u>	3.82M	1380.0	52.8	81.0	<u>77.6</u>	95.4	1.00
BR+DF (Ours)	✓	✓	20/20	20/20 ₊₂	77.0 _{+2.0}	3.72M _{−10%}	1365.9	<u>51.9</u>	<u>80.9</u> _{+1.7}	76.7	96.7 _{+0.9}	1.00 _{+0.10}

Table 5: **Ablation on Budget Router (BR) and Deadline Framer (DF)**. A ✓ indicates the module is enabled. Red subscripts on the final row mark gains over the no-module baseline (top row) in absolute units except Avg. Tokens, which is a relative reduction. Runtime metrics are in Appendix Table 8.

Gate, and pass@1 among all 0xClaw variants. The remaining cost is wall-clock: about $2.5\times$ ARIS, attributable to router inference and longer controlled execution. 0xClaw is therefore best suited to token-constrained or quality-critical settings where the wall-clock overhead is amortized over a fixed task budget. Latency-sensitive deployments can mitigate the gap by distilling the router to a smaller policy and batching router inference, as discussed in Section 7.

6 Conclusion

We present 0xClaw, a seven-phase agent framework that decouples resource allocation from behavioral framing under hard deadlines: the Budget Router learns per-phase control tuples via a LoRA-adapted Qwen3.5-2B trained on weakly supervised HackRep traces, and the Deadline Framer converts each tuple with cumulative spend into a qualitative deadline instruction for the main agent. On 20 real hackathon tasks, the system achieves a perfect 1.00 pass@1 and a mean quality score of 77.0, essentially matching the strongest baseline, Claude Code (77.1), while using 59% fewer tokens. This separation improves both quality and token efficiency on open-ended agent tasks, with no changes to the underlying agent.

7 Limitations

Budget Router labels are weak-supervision targets derived from repository-level signals in HackRep (Halmans et al., 2026), the largest publicly available hackathon corpus. They are calibrated to Devpost-style web and software competitions; transfer to hardware-heavy or multi-week tracks is future work. The benchmark contains 20 tasks. Wilson 95% confidence intervals on pass@1 and paired Wilcoxon tests on per-task **Quality** (Section 5.3) address the finite-sample concern, and we release per-task scores for reproduction. 0xClaw spends about $2.5\times$ the wall-clock time of ARIS,

driven by per-phase router inference and longer controlled execution; it suits token-constrained or quality-critical settings rather than latency-sensitive single-run use, with router distillation and batched inference as follow-ups. Phases run in fixed monotonic order, which enforces structural firebreaks and keeps per-phase control tractable. Tasks that revisit early decisions, such as research-style hypothesis loops, fall outside this design; bounded backtracking is a natural extension. **Quality** comes from a frozen LLM judge (Claude Opus 4.7) on anonymized, order-randomized outputs. The deterministic metrics Gate and pass@1 corroborate it, so the judge is not the sole arbiter.

8 Ethical Considerations

Training data for the Budget Router come from the publicly released HackRep corpus, from which we extract aggregate repository signals rather than individual user identities. The hackathon benchmark uses publicly available competition pages and submissions, with no human subjects involved. Agent outputs are anonymized before LLM-judge scoring, and the trained adapter, code, and evaluation scripts are released with the paper.

9 Acknowledgments

Many thanks to UK AI Agent Lab and Z.ai for early hackathon product support. Thanks to Minghan Xiao (Co-founder of UK AI Agent Lab) and Damian Zhang (CEO of Z.ai UK) for early discussions. Thanks to the Imperial College Centre for Cryptocurrency Research and Engineering for device support. Use of AI assistants only in proof-reading and L^AT_EX debugging in this paper. The 0xClaw (research stage and hackathon stage both) project lead is Junhao Song, who is a computing research PhD student fully funded by the Hitachi-Imperial Centre for Decarbonisation and Natural Climate Solutions, a collaboration between Hitachi Ltd, Hitachi Europe and Imperial College London.

References

- Aider-AI. 2024. Polyglot benchmark. <https://github.com/Aider-AI/polyglot-benchmark>. GitHub repository, accessed 22 August 2026.
- Anthropic. 2026. [Claude code: Agentic coding tool](#). Accessed: 2026-08-28.
- Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. 2025. [ResearchAgent: Iterative research idea generation over scientific literature with large language models](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6709–6738, Albuquerque, New Mexico. Association for Computational Linguistics.
- Federico Cassano, Luisa Li, Akul Sethi, Noah Shinn, Abby Brennan-Jones, Jacob Ginesin, Edward Berman, George Chakraborty, Anton Lozhkov, Carolyn Jane Anderson, and 1 others. 2023. Can it edit? evaluating the ability of large language models to follow code editing instructions. *arXiv preprint arXiv:2312.12450*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30.
- Daniel Nygaard Ege, Mark Goudswaard, James Gopsill, Martin Steinert, and Ben Hicks. 2024. What, how and when should i prototype? an empirical study of design team prototyping practices at the idea challenge hackathon. *Design Science*, 10:e22.
- Jeanette Falk, Yiyi Chen, Janet Rafner, Mike Zhang, Johannes Bjerva, and Alexander Nolte. 2025. [How do hackathons foster creativity? towards automated evaluation of creativity at scale](#). In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–23. Association for Computing Machinery.
- GitHub Next. 2026. [Github copilot cli](#). Accessed: 2026-08-28.
- Sjoerd Halmans, Lavinia Paganini, Alexander Serebrenik, and Alexander Nolte. 2026. Hackrep: A large-scale dataset of github hackathon projects. *arXiv preprint arXiv:2603.29672*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. [Token-budget-aware LLM reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiaowu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2024. MetaGPT: Meta programming for multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pages 23247–23275.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.
- Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixin Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. [Aide: Ai-driven exploration in the space of code](#). *Preprint*, arXiv:2502.13138.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.
- Hao Kang, Qingru Zhang, Han Cai, Weiyan Xu, Tushar Krishna, Yilun Du, and Tsachy Weissman. 2025. [Win fast or lose slow: Balancing speed and accuracy in latency-sensitive decisions of LLMs](#). *Preprint*, arXiv:2505.19481.
- Ron Kohavi. 1995. [The power of decision tables](#). In *Machine Learning: ECML-95*, pages 174–189. Springer.
- Xiaozhe Li, Tianyi Lyu, Yizhao Yang, Liang Shan, Siyi Yang, Ligao Zhang, Zhuoyi Huang, Qingwen Liu, and Yang Li. 2026. [Escaping the context bottleneck: Active context curation for llm agents via reinforcement learning](#). *Preprint*, arXiv:2604.11462.
- Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. 2025. [Selfbudgeter: Adaptive token allocation for efficient llm reasoning](#). *Preprint*, arXiv:2505.11274.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.
- Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. 2025. [Budget-aware tool use enables effective agent scaling](#). *Preprint*, arXiv:2511.17006.
- Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. [The ai scientist: Towards fully automated open-ended scientific discovery](#). *Preprint*, arXiv:2408.06292.

- Jean Metz, Newton Spolaôr, Everton A Cherman, and Maria C Monard. 2015. Comparing published multi-label classifier performance measures to the ones obtained by a simple multi-label baseline classifier. *arXiv preprint arXiv:1503.06952*.
- Nous Research. 2026. [Hermes agent](#). Open-source software. Accessed: 2026-05-25.
- OpenAI. 2026. [Codex: Openai coding agent](#). Accessed: 2026-08-28.
- OpenClaw Contributors. 2026. [Openclaw: Personal ai assistant](#). Open-source software. Accessed: 2026-05-25.
- OpenCode Contributors. 2026. [Opencode: Open ai coding agent](#). Open-source software. Accessed: 2026-08-28.
- Ivan Pasichnyk. 2026. [The Yerkes-Dodson curve for AI agents: Emergent cooperation under environmental pressure in multi-agent LLM simulations](#). *Preprint*, arXiv:2603.07360.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 15174–15186.
- Yansheng Qiu, Haoquan Zhang, Zhaopan Xu, Ming Li, Diping Song, Zheng Wang, and Kaipeng Zhang. 2025. AI Idea Bench 2025: AI research idea generation benchmark. *arXiv preprint arXiv:2504.14191*.
- Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB endowment. International conference on very large data bases*, volume 11, page 269.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.
- Xubin Ren and HKU Data Intelligence Lab. 2026. [Nanobot: A lightweight open-source ai agent](#). Open-source software. Accessed: 2026-05-26.
- Neil K. R. Sehgal, Sharath Chandra Guntuku, and Lyle Ungar. 2026. [Real-time deadlines reveal temporal awareness failures in llm strategic dialogues](#). *Preprint*, arXiv:2601.13206.
- Guobin Shen, Dongcheng Zhao, Aorigele Bao, Xiang He, Yiting Dong, and Yi Zeng. 2025. [Stressprompt: does stress impact large language models and human performance similarly?](#) In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’25/IAAI’25/EAAI’25. AAAI Press.
- Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2025. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. In *International Conference on Learning Representations*, volume 2025, pages 94003–94092.
- Herbert A. Simon. 1955. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118.
- Junhao Song, Ziqian Bi, Xinliang Chia, William Knottenbelt, and Yudong Cao. 2026a. [Data verification is the future of quantum computing copilots](#). *Preprint*, arXiv:2602.04072.
- Junhao Song, Lluís Guasch, Xilin He, Zhongyu Yang, Yingfang Yuan, Weicheng Xie, Linlin Shen, Haijun Lin, Shizhe Liu, Wei Pang, and Siyang Song. 2026b. [Conversational human audio-visual talking dialogue generation](#). *Preprint*, arXiv:2607.02799.
- Junhao Song, Yingfang Yuan, and Wei Pang. 2024. [SAIS: A novel bio-inspired artificial immune system based on symbiotic paradigm](#). In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO ’24 Companion)*, pages 2115–2118, New York, NY, USA. Association for Computing Machinery.
- Junhao Song, Yu Zhou, William Knottenbelt, and Yudong Cao. 2026c. [Position: Quantum program generation must prioritize validity over probabilistic scaling](#). *Preprint*, arXiv:2607.15313.
- Aditya Bharat Soni, Boxuan Li, Xingyao Wang, Valerie Chen, and Graham Neubig. 2026. Coding agents with multimodal browsing are generalist problem solvers. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 6052–6069.
- Jiabing Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. 2026. AI-Researcher: Autonomous scientific innovation. *Advances in Neural Information Processing Systems*, 38:9481–9520.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. [Large language models are not fair evaluators](#). *Preprint*, arXiv:2305.17926.
- Xinglin Wang, Zishen Liu, Shaoxiong Feng, Peiwen Yuan, Yiwei Li, Jiayi Shi, Yueqi Zhang, Chuyi Tan, Ji Zhang, Boyuan Pan, Yao Hu, and Kan Li. 2026. [On time, within budget: Constraint-driven online resource allocation for agentic workflows](#). *Preprint*, arXiv:2605.06110.

- Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangu Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2025. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pages 65882–65919.
- Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.
- Edwin B. Wilson. 1927. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212.
- Yizhe Xie, Congcong Zhu, Xinyue Zhang, Tianqing Zhu, Dayong Ye, Minfeng Qi, Huajie Chen, and Wanlei Zhou. 2026. [From spark to fire: Modeling and mitigating error cascades in LLM-based multi-agent collaboration](#). *Preprint*, arXiv:2603.04474.
- Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. [The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search](#). *Preprint*, arXiv:2504.08066.
- John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. [Swe-agent: Agent-computer interfaces enable automated software engineering](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 50528–50652. Curran Associates, Inc.
- Ruofeng Yang, Yongcan Li, and Shuai Li. 2026. Aris: Autonomous research via adversarial multi-agent collaboration. *arXiv preprint arXiv:2605.03042*.
- Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. [MasRouter: Learning to route LLMs for multi-agent systems](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15549–15572, Vienna, Austria. Association for Computational Linguistics.
- Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025a. [Multi-agent architecture search via agentic supernet](#). *Preprint*, arXiv:2502.04180.
- Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2025b. [Aflow: Automating agentic workflow generation](#). *Preprint*, arXiv:2410.10762.
- Yunxiang Zhang, Muhammad Khalifa, Shitanshu Bhushan, Grant Murphy, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. 2026. Mlrc-bench: Can language agents solve machine learning research challenges? *Advances in Neural Information Processing Systems*, 38.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Appendix

A.1 Budget Router Setup Details

HackRep filtering and split. HackRep contains 100,356 hackathon GitHub repositories. After filtering for README length, commit count, abnormal duration, and archive status, 56,311 valid projects remain. We split projects 90/5/5 into 50,669/2,819/2,823 for train, validation, and test. Each project is segmented into the seven phases of research, idea, selection, planning, coding, testing, and documentation.

Weak-supervision labels. HackRep has no token-budget labels. We estimate a per-phase workload score from code size, project duration, README structure, API and deployment complexity, test signals, commit churn, and team complexity. Subtracting the project-level seven-phase mean from each phase prevents the trivial all-high-budget solution. Scores are mapped to a discrete control tuple. A coding-phase label, for example, looks like:

`<PHASE_CODING><TOKEN_1.0X><TIME_MEDIUM>
<THINK_LOW><REPAIR_1>`

TOKEN scales the per-call max-tokens cap, TIME scales the timeout, THINK sets reasoning effort, and REPAIR caps repair rounds.

Training. Training and validation instances are subsampled from the project-level splits: we use 50,000 phase-level instruction examples for training, 4,096 for validation, and a fixed 1,000-instance subset of the test split for evaluation, with a maximum sequence length of 1,024. The adapter is trained for one epoch with LoRA rank 16, alpha 32, dropout 0.05, learning rate 2×10^{-4} , gradient accumulation 16, and a fixed seed. Training runs on an NVIDIA RTX 5090 (32GB) workstation with an Intel Core Ultra 9 285K and 64GB RAM.

Router baselines. **Static default** emits the same control tuple for every phase, a fixed-budget sanity baseline (Metz et al., 2015). **Phase majority** predicts the per-phase majority tuple of the training split (Metz et al., 2015). **Feature heuristic** applies labeling-function-style rules over prompt-visible repository features such as duration, repo size, README length, commits, contributors, language, topics, and keyword counts (Ratner et al., 2017). **Lightweight ML** is a pure-Python backoff decision-table classifier that discretizes the same features, builds feature-conjunction tables, and backs off to less specific signatures and finally to global majority when support is too low (Kohavi, 1995). Qwen3.5-2B zero-shot and few-shot

prompting probe the foundation model without fine-tuning.

Deadline Framer. The framer needs no training. It reads the router’s control tuple together with the cumulative execution log, maps $\text{TOKEN} \times \text{TIME}$ to a discrete pressure tier, and renders a natural-language deadline instruction together with a structured policy summary.

A.2 Evaluation Details

Hackathon protocol. Inputs to each agent contain only the competition URL, the task description, and the expected outputs. We exclude winning entries, winner URLs, submission galleries, and any top-1 reference to keep the agent’s outputs uncontaminated. Winner information is used only at evaluation time to verify that the agent does not regenerate the winning solution. None of the 20 competitions or their repositories appear in HackRep, so the router’s training data and the benchmark share no source repositories. Each baseline runs its native agent loop, and we ask it to organize the output into the same seven phase artifacts at submission time without modifying its internal control flow, prompts, or tool selection.

LLM judge. The judge is Claude Opus 4.7 with temperature 0 and thinking disabled. Outputs are anonymized to remove agent identity before judging (Zheng et al., 2023), and we randomize within-batch order to mitigate position bias (Wang et al., 2023). A second judge, GPT-5.5, keeps Claude Code strongest (85.7) and places 0xClaw-BR+DF level with ARIS (84.8, Table 7). The deterministic non-judge metrics Gate and pass@1 triangulate the judge-driven Quality result and remain unaffected by judge preferences.

Quality components. Quality scores combine Relative Improvement to Human and LLoC (Zhang et al., 2026), Novelty (Qiu et al., 2025; Si et al., 2025), Usefulness (covering technical feasibility) (Falk et al., 2025), and Semantic Alignment and Distance (Falk et al., 2025; Reimers and Gurevych, 2019).

Gate and pass@k. Gate is the fraction of tasks where the submission contains all seven required artifacts and the generated project builds and runs a smoke test, reported as $n/20$. Following Chen et al. (2021), we define pass@1 and pass@k as the fraction of tasks where the agent’s first (or any of the first k) end-to-end run produces a Gate-passing submission.



Figure 6: **Deployment surfaces of 0xClaw.** Top left: a messaging-app channel that reports phase progress, locked scope, and per-phase token usage. Top right: the command-line interface, where `/status` reports the seven-phase pipeline and per-phase completion. Bottom: example front-ends from generated projects.

A.3 Deployment and Training Artifacts

Figure 6 summarizes the deployment surfaces of 0xClaw. Figure 7 reports the training dynamics of the Budget Router LoRA tuning stage. The loss decreases sharply at the beginning of training and then enters a slower convergence regime. To avoid the initial drop obscuring later-stage behavior, we additionally provide a zoomed-in view after the early rapid descent, where both the smoothed training loss and validation loss continue to decrease and stabilize.

Table 8 reports per-agent wall-clock time. Wall-clock is not used as the main sorting criterion because it is affected by network, region, service

provider, local machine state, Docker and sub-process overhead, and API rate limiting. For the 0xClaw series, we additionally extract the agent’s actual execution-phase time and the outer scheduling overhead. For the Budget Router, the local Qwen3.5-2B load and decision times come from a separate calibration run and represent the additional cost of local deployment rather than API model call time.

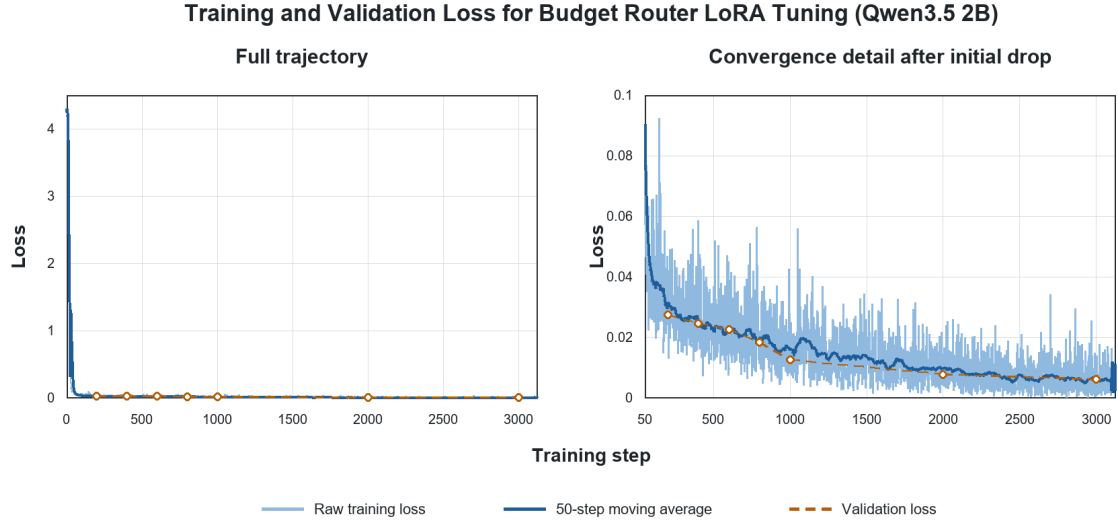


Figure 7: **Budget Router LoRA training and validation loss.** The left panel shows the full training trajectory. The right panel zooms into the post-drop region to make the convergence trend visible.

Model	Prompting	Parse	Exact	Token	Time	Think	Repair	Token MAE	Average
OpenAI GPT-5.4-mini	naive zero-shot	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
OpenAI GPT-5.4-mini	schema zero-shot	1.0000	0.0145	0.1474	0.5642	0.5575	0.6822	1.3617	0.3932
OpenAI GPT-5.4-mini	few-shot	0.9404	0.0763	0.4461	0.3744	0.6786	0.5990	0.5876	0.4349
DeepSeek V4 Flash	naive zero-shot	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
DeepSeek V4 Flash	schema zero-shot	1.0000	0.0026	0.2678	0.6665	0.2269	0.4957	0.9669	0.3319
DeepSeek V4 Flash	few-shot	0.9906	0.2375	0.4797	0.5747	0.6821	0.7481	0.5288	0.5444
Z.ai GLM-5.1	naive zero-shot	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Z.ai GLM-5.1	few-shot	0.9656	0.1155	0.4201	0.3344	0.6781	0.7049	0.6444	0.4506
Qwen3.5-2B raw	naive zero-shot	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Qwen3.5-2B raw	schema zero-shot	0.9931	0.0029	0.0640	0.7066	0.2490	0.6922	1.7503	0.3429
Qwen3.5-2B raw	few-shot	1.0000	0.3486	0.4393	0.7620	0.6844	0.6927	0.5607	0.5854

Table 6: **Cross-model prompting baselines for Budget Router.** Values are on the original 0–1 scale.

Setting	Direct weak labels	Unlabeled pseudo-labels	Joint Exact Match $\uparrow$
Matched Weak-SFT	10,000	0	22.11%
Weak-SFT-guided UST	5,000	5,000	29.49%

Table 9: **Weak-supervision comparison.** Joint exact-match performance of matched Weak-SFT and Weak-SFT-guided uncertainty-aware self-training (UST).

To test whether the router can improve beyond direct weak labels, we run a controlled comparison between matched weakly supervised SFT and weak-SFT-guided uncertainty-aware self-training (UST) on a subset of the full dataset (Table 9). The Weak-SFT baseline uses 10,000 direct weak labels; UST replaces half of them with 5,000 high-confidence pseudo-labels derived from unlabeled item trajectories. Under an identical training bud-

get, UST raises the Joint Budget Decision Exact Match score from 22.11% to 29.49%. The Budget Router need not rely solely on directly aggregated HackRep labels but can further refine its budgeting strategy through self-training on unlabeled item trajectories.

A.4 Qualitative Examples

Figures 8 and 9 show qualitative examples of the projects 0xClaw produced across the 20 hackathon tasks. Figure 8 is a montage of front-ends that illustrates the range of domains and interface styles. Figure 9 zooms into six representative projects and pairs each project brief with the agent’s design pipeline and the resulting interface.

Agent	Complete $\uparrow$	Gate $\uparrow$	Quality $\uparrow$	Avg. Tokens $\downarrow$	LLoC	Novelty $\uparrow$	Usefulness $\uparrow$	Align $\uparrow$	Distance $\uparrow$	pass@1 $\uparrow$
OpenClaw	20/20	20/20	80.8	1.68M	757.9	56.4	82.8	79.1	97.0	1.00
OpenHands	20/20	20/20	81.3	2.19M	827.5	57.5	82.9	79.1	96.4	1.00
OpenHands-Versa	18/20	18/20	78.7	2.24M	632.2	56.0	80.6	78.0	97.2	0.90
AI-Researcher	17/20	15/20	74.6	1.15M	1478.6	49.2	76.2	73.6	97.9	0.75
ARIS	20/20	20/20	84.8	4.32M	1604.7	57.3	86.2	80.3	97.1	1.00
Hermes-agent	19/20	18/20	82.6	3.91M	1244.7	54.8	84.2	79.8	96.5	0.90
OpenCode	20/20	19/20	78.0	0.89M	442.5	48.4	80.0	76.9	97.6	<u>0.95</u>
Copilot CLI	20/20	19/20	81.8	<u>1.04M</u>	638.0	51.6	84.1	79.7	96.9	<u>0.95</u>
Claude Code	20/20	19/20	85.7	9.17M	1643.2	57.6	87.4	81.2	97.1	<u>0.95</u>
Codex	20/20	14/20	74.7	5.00M	651.1	44.4	77.5	76.0	96.7	0.70
0xClaw-Native	20/20	20/20	84.3	4.13M	1245.5	57.6	86.3	80.4	95.8	1.00
0xClaw-DF-only	20/20	20/20	84.2	3.25M	1400.5	56.6	86.6	79.2	95.0	1.00
0xClaw-BR-only	20/20	20/20	<u>85.6</u>	3.82M	1380.0	59.7	87.4	<u>80.6</u>	95.4	1.00
0xClaw-BR+DF (Ours)	20/20	20/20	84.8 _{+0.0}	3.72M _{-14%}	1365.9	57.8 _{+0.5}	86.7 _{+0.5}	79.5	96.7	1.00 _{+0.00}

Table 7: **Hackathon benchmark with a second LLM judge.** Agent results across 20 tasks evaluated by GPT-5.5. Red subscripts on the final row mark gains over ARIS, the second-strongest baseline, in absolute units except Avg. Tokens, which is a relative reduction. BR and DF denote Budget Router and Deadline Framer. Runtime metrics are in Appendix Table 8; ablations in Table 5.

Agent	Avg. Time(s)	Agent Phase(s)	Outer Overhead(s)	Outer Overhead Rate	Est. Router Load(s)	Est. Router Decide(s)
OpenClaw	336.3	—	—	—	—	—
OpenHands	518.4	—	—	—	—	—
OpenHands-Versa	517.3	—	—	—	—	—
AI-Researcher	307.6	—	—	—	—	—
ARIS	458.1	—	—	—	—	—
Hermes-agent	529.0	—	—	—	—	—
0xClaw-Native	644.8	630.0	14.8	2.6%	—	—
0xClaw-DF-only	616.8	604.1	12.8	2.2%	—	—
0xClaw-BR-only	1059.3	606.3	453.0	43.7%	32.4	291.1
0xClaw-BR+DF	1179.9	776.0	403.9	35.4%	32.4	291.1

Table 8: **Runtime diagnostics.** Wall-clock time is reported separately because it is affected by network, region, machine state, and local process overhead. Budget Router load and decision times come from a separate local calibration. The overhead rate is averaged over runs, so it does not equal the ratio of the reported column means.

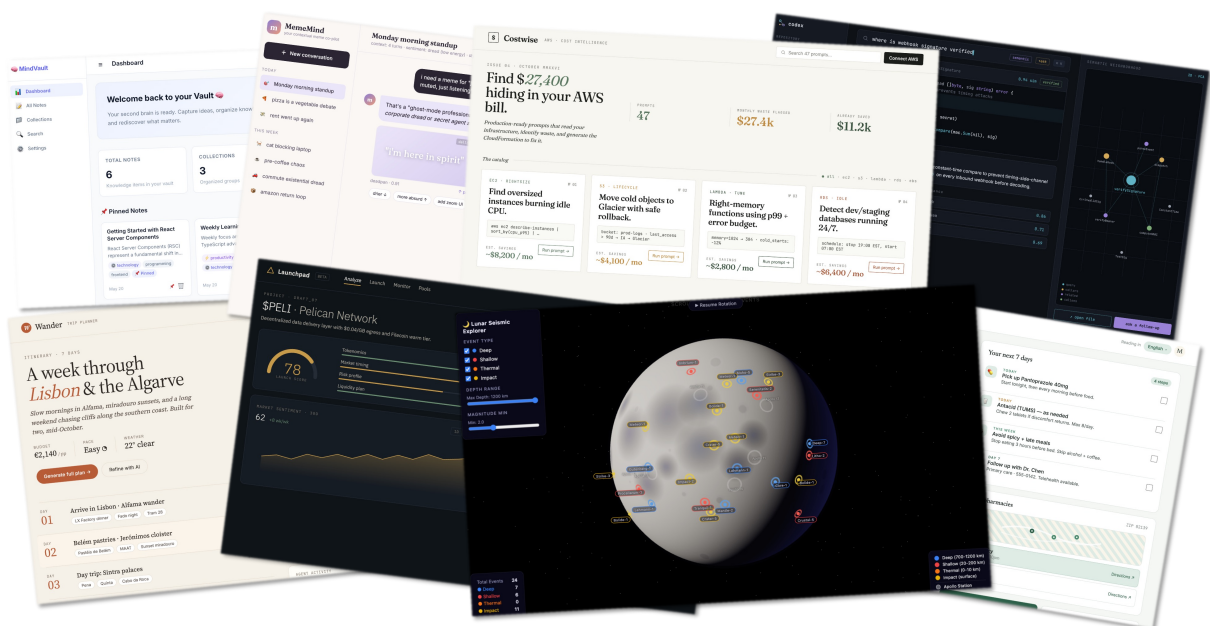


Figure 8: **Project front-end gallery.** A montage of front-ends from projects produced by 0xClaw across the 20 hackathon tasks, covering note-taking, cloud-cost analysis, code search, trip planning, a token launchpad, and lunar seismic visualization.

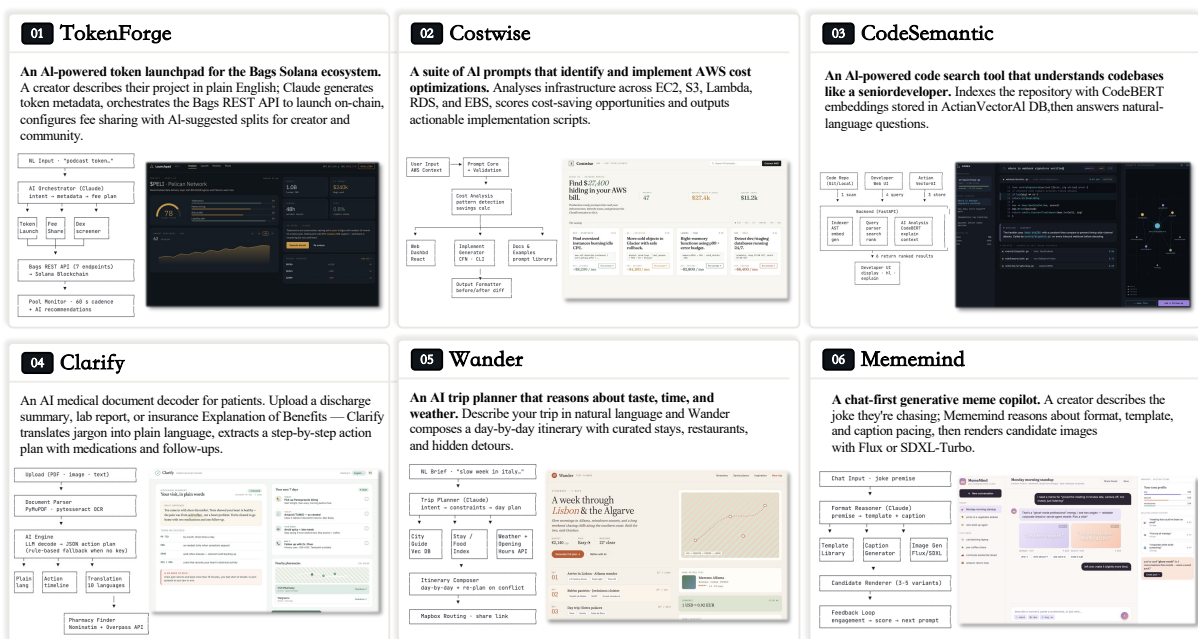


Figure 9: **Representative 0xClaw projects.** Six end-to-end projects spanning a Solana token launchpad, AWS cost optimization, semantic code search, medical document decoding, trip planning, and meme generation. Each panel shows the project brief, the agent’s design pipeline, and the resulting interface.