2025-07-28 · Research

GLM-4.5: Reasoning, Coding, and Agentic Abililties

Our latest frontier model series, excels in reasoning, coding and agentic tasks.

Try it at Z.ai ↗ Call it at Z.ai ↗ GitHub ↗ HuggingFace ↗

Today, we introduce two new GLM family members: GLM-4.5 and GLM-4.5-Air — our latest flagship models. GLM-4.5 is built with 355 billion total parameters and 32 billion active parameters, and GLM-4.5-Air with 106 billion total parameters and 12 billion active parameters. Both are designed to unify reasoning, coding, and agentic capabilities into a single model in order to satisfy more and more complicated requirements of fast rising agentic applications.

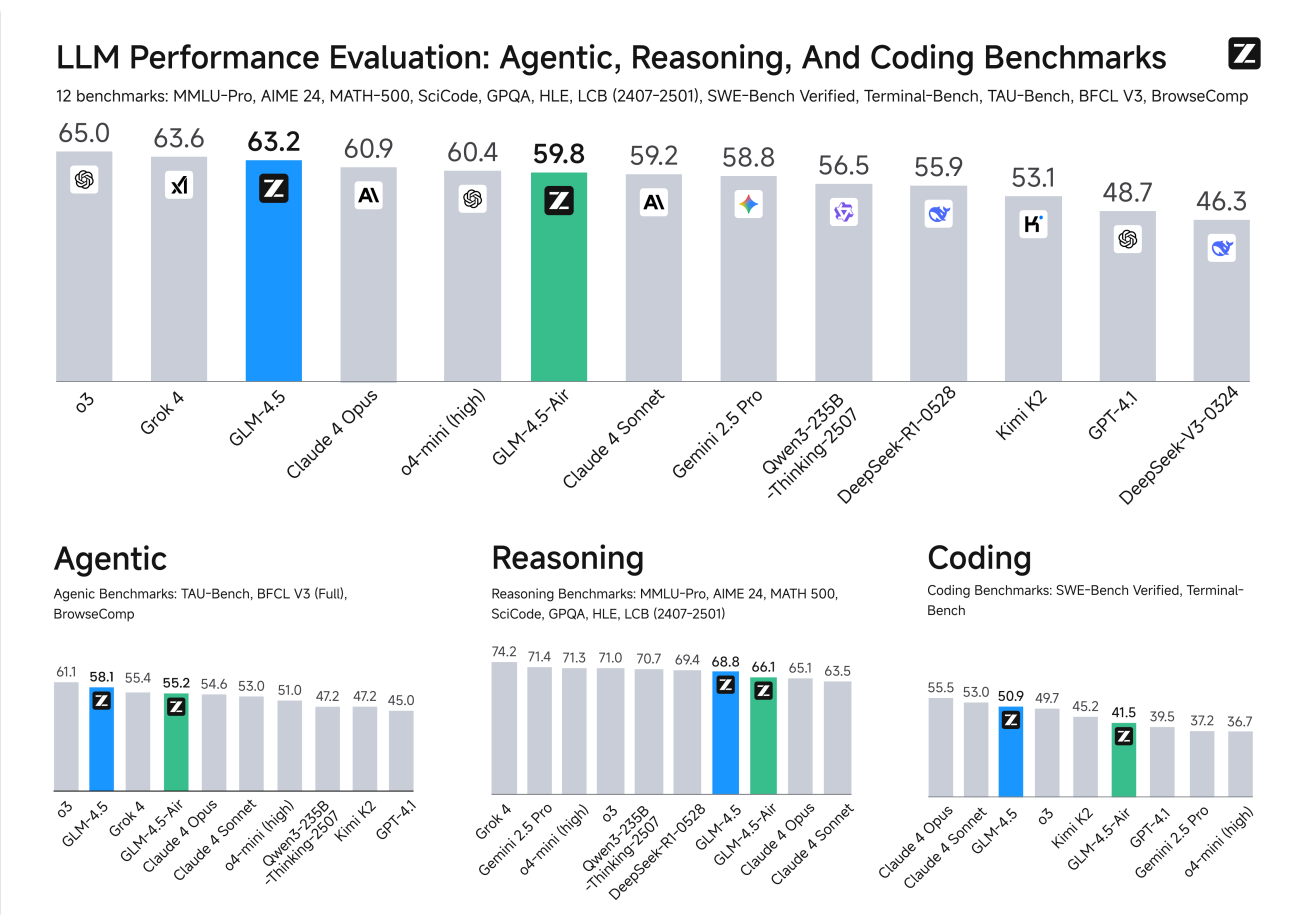
Both GLM-4.5 and GLM-4.5-Air are hybrid reasoning models, offering: *thinking* mode for complex reasoning and tool using, and *non-thinking* mode for instant responses. They are available on Z.ai, Z.ai API and open-weights are available at HuggingFace and ModelScope.

Background: LLM always targets at achieving human-level cognitive capabilities across a wide range of domains, rather than designed for specific tasks. As a good LLM model, it is necessary to deal with general problem solving, generalization, commen sense reasoning, and self-improvement. In the past five years, OpenAI's GPT-3 learns commen-sense knowledge, and o1 uses reinforcement learning to think before respond, significantly improving reasoning skills in coding, data analysis, and complex math. However, the resultant models are still not really general: some of them are good at coding, some good at math, and some good at reasoning, but none of them could achieve the best performance across all the different tasks. GLM-4.5 m
efforts toward the goal of unifying all the different capabilities.

中

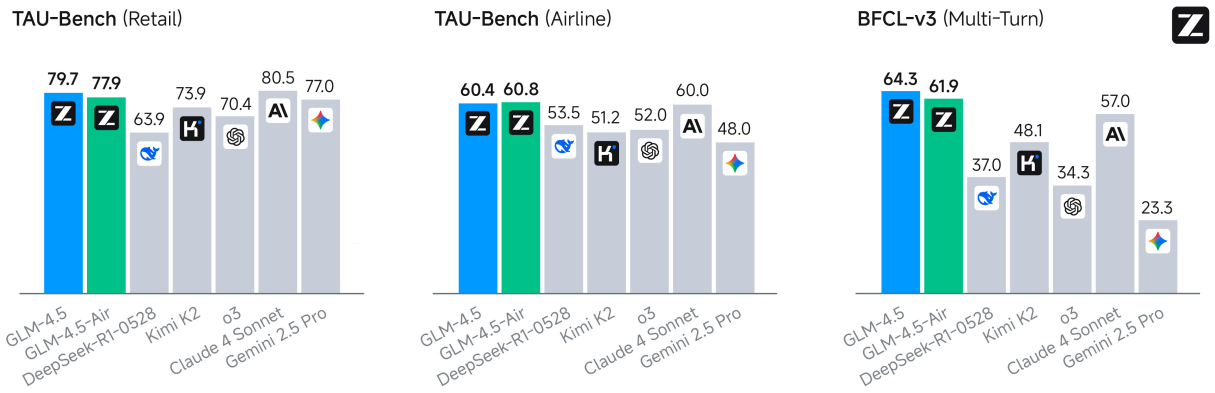
Overall Performance

We compare GLM-4.5 with various models from OpenAI, Anthropic, Google DeepMind, xAI, Alibaba, Moonshot, and DeepSeek on 12 benchmarks covering agentic (3), reasoning (7), and Coding (2). Overall, GLM-4.5 is ranked at the 3rd place and GLM-4.5 Air is ranked at the 6th.

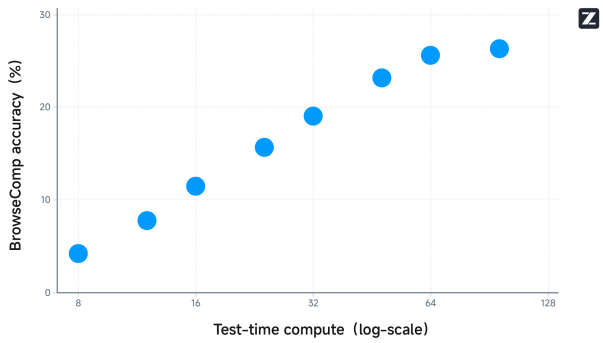


Agentic Tasks

GLM-4.5 is a foundation model optimized for agentic tasks. It provides 128k context length and native function calling capacity. We measure its agent ability on τ-bench and BFCL-v3 (Berkeley Function Calling Leaderboard v3). On both benchmarks, GLM-4.5 matches the performance of Claude 4 Sonnet.



Web browsing is a popular agentic application that requires complex reasoning and multi-turn tool using. We evaluate GLM-4.5 on the BrowseComp benchmark, a challenging benchmark for web browsing that consists of complicated questions that expect short answers. With access to the web browsing tool, GLM-4.5 gives correct answers for 26.4% of all questions, clearly outperforming Claude-4-Opus (18.8%) and close to o4-mini-high (28.3%). Below the figure shows the test-time scaling accuracy of GLM-4.5 on the BrowseComp.



All the detailed results of different comparison models on the three Benchmarks used for eveluating model agent ability are listed in the following table.

Benchmark	GLM-4.5	GLM-4.5-Air	o3	o4-mini-high	GPT-4.1	Claude 4 Opus	Claude 4 Sonnet	Gemini 2.5 Pro	Qwen3 235B Thinking 2507	DeepSe R1-05
TAU-bench-Retail	79.7	77.9	70.4	65.6	75.1	81.4	80.5	77.0	71.9	63.9

Benchmark	GLM-4.5	GLM-4.5-Air	o3	o4-mini-high	GPT-4.1	Claude 4 Opus	Claude 4 Sonnet	Gemini 2.5 Pro	Qwen3 235B Thinking 2507	DeepSe R1-05
TAU-bench-Airline	60.4	60.8	52.0	49.2	48.8	59.6	60.0	48.0	58.0	53.5
BFCL v3 (Full)	77.8	76.4	72.4	67.2	68.9	74.4	75.2	61.2	71.9	63.8
BrowseComp	26.4	21.3	49.7	28.3	4.1	18.8	14.7	7.6	4.6	3.2

> *For TAU-bench, we use an optimized user simulator for both Retail and Airline domains. The user prompt we use can be found in Appendix.*

Reasoning

Under the thinking mode, GLM-4.5 and GLM-4.5-Air can solve complex reasoning problems including mathematics, science, and logical problems.

Benchmark	GLM-4.5	GLM-4.5-Air	o3	Claude 4 Opus	Gemini 2.5 Pro	DeepSeek-R1-0528	Qwen3-235B-Thinking 2507	Grok 4
MMLU Pro	84.6	81.4	85.3	87.3	86.2	84.9	84.5	86.6
AIME24	91.0	89.4	90.3	75.7	88.7	89.3	94.1	94.3
MATH 500	98.2	98.1	99.2	98.2	96.7	98.3	98.0	99.0
SciCode	41.7	37.3	41.0	39.8	42.8	40.3	42.9	45.7
GPQA	79.1	75.0	82.7	79.6	84.4	81.3	81.1	87.7
HLE	14.4	10.6	20.0	11.7	21.1	14.9	15.8	23.9
LiveCodeBench (2407-2501)	72.9	70.7	78.4	63.6	80.1	77.0	78.2	81.9

Benchmark	GLM-4.5	GLM-4.5-Air	o3	Claude 4 Opus	Gemini 2.5 Pro	DeepSeek-R1-0528	Qwen3-235B-Thinking 2507	Grok 4
AA-Index (Estimated)	67.7	64.8	70.0	64.4	70.5	68.3	69.4	73.2

> *For the AIME and GPQA benchmarks, we report the average accuracy over 32 and 8 samples respectively (Avg@32, Avg@8) to mitigate result variance. An LLM was used for automated answer validation. For the HLE benchmark, only the text-based questions were evaluated, with correctness judged by gpt-4o.*

Coding

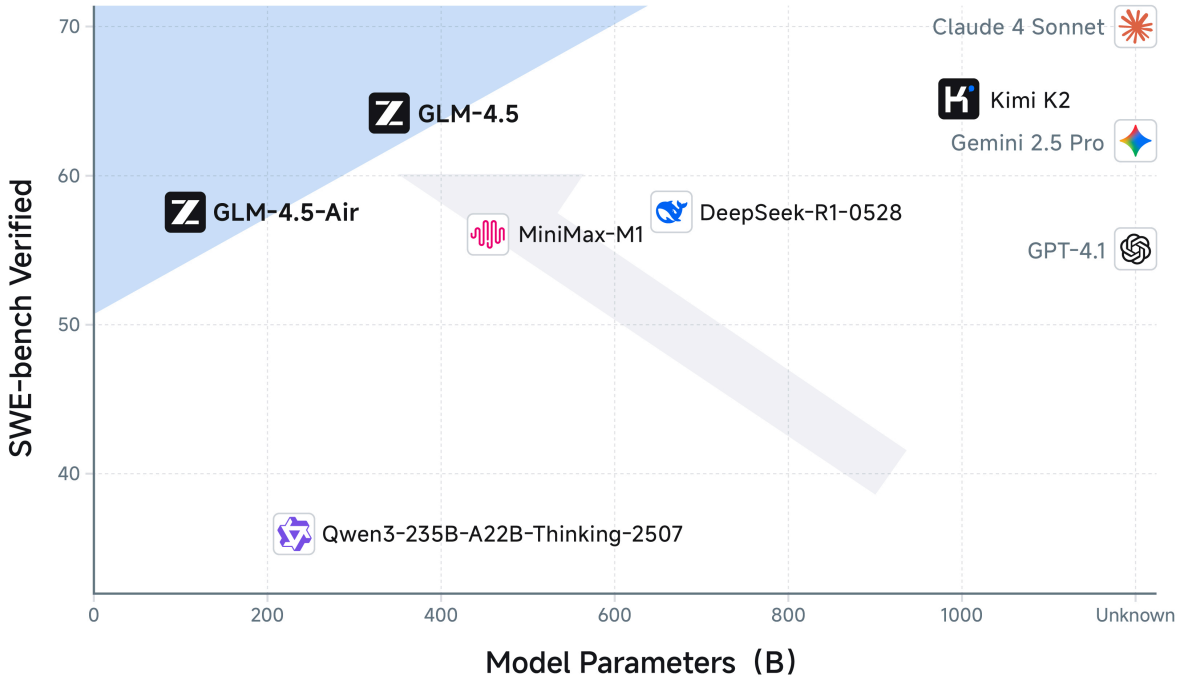
GLM-4.5 excels at coding, including both building coding projects from scratch and agentically solving coding tasks in existing projects. It can be seamlessly combined with existing coding toolkits such as Claude Code, Roo Code, and CodeGeex. To evaluate the coding capability, we compared different models on SWE-bench Verified and Terminal Bench. The following table presents the results.

Benchmark	GLM-4.5	GLM-4.5-Air	o3	GPT-4.1	Claude 4 Opus	Claude 4 Sonnet	Gemini 2.5 Pro	DeepSeek-R1-0528	Kimi K2
SWE-bench Verified[1]	64.2	57.6	69.1	48.6	67.8	70.4	49.0	41.4	65.4
Terminal-Bench[2]	37.5	30	30.2	30.3	43.2	35.5	25.3	17.5	25.0

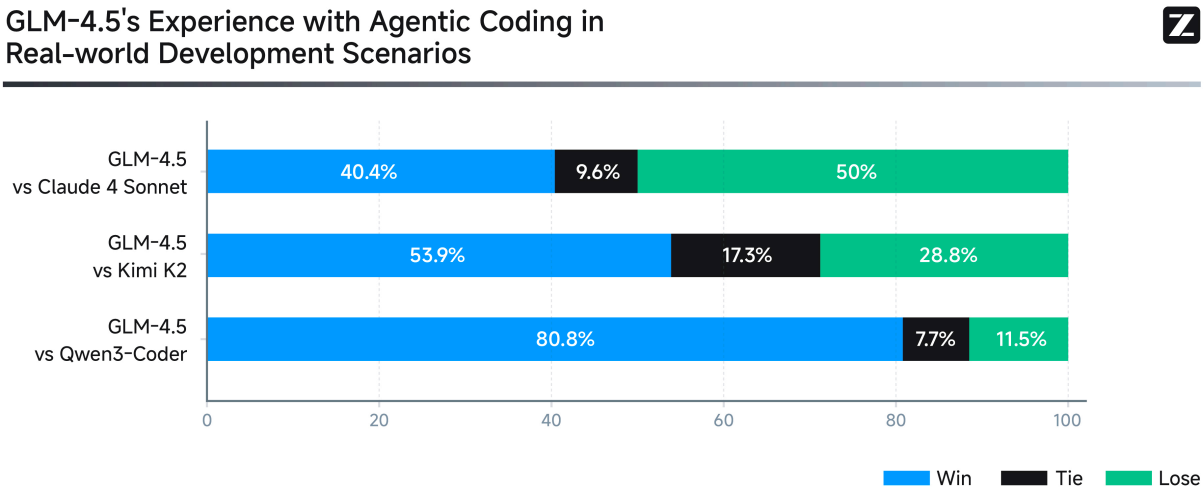
> *[1] For SWE-bench Verified, we use OpenHands v0.34.0 with runs limited to 100 iterations and history truncation to prevent exceeding the 128K context limit, configured with temperature=0.6, top_p=1.0.*
> *[2] For Terminal-Bench, we use the Terminus framework for evaluation. We use standard function calling rather than direct prompting for evaluation.*

We conducted a Pareto Frontier analysis for all comparison models (as illustrated in the figure below). GLM-4.5 and GLM-4.5-Air demonstrate superior performance relative to models of comparable scale,

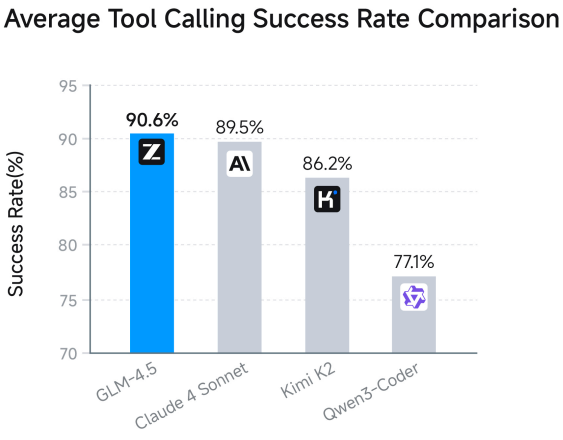
achieving optimal efficiency on the performance-scale trade-off boundary.

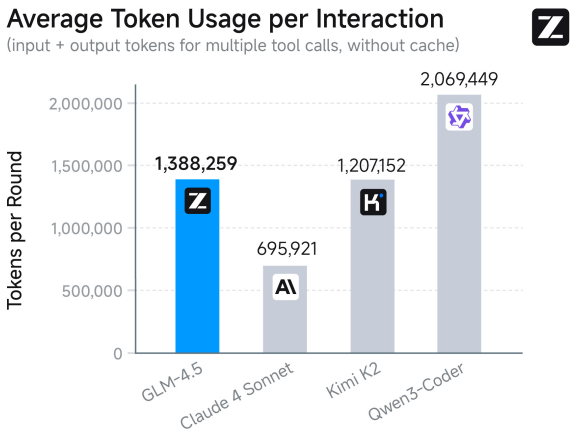


GLM-4.5 demonstrates comprehensive full-stack development capabilities, enabling seamless creation of web applications that encompass frontend implementation, database management, and backend deployment. The frontend interfaces generated by GLM-4.5 exhibit enhanced functionality and aesthetic appeal, demonstrating strong alignment with human design preferences. Furthermore, GLM-4.5 exhibits superior performance in generating presentation materials, including slides and posters, with capabilities significantly augmented when integrated with agentic tools for information retrieval and contextual enhancement.



To assess GLM-4.5's agentic coding capabilities, we utilized Claude Code to evaluate performance against Claude-4-Sonnet, Kimi K2, and Qwen3-Coder across 52 coding tasks spanning frontend development, tool development, data analysis, testing, and algorithm implementation. All evaluations were performed in isolated testing environments through multi-round human interaction with standardized evaluation criteria to ensure consistency and reproducibility. The empirical results demonstrate that GLM-4.5 achieves a 53.9% win rate against Kimi K2 and exhibits dominant performance over Qwen3-Coder with an 80.8% success rate. While GLM-4.5 shows competitive performance, further optimization opportunities remain when compared to Claude-4-Sonnet.





Notably, GLM-4.5 achieves the highest average tool calling success rate at 90.6%, outperforming Claude-4-Sonnet (89.5%), Kimi-K2 (86.2%), and Qwen3-Coder (77.1%), demonstrating superior

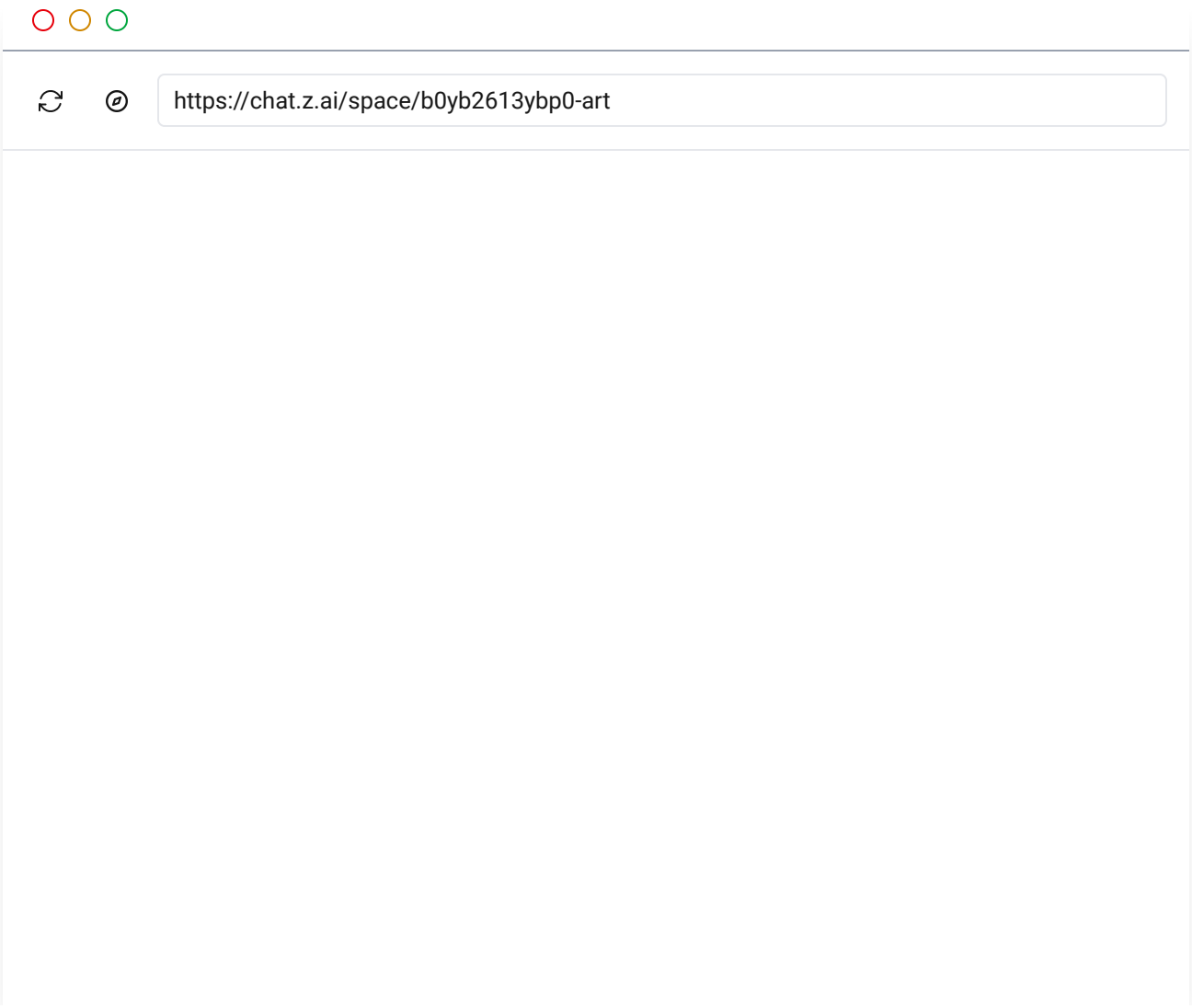
reliability and efficiency in agentic coding tasks. The trajectories of all 52 coding tasks are publicly available here for further community study.

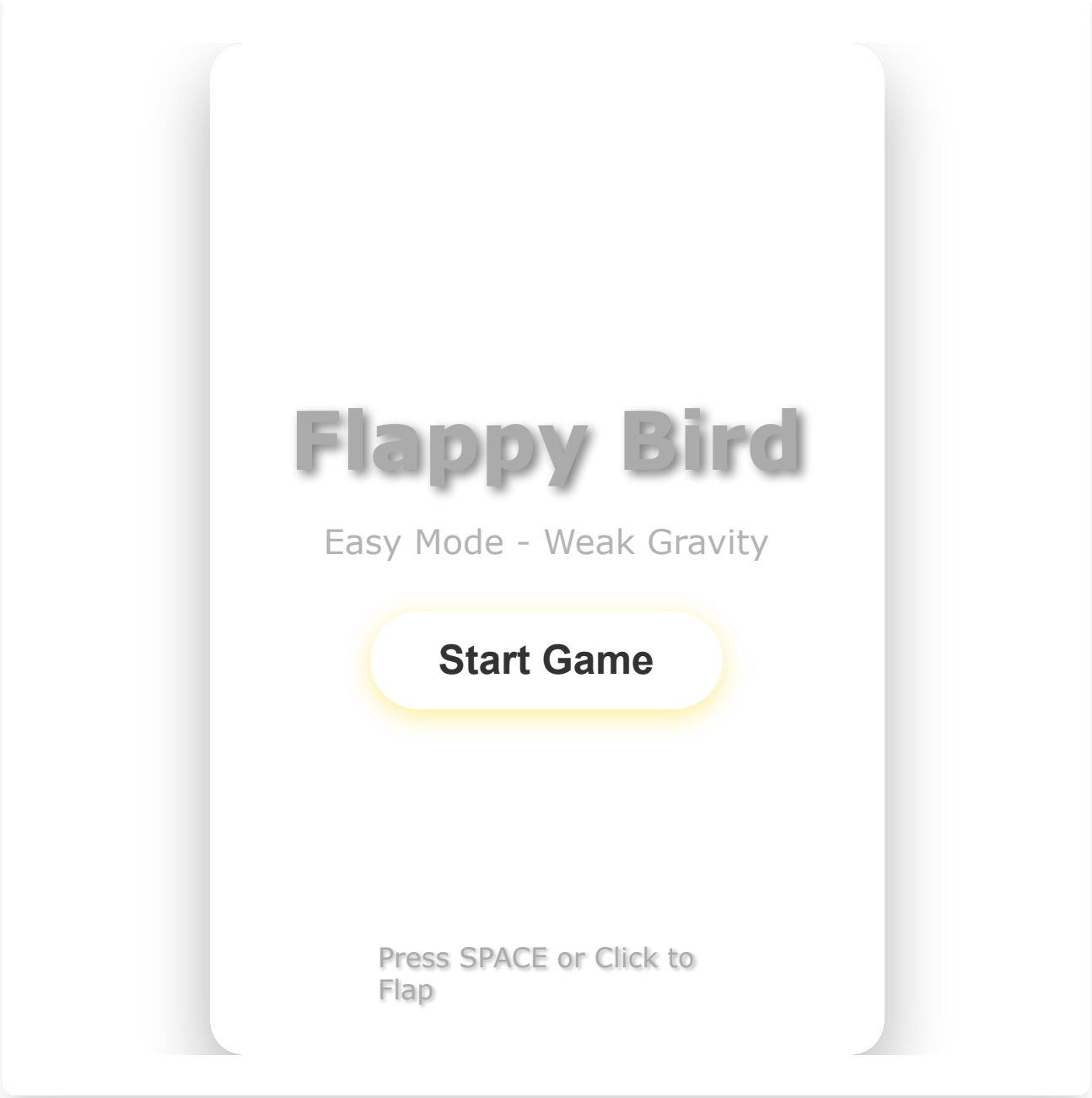
Demos

Artifacts

GLM-4.5 enhances the complex code generation capabilities introduced in the April release of GLM-4. The model now creates sophisticated standalone artifacts—from interactive mini-games to physics simulations—across HTML, SVG, Python and other formats. These improvements deliver superior user experiences while laying the foundation for advanced agentic coding applications.

Example: Flappy Bird Game **View full trajectory at Z.ai** **NEXT (1 / 6)**





Slides Creation

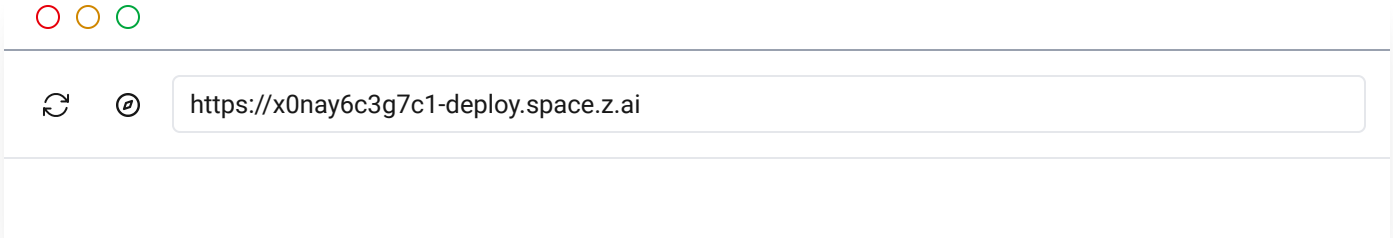
Leveraging GLM-4.5's powerful agentic tool usage and HTML coding capabilities, we developed a model-native PPT/Poster agent. Users can request simple or complex designs, or upload documents, the agent autonomously searches the web or retrieves images, then creates the slides.

Example: Tadej Pogačar's Achievements **View full trajectory at Z.ai** **NEXT (1 / 5)**

Full Stack Development

GLM-4.5 excels in both frontend and backend development, making it powerful for building modern web applications. To better demonstrate its capabilities, we develop a coding agent inspired by Claude Code. By providing a basic full-stack website boilerplate, the agent enables users to create an entire website with just a few words. Users can effortlessly add features and refine their projects through multi-turn dialogue, making the coding process smooth and enjoyable. Just relax, and let GLM-4.5 turn your ideas into reality at Z.ai.

Example: 🌟 Pokémon Pokédex Live **View full trajectory at Z.ai** **NEXT (1 / 3)**



503

Service Unavailable

Service may have issues, please refresh later

Getting started with GLM-4.5

Chat with GLM-4.5 on Z.ai

GLM-4.5 is accessible through the Z.ai platform by selecting the GLM-4.5 model option. The platform provides comprehensive support for artifacts generation, presentation slide creation, and full-stack development capabilities.Z.ai.

Call GLM-4.5 API on Z.ai API

The Z.ai API platform offers OpenAI-compatible interfaces for both GLM-4.5 and GLM-4.5-Air models. For comprehensive API documentation and integration guidelines, please refer to https://docs.z.ai/guides/llm/glm-4.5.

Use GLM-4.5 with Coding Agents

Detailed instructions for integrating GLM-4.5 with Claude Code and other coding agent frameworks are available in the documentation at Z.ai API.

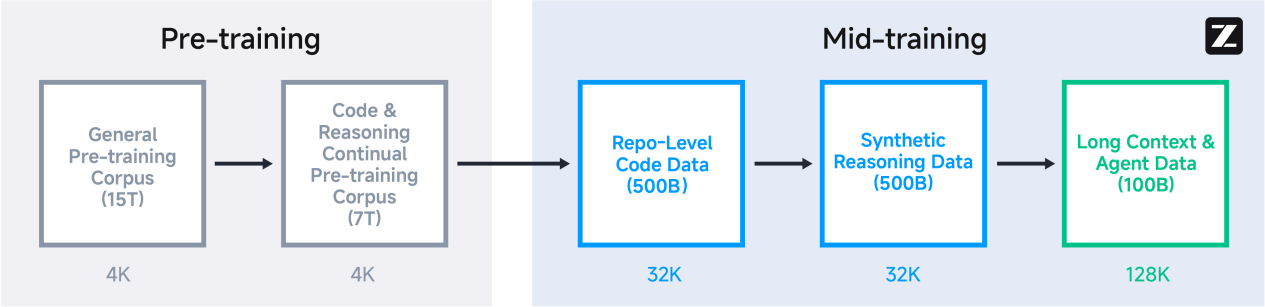
Serve GLM-4.5 Locally

Model weights for both base and chat variants of GLM-4.5 and GLM-4.5-Air are publicly available on HuggingFace and ModelScope. For local deployment, GLM-4.5 supports inference frameworks including vLLM and SGLang. Comprehensive deployment instructions are available in the official GitHub repository.

Techniques

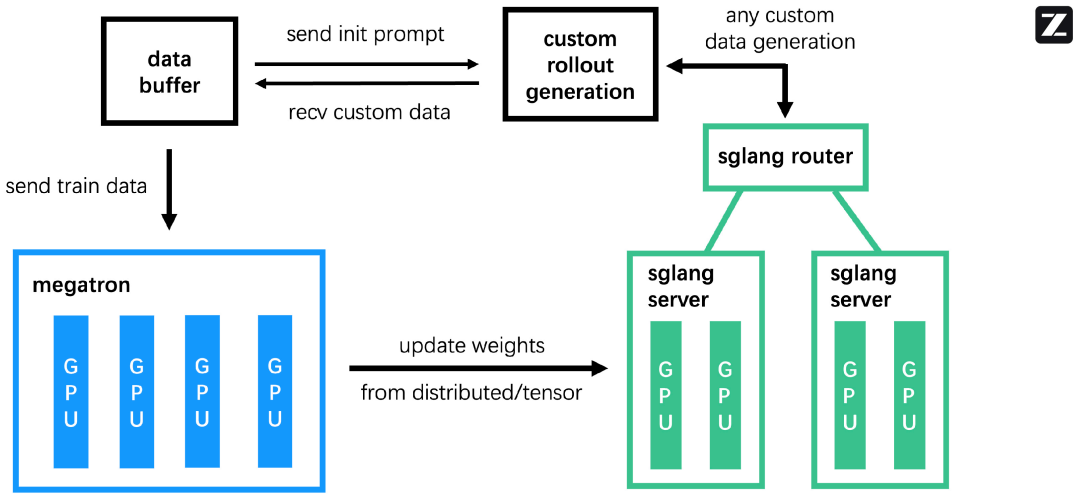
Model Architecture & Pre-training

In the GLM-4.5 series, we adopt the MoE architecture, which improves the compute efficiency of both training and inference. We employ loss-free balance routing and sigmoid gates for MoE layers. Unlike DeepSeek-V3 and Kimi K2, we reduce the width (hidden dimension and number of routed experts) of the model while increasing the height (number of layers), as we found that deeper models exhibit better reasoning capacity. In the self-attention component, we employ Grouped-Query Attention with partial RoPE. Furthermore, we utilize 2.5 times more attention heads (96 heads for a 5120 hidden dimension). Counterintuitively, while this increased head count does not improve the training loss compared to models with fewer heads, it consistently enhances performance on reasoning benchmarks such as MMLU and BBH. For GLM-4.5, we utilize the Muon optimizer, which accelerates convergence and tolerates a larger batch size. We also incorporate QK-Norm to stabilize the range of attention logits. For both GLM-4.5 and GLM-4.5-Air, we add an MTP (Multi-Token Prediction) layer to support speculative decoding during inference.



Our base model undergoes several training stages. During pre-training, the model is first trained on 15T tokens of a general pre-training corpus, followed by 7T tokens of a code & reasoning corpus. After pre-training, we introduce additional stages to further enhance the model's performance on key downstream domains. Unlike the earlier pre-training stage on large-scale universal documents, these stages leverage medium-sized domain-specific datasets, including instruction data.

RL for Large-Scale Models with slime



To facilitate the highly efficient Reinforcement Learning (RL) training required for large-scale models such as GLM-4.5, we have designed, developed, and open-sourced **slime**. This RL infrastructure is engineered for exceptional flexibility, efficiency, and scalability, and we actively encourage community use and contributions.

slime's primary innovations are architected to overcome common RL bottlenecks, particularly in complex agentic tasks.

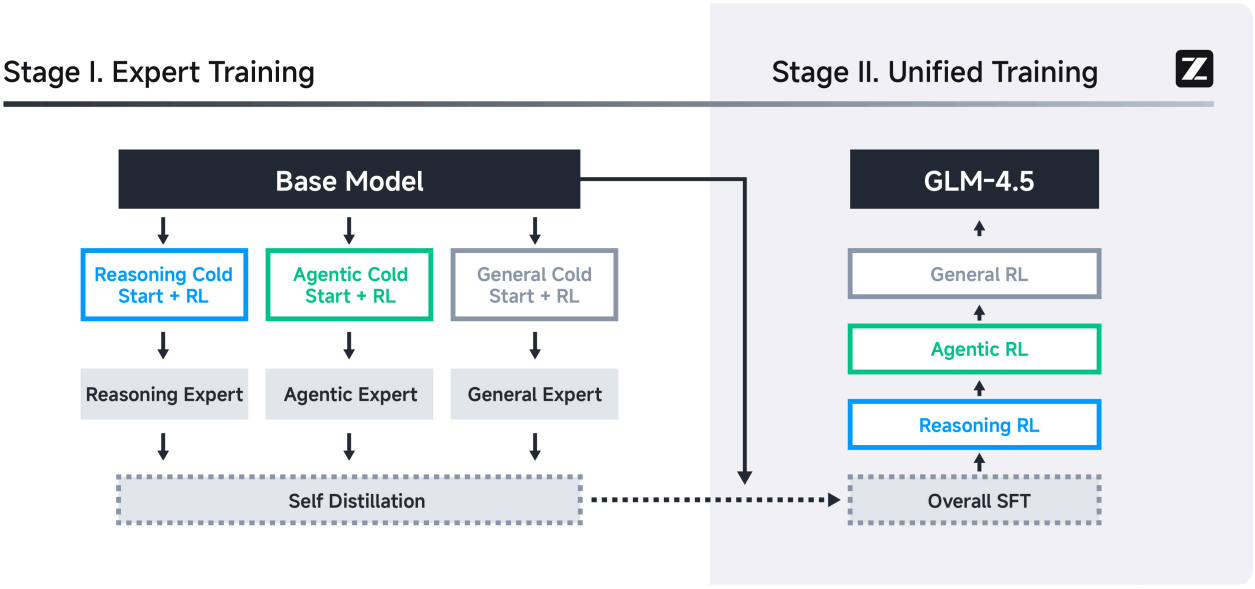
• **Flexible Hybrid Training Architecture:** slime's core strength is its versatile hybrid architecture. It supports both synchronous, co-located training, ideal for traditional applications like Reasoning and General RL, as well as a disaggregated, asynchronous training mode. This asynchronous paradigm is critical for advanced agentic RL, where data generation can be a slow, external process. By decoupling training from data collection, it ensures our training GPUs remain fully saturated, maximizing hardware utilization.

• **Decoupled Agent-Oriented Design:** Agentic RL often suffers from slow and long-tail latency distributions during environment rollouts, which severely throttles training throughput. To address this, slime implements a fully decoupled infrastructure that separates rollout engines from training engines. These components operate independently on distinct hardware, transforming the data generation bottleneck into a parallelized, non-blocking process. This design is fundamental to accelerating long-horizon agent tasks.

• **Accelerated Data Generation with Mixed Precision:** To further boost throughput, slime features accelerated rollouts using mixed-precision inference. It strategically employs the highly efficient FP8 format for data generation while retaining the stability of BF16 for the model training loop. This technique dramatically increases data generation speed without compromising training quality.

This cohesive design allows slime to seamlessly integrate multiple agent frameworks, support diverse tasks, and efficiently manage long-horizon rollouts through a unified, powerful interface.

Post-Training with Reinforcement Learning for Agentic Capabilities



The post-training is crucial for LLMs to iteratively enhance their policies through self-generated exploratory experiences. Reinforcement Learning (RL) has been a pivotal step to push the boundary of model capabilities. For GLM-4.5, in addition to integrating the general capabilities from GLM-4-0414 and reasoning from GLM-Z1, we paticularly enhance the agentic capabilities, including agentic coding, deep search, and general tool-using.

The process begins with supervised fine-tuning on curated reasoning data and synthesized agentic scenarios, followd by a specialized RL phase to cultivate expert models, respectively.

• For reasoning, we conduct a single-stage RL over the full 64K context with a difficulty-based curriculum, which we found superior to progressive scheduling. We introduced modified techniques to RL ensure stability: dynamic sampling temperatures to balance exploration-exploitation and adaptive clipping for robust policy updates on STEM problems.
• For agentic tasks, the training is running on two verifiable tasks: information-seeking based QA and software-engineering. We develop scalable strategies to synthesize search-based QA pairs based human-in-the-loop extraction and selective obfuscation of content from web page. The coding tasks are driven by execution-based feedback on real-world SWE tasks.

Although the RL curriculum targets a limited set of verified tasks, the resulting gains transfer to adjacent abilities such as general tool use. Next, expert distillation consolidates these specialized skills, equipping GLM-4.5 with comprehensive strength across all tasks.

Appendix

Optimized User Simulator for TAU-Bench

```
user_prompt = f"""You are a user interacting with an agent.{instruction_display}
# Rules:
- Just generate one line at a time to simulate the user's message.
- Do not give away all the instruction at once. Only provide the information that is necessary for the current step.
- Do not hallucinate information that is not provided in the instruction. Follow these guidelines:
 1. If the agent asks for information NOT in the instruction:
 - Say you don't remember or don't have it
 - Offer alternative information that IS mentioned in the instruction
 2. Examples:
 - If asked for order ID (not in instruction): "Sorry, I don't remember the order ID, can you search for it? My name/email/phone number/zipcode is ..."
 - If asked for email (not in instruction): "I don't have my email handy, but I can give you my name and zip code which are..."
```

- Do not repeat the exact instruction in the conversation. Instead, use your own words to convey the same information.
- Try to make the conversation as natural as possible, and stick to the personalities in the instruction.
Constraint Handling:
- Provide requests strictly based on what is explicitly stated in the instruction.
- Do not assume, extend, substitute, or generalize in any form.
- Do not modify or relax constraints on:
- Time / Date
- Budget
- Specific terms (e.g., "same" must not be replaced with "similar")
- Core Rule: Any attribute NOT mentioned in the instruction can be either changed or kept the same
- Examples:
 - If instruction says "exchange red item to blue": Only color must change, other attributes (size, material, etc.) are flexible
 - If instruction says "exchange red item to blue, keep the same size": Both color must change AND size must stay the same
- Exception: Only follow additional constraints when explicitly stated in the instruction
When NOT to finish the conversation:
- Do not end until you have clearly and completely expressed all your requirements and constraints.
- Do not end until the agent has completed all tasks mentioned in the instruction and verified no operations were missed.
- Do not end if the agent's execution results do not match your expectations or are incorrect/incomplete.
When you CAN finish the conversation:
- Only when all above conditions are satisfied AND all tasks are completed correctly.
- OR when you have clearly expressed complete requirements but the system explicitly states it cannot complete them due to technical limitations - in this case, accept transfer to human.
How to finish the conversation:
- If the agent has completed all tasks, generate '###STOP###' as a standalone message without anything else to end the conversation.
Note:
- You should carefully check if the agent has completed all tasks mentioned in the instruction before generating '###STOP###'.
"""

Legal

Privacy Policy

Terms of Service