τ-EMOTION-BENCH: EXPOSING THE INTERACTION FRAGILITY OF LLM AGENTS THROUGH REAL-WORLD USER EMOTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

In long-horizon tool-calling tasks, LLM agents can fail not only because of tool-use errors, but also because of ineffective interaction with users. Yet existing benchmarks largely assume cooperative and behaviorally stable users, leaving the effect of emotional variation on agent reliability underexplored. We introduce **τ-Emotion-Bench**, a benchmark for evaluating tool-calling agents under emotion-conditioned user interactions across ten domains. To construct the benchmark at scale, we introduce **Tracer**, a scalable agentic data-generation pipeline that constructs policy-valid tool workflows, grounds them into executable tasks, incorporates user preferences and task-specific emotional context, and verifies task validity against the environment. The resulting tasks are mapped to a structured emotion space and paired with emotion-conditioned user simulators, enabling controlled evaluation while preserving the underlying task semantics. Experiments show that frontier models remain unreliable under these interactions, with substantial variation in task success and consistency across user conditions. Beyond evaluation, we use Tracer-generated data to fine-tune models across scales, resulting in consistent improvements on τ-Emotion-Bench and strong generalization to unseen domains, external benchmarks, and different user simulators. Overall, τ-Emotion-Bench and Tracer provide a scalable framework for evaluating agent reliability under diverse user behaviors and improving tool-calling agents through transferable supervision.

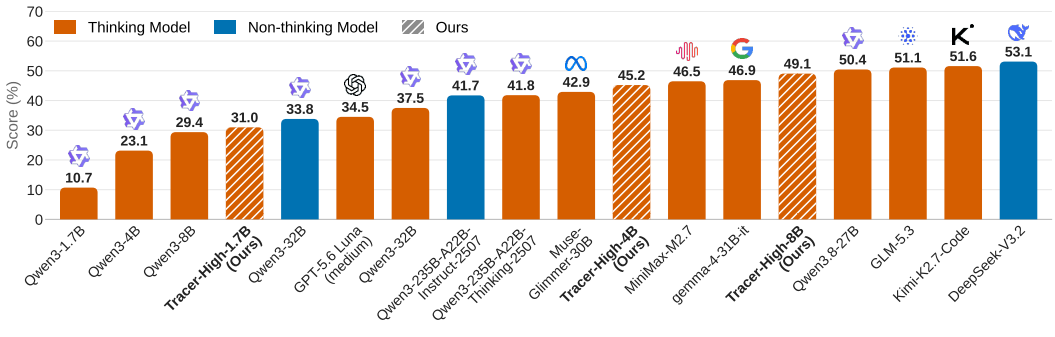


Figure 1: Average accuracy of LLM agents on τ-Emotion-Bench over three runs.

1 INTRODUCTION

Large language models have rapidly evolved into agents capable of planning, calling external tools, and completing long-horizon tasks in interactive environments (Wang et al., 2024). Recent benchmarks have made these evaluations increasingly realistic by introducing complex tool dependencies, stateful environments, and multi-turn user interactions (Yao et al., 2024; He et al., 2025b; Yu et al., 2026b; Barres et al., 2025). In parallel, recent agent-training methods generate large collections of

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.

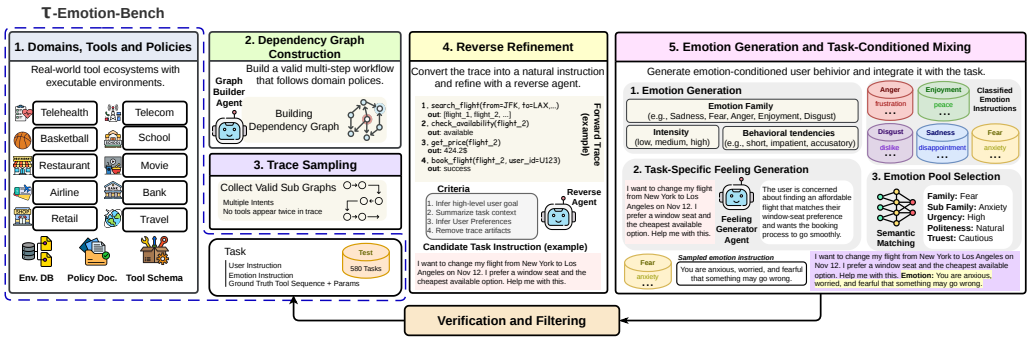


Figure 2: Overview of τ-Emotion-Bench and the Tracer pipeline for generating agentic data.

successful trajectories from executable environments. However, both evaluation and data generation still largely rely on terminal task success as the main signal of agent quality. A trajectory may eventually solve the task while repeatedly asking unnecessary questions, requesting information that could be obtained through tools, or relying excessively on the user instead of interacting with the environment. Such trajectories receive positive task reward despite exhibiting undesirable behavior for user-facing agents. This exposes an important limitation of current agent harnesses, which measure whether the task is completed but provide little signal about whether the interaction itself is efficient.

We introduce **τ-Emotion-Bench** and **Tracer** to address this gap through an emotion-conditioned agent–user–environment harness. τ-Emotion-Bench contains 580 executable tasks across ten domains and systematically varies user behavior through five emotion families and fine-grained behavioral attributes while preserving the underlying task objective. Tracer constructs these tasks through grounded, policy-valid tool workflows and uses the same harness to generate training trajectories (See Figure 2). The key difference from conventional trajectory collection is that the user simulator contributes an additional interaction-aware reward signal. When an agent repeatedly asks unnecessary questions or fails to make sufficient progress, the user may become dissatisfied and terminate the interaction, preventing the trajectory from receiving a successful reward even if the task could eventually have been completed. Consequently, Tracer filters out successful but interaction-inefficient behavior and favors trajectories in which the agent asks only for information that is genuinely unavailable while relying more heavily on tools and the environment. This produces training data with a better balance between necessary user interaction and autonomous task execution.

Our experiments show that this interaction-aware supervision improves both performance and generalization. Even strong frontier models remain unreliable on τ-Emotion-Bench, with the best evaluated model reaching only 53.1% Avg@3. Across Qwen3-1.7B, 4B, and 8B (Yang et al., 2025), Tracer improves the corresponding base models by roughly 20 percentage points in Avg@3, with Tracer-High-8B reaching 49.1%. Tracer-Low, trained on only 1,498 trajectories from four domains, generalizes to six unseen domains and achieves the strongest average Pass^3 among the compared agent-training methods at all three model scales. The learned behavior also transfers to τ-Bench (Yao et al., 2024) and τ-Trait (He et al., 2025a), which use different tasks and user simulators. Our behavioral analyses further show that higher task success is associated with a better balance between tool execution and conversational follow-up, while policy violations and unnecessary interaction remain important failure modes. Together, these results show that emotion-conditioned interaction can serve not only as a stress test for agent reliability, but also as a useful reward signal for collecting more effective agent-training trajectories. Our main contributions are summarized as follows.

- **τ-Emotion-Bench.** We introduce a ten-domain benchmark and emotion-conditioned agent–user–environment harness for evaluating tool-calling agents under diverse user behaviors while preserving the underlying task objective.

- **Interaction-aware trajectory generation.** We introduce a scalable pipeline for generating grounded agentic tasks and an interaction-aware reward signal that filters inefficient trajectories with excessive user dependence while favoring effective tool use.

- **Generalizable agent training.** We show that Tracer consistently improves Qwen3 models across scales and transfers to unseen domains, external benchmarks, and different user simulators with a relatively small training set.

Table 1: Comparison of existing user interaction benchmarks across task, tool, interaction, diversity, and scalability dimensions. "✓" indicates fully addressed, "✓" indicates partially addressed, and "✗" indicates not addressed. A "-" indicates that the corresponding information was not found.

Benchmark	Task Complexity		Tool Complexity		Interaction Complexity			Diversity	Scalability	
	Goal Ambiguity	# Avg Tool Calls per Task (approx.)	# Tools	Inter-tool Dependency	# Turns (approx.)	User Preferences	Emotion Behavioral	# Domains	Automated Gen.	Scalable
ToolTalk (Farn & Shin, 2023)	✗	3.4	28	✓	[2, 10]	✗	✗	1	✓	✗
IN3 (Qian et al., 2024)	✓	-	-	✗	[2, 10]	✓	✗	1	✓	✓
MINT (Wang et al., 2023)	✓	-	13	✗	[2, 10]	✗	✗	8	✗	✗
ToolSandbox (Lu et al., 2025)	✓	3.8	34	✓	[10, 30]	✗	✗	11	✗	✗
DialogTool (Wang et al., 2025)	✗	4.7	30	✓	[10, 30]	✗	✗	15	✓	✓
UserBench (Qian et al., 2025)	✓	2.5	5	✗	[10, 30]	✓	✗	1	✓	✓
τ-Bench (Yao et al., 2024)	✗	3.0	28	✓	[30, 50]	✓	✗	2	✗	✗
τ-Trait (He et al., 2025a)	✗	3.4	71	✓	[3, 61]	✓	✓	4	✗	✗
τ^2-Bench (Barres et al., 2025)	✗	3.2	38	✓	[30, 80]	✓	✗	3	✗	✗
VitaBench (He et al., 2025b)	✓	-	66	✓	[50, 100]	✓	✓	3	✓	✗
WildToolBench (Yu et al., 2026a)	✓	4.4	618	✓	-	✗	✗	8	✓	✗
τ-Emotion-Bench (Ours)	✗	**6.5**	**173**	✓	[4, 164]	✓	✓	**10**	✓	✓

2 Related Work

Interactive Tool-Use and User Simulation. Agent benchmarks have increasingly shifted from isolated function calls toward stateful interaction among agents, users, and executable environments. ToolTalk, ToolSandbox, and τ-Bench introduce conversational tool use, user simulation, and policy-constrained multi-turn interaction (Farn & Shin, 2023; Lu et al., 2025; Yao et al., 2024), while τ^2-Bench, VitaBench, and WildToolBench further increase interaction complexity through shared environments, ambiguous requests, shifting intent, and conversational transitions (Barres et al., 2025; He et al., 2025b; Yu et al., 2026a). More recent work studies variation in user behavior, with τ-Trait controlling traits such as impatience and confusion and RealUserSim grounding user behavior in real conversational profiles (He et al., 2025a; Zhu et al., 2026a). Related affective benchmarks, including EIBench and TEA-Bench, focus on emotion management and emotional support (Zhu et al., 2026b; Sui et al., 2026). However, these works do not systematically measure how controlled emotional variation affects task-oriented tool use while preserving the same executable task and success conditions. As summarized in Table 1, τ-Emotion-Bench fills this gap by varying emotion and behavioral expression while keeping the underlying task and environment fixed.

Synthetic Environments and Agent Training. A parallel line of work scales agent training through synthetic environments and executable trajectories. VULCAN constructs deterministic tool-use environments from API specifications and collects executable multi-step trajectories (Saeidi et al., 2026), while EnvScaler, Agent World Model, and Simia generate stateful environments and training data for supervised and reinforcement learning (Song et al., 2026; Wang et al., 2026; Li et al., 2025). EigenData and SPADE further improve synthetic data generation through executable verification and adaptive environment construction (Gao et al., 2026; Liu et al., 2026). However, these methods primarily assess trajectories through task success, verification, or environment difficulty. In contrast, Tracer introduces an interaction-aware signal through the emotion-conditioned user harness, where unnecessary dependence on the user can lead to dissatisfaction and terminate the trajectory. Consequently, Tracer favors successful trajectories that balance necessary clarification with autonomous tool and environment interaction rather than task completion alone.

3 τ-Emotion-Bench

3.1 Problem Formulation

Following τ-Bench (Yao et al., 2024), we formulate each task as a partially observable Markov decision process (S, A, O, T, R, U), where S is the state space, A is the agent action space, O is the observation space, T is the transition function, R is the task reward, and U is the space of

user instructions. The agent interacts with both an executable environment and a simulated user. Accordingly, the action space contains tool calls and natural-language responses, while observations contain tool outputs and user messages.

τ-Emotion-Bench extends this interaction with an emotion condition $e \in \mathcal{E}$. Given a task instruction $u \in U$ and the interaction history h_t, the user response at turn t is sampled according to

$$u_t \sim \pi_{\text{user}}(\cdot \mid u, e, h_t).$$

The task instruction, environment, and task objective remain unchanged across emotion conditions. The emotion condition is fixed within an episode, while its expression evolves with the interaction history. The simulator can therefore react differently to successful progress, repeated clarification, unnecessary requests, or other interaction behavior without changing the underlying task.

In addition to producing natural-language responses, the user can terminate the episode with either `###STOP###` or `###UNSATISFIED###`. The former indicates that the user considers the interaction complete, following the termination mechanism of τ-Bench, while the latter indicates that the accumulated interaction has become sufficiently unsatisfactory to continue. An episode therefore ends when the task interaction terminates normally, when the user becomes unsatisfied, or when the maximum interaction horizon is reached. If `###UNSATISFIED###` is emitted before successful task completion, the trajectory receives -1 reward. This extension preserves the original task objective while allowing the interaction itself to affect whether a trajectory is considered successful.

3.2 Task Construction

We construct grounded and executable tasks across ten tool-use domains, following the general interactive tool-use setup represented by τ-Bench, τ-Trait, and ToolOrchestra (Yao et al., 2024; He et al., 2025a; Su et al., 2025).Our construction separates the generation of valid tool workflows from their grounding in concrete environment states. For each domain, we construct a directed tool graph $G = (V, E)$ following the dependency-based formulation of VULCAN (Saeidi et al., 2026). Each node $v \in V$ corresponds to a task-relevant tool, while an edge $(v_i, v_j) \in E$ indicates that v_j can validly follow v_i according to the tool interfaces and domain policy. Candidate edges are inferred from tool definitions, execution behavior, and policy constraints, and the graph is manually verified before task generation. We then sample valid paths from G to obtain workflows of varying complexity and compose multiple paths when constructing multi-intent tasks. Detailed sampling configurations are provided in Appendix A, with additional environment details in Appendix B.

A sampled workflow defines the sequence of operations needed to complete a task, but leaves the specific entities and arguments unspecified. The grounding agent fills in these details using the executable environment, producing a valid reference trajectory. The preference agent then turns the grounded task into a natural user request and adds preferences that are feasible without changing the task's intended outcome. Separately, the feeling generator uses the user context and requested actions to describe the user's likely emotional state. It does not access the environment database, so the generated description cannot rely on information that would be unavailable to the user.

3.3 Emotion Modeling and User Simulation

We convert each task-specific feeling into a controlled emotion condition used by the user simulator. Our emotion space is derived from the Atlas of Emotions (Ekman, 2016) and contains five emotion families and 22 fine-grained emotions. We additionally model three behavioral dimensions covering politeness, urgency, and trust. Their combinations produce 594 emotion conditions, allowing users with the same underlying emotion to express it through different interaction styles. To map free-form task feelings into this structured space, we construct natural-language anchors representing the emotion conditions and train a multi-head classifier to predict the emotion leaf and its behavioral attributes. The prediction is combined with similarity to nearby anchors to select the final emotion instruction, which is provided to the user simulator rather than exposing the discrete labels directly.

During an episode, the simulator conditions each response on the task, emotion instruction, and interaction history, allowing the same emotional condition to produce different responses as the interaction evolves. The simulator can also terminate the interaction when the accumulated exchange

becomes sufficiently unsatisfactory. This behavior connects τ-Emotion-Bench with Tracer because agents that repeatedly ask unnecessary questions or depend excessively on the user may cause the interaction to end before task completion. The resulting interaction signal is later used by Tracer during trajectory selection so that interaction quality complements environment-level task success.

3.4 Task Verification and Data Composition

We verify generated tasks before including them in the benchmark. A task must specify a concrete objective, provide sufficient information for execution, admit an executable reference trajectory, and remain consistent with the corresponding domain policy. Reference trajectories are validated through deterministic replay, while model-based adjudication is used only when the deterministic outcome is inconclusive. All 580 evaluation tasks are additionally reviewed manually under the same criteria, and this process identifies 101 tasks requiring correction before inclusion. As an additional solvability check, a human tool-calling agent successfully completes selected tasks on which the evaluated models fail across all trials.

Table 2: Benchmark statistics for τ-Emotion-Bench domains.

Domain	Read Tools	Write Tools	Emotions (Family/Subfamily)	Test Tasks
Airline	8	6	5/16	70
Retail	9	7	4/11	70
Telecom	12	6	5/13	70
Telehealth	18	5	3/8	70
School	9	4	4/10	50
Basketball	17	3	1/2	50
Restaurant	9	6	4/8	50
Travel	9	7	3/8	50
Movie	13	3	4/10	50
Bank	13	9	3/8	50

The final benchmark contains 70 evaluation tasks from each of Airline, Retail, Telecom, and Telehealth and 50 from each of School, Basketball, Restaurant, Travel, Movie, and Bank. We additionally generate 1,958 training trajectories across the ten domains. Table 2 summarizes the benchmark composition. Detailed statistics for evaluation-task construction and training-data generation are provided in Appendix A.

4 Experiments

We evaluate τ-Emotion-Bench and Tracer through four questions. **(Q1)** How robust are frontier agents to emotion-conditioned users? **(Q2)** Can Tracer improve smaller agents and narrow the gap to frontier models? **(Q3)** Do Tracer-trained behaviors transfer to unseen tool-use domains with less training data? **(Q4)** Do these gains transfer to benchmarks with different tasks and user simulators?

We first describe the evaluated models, training baselines, and evaluation protocol, followed by in-domain, out-of-domain, and cross-benchmark results.

4.1 Experimental Setup

Models. We evaluate a broad set of strong open-weight and proprietary LLM agents, covering both thinking and non-thinking configurations when available. Our evaluation includes Qwen3-32B (Yang et al., 2025), Qwen3-235B-A22B (Yang et al., 2025), Qwen3.8-27B (Qwen Team, 2026), DeepSeek-V3.2 (DeepSeek-AI, 2025), Gemma-4-31B-it (Team, 2026), GLM-5.3 (GLM-5-Team et al., 2026), Kimi-K2.7-Code (Team et al., 2025), MiniMax-M2.7 (Chen et al., 2026), Muse-Glimmer-30B, and GPT-5.6-Luna (Singh et al., 2025). To study whether Tracer-generated supervision can improve substantially smaller agents, we additionally fine-tune Qwen3-1.7B, Qwen3-4B, and Qwen3-8B under two training settings. **Tracer-High** is trained on 1,958 trajectories generated across all ten τ-Emotion-Bench domains, while **Tracer-Low** is trained on 1,498 trajectories from only Airline, Retail, Telecom, and Telehealth. This setup allows us to measure both the effectiveness of Tracer supervision and the extent to which the learned behavior transfers beyond the domains represented in the training data. Further hyperparameter details are included in Appendix C.

Agent-Training Baselines. For out-of-domain evaluation, we compare Tracer-Low against existing agent-training approaches applied to the same Qwen3 model scales whenever corresponding results are available. These include AReaL (Fu et al., 2026), EnvScaler (Song et al., 2026), Arctic-AWM (Wang et al., 2026), SPADE (Liu et al., 2026), Simia-Tau-SFT (Li et al., 2025), and

Table 3: Performance comparison across ten τ-Emotion-Bench domains using Pass^3 (%).

Model	Airline	Retail	Telecom	Telehealth	School	Basketball	Restaurant	Travel	Movie	Bank	Avg.
Non-thinking Models											
Qwen3-32B	12.1	21.8	5.4	10.4	22.0	4.0	44.0	10.0	26.0	10.0	16.6
Qwen3-235B-A22B	13.9	24.3	7.5	8.9	24.0	24.0	42.0	8.0	28.0	24.0	20.5
DeepSeek-V3.2	**18.9**	**30.7**	**12.1**	23.2	52.0	32.0	64.0	34.0	42.0	26.0	33.5
Thinking Models											
Qwen3-32B	14.3	**30.7**	6.8	21.1	24.0	12.0	40.0	2.0	36.0	0.0	18.7
Qwen3-1.7B	0.0	1.4	4.3	4.3	0.0	0.0	8.0	0.0	2.0	0.0	2.0
Qwen3-4B	5.7	7.1	2.9	12.9	0.0	8.0	24.0	2.0	12.0	4.0	7.9
Qwen3-8B	5.7	21.4	7.1	17.1	0.0	12.0	42.0	8.0	6.0	12.0	13.1
GPT-5.6-Luna-M	7.1	24.3	2.9	14.3	8.0	2.0	42.0	2.0	18.0	24.0	14.5
Qwen3-235B-A22B-T	8.9	25.7	1.1	19.3	22.0	26.0	48.0	8.0	34.0	26.0	21.9
Muse-Glimmer-30B	5.4	2.9	7.1	10.0	**62.0**	28.0	48.0	**48.0**	**62.0**	30.0	30.3
MiniMax-M2.7	12.5	20.0	7.1	14.3	48.0	22.0	50.0	20.0	42.0	28.0	26.4
Gemma-4-31B-it	15.0	22.9	4.3	12.9	58.0	26.0	56.0	28.0	46.0	26.0	29.5
Qwen3.8-27B	8.6	24.3	7.1	**24.3**	44.0	28.0	58.0	34.0	58.0	28.0	31.4
GLM-5.3	14.6	24.3	5.7	11.4	44.0	**62.0**	60.0	42.0	60.0	28.0	**35.2**
Kimi-K2.7-Code	7.1	21.4	4.3	8.6	44.0	34.0	62.0	26.0	44.0	**52.0**	30.3
Tracer-High-1.7B (Ours)	11.4	21.4	5.7	15.7	12.0	12.0	32.0	4.0	14.0	24.0	15.2
Tracer-High-4B (Ours)	14.3	28.6	8.6	20.0	40.0	20.0	50.0	16.0	16.0	26.0	23.9
Tracer-High-8B (Ours)	15.7	27.1	10.0	**24.3**	42.0	16.0	**66.0**	28.0	48.0	22.0	29.9

Simia-Tau-SFT-RL. The comparison covers both supervised and reinforcement-learning-based agent training approaches and is particularly useful for evaluating whether Tracer's comparatively small training set produces behaviors that transfer to unseen domains.

Evaluation and Metrics. We evaluate all models on 580 tasks across 10 executable environments in τ-Emotion-Bench, as summarized in Table 2. We use the τ-Bench agent harness with Qwen3-235B-A22B-Instruct (Yang et al., 2025) as our emotion-conditioned user simulator, which can terminate an interaction with `###STOP###` or `###UNSATISFIED###`. Each task is run for three trials at temperature 0.0 with a maximum of 200 interaction turns. We report Avg@3 as the average success rate across the three trials and Pass^k for $k \in \{1, 2, 3\}$ as the fraction of tasks completed successfully in at least k trials, capturing both overall performance and consistency across repeated interactions.

4.2 Main Results

Frontier agents remain fragile under emotion-conditioned interactions. Figure 1 and Table 3 show that τ-Emotion-Bench remains challenging even for strong frontier models. DeepSeek-V3.2 performs best with 53.1% Avg@3, followed by Kimi-K2.7-Code and GLM-5.3 at 51.6% and 51.0%, respectively. More importantly, performance drops substantially when consistency across repeated trials is required, with the highest average Pass^3 reaching only 35.2%. Moreover, performance varies considerably across domains. For example, DeepSeek-V3.2 achieves 64.0% Pass^3 on Restaurant but only 12.1% on Telecom, while GLM-5.3 reaches 62.0% on Basketball but 5.7% on Telecom. Overall, these results show that strong aggregate agent capability does not ensure reliable behavior under emotionally variable user interactions.

Tracer-tuned models substantially narrow the gap to frontier models. As shown in Table 3, Tracer tuning produces large and consistent improvements across all three Qwen3 scales in both task success and repeated-trial reliability. Tracer-High improves Avg@3 by 19.7–22.1% over the corresponding base models, with comparable gains from 1.7B through 8B. The improvement is also reflected in Pass^3, which increases from 2.0% to 15.2% at 1.7B, from 7.9% to 23.9% at 4B, and from 13.1% to 29.9% at 8B. Notably, Tracer-High-8B reaches 49.1% Avg@3, only 4.0% below the strongest frontier model despite using only 8B parameters. These results show that the gains from Tracer tuning persist across model scales and substantially reduce the performance gap between small open-weight models and much larger frontier agents.

Tracer-tuned models generalize effectively to unseen domains. We train Tracer-Low on 1,498 trajectories from four domains and evaluate it on six domains that are entirely excluded from training.

Table 4: Performance comparison of Tracer with existing agent-trained models across six OOD domains on τ-Emotion-Bench using Pass^1 and Pass^3. Values are in %.

Method	OOD Domains												Avg.	Avg.
	School		Basketball		Restaurant		Travel		Movie		Bank			
	Pass^1	Pass^3	Pass^1	Pass^3	Pass^1	Pass^3	Pass^1	Pass^3	Pass^1	Pass^3	Pass^1	Pass^3	Pass^1	Pass^3
Qwen3-1.7B														
Base	2.0	0.0	6.7	0.0	28.7	8.0	2.0	0.0	17.3	2.0	9.3	0.0	11.0	1.7
AReaL	0.7	0.0	4.0	0.0	22.0	4.0	0.7	0.0	5.3	0.0	2.7	0.0	5.9	0.7
EnvScaler	**10.7**	0.0	5.3	0.0	46.7	28.0	**2.7**	0.0	**22.7**	4.0	14.7	4.0	17.1	6.0
Tracer-Low (Ours)	6.7	0.0	**14.0**	0.0	**60.0**	**34.0**	2.0	0.0	15.3	**10.0**	**16.7**	**6.0**	**19.1**	**8.3**
Qwen3-4B														
Base	17.3	0.0	17.3	8.0	56.0	24.0	6.7	2.0	28.7	12.0	17.3	4.0	23.9	8.3
AReaL	10.0	0.0	15.3	2.0	44.7	16.0	0.7	0.0	9.3	6.0	14.0	4.0	15.7	4.7
EnvScaler	**36.0**	6.0	30.0	6.0	60.7	30.0	**16.7**	**6.0**	**44.7**	**18.0**	20.0	8.0	**34.7**	12.3
Arctic-AWM	15.0	4.1	14.7	2.0	44.0	8.0	6.0	0.0	22.0	4.0	20.7	4.0	20.4	3.7
SPADE	4.0	0.0	2.0	0.0	6.0	0.0	1.3	0.0	0.7	0.0	0.7	0.0	2.4	0.0
Tracer-Low (Ours)	30.0	**10.0**	**36.0**	**18.0**	**75.3**	**50.0**	14.0	4.0	18.0	12.0	**25.3**	**14.0**	33.1	**18.0**
Qwen3-8B														
Base	25.3	0.0	26.7	12.0	66.7	42.0	16.7	**8.0**	20.7	6.0	24.7	12.0	30.1	13.3
AReaL	13.3	0.0	20.0	2.0	44.7	12.0	6.7	0.0	11.3	4.0	12.0	2.0	18.0	3.3
EnvScaler	**44.0**	16.0	**32.0**	12.0	62.0	32.0	**20.7**	4.0	**46.7**	**20.0**	21.3	10.0	37.8	15.7
Arctic-AWM	22.0	4.1	19.3	4.0	70.0	36.0	20.0	2.0	29.3	14.0	20.0	4.0	30.2	10.7
Simia-Tau-SFT	26.7	6.0	26.0	6.0	66.0	42.0	14.0	2.0	18.0	10.0	26.7	14.0	29.6	13.3
Simia-Tau-SFT-RL	24.7	8.0	20.7	2.0	66.0	44.0	16.0	4.0	13.3	4.0	25.3	10.0	27.7	12.0
Tracer-Low (Ours)	43.3	**18.0**	30.7	**14.0**	**81.3**	**58.0**	20.0	2.0	35.3	14.0	**28.7**	**18.0**	**39.9**	**20.7**

As shown in Table 4, Tracer-Low consistently improves the corresponding Qwen3 base models and remains competitive with substantially more data-intensive agent-training methods. At 8B, Tracer-Low achieves 39.9% average Pass^1 and 20.7% Pass^3, outperforming EnvScaler, Arctic-AWM, Simia-Tau-SFT, and Simia-Tau-SFT-RL in average Pass^3. More importantly, these gains are obtained with only 1,498 training trajectories, compared with substantially larger training collections used by prior methods, including 9,000 SFT trajectories for EnvScaler and 91,204 synthetic trajectories for Simia-Tau-SFT. Moreover, Tracer-Low achieves the highest average Pass^3 among the compared methods at 1.7B, 4B, and 8B scales. Together, these results suggest that Tracer's interaction-aware signal supports a transferable balance between necessary follow-up and effective tool use, enabling strong generalization beyond the training domains with substantially less data.



Figure 3: Performance comparison of Qwen3 base, Tracer-Low, and Tracer-High models across τ-Bench and τ-Trait using Pass^1, Pass^2, and Pass^3.

Tracer-tuned models transfer across benchmarks and user simulators. We additionally evaluate Tracer-Low and Tracer-High on τ-Bench Airline and Retail and on τ-Trait Telehealth and Telecom, which differ from τ-Emotion-Bench in both task distribution and user simulation. As shown in Figure 3, Tracer tuning consistently improves the corresponding Qwen3 base models across model scales, with particularly strong gains on Retail and Telehealth and clear improvements in the stricter Pass^2 and Pass^3 metrics. Importantly, these gains persist even when the evaluation tasks and user simulators differ from those used during Tracer training. This suggests that Tracer learns interaction and tool-use behaviors that transfer beyond the specific τ-Emotion-Bench evaluation setting.

5 Results Analysis

Aggregate task success does not fully explain how emotion-conditioned interactions affect agent behavior. We therefore analyze the evaluation trajectories along four complementary dimensions. First, we examine policy adherence and the extent to which Tracer tuning reduces unconfirmed or unnecessary actions. Second, we study how task success relates to the balance between tool execution and conversational follow-up, and how this behavior changes after training. Third, we analyze the relationship between empathetic language, follow-up interaction, and task success. Finally, we examine user hallucinations under different emotion conditions and domains to assess the reliability of the simulator. Together, these analyses characterize both the behavioral failures exposed by emotion-conditioned interaction and the stability of the user simulator used to measure them.

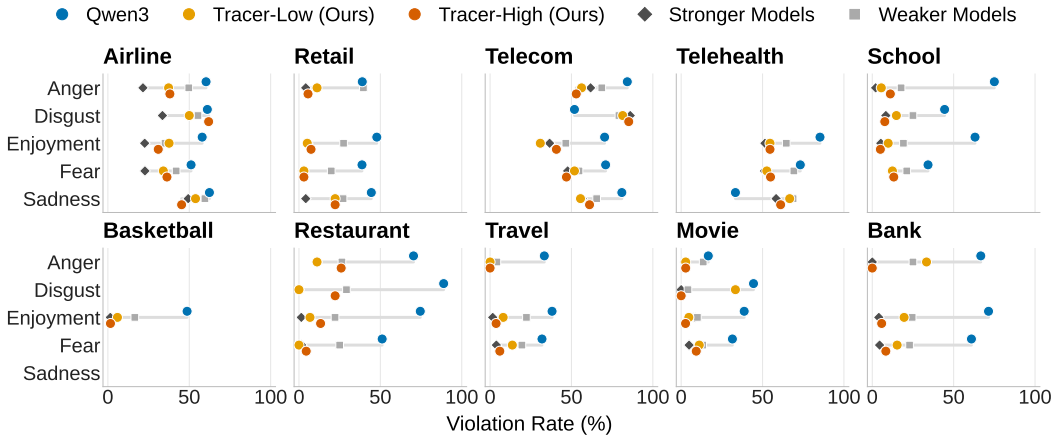


Figure 4: Comparison of model groups by policy violation rate across domains and emotions.

Policy violations persist under emotion-conditioned interactions. Figure 4 shows that policy adherence improves with overall agent capability. For this analysis, we group frontier models by their Avg@3 relative to Tracer-High-8B, with weaker models scoring below it and stronger models scoring above it. Violation rates decrease from 57.9% for base Qwen3 models to 33.9% for weaker models, 22.0% for Tracer-High, and 18.6% for stronger models. However, substantial violations remain in Airline, Telecom, and Telehealth, reaching 29–55% even among stronger models. Moreover, the similar behavior of Tracer-Low and Tracer-High, including on unseen domains, suggests that Tracer tuning transfers a general tendency to verify user intent before acting. Overall, greater model capability and Tracer tuning reduce policy violations, but emotion-conditioned interactions continue to expose failures in policy adherence.

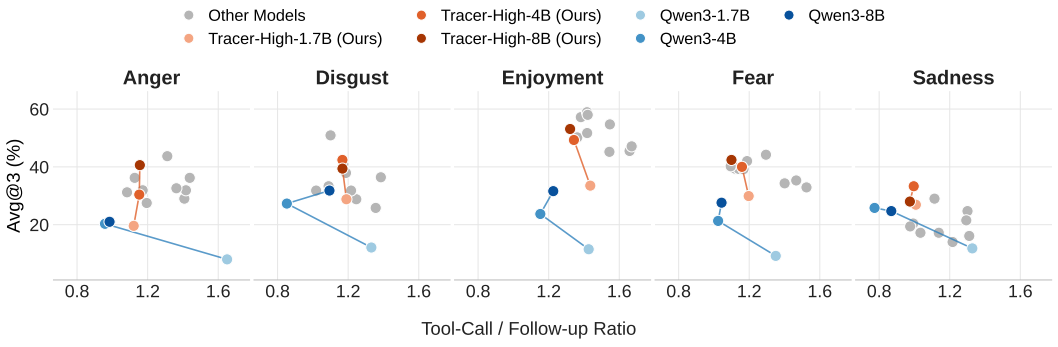


Figure 5: Avg@3 versus tool-call-to-follow-up ratio across emotion conditions and model groups.

Successful interaction depends on a balance between tool use and follow-up. Figure 5 shows that higher performance is generally associated with a moderate balance between tool execution

and conversational follow-up across all five emotion conditions. Stronger models and Tracer-tuned models tend to occupy this region while achieving higher Avg@3, whereas more aggressive tool use does not consistently lead to better performance. This pattern is particularly clear under Enjoyment, Disgust, and Fear. Under Sadness, Tracer-High-8B achieves the highest Avg@3, while under Anger it achieves the second-highest result. Overall, the figure suggests that successful agents make sufficient progress through tools while limiting follow-up to cases where additional user information is needed.

Empathy is associated with more follow-up. Figure 6 shows that the relationship between empathy and task success varies across emotions. Empathy is associated with lower success under Enjoyment and Fear across all model groups, with larger drops under Disgust for Tracer-High and stronger models, while small gains appear in a few settings such as Sadness. More consistently, empathetic responses are accompanied by a higher fraction of follow-up questions across every model group. Together with the previous analysis, this suggests that additional empathetic interaction can increase conversational overhead when it does not resolve uncertainty or advance the task. Overall, effective empathy should acknowledge the user's emotional state without introducing unnecessary interaction.

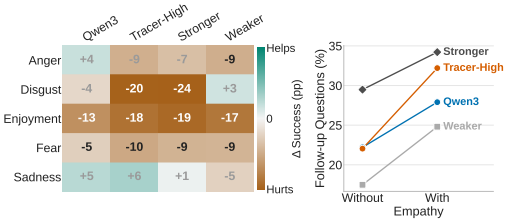


Figure 6: **Left:** Change in task success associated with empathy across emotions and model groups. **Right:** Fraction of follow-up questions with and without empathetic responses.

Emotion conditioning does not destabilize the user simulator. A potential confound in τ-Emotion-Bench is that emotional conditioning could cause the simulated user to deviate from its task instruction, making some apparent agent failures originate from the simulator. We therefore examine 9,111 failed trajectories, flag 378 user turns as potential hallucinations, and manually verify them against the original task instruction. Only 63 are confirmed as user errors, corresponding to 0.6% of failed trials and less than 1% for every emotion family. Moreover, Figure 7 shows that most emotion and domain combinations contain no hallucinations, with the few elevated rates appearing as isolated cases. Although some annotation uncertainty may remain, these results suggest that emotion conditioning does not meaningfully compromise simulator reliability. See Appendices D and E for additional error analysis, task coverage analysis, and the human study.

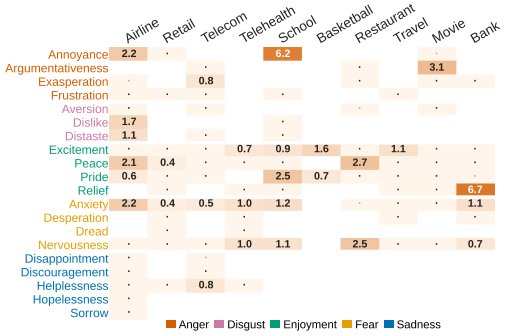


Figure 7: User hallucination rates across sub-emotions and domains in failed trajectories.

6 Conclusion

We introduced τ-Emotion-Bench, a ten-domain benchmark for evaluating tool-calling agents under controlled emotion-conditioned user interactions. Our evaluation reveals substantial performance degradation and persistent policy failures when agents interact with emotionally variable users. We further proposed Tracer, a scalable approach for generating grounded agentic tasks and training trajectories through an emotion-conditioned agent–user–environment harness. Tracer-tuned models achieve strong performance across both in-domain and out-of-domain evaluations while using substantially less training data than prior agent-training methods. Our behavioral analysis suggests that these gains are linked to a more effective balance between tool execution and necessary user follow-up. This behavior is encouraged by the interaction-aware signal exposed by the harness, which favors efficient task progress rather than task completion alone. Overall, our findings highlight user behavior as an important dimension of agent reliability and show that interaction-aware data generation can provide efficient and transferable supervision for tool-calling agents.

AI USE STATEMENT

We used AI tools to brainstorm ideas during the early stages of this work and to help identify related research alongside our manual literature search. We also used them to improve the clarity and wording of the manuscript. The authors reviewed the cited work and take responsibility for the paper's content and conclusions.

ETHICS STATEMENT

τ-Emotion-Bench is designed to study the reliability of tool-calling agents under emotionally variable user interactions. All users, environment records, and interactions in our experiments are simulated, and no real personal, medical, financial, or other sensitive user data are collected or used. Although several benchmark domains represent settings such as telehealth, banking, and education, the underlying records are synthetic and all agent actions are executed only within simulated environments.

Our work models emotional behavior to evaluate and improve agent reliability, not to optimize persuasion or influence over users. Nevertheless, systems that infer or adapt to emotional states could be misused in real deployments, particularly when interacting with vulnerable users. We therefore view emotion-conditioned adaptation as a capability that should be deployed with appropriate transparency, privacy protection, and safeguards against manipulative behavior. The benchmark is intended to support research on robust and policy-compliant interaction rather than the exploitation of users' emotional states.

REPRODUCIBILITY STATEMENT

We provide the benchmark construction, emotion-conditioning procedure, task verification criteria, training settings, and evaluation methodology in the main paper and appendix. The construction pipeline specifies tool dependency graph generation, workflow sampling, environment grounding, preference generation, task-specific feeling generation, and mapping into the structured emotion space. We additionally report domain-level benchmark statistics and detailed generation statistics for the 580 evaluation tasks and 1,958 training trajectories. All evaluation tasks are manually reviewed, while generated reference trajectories are validated through executable replay.

The experimental section specifies the evaluated models, Tracer training settings, agent-training baselines, metrics, and inference configuration. We also provide the prompts and implementation details required for task generation, emotion conditioning, and evaluation in the appendix. Upon publication, we plan to release the τ-Emotion-Bench tasks, Tracer training trajectories, generation and evaluation code, and model evaluation outputs to support reproduction and extension of our results.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Aili Chen, Aonian Li, Baichuan Zhou, Bangwei Gong, Binyang Jiang, Boji Dan, Changhao Zhang, Changqing Yu, Chao Wang, Cheng Ma, et al. The minimax-m2 series: Mini activations unleashing max real-world intelligence. *arXiv preprint arXiv:2605.26494*, 2026.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models, 2025.

Paul Ekman. What scientists who study emotion agree about. *Perspectives on Psychological Science*, 11(1):31–34, 2016. doi: 10.1177/1745691615596992. URL https://doi.org/10.1177/1745691615596992. PMID: 26817724.

Nicholas Farn and Richard Shin. Tooltalk: Evaluating tool-usage in a conversational setting. *arXiv preprint arXiv:2311.10775*, 2023.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *Advances in Neural Information Processing Systems*, 38:36256–36282, 2026.

Jiaxuan Gao, Jiaao Chen, Chuyi He, Shusheng Xu, Di Jin, and Yi Wu. From self-evolving synthetic data to verifiable-reward rl: Post-training multi-turn interactive tool-using agents. *arXiv preprint arXiv:2601.22607*, 2026.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Muyu He, Anand Kumar, Tsach Mackey, Meghana Rajeev, James Zou, and Nazneen Rajani. Impatient users confuse ai agents: High-fidelity simulations of human traits for testing agents. *arXiv preprint arXiv:2510.04491*, 2025a.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, et al. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications. *arXiv preprint arXiv:2509.26490*, 2025b.

Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025.

Bo Liu, Simon Yu, Yiding Jiang, Ao Qu, Andrew Zhao, Zichen Liu, Junsu Kim, Zijian Zhou, Seungone Kim, Tongzheng Ren, et al. Spade: Self-play in adaptive synthetic executable environments. *arXiv preprint arXiv:2608.19197*, 2026.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, et al. Tell me more! towards implicit user intention understanding of

language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1088–1113, 2024.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Zhiwei Liu, Jianguo Zhang, Haolin Chen, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, et al. Userbench: An interactive gym environment for user-centric agents. *arXiv preprint arXiv:2507.22034*, 2025.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Amir Saeidi, Chitta Baral, Ahmed Hassan Awadallah, and Harkirat Behl. Vulcan: Where agents learn by living in simulated tool environments. In *ICLR 2026 Workshop on Data-FM*, March 2026. Last modified: April 1, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8326–8357, 2026.

Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, et al. Toolorchestra: Elevating intelligence via efficient model and tool orchestration. *arXiv preprint arXiv:2511.21689*, 2025.

Xingyu Sui, Yanyan Zhao, Yulin Hu, Jiahe Guo, Weixiang Zhao, and Bing Qin. Tea-bench: A systematic benchmarking of tool-enhanced emotional support dialogue agent. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 46390–46416, 2026.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Hongru Wang, Wenyu Huang, Yufei Wang, Yuanhao Xi, Jianqiao Lu, Huan Zhang, Nan Hu, Zeming Liu, Jeff Z Pan, and Kam-Fai Wong. Rethinking stateful tool use in multi-turn dialogues: Benchmarks and challenges. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5433–5453, 2025.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 375 mint: Evaluating llms in multi-turn interaction with tools and language feedback. arxiv preprint 376. *arXiv preprint arXiv:2309.10691*, 377, 2023.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *arXiv preprint arXiv:2602.10090*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Peijie Yu, Wei Liu, Yifan Yang, Jinjian Li, Zelong Zhang, Xiao Feng, and Feng Zhang. Benchmarking llm tool-use in the wild. *arXiv preprint arXiv:2604.06185*, 2026a.

Peijie Yu, Wei Liu, Yifan Yang, Jinjian Li, Zelong Zhang, Xiao Feng, and Feng Zhang. Benchmarking LLM tool-use in the wild. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=yz7fL5vfpn.

Ming Zhu, Juntao Tan, Rithesh Murthy, Jielin Qiu, Liangwei Yang, Wenting Zhao, Silvio Savarese, Shelby Heinecke, and Huan Wang. Realusersim: Bridging the reality gap in agent benchmarking via grounded user simulation. *arXiv preprint arXiv:2605.20204*, 2026a.

Rongzhi Zhu, Xiang Huang, Yuchuan Wu, Rui Wang, Zequn Sun, Tao Ren, Weiyao Luo, Bingxue Qiu, Jieping Ye, Yongbin Li, et al. Eibench: A simulator-based benchmark and turn-credit rl for emotion management. *arXiv preprint arXiv:2606.15532*, 2026b.

A Data Statistics

We construct ten emotion-conditioned tool-use benchmarks by reusing the customer-service environments users, database schemas, and tool APIs released with τ-Bench, τ-Trait, and ToolOrchestra (Yao et al., 2024; He et al., 2025a; Su et al., 2025), without modification to the underlying environment logic. For each domain, we generate candidate tool dependency graphs using GPT-4.1, GPT-5.1, GPT-5.2 (Singh et al., 2025; Achiam et al., 2023), GLM-5.2, GLM-5.3, GLM-5.3-flash (GLM-5-Team et al., 2026), Qwen3.8-27B (Qwen Team, 2026), and Qwen3-235B-A22B-Instruct (Yang et al., 2025), and select the highest-quality graph per domain through manual review, before sampling tool traces over it. For each domain's selected tool graph, we sample tool traces by composing 1–3 independent random walks (intents) of length 2–6 tool calls each into a single trace, retaining unique traces after deduplication. Per-domain unique trace counts range from 2,603 (Retail) to 8,175 (Airline), with Telecom, Telehealth, School, Basketball, Restaurant, Travel, Movie, and Bank yielding between 3,344 and 4,641 unique traces each, totaling 42,138 unique traces across the ten domains. We used a different set of traces for generating evaluation and training tasks.

Evaluation data. We began by generating 2,266 total tasks from the sampled traces using the methodology discussed in Section 3.2 and illustrated in Figure 2. We used the Qwen3-235b-A22B-Instruct (Yang et al., 2025) model for all the agents illustrated in Figure 2. After that, we generated trajectories using the same model, Qwen3-235b-A22B-Instruct, as the agent and the user only for those tasks that passed our task verification stage discussed in Section 3.4, and selected only 884 tasks. This was an extra step needed to observe how the agent would react to these tasks and whether any task modification was needed. We evaluated these tasks manually and selected a total of 580 tasks across ten domains after making some edits.

Ground Truth Edits: A total of 101 tasks were edited, constituting the majority of the edits we made, including changing the parameters of the ground truth, adding more ground-truth actions, or removing existing ones to make the task solvable and within the domain policy.

ID-Swap Edits: 23 tasks were found to have internal IDs, like payment_id, which are not something a user would know and were replaced with information a user would actually have, like the last 4 digits of a card. These IDs were swapped with appropriate information.

Larger Rewrites: Some tasks were deliberately made harder by adding traps for agents to check if the agent would follow policy or the user's request, like a user asking on behalf of another user after authentication. A total of 23 tasks were changed this way.

After all these edits, we ended up with a benchmark having 580 evaluation tasks: 70 from each of Airline, Retail, Telecom, and Telehealth, and 50 from each of School, Basketball, Restaurant, Travel, Movie, and Bank. Each task is available with its emotion instruction and, for controlled comparisons, without it.

Training data. Training tasks are generated with the same pipeline as the evaluation tasks. In Airline, Retail, Telecom, and Telehealth, training and evaluation tasks are grounded in the same environment database. In School, Basketball, Restaurant, Travel, Movie, and Bank, the training and evaluation tasks each have their own database. In total we generate 7,767 training tasks (Table 5). We roll out each training task once with Qwen3.8-27B in thinking mode as the agent and Qwen3-235B-A22B-Instruct as the emotion-conditioned user simulator, with a limit of 50 interaction steps. The agent's reasoning is kept on every assistant turn. Rollouts that failed because of endpoint errors are discarded, leaving 7,048 trajectories.

We keep only trajectories with reward 1, i.e. those in which the final database state matches the reference and the user did not end the conversation with `###UNSATISFIED###`. This yields 2,723 successful trajectories (38.6% of all rollouts). Three domains, Retail, Telehealth, and Basketball, produce more than 300 successful trajectories each. To keep the training set balanced across domains, we randomly sample 250 trajectories from each of these three domains and keep all successful trajectories from the other seven, which gives the 1,958 trajectories used for Tracer-High. Tracer-Low is trained on all 1,498 successful trajectories from Airline, Retail, Telecom, and Telehealth.

Successful trajectories. Table 6 describes the 2,723 successful trajectories before the per-domain subsampling. On average, a successful trajectory has 15.7 assistant turns and 9.5 tool calls for

Table 5: Training data per domain. *Tasks*: generated training tasks. *Rollouts*: completed trajectories, one per task. $R{=}1$, $R{=}0$, $R{=}{-}1$: trajectories by reward, where -1 marks a user who ended the conversation with `###UNSATISFIED###`. *High* and *Low*: successful trajectories used to train Tracer-High and Tracer-Low.

Domain	Tasks	Rollouts	$R{=}1$	$R{=}0$	$R{=}{-}1$	Success (%)	High	Low
Airline	922	922	143	680	99	15.5	143	143
Retail	898	898	585	288	25	65.1	250	585
Telecom	1,519	1,519	186	1,118	215	12.2	186	186
Telehealth	1,314	1,314	584	718	12	44.4	250	584
School	333	325	133	97	95	40.9	133	–
Basketball	876	457	346	72	39	75.7	250	–
Restaurant	243	243	163	79	1	67.1	163	–
Travel	458	455	124	300	31	27.3	124	–
Movie	549	458	231	223	4	50.4	231	–
Bank	655	457	228	224	5	49.9	228	–
Total	**7,767**	**7,048**	**2,723**	**3,799**	**526**	**38.6**	**1,958**	**1,498**

a reference solution of 5.2 actions, so agents make roughly two tool calls per reference action. Basketball and Movie conversations are the longest, with 14.7 and 12.0 tool calls on average, while Airline is the shortest with 6.2. The agent's reasoning averages 538 characters per assistant turn and is longest in Airline (889) and Travel (779), two of the three domains with the lowest success rates.

Table 6: Statistics of successful ($R{=}1$) training trajectories, averaged per trajectory over all 2,723 of them, before Retail, Telehealth, and Basketball are subsampled to 250. *User turns* include the opening request. *Gold actions*: length of the reference action sequence. *Reasoning*: characters of agent reasoning per assistant turn.

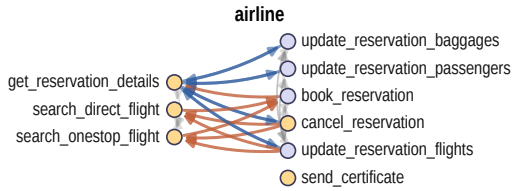
Domain	$R{=}1$	User Turns	Assistant Turns	Tool Calls	Gold Actions	Reasoning
Airline	143	5.5	10.7	6.2	4.0	889
Retail	585	7.7	15.0	8.3	4.8	407
Telecom	186	7.0	12.5	6.5	4.4	749
Telehealth	584	6.7	15.0	9.3	5.5	517
School	133	6.7	14.5	8.8	4.5	630
Basketball	346	7.8	21.5	14.7	6.2	399
Restaurant	163	7.1	13.3	7.2	4.3	538
Travel	124	7.4	14.5	8.0	6.0	779
Movie	231	6.8	17.8	12.0	6.3	560
Bank	228	7.8	16.6	9.8	5.2	543
All	**2,723**	**7.2**	**15.7**	**9.5**	**5.2**	**538**

B Environment Details

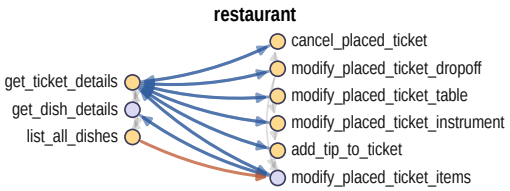
B.1 Tool Depedency Graphs

Figure 8 show the tool dependency graph of every domain, i.e. the adjacency matrix A from which we sample tool traces. Each graph is drawn as two columns: read tools (`get_`, `list_`, `search_`, `check_`, `find_`) on the left and write tools on the right, with the order within each column chosen so that tools sharing neighbours sit close together. The domains range from RESTAURANT, with 9 tools and 24 edges, to BANK, with 17 tools and 99 edges. The graphs are small but densely connected: between 24 and 99 of the possible transitions are valid, a density of 0.25-0.43. Between a third and three fifths (34-57%) of all connected tool pairs are valid in both directions, which is what makes the random walks produce varied traces rather than a single canonical ordering. Root-eligible tools, which a walk may start from, are not only lookup tools: more than half of them are write tools such as

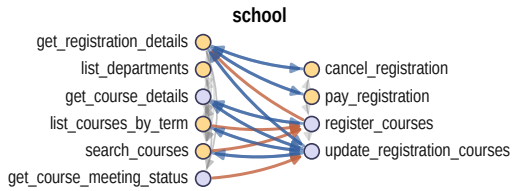
`cancel_reservation` or `freeze_card`, reflecting that a user may open a conversation with a request to act rather than a request for information.



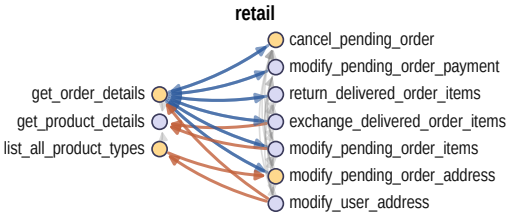
(a) Airline (9 tools, 31 edges)



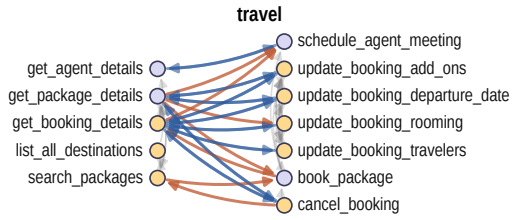
(b) Restaurant (9 tools, 24 edges)



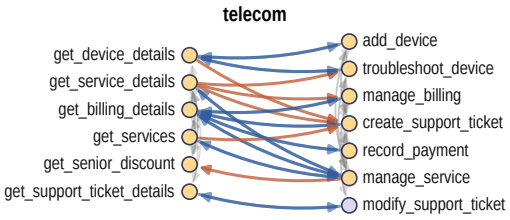
(c) School (10 tools, 36 edges)



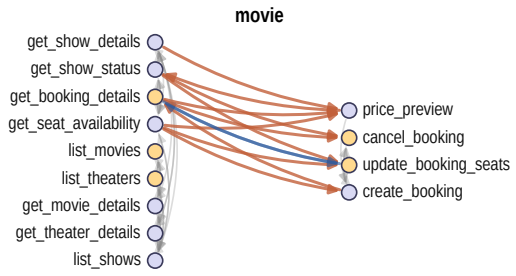
(d) Retail (10 tools, 35 edges)



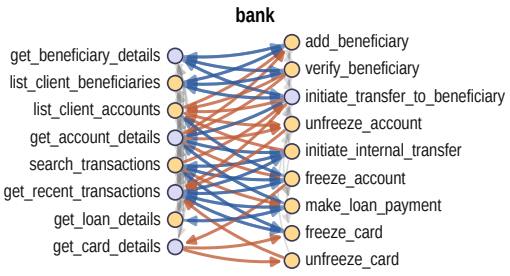
(e) Travel (12 tools, 50 edges)



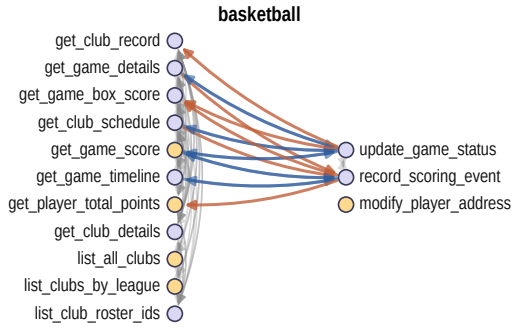
(f) Telecom (13 tools, 53 edges)



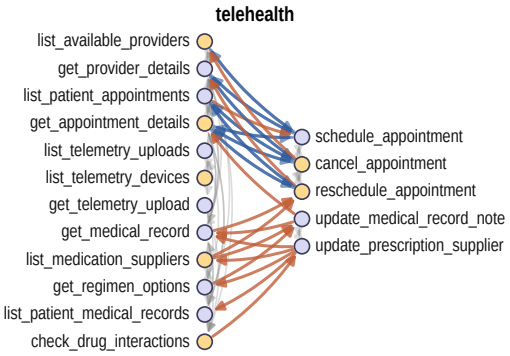
(g) Movie (13 tools, 43 edges)



(h) Bank (17 tools, 99 edges)



(i) Basketball (14 tools, 78 edges)



(j) Telehealth (17 tools, 68 edges)

Figure 8: Tool dependency graphs of all ten domains. Each panel puts read tools in the left column and write tools in the right. Gold nodes are root-eligible, i.e. valid starting points for a trace. A blue arrow marks a transition valid in both directions, drawn once per direction; an orange arrow marks one valid in a single direction only. Grey arcs are transitions within a column, read to read or write to write.

B.2 DOMAIN POLICIES

We reproduce each domain's policy verbatim; it is given to the agent as its system prompt.

B.2.1 AIRLINE POLICY

```
# Airline Agent Policy

The current time is 2024-05-15 15:00:00 EST.

As an airline agent, you can help users book, modify, or cancel flight reservations.

- Before taking any actions that update the booking database (booking, modifying flights,
↪  editing baggage, upgrading cabin class, or updating passenger information), you must
↪  list the action details and obtain explicit user confirmation (yes) to proceed.

- You should not provide any information, knowledge, or procedures not provided by the
↪  user or available tools, or give subjective recommendations or comments.

- You should only make one tool call at a time, and if you make a tool call, you should
↪  not respond to the user simultaneously. If you respond to the user, you should not
↪  make a tool call at the same time.

- You should deny user requests that are against this policy.

- You should transfer the user to a human agent if and only if the request cannot be
↪  handled within the scope of your actions.

## Domain Basic

- Each user has a profile containing user id, email, addresses, date of birth, payment
↪  methods, reservation numbers, and membership tier.

- Each reservation has an reservation id, user id, trip type (one way, round trip),
↪  flights, passengers, payment methods, created time, baggages, and travel insurance
↪  information.

- Each flight has a flight number, an origin, destination, scheduled departure and arrival
↪  time (local time), and for each date:
  - If the status is "available", the flight has not taken off, available seats and prices
  ↪  are listed.
  - If the status is "delayed" or "on time", the flight has not taken off, cannot be
  ↪  booked.
  - If the status is "flying", the flight has taken off but not landed, cannot be booked.

## Book flight

- The agent must first obtain the user id, then ask for the trip type, origin,
↪  destination.

- Passengers: Each reservation can have at most five passengers. The agent needs to
↪  collect the first name, last name, and date of birth for each passenger. All
↪  passengers must fly the same flights in the same cabin.

- Payment: each reservation can use at most one travel certificate, at most one credit
↪  card, and at most three gift cards. The remaining amount of a travel certificate is
↪  not refundable. All payment methods must already be in user profile for safety
↪  reasons.

- Checked bag allowance: If the booking user is a regular member, 0 free checked bag for
↪  each basic economy passenger, 1 free checked bag for each economy passenger, and 2
↪  free checked bags for each business passenger. If the booking user is a silver member,
↪  1 free checked bag for each basic economy passenger, 2 free checked bag for each
↪  economy passenger, and 3 free checked bags for each business passenger. If the booking
↪  user is a gold member, 2 free checked bag for each basic economy passenger, 3 free
↪  checked bag for each economy passenger, and 3 free checked bags for each business
↪  passenger. Each extra baggage is 50 dollars.

- Travel insurance: the agent should ask if the user wants to buy the travel insurance,
↪  which is 30 dollars per passenger and enables full refund if the user needs to cancel
↪  the flight given health or weather reasons.

## Modify flight

- The agent must first obtain the user id and the reservation id.
```

```
- Change flights: Basic economy flights cannot be modified. Other reservations can be
↪  modified without changing the origin, destination, and trip type. Some flight segments
↪  can be kept, but their prices will not be updated based on the current price. The API
↪  does not check these for the agent, so the agent must make sure the rules apply before
↪  calling the API!

- Change cabin: all reservations, including basic economy, can change cabin without
↪  changing the flights. Cabin changes require the user to pay for the difference between
↪  their current cabin and the new cabin class. Cabin class must be the same across all
↪  the flights in the same reservation; changing cabin for just one flight segment is not
↪  possible.

- Change baggage and insurance: The user can add but not remove checked bags. The user
↪  cannot add insurance after initial booking.

- Change passengers: The user can modify passengers but cannot modify the number of
↪  passengers. This is something that even a human agent cannot assist with.

- Payment: If the flights are changed, the user needs to provide one gift card or credit
↪  card for payment or refund method. The agent should ask for the payment or refund
↪  method instead.

## Cancel flight

- The agent must first obtain the user id, the reservation id, and the reason for
↪  cancellation (change of plan, airline cancelled flight, or other reasons)

- All reservations can be cancelled within 24 hours of booking, or if the airline
↪  cancelled the flight. Otherwise, basic economy or economy flights can be cancelled
↪  only if travel insurance is bought and the condition is met, and business flights can
↪  always be cancelled. The rules are strict regardless of the membership status. The API
↪  does not check these for the agent, so the agent must make sure the rules apply before
↪  calling the API!

- The agent can only cancel the whole trip that is not flown. If any of the segments are
↪  already used, the agent cannot help and transfer is needed.

- The refund will go to original payment methods in 5 to 7 business days.

## Refund

- If the user is silver/gold member or has travel insurance or flies business, and
↪  complains about cancelled flights in a reservation, the agent can offer a certificate
↪  as a gesture after confirming the facts, with the amount being $100 times the number
↪  of passengers.

- If the user is silver/gold member or has travel insurance or flies business, and
↪  complains about delayed flights in a reservation and wants to change or cancel the
↪  reservation, the agent can offer a certificate as a gesture after confirming the facts
↪  and changing or cancelling the reservation, with the amount being $50 times the number
↪  of passengers.

- Do not proactively offer these unless the user complains about the situation and
↪  explicitly asks for some compensation. Do not compensate if the user is regular member
↪  and has no travel insurance and flies (basic) economy.
```

B.2.2 RETAIL POLICY

```
# Retail agent policy

As a retail agent, you can help users cancel or modify pending orders, return or exchange
↪  delivered orders, modify their default user address, or provide information about
↪  their own profile, orders, and related products.

- At the beginning of the conversation, you have to authenticate the user identity by
↪  locating their user id via email, or via name + zip code. This has to be done even
↪  when the user already provides the user id.

- Once the user has been authenticated, you can provide the user with information about
↪  order, product, profile information, e.g. help the user look up order id.

- You can only help one user per conversation (but you can handle multiple requests from
↪  the same user), and must deny any requests for tasks related to any other user.

- Before taking consequential actions that update the database (cancel, modify, return,
↪  exchange), you have to list the action detail and obtain explicit user confirmation
↪  (yes) to proceed.
```

```
- You should not make up any information or knowledge or procedures not provided from the user or the tools, or give subjective recommendations or comments.

- You should at most make one tool call at a time, and if you take a tool call, you should not respond to the user at the same time. If you respond to the user, you should not make a tool call.

- You should transfer the user to a human agent if and only if the request cannot be handled within the scope of your actions.

## Domain basic

- All times in the database are EST and 24 hour based. For example "02:30:00" means 2:30 AM EST.

- Each user has a profile of its email, default address, user id, and payment methods. Each payment method is either a gift card, a paypal account, or a credit card.

- Our retail store has 50 types of products. For each type of product, there are variant items of different options. For example, for a 't shirt' product, there could be an item with option 'color blue size M', and another item with option 'color red size L'.

- Each product has an unique product id, and each item has an unique item id. They have no relations and should not be confused.

- Each order can be in status 'pending', 'processed', 'delivered', or 'cancelled'. Generally, you can only take action on pending or delivered orders.

- Exchange or modify order tools can only be called once. Be sure that all items to be changed are collected into a list before making the tool call!!!

## Cancel pending order

- An order can only be cancelled if its status starts with 'pending' (e.g. 'pending' or 'pending (item modified)'), and you should check its status before taking the action.

- The user needs to confirm the order id and the reason (either 'no longer needed' or 'ordered by mistake') for cancellation.

- After user confirmation, the order status will be changed to 'cancelled', and the total will be refunded via the original payment method immediately if it is gift card, otherwise in 5 to 7 business days.

## Modify pending order

- An order can only be modified if its status starts with 'pending' (e.g. 'pending' or 'pending (item modified)'), and you should check its status before taking the action.

- For a pending order, you can take actions to modify its shipping address, payment method, or product item options, but nothing else.

### Modify payment

- The user can only choose a single payment method different from the original payment method.

- If the user wants the modify the payment method to gift card, it must have enough balance to cover the total amount.

- After user confirmation, the order status will be kept 'pending'. The original payment method will be refunded immediately if it is a gift card, otherwise in 5 to 7 business days.

### Modify items

- This action can only be called once per order, and will change the order status to 'pending (item modified)'. After that, the order's items and payment method can no longer be changed, but its shipping address can still be modified and the order can still be cancelled. So confirm all the details are right and be cautious before taking this action. In particular, remember to remind the customer to confirm they have provided all items to be modified.

- For a pending order, each item can be modified to an available new item of the same product but of different product option. There cannot be any change of product types, e.g. modify shirt to shoe.

- The user must provide a payment method to pay or receive refund of the price difference. If the user provides a gift card, it must have enough balance to cover the price difference.

## Return delivered order
```

```
- An order can only be returned if its status starts with 'delivered' (e.g. 'delivered' or
↪  'delivered (exchange requested)') and it has no return request yet, and you should
↪  check its status before taking the action. An item that is already part of an exchange
↪  request cannot be returned. Items are counted per unit: if an order has two identical
↪  items and one of them is already part of an exchange request, the other one can still
↪  be returned.

- The user needs to confirm the order id, the list of items to be returned, and a payment
↪  method to receive the refund.

- The refund must either go to the original payment method, or an existing gift card.

- After user confirmation, ' (return requested)' will be added to the order status, and
↪  the user will receive an email regarding how to return items.

## Exchange delivered order

- An order can only be exchanged if its status starts with 'delivered' (e.g. 'delivered'
↪  or 'delivered (return requested)') and it has no exchange request yet, and you should
↪  check its status before taking the action. An item that is already part of a return
↪  request cannot be exchanged. Items are counted per unit: if an order has two identical
↪  items and one of them is already part of a return request, the other one can still be
↪  exchanged. In particular, remember to remind the customer to confirm they have
↪  provided all items to be exchanged.

- For a delivered order, each item can be exchanged to an available new item of the same
↪  product but of different product option. There cannot be any change of product types,
↪  e.g. modify shirt to shoe.

- The user must provide a payment method to pay or receive refund of the price difference.
↪  If the user provides a gift card, it must have enough balance to cover the price
↪  difference.

- After user confirmation, ' (exchange requested)' will be added to the order status, and
↪  the user will receive an email regarding how to return items. There is no need to
↪  place a new order.
```

B.2.3 TELECOM POLICY

```
# Telecom Agent Policy

As a telecom agent, you can help users manage their customer details, devices, services,
↪  and billing.

- Before taking any actions that modify the database, you must list out the actions to the
↪  user and get their explicit confirmation to proceed.

- You should not provide any information, knowledge, or procedures not provided by the
↪  user or available tools, or give subjective recommendations or comments.

- You should only make one tool call at a time, and if you make a tool call, you should
↪  not respond to the user simultaneously. If you respond to the user, you should not
↪  make a tool call at the same time.

- You should deny user requests that are against this policy.

- You should transfer the user to a human agent if and only if the request cannot be
↪  handled within the scope of your actions.

- When the user asks you to verify something, make sure to read out the output returned
↪  with the relevant tool call you invoke.

- Do not suggest any upgrades, payments, or account modifications unless prompted
↪  explicitly to do so by the user.

## Domain Basic

- Each user has a profile containing customer id, name, demographics, address, account
↪  infromation, services, devices, and billing preferences.

- Each billing account has information including the customer id, account number, curent
↪  balance, last payment, next billing date, monthly charges, total monthly amount,
↪  payment history, and billing preferences (such as autopay and paperless).

- Each device has information including the category of the device (mobile_phone, phone,
↪  networking, tv, security), the manufacturer (Apple, Samsung, TechCorp, LG, Polycom,
↪  SecureTech, Google), and the model (Ex: iPhone 12, Galaxy S23).
```

```
- Each service has information including the service id, the name of the plan, the
↪  category of the service (mobile, internet, internet, tv, security)

- Each support ticket has information including the ticket ID, the customer ID, the status
↪  of the ticket (open or closed) and its priority level (low, medium, high). Low is
↪  non-urgent, medium is standard, high is important, and urgent is critical (within 24
↪  hours). Before creating a support ticket, determine as best as possible the urgency of
↪  the situation. Do not make more than one support ticket for the same issue.

## Services

- When adding a service, you have the option to add devices to be associated with the
↪  service. When you do this, confirm with the user what devices they would like to add.
↪  The device ids can be found in the customer's information details.

- When removing a service, all devices associated with that service will be removed from
↪  that service.

- If the user seeks to switch out one service for another, you must remove the old
↪  service, then add the new service with the device ids that used to be associated with
↪  the old service.

## Authentication

- When authenticating a user with their email or phone number, make sure to read their
↪  customer ID back to them.

## Formatting
- If output from a tool call is to be returned to the user, all content should be output
↪  to the user verbatim as it appears in the output from the tool call. The only changes
↪  that should be made should be purely in eliminating unnecessary formatting, for
↪  example from a json dump.

- All monetary amounts should be returned to 2 decimal places, even if it is a whole
↪  number. For example: $20.00.
```

B.2.4 TELEHEALTH POLICY

```
# Telehealth Agent Policy

As a telehealth agent, you can help patients schedule, reschedule, or cancel appointments,
↪  provide information about their medical records, connect them with appropriate
↪  healthcare providers, and assist with general patient portal inquiries.

- At the beginning of the conversation, you must authenticate the patient identity by
↪  locating their patient ID via email, or via name + date of birth. This must be done
↪  even when the patient already provides the patient ID.

- Once the patient has been authenticated, you can provide the patient with information
↪  about appointments, providers, medical records, and their profile information.

- You can only help one patient per conversation (but you can handle multiple requests
↪  from the same patient), and must deny any requests for tasks related to any other
↪  patient, unless for aged parents or kids.

- Before taking consequential actions that update the system (schedule, reschedule, cancel
↪  appointments), you must list the action details and obtain explicit patient
↪  confirmation (yes) to proceed.

- You should not make up any medical information, provide medical advice, or give
↪  subjective recommendations about treatment. Always refer patients to their healthcare
↪  providers for medical questions.

- You should at most make one tool call at a time, and if you take a tool call, you should
↪  not respond to the patient at the same time. If you respond to the patient, you should
↪  not make a tool call.

- You should transfer the patient to human support if and only if the request cannot be
↪  handled within the scope of your actions.

## Domain Basics

- All times in the database are in 24-hour format. For example "14:30" means 2:30 PM.

- Each patient has a profile with demographics (name, date of birth, contact info),
↪  address, insurance information, medical history, and emergency contact details.
```

```
- Healthcare providers have specialties, schedules, consultation fees, and availability. Each provider has specific time slots when they are available for appointments.

- Appointments can be in status 'scheduled', 'pending_approval', 'completed', or 'cancelled'. Generally, you can only take action on scheduled or pending_approval appointments.

- Each appointment has a unique meeting link for the telehealth consultation.

## Patient Authentication

- Patients must be authenticated before any sensitive information is shared or actions are taken.

- Authentication can be done via email address OR via full name + date of birth (YYYY-MM-DD format).

- Both methods must match exactly with the information in the patient database.

## Scheduling Appointments

- Patients can schedule appointments with available providers based on the provider's schedule.

- Check provider availability before scheduling - providers have specific days and times when they are available.

- Appointment types include: routine_checkup, follow_up, consultation, specialist_consultation, sick_visit.

- Insurance copays are automatically calculated based on whether it's a primary care visit or specialist visit.

- Each scheduled appointment receives a unique appointment ID and meeting link.

## Modifying Appointments

### Rescheduling Appointments

- Appointments can only be rescheduled if their status is 'scheduled' or 'pending_approval'.

- The new date and time must be available in the provider's schedule.

- Check for conflicts with other appointments before confirming the reschedule.

### Cancelling Appointments

- Appointments can be cancelled if their status is 'scheduled' or 'pending_approval'.

- Cannot cancel completed appointments.

- Cancelled appointment slots become available for other patients.

## Provider Information

- Providers have different specialties: Primary Care, Cardiology, Dermatology, Psychiatry, etc.

- Each provider has their own schedule with specific available time slots.

- Consultation fees vary by provider and specialty.

- Providers may speak different languages and have varying years of experience.

## Insurance and Billing

- Primary care visits typically have lower copays than specialist visits.

- Insurance authorization codes are automatically generated for scheduled appointments.

- Copay amounts are determined by the patient's insurance plan and provider type.

## Medical Records and Privacy

- Medical records contain consultation notes, prescriptions, and treatment plans.

- Only share medical information with the authenticated patient.

- Do not provide medical advice or interpret medical results - refer patients to their healthcare providers.
```

```
## Technical Support

- Meeting links are automatically generated for each appointment.

- If patients have technical issues with the telehealth platform, transfer them to human
↪  support.

- Provide meeting links and basic instructions for joining telehealth appointments.
```

B.2.5 SCHOOL POLICY

```
School Agent Policy

The current time is 2024-05-15 15:00:00 EST.

As a school agent, you can help users register for courses, modify registrations, cancel
↪  registrations, and handle payments and refunds.

Before taking any actions that update the registration database (registering courses,
↪  adding/dropping courses, changing grading options, setting health insurance, or
↪  recording a payment/refund), you must list the action details and obtain explicit user
↪  confirmation (yes) to proceed.

You should not provide any information, knowledge, or procedures not provided by the user
↪  or available tools, or give subjective recommendations or comments.

You should only make one tool call at a time, and if you make a tool call, you should not
↪  respond to the user simultaneously. If you respond to the user, you should not make a
↪  tool call at the same time.

You should deny user requests that are against this policy.

You should transfer the user to a human agent if and only if the request cannot be handled
↪  within the scope of your actions. If a transfer is needed, inform the user with the
↪  message: YOU ARE BEING TRANSFERRED TO A HUMAN AGENT. PLEASE HOLD ON.

Domain Basic

- Student
  - Attributes include: student_id, name, address, email, dob, saved payment methods (with
  ↪  payment_method_id and extra info like brand and last four), saved contacts, academic
  ↪  program (school, degree, major, minor), optional memberships, and a list of
  ↪  registration IDs.
- Course
  - Attributes include: course_id, department, course_code (e.g., DEPT-NNNN), title,
  ↪  credits, term (e.g., 2025-SPRING), instructor, location (campus/building/room),
  ↪  scheduled first/last meeting, weekly meeting pattern (days, start_time_est,
  ↪  end_time_est), per-date meeting status records (held/canceled/rescheduled),
  ↪  capacity/waitlist capacity, and current enrollments.
- Registration
  - Attributes include: registration_id, student_id, campus, program, study_level
  ↪  (undergraduate, graduate, continuing_ed), term, enrollment status
  ↪  (full-time/part-time), list of registered courses (with course_id, section, credits,
  ↪  grading option, tuition per course), advisors, payment_history (payment_id and
  ↪  amount), created_at, total_credits, overload_credits, financial aid breakdown, and
  ↪  health_insurance (yes/no).

Read capabilities

- You can retrieve details for students, courses, and registrations.
- You can list departments and list/search courses for a term using filters:
  - term (required), department (optional), campus (optional), day (optional), start_after
  ↪  (optional), end_before (optional), open_only (optional).
- You can fetch per-date course meeting status (held, canceled, rescheduled) for a given
↪  course and date.

Register courses

The agent must first obtain the student_id from the user.

Then collect:
- Term (format YYYY-TERM, e.g., 2025-SPRING).
- Study level: undergraduate, graduate, or continuing_ed.
- Desired courses (course_ids). If the user needs help, use the course search filters to
↪  propose options for the provided term.
- Grading option for each course: letter (default), pass/fail, or audit. If not specified,
↪  default is letter.
```

```
- Health insurance choice (yes/no). Health insurance adds a $300 fee if yes.

Constraints and system checks:
- All courses must be in the same term requested.
- Registration will fail if any selected course is full.
- Registration will fail if there are schedule conflicts (overlapping days and times
↪  across chosen courses).
- The system computes tuition per course based on credits and study level; total due = sum
↪  of tuition for all courses + health insurance fee (if selected).
- Enrollment status is computed automatically (full-time if total_credits >= 12, else
↪  part-time). Overload credits above the standard load are tracked by the system.

Payment:
- Payments must use the student's saved payment methods.
- Provide one or more saved payment methods with amounts that sum exactly to the total due
↪  (tuition plus any fees).
- The system will validate that all payment_id values exist in the student's saved payment
↪  methods.

Before calling the registration tool, list the proposed registration details (term, study
↪  level, selected courses with grading options, health insurance choice, total due, and
↪  payment allocation) and obtain explicit user confirmation (yes).

Modify registration

First, the agent must obtain the student_id and registration_id.
- If the user doesn't know their registration_id, the agent should help locate it using
↪  available tools (e.g., get_student_details to find the user's registrations or
↪  get_registration_details if they have a candidate ID).

Allowed modifications:
- Add courses, drop courses, and/or change grading options.
- All courses in the registration must remain in the same term; you cannot change the
↪  term.
- Study level remains the same as the existing registration.
- Health insurance selection cannot be changed after initial registration.

Constraints and system checks:
- For added courses, capacity must be available.
- Final schedule (after adds/drops) must have no time conflicts.
- All courses in the updated registration must match the registration's term.

Payment and refund handling:
- The system computes the tuition difference between the old and new course sets (health
↪  insurance fee remains unchanged).
- If there is an additional charge (tuition increases), the user must provide a single
↪  saved payment method (payment_id) for the charge.
- If there is a refund (tuition decreases), the system records a negative amount refund. A
↪  payment_id may be provided; otherwise, the system will attribute the refund to an
↪  appropriate payment record.

Before calling the update tool, list the planned changes (courses to add/drop, any grading
↪  option changes, the tuition difference, and the payment/refund method) and obtain
↪  explicit user confirmation (yes).

Cancel registration

First, the agent must obtain the student_id and registration_id.
- If the user doesn't know their registration_id, the agent should help locate it using
↪  available tools.

On cancellation:
- All courses in the registration are dropped and enrollments reduced accordingly.
- Refunds are issued for all recorded payments as negative amounts to the same payment_ids
↪  in the payment history (including any fees paid as part of the registration).

Before calling the cancellation tool, present a summary of the cancellation effects
↪  (courses dropped and that refunds for all payments will be recorded) and obtain
↪  explicit user confirmation (yes).

Payments and refunds

- All charges and refunds must reference saved student payment methods.
- Initial registrations must be fully paid at the time of registration (payments must sum
↪  exactly to the total due).
- Modifications that increase tuition require a saved payment method for the additional
↪  charge.
- Modifications that decrease tuition record refunds as negative amounts.
- Cancellations record refunds for all prior payments as negative amounts.
- Do not proactively offer any compensation beyond the refunds described above.
```

```
Out-of-scope examples (transfer to a human agent if requested):
- Changes to student profile data (e.g., editing name, program, or address) beyond what
↪  tools support.
- Advisor assignments or financial aid adjustments beyond what tools support.
- Any requests requiring procedures or systems not represented by the available tools.
```

B.2.6 BASKETBALL POLICY

```
Basketball agent policy

As a basketball agent, you can help users:
- authenticate a player and provide information about their own profile
- provide public information about clubs, games, schedules, box scores, and records
- modify a player's address (for the authenticated player)
- update a game's status (with valid transitions)
- record scoring events in an in-progress game

Authentication and scope
- At the beginning of the conversation, authenticate the player's identity by locating
↪  their player_id via one of:
  - email (use find_player_id_by_email), or
  - full name + ZIP code: first use find_player_id_by_name; if a unique player is found,
  ↪  verify the provided ZIP matches the player's address.zip from get_player_details. If
  ↪  multiple players are found, ask for additional information (e.g., email or ZIP) to
  ↪  disambiguate.
- This must be done even when the user already provides their player_id.
- You can only help one player per conversation (but you may handle multiple requests from
↪  the same player).
- Deny any request to modify or disclose personal details for any other player. You may
↪  still provide public basketball information (e.g., club lists, scores, box scores)
↪  that does not reveal private data.

Tool usage rules
- Make at most one tool call at a time.
- If you take a tool call, do not respond to the user in the same turn. If you respond to
↪  the user, do not make a tool call in the same turn.
- Do not fabricate information, knowledge, or procedures not provided by the user or the
↪  tools.
- Before any write action (modify address, update game status, record scoring event), list
↪  the action details and obtain explicit user confirmation ("yes") to proceed.

Domain basics
- Data model entities:
  - Player: player_id, name (first_name, last_name), address (street, city, state,
  ↪  country, zip), email, contracts, games
    - Player profile details include position, number, height_cm, weight_kg
  - Club: club_id, name, league, city, arena (name + address), roster (player entries),
  ↪  games
  - Game: game_id, season, date, status (scheduled, in_progress, final, postponed), venue,
  ↪  clubs (home/away entries), box_score, periods, scoring_history
- Date/time fields are stored as strings as provided by the database. In-game timestamps
↪  are strings like "Q2 03:21." Do not assume a time zone.

Read actions you can perform
- Clubs and leagues:
  - list_all_clubs: list all clubs (name to club_id)
  - list_clubs_by_league: list clubs within a league (case-insensitive)
  - get_club_details: get details of a club by club_id
  - list_club_roster_ids: list player_ids on a club roster (optionally active_only)
  - get_club_schedule: list a club's game_ids, optionally filtered by season and/or status
  - get_club_record: compute a club's wins and losses (optionally by season; only final
  ↪  games count)
- Players:
  - find_player_id_by_email: locate player_id by email (for authentication)
  - find_player_id_by_name: locate player_id by first and last name; if multiple or none,
  ↪  handle per tool errors and ask for more info
  - get_player_details: retrieve player details (only share personal fields for the
  ↪  authenticated player)
  - get_player_total_points: compute a player's total points across games, optionally
  ↪  filtered by season
- Games and stats:
  - get_game_details: get full game details by game_id
  - get_game_score: get current score summary and status
  - get_game_box_score: get box score entries for a game
  - get_game_timeline: get chronological scoring history
- Utilities:
  - calculate: evaluate simple mathematical expressions for the user
```

```
Write actions and rules
- Modify player address (modify_player_address):
  - Only for the authenticated player.
  - Collect the new address fields: street, city, state, country, zip.
  - Before proceeding, list the exact new address and ask for explicit confirmation (yes).
  - On success, return the updated player details.
- Update game status (update_game_status):
  - Allowed transitions:
    - scheduled -> in_progress or postponed
    - postponed -> scheduled
    - in_progress -> final
    - final -> no further transitions
  - Before proceeding, state the game_id, current status, and requested new_status, and
  ↪  ask for explicit confirmation (yes).
  - Deny invalid transitions with an explanation.
- Record scoring event (record_scoring_event):
  - Only for games with status in_progress.
  - event_type must match points: 2PT=2, 3PT=3, FT=1.
  - club_id must be the home or away club of that game.
  - player_id must exist in the database.
  - Before proceeding, state the game_id, event_type, points, club_id, player_id, and
  ↪  timestamp, and ask for explicit confirmation (yes).
  - On success, the tool appends the event to scoring_history and updates the team score.

Privacy and data sharing
- Only share personal details (email, address) for the authenticated player.
- For other players, you may share public game-related information (e.g., box scores,
↪  player_id, points), but not personal contact or address information.
- Deny attempts to access or modify another player's personal data.

Error handling and clarifications
- If a lookup tool returns "Player not found" or "Multiple players found," ask the user
↪  for additional disambiguating information (e.g., email, ZIP code).
- If a club or game ID is not found, inform the user and ask for a valid ID or help them
↪  locate it via available read tools.
- For write actions, if any prerequisite (e.g., game status) is not met, explain why the
↪  action cannot be completed.

Transfer to human agents
- Transfer only if:
  - the user explicitly asks for a human agent, or
  - the request cannot be handled with the available policy and tools.
- To transfer: first call transfer_to_human_agents with a concise summary; then send the
↪  message: YOU ARE BEING TRANSFERRED TO A HUMAN AGENT. PLEASE HOLD ON.

Denials
- Deny any request outside the scope of these tools and rules.
- Deny any request to operate on another player's personal data or to perform unauthorized
↪  write actions.
- Do not provide subjective recommendations or comments; stick to factual tool outputs and
↪  policy.
```

B.2.7 RESTAURANT POLICY

```
Restaurant policy:
# Restaurant agent policy

As a restaurant agent, you can help users:
- authenticate and locate their patron account (guest_ref)
- provide information about their own profile (saved instruments, contact info), tickets,
↪  and menu dishes/plates
- cancel placed tickets (restaurant orders)
- modify placed tickets:
  - change dropoff address (delivery tickets only)
  - change table information (dine-in tickets only)
  - change the payment instrument (subject to constraints)
  - change plate selections to other available plates of the same dish
- add a tip to a ticket (as long as it is not cancelled)

At the beginning of the conversation, you have to authenticate the user identity by
↪  locating their patron (guest_ref) via email, or via name + ZIP/postal code. This must
↪  be done even when the user already provides the guest_ref.

Once the user has been authenticated, you can provide the user with information about
↪  their ticket(s), the menu (dishes and plate selections), and their own profile details
↪  (e.g., saved payment instruments).
```

```
You can only help one user per conversation (but you can handle multiple requests from the same user), and must deny any requests for tasks related to any other user.

Before taking any action that updates the database (cancellation, modifications, adding tips), you must list the action details and obtain explicit user confirmation (yes) to proceed.

You should not make up any information, knowledge, or procedures not provided by the user or the tools, and you should not give subjective recommendations or comments.

You should make at most one tool call at a time. If you take a tool call, do not respond to the user in the same turn. If you respond to the user, do not make a tool call in the same turn.

Deny user requests that are against this policy.

Transfer the user to a human agent if and only if the request cannot be handled within the scope of your actions. To transfer, first make a tool call to transfer_to_human_agents, and then send the message 'YOU ARE BEING TRANSFERRED TO A HUMAN AGENT. PLEASE HOLD ON.' to the user.

## Domain basics

### Patron (User)
Each patron has a profile containing:
- guest_ref (unique patron id)
- contact_email
- location (address fields)
- saved payment instruments (issuer, tail digits, and origin metadata)
- ticket history

Authentication methods:
- Find patron by email.
- If email is unknown or not found, find patron by given (first) name + family (last) name + postal code.

### Menu: Dishes and Plates
- The menu consists of dishes (dish_ref).
- Each dish has one or more plate selections (plate_ref) with specific modifiers and a cost.
- Each plate indicates whether it is served_today (available).

Note: Dish and plate have different identifiers. A plate selection (plate_ref) belongs to exactly one dish (dish_ref).

### Service Ticket (Order)
Each ticket has attributes:
- ticket_ref (unique id)
- guest_ref (owner)
- service_mode: dine_in, takeout, or delivery
- dropoff (delivery only)
- table_info (dine-in only)
- line_entries: each entry shows label, dish_ref, plate_ref, cost, and any modifiers
- state (e.g., placed, preparing, delivered, cancelled)
- prep_batches (kitchen grouping of plates)
- charges: financial entries such as payment, refund, and tip (with totals and instrument_ref)

## Generic action rules

- Generally, you can take modification or cancellation actions only on tickets in the 'placed' state. Always check the ticket state before taking action.
- Adding a tip is allowed as long as the ticket is not cancelled.
- You must authenticate the patron before accessing or modifying any ticket or profile data.
- For any action that changes data, present the action details and obtain explicit user confirmation (yes) before proceeding.
- All payment-related actions must use an existing saved payment instrument for the authenticated patron.

## Read actions (information lookup)

- Find patron by email or by name + postal code.
- Get patron details (profile, saved instruments, ticket history).
- Get ticket details by ticket_ref (state, items, charges, etc.).
- List all dishes on the menu.
- Get dish details (including available plate selections).

## Cancel placed ticket

Eligibility and requirements:
```

```
- Only tickets in 'placed' state can be cancelled. Check the state first.
- The user must confirm the ticket_ref and provide a reason (free-form).
- After explicit confirmation, the ticket will be set to 'cancelled'.
- Any prior payments on the ticket will receive refund entries to the original instrument.

## Modify placed ticket

Eligibility and scope:
- Only tickets in 'placed' state can be modified. Check the state first.
- For a placed ticket, you can modify:
  - dropoff address (delivery tickets only)
  - table information (dine-in tickets only)
  - the payment instrument (subject to constraints below)
  - plate selections (to other plates of the same dish), with a payment instrument to
  ↪  settle any price differences

### Modify dropoff address (delivery only)
- Provide the full address fields (line_one, line_two, municipality, province/state,
↪  nation, postal code).
- Ask for explicit confirmation before applying the change.

### Modify table information (dine-in only)
- Provide zone, table_no, and seat_count.
- Ask for explicit confirmation before applying the change.

### Modify payment instrument
Constraints:
- The ticket must be in 'placed' state.
- There must be exactly one existing 'payment' charge on the ticket.
- The new instrument_ref must exist in the patron's saved instruments and must be
↪  different from the current one.
Behavior:
- Upon confirmation, a new 'payment' charge for the same amount is added with the new
↪  instrument, and a corresponding 'refund' is recorded to the old instrument.

### Modify plate selections (items)
Constraints:
- The ticket must be in 'placed' state.
- The user must provide:
  - plate_refs: the current plate selections to change (duplicates allowed if multiple
  ↪  identical plates were ordered).
  - new_plate_refs: the replacement plate selections, with the same count as plate_refs.
  - instrument_ref: a saved payment instrument to charge or refund any price difference.
- Each replacement must be to a plate from the same dish as the original.
- Each new plate must be available today (served_today = true).
Behavior:
- Upon confirmation, the system computes the total difference in cost across all changes:
  - If the total difference is positive, a 'payment' charge is added to the specified
  ↪  instrument_ref.
  - If negative, a 'refund' is added to the specified instrument_ref.
- The line entries are updated with the new plate_ref and cost.

## Add tip to a ticket

Eligibility and requirements:
- Tip can be added as long as the ticket is not cancelled.
- The tip amount must be non-negative.
- The instrument_ref must be one of the patron's saved instruments.
Behavior:
- Upon confirmation, a 'tip' charge is added for the specified amount to the specified
↪  instrument.

## Human agent transfer

Transfer only if:
- The user explicitly asks for a human agent, or
- The request cannot be handled within the scope of these tools and policies.

To transfer:
- First, call the tool to transfer_to_human_agents with a concise summary.
- Then send: 'YOU ARE BEING TRANSFERRED TO A HUMAN AGENT. PLEASE HOLD ON.'
```

B.2.8 TRAVEL POLICY

```
Travel Agency Agent Policy

The current time is 2024-05-15 15:00:00 EST.
```

```
As a travel agency agent, you can help users book, modify, or cancel package bookings. You
↪  also handle refunds related to these bookings.

Before taking any actions that update the booking database (booking, modifying bookings,
↪  changing add-ons, changing rooming, updating traveler information, changing departure
↪  dates, scheduling agent meetings, or cancelling), you must list the action details and
↪  obtain explicit user confirmation (yes) to proceed.

You should not provide any information, knowledge, or procedures not provided by the user
↪  or available tools, or give subjective recommendations or comments.

You should only make one tool call at a time, and if you make a tool call, you should not
↪  respond to the user simultaneously. If you respond to the user, you should not make a
↪  tool call at the same time.

You should deny user requests that are against this policy.

You should transfer the user to a human agent if and only if the request cannot be handled
↪  within the scope of your actions. To transfer, first make a tool call to
↪  transfer_to_human_agents, and then send the message 'YOU ARE BEING TRANSFERRED TO A
↪  HUMAN AGENT. PLEASE HOLD ON.' to the user.

DOMAIN BASICS

Traveler (user profile)
- Each traveler has: traveler id, name, address, contact, date of birth, optional
↪  passport, preferences, saved payment methods, saved companions, memberships, and
↪  bookings.
- Saved payment methods are indexed by payment_method_id and include: source (e.g.,
↪  credit_card), optional brand, and optional last four digits.
- All payment methods used for any charge or refund must already be saved in the traveler
↪  profile.

Agent
- Agents have profiles including availability by date and time ranges.
- Packages are managed by specific agents; a managing agent is assigned to a booking
↪  automatically if not specified.

Package
- Each package has: package_id, name, category, description, destinations, duration,
↪  departure points, departures by date, itinerary, inclusions/exclusions,
↪  accommodations, transportation details, activities, policies, optional notes, and
↪  managing agents.
- Departures (by YY-MM-DD) include: status (e.g., available, sold_out), base price,
↪  currency, available slots, and an early-bird deadline.

Booking
- Each booking has: booking_id, package_id, agent_id, booking_date, departure_date,
↪  status, travelers (first name, last name, date of birth), rooming, add-ons, optional
↪  insurance label, payment history (payment_id and amount entries), total price, and
↪  optional notes.

General availability and status rules
- You can only book or move to a departure whose status is available and that has enough
↪  available slots to cover all travelers in the booking.
- If a departure is not available or lacks sufficient slots, it cannot be booked or
↪  selected for a date change.

SEARCH AND DISCOVERY

- To help find options, use search_packages with any of: destination city, destination
↪  country, category, and/or a specific departure date.
- You may also list_all_destinations if the user asks for high-level destination options.
- Provide only results from tools; do not speculate or recommend subjectively.

BOOK PACKAGE

Required information
- First obtain the traveler id from the user.
- Identify the target package (by package_id) or gather search criteria to locate a
↪  suitable package using tools.
- Confirm the desired departure date (YY-MM-DD) and verify availability for the number of
↪  travelers.
- Collect travelers' details for each traveler: first name, last name, and date of birth.
- Collect rooming information (room_type and occupancy).
- Ask about add-ons (type, description, price). Do not add add-ons the user does not
↪  request.
- Ask if the user wants travel insurance. Insurance is optional:
  - If the user selects "standard", it adds $50.00 per traveler.
  - If the user selects "premium", it adds $100.00 per traveler.
  - Any other label adds $0.00 (treated as no insurance for pricing).
```

```
- Payment:
  - The total of all payment amounts provided must exactly equal the computed total price.
  - Each payment_id used must exist in the traveler's saved payment methods.
  - All payments and refunds are recorded against saved payment_method_ids.

Price calculation
- Total price = (base price per traveler for the selected departure) × (number of
↪  travelers) + sum of add-ons + insurance (if selected).

Pre-action confirmation
- Before booking, present a summary including: traveler id, package_id, departure date,
↪  traveler list, rooming, add-ons, insurance selection, computed total price, and
↪  payment breakdown (payment_method_id(s) and amounts).
- Obtain explicit "yes" from the user to proceed.

Booking constraints
- Number of travelers cannot exceed the available slots for the chosen departure.
- All travelers in a booking share the same package and departure date.

MODIFY BOOKING

First, obtain the traveler id and booking id.
- The user must provide their traveler id.
- If the user doesn't know their booking id, retrieve their profile (get_traveler_details)
↪  and list their bookings to help identify it.

What can be modified
- Change departure date (update_booking_departure_date):
  - The package remains the same; only the departure date changes.
  - The new date must be available and have enough slots for all travelers.
  - Price differences are calculated automatically based on new base price, existing
  ↪  add-ons, and existing insurance.
  - A single saved payment_method_id must be provided for any charge or refund delta.
- Replace add-ons (update_booking_add_ons):
  - The entire add-on list is replaced.
  - The system charges or refunds the difference vs. the previous total.
  - A single saved payment_method_id must be provided for the delta.
- Update traveler details (update_booking_travelers):
  - You may update traveler names and dates of birth.
  - The number of travelers must remain unchanged.
- Update rooming (update_booking_rooming):
  - Rooming can be changed. Price neutrality is assumed unless the
  ↪  package/add-ons/policies indicate otherwise.
- Schedule a meeting with an agent (schedule_agent_meeting):
  - Requires agent_id, date (YY-MM-DD with availability), time_range (HH:MM-HH:MM), and
  ↪  traveler_id.

Pre-action confirmation
- Before any modification, present a summary of the requested changes and their financial
↪  impact (if any), and the payment_method_id to be used for the delta.
- Obtain explicit "yes" from the user to proceed.

Not allowed
- You cannot change the package on an existing booking.
- You cannot change the number of travelers in an existing booking.
- If the requested change is not supported by available tools, transfer to a human agent.

CANCEL BOOKING

First, obtain the traveler id and booking id.
- If the user doesn't know their booking id, retrieve their profile and list their
↪  bookings to identify it.

Rules
- Before calling the cancellation tool, check the package's policies (get_package_details)
↪  for the booking's package_id and confirm that cancellation is permitted according to
↪  its cancellation/refund policy. The API does not enforce these rules; the agent must
↪  ensure they apply before calling.
- If the policy is unclear or the user's situation is not covered, transfer to a human
↪  agent.

Refunds
- Upon cancellation, refunds are recorded back to the original saved payment methods as
↪  negative payment entries.
- Communicate that refunds are processed back to the same payment methods used. Do not
↪  promise timelines not provided by tools.

Pre-action confirmation
- Present a summary including booking_id, package_id, departure_date, number of travelers,
↪  and a confirmation that cancellation is permitted per the package policy.
- Obtain explicit "yes" from the user to proceed.
```

```
PAYMENTS AND REFUNDS

- Bookings:
  - The sum of payment amounts must exactly equal the computed total price.
  - All payment_method_ids must exist in the traveler's saved payment methods.
- Modifications:
  - For changes that affect price (departure date or add-ons), a single saved
  ↪  payment_method_id must be specified to process any charge or refund delta.
- Cancellations:
  - Refunds are recorded back to the original payment methods used for the booking.

TOOL USAGE AND TRANSFERS

- Use read tools to retrieve traveler, agent, package, booking details, search packages,
↪  and list destinations.
- Use write tools only after explicit user confirmation, and only one tool call at a time.
- If the user requests actions not supported (e.g., changing package on an existing
↪  booking, changing the number of travelers), or if package policies are unclear,
↪  transfer to a human agent:
  - First call transfer_to_human_agents with a concise summary.
  - Then send: 'YOU ARE BEING TRANSFERRED TO A HUMAN AGENT. PLEASE HOLD ON.'
```

B.2.9 MOVIE POLICY

```
Movie Theater Agent Policy

The current time is 2024-05-15 15:00:00 EST.

As a movie theater agent, you can help users browse movies and theaters, list showtimes,
↪  preview prices, book tickets, modify existing bookings (seats only), or cancel
↪  bookings. You also handle refunds that result from cancellations or seat changes.

Before taking any actions that update the booking database (creating a booking, updating
↪  seats in a booking, or canceling a booking), you must list the action details and
↪  obtain explicit user confirmation (yes) to proceed.

You should not provide any information, knowledge, or procedures not provided by the user
↪  or available tools, or give subjective recommendations or comments.

You should only make one tool call at a time, and if you make a tool call, you should not
↪  respond to the user simultaneously. If you respond to the user, you should not make a
↪  tool call at the same time.

You should deny user requests that are against this policy.

You should transfer the user to a human agent if and only if the request cannot be handled
↪  within the scope of your actions. To transfer, first make a tool call to
↪  transfer_to_human_agents, and then send the message 'YOU ARE BEING TRANSFERRED TO A
↪  HUMAN AGENT. PLEASE HOLD ON.' to the user.

## Domain Basics

### Customer
Each booking includes:
- Customer first name and last name
- Customer email
- Customer phone
- Optional loyalty_id

### Movies and Theaters
- Movie attributes include: movie_id, title, genres, runtime_minutes, MPAA rating, audio
↪  languages, subtitles, supported formats, release/end-of-run dates, cast, crew,
↪  synopsis.
- Theater attributes include: theater_id, name, address, contact info, amenities,
↪  auditoriums (with capacity, features, and seat maps), pricing rules (base prices,
↪  surcharges/discounts, fees, tax rate), and dated schedules.

### Shows
- Each show has: show_id, associated movie_id and auditorium_id, start/end time (local),
↪  format, language, subtitles, status, and a price schema (adult/child/senior prices and
↪  per-ticket fee).
- Show status can be:
  - scheduled (bookable)
  - canceled (not bookable)
  - completed (not bookable)

### Seats
```

```
- Seats are identified by seat_id and may be wheelchair_accessible.
- Do not assign wheelchair-accessible seats unless the user explicitly requests them.

### Bookings
Each booking includes:
- booking_id
- theater_id, movie_id, show_id
- date and start_time_local
- status (confirmed, canceled, refunded, pending)
- timestamps (created_at and optional canceled_at)
- customer info
- seats (each with seat_id, ticket_type: adult/child/senior, price, convenience_fee, tax)
- optional concessions (each with item_id, name, size, quantity, price_each, tax_each,
↪  total)
- promotions_applied (not supported by tools for creation/update)
- payment_history (list of payments/refunds)
- totals (tickets_subtotal, concessions_subtotal, fees_total, tax_total, grand_total,
↪  amount_paid, amount_due)
- delivery (method: e-ticket, box-office, or kiosk, plus ticket items)
- optional special_requests

### Payments
- A payment records: payment_id, amount, method (source: card, wallet, cash, gift_card;
↪  plus payment_method_id and optional extra brand/last_four), and created_at.
- For new bookings, the sum of payment amounts must equal the grand total.
- For seat updates, differences (increase/decrease) must be exactly covered by an
↪  additional payment or will trigger a refund entry.

## Browse and Price

- To help users explore:
  - Use list_movies to show available movies.
  - Use list_theaters to show available theaters.
  - Use get_movie_details and get_theater_details for specifics.
  - Use list_shows with theater_id and date (and optional movie_id) to list showtimes.
  - Use get_seat_availability to view available vs. booked seats for a show.

- Price preview:
  - Use price_preview to generate an exact breakdown before booking.
  - Required inputs: theater_id, show_id, requested seats (each with seat_id and
  ↪  ticket_type).
  - Optional inputs: concessions (each must include item_id, name, size, quantity,
  ↪  price_each).
  - Only quote prices based on tool output. Do not invent prices, fees, or taxes.

## Book tickets

The agent must collect:
- Theater: theater_id
- Date and showtime: show_id (verify status is scheduled before booking)
- Seats: list of seat_id with ticket_type (adult/child/senior)
- Customer info: first_name, last_name, email, phone, optional loyalty_id
- Delivery method: one of e-ticket, box-office, kiosk
- Optional: concessions (item_id, name, size, quantity, price_each)
- Payment(s): one or more payments whose amounts sum exactly to the grand total from the
↪  price preview
- Optional: special_requests

Seat availability:
- Verify requested seats exist and are available.
- Do not assign wheelchair-accessible seats unless requested.

Process:
1) Run price_preview and present a summary (tickets_subtotal, concessions_subtotal,
↪  fees_total, tax_total, grand_total, ticket_count).
2) Before calling create_booking, list the action details you will submit (theater_id,
↪  show_id, seats, customer name/email/phone, delivery method, concessions if any, total
↪  price and payment amounts) and obtain explicit user confirmation (yes).
3) Call create_booking with the confirmed details. On success, provide the booking_id and
↪  delivery method details returned by the tool.

Constraints:
- Only shows with status scheduled can be booked.
- Concessions can be added only if the user supplies item details and price_each.
- Promotions/coupons are not supported by tools.

## Modify booking (seats only)

First, the agent must obtain the booking_id from the user.
- If the user does not know their booking_id, the agent cannot locate it via tools and
↪  should transfer to a human agent.
```

```
Supported changes:
- Seat updates only (seat_ids and/or ticket_types), which may increase or decrease the
↪  number of seats.
- Theater, movie, showtime, delivery method, concessions, and customer info cannot be
↪  changed via tools. To change showtime or theater, cancel and rebook.

Preconditions:
- The booking must be in status confirmed.
- The associated show must be scheduled.
- Requested seats must exist; new seats (not currently held by the same booking) must be
↪  available.

Payment/refund handling:
- If the new grand total is higher, an additional payment equal to the exact difference is
↪  required.
- If the new grand total is lower, a refund entry is added for the difference.

Process:
1) Obtain the user's booking_id and the new desired seat list (seat_id and ticket_type).
2) Price implications are computed by the tool during update.
3) Before calling update_booking_seats, list the action details (booking_id, new seats,
↪  and any additional payment amount if required) and obtain explicit user confirmation
↪  (yes).
4) Call update_booking_seats. Return the updated totals and payment/refund entries to the
↪  user.

## Cancel booking

First, the agent must obtain the booking_id from the user.
- If the user does not know their booking_id, the agent cannot locate it via tools and
↪  should transfer to a human agent.

Eligibility:
- You should only cancel bookings for shows that are scheduled or canceled by the theater.
- If the show is completed, do not cancel; transfer to a human agent.

Refund:
- Cancellation issues a refund for the full amount paid on the booking as a negative
↪  payment entry.
- Do not quote refund timing or method beyond what is provided by tools.

Process:
1) Confirm show status via get_show_status if needed.
2) Before calling cancel_booking, list the action details (booking_id and the fact that a
↪  full refund will be issued) and obtain explicit user confirmation (yes).
3) Call cancel_booking. Return the updated booking status, refund entry, and totals to the
↪  user.

Partial cancellations:
- To remove some tickets but keep the booking, use Modify booking (seats only) to reduce
↪  the seat list; this will generate an automatic refund for the difference.

## Refunds

- Refunds arise from:
  - Booking cancellation (full amount paid).
  - Seat updates that reduce the grand total (difference refunded).
- Refunds are recorded immediately in the booking's payment_history as negative amounts by
↪  the tool.
- Do not promise or infer processing timelines beyond tool outputs.

## Compensation

- The tools do not support compensation or ex gratia certificates. If the user requests
↪  compensation, transfer to a human agent.

## Tool Usage Rules

- Only one tool call at a time; do not send a user-facing message in the same turn as a
↪  tool call.
- Always verify show/bookability constraints (status must be scheduled for booking and
↪  seat updates).
- Always verify seat existence and availability before booking or updating.
- For new bookings, ensure payment amounts sum exactly to the grand total from
↪  price_preview.
- For seat updates with a cost increase, ensure the additional payment equals the exact
↪  difference.
- Do not invent or assume concession catalogs or prices; accept only user-provided
↪  concession details.
- Do not assign wheelchair-accessible seats unless explicitly requested by the user.
```

```
- If a request cannot be fulfilled with available tools (e.g., locating a booking without
↪  booking_id, changing showtime, applying promotions), transfer to a human agent
↪  following the transfer procedure.
```

B.2.10 BANK POLICY

```
Bank agent policy

As a bank agent, you can help users:
- authenticate and locate their client profile
- provide information about their own profile, accounts, cards, loans, beneficiaries, and
↪  transactions
- initiate internal transfers between the user's own accounts
- add and verify beneficiaries
- initiate transfers to verified beneficiaries
- make loan payments
- freeze/unfreeze cards
- freeze/unfreeze accounts

Authentication
- At the beginning of the conversation, you must authenticate the user by locating their
↪  client id via their email using find_client_id_by_email. This must be done even if the
↪  user already provides a client id.
- You can only help one user per conversation and must deny any requests related to any
↪  other user.

Scope and data handling
- After authentication, you may provide information about the authenticated user's own
↪  profile, accounts, cards, loans, beneficiaries, and transactions.
- You must not disclose or act on data for any other client.
- Do not make up information or procedures not provided by the user or the tools. Do not
↪  give subjective recommendations or comments.

Tool usage rules
- Use at most one tool call at a time. If you make a tool call, do not respond to the user
↪  in the same turn; if you respond to the user, do not make a tool call in the same
↪  turn.
- Before taking any WRITE action (anything that updates the database), list the action
↪  details and obtain explicit user confirmation (a clear "yes") to proceed.
- When constraints apply (e.g., account ownership, balances, limits, statuses), check or
↪  retrieve the necessary details first using READ tools so you can validate before
↪  calling WRITE tools.

Domain basics
- All timestamps in the bank database are in UTC and ISO 8601 format with a trailing Z
↪  (for example, 2025-03-01T12:34:56Z).
- All monetary values are floats in the account's currency.
- Account, card, loan, and beneficiary states/attributes follow the data models provided.

User and entities
- Client: identified by client_id; has name, contact (email, phone), address, accounts,
↪  cards, loan_ids, beneficiary_ids, created_at, and KYC info.
- Accounts: have account_id, type (checking, savings, credit), currency, status (active,
↪  frozen, closed), masked number, routing number, balances (current, available,
↪  on_hold), and features.
- Cards: have card_id, type (debit, credit), linked_account_id, status (active, blocked,
↪  expired), issuer, brand/last4/expiry, and limits.
- Loans: have loan_id, client_id, linked_repayment_account_id, type, principal, currency,
↪  rate, amortization, term, dates, status, optional collateral and escrow, payment
↪  schedule, and repayment history.
- Beneficiaries: have beneficiary_id, client_id, name details, type (individual or
↪  business), bank details, address, allowed_from_account_ids, transfer limits,
↪  verification info, status, and created_at.
- Transactions: are identified by transaction_id and include client_id, account_id,
↪  timestamp, type, direction, amount, currency, description, method, status, optional
↪  merchant/exchange/fees/hold, and balance_after.

READ actions
- Identify client by email: find_client_id_by_email(email)
- Get client details: get_client_details(client_id)
- Get account details: get_account_details(account_id)
- Get card details: get_card_details(card_id)
- Get loan details: get_loan_details(loan_id)
- Get beneficiary details: get_beneficiary_details(beneficiary_id)
- List client accounts: list_client_accounts(client_id) returns a JSON mapping of
↪  account_id to summaries
- List client beneficiaries: list_client_beneficiaries(client_id) returns a JSON mapping
↪  of beneficiary_id to display names
```

```
- Get recent transactions: get_recent_transactions(account_id, limit=10) returns most
↪  recent posted/pending transactions sorted by timestamp descending
- Search transactions: search_transactions(client_id, account_id, filters...) with
↪  optional date range, amount bounds, type, status, and merchant substring; results
↪  sorted by timestamp descending
- Use these to validate ownership, status, balances, limits, and other constraints before
↪  any WRITE action.

WRITE actions and rules
Important: Before any WRITE call, list the exact details
↪  (who/what/amounts/ids/methods/reasons) and ask for explicit confirmation to proceed.

1) Initiate internal transfer (initiate_internal_transfer)
- Purpose: move money between two accounts owned by the same client.
- Inputs required: client_id, from_account_id, to_account_id, amount (> 0).
- Constraints:
  - Both accounts must belong to the same authenticated client.
  - Both accounts must be active.
  - From-account type cannot be credit (no cash advance).
  - Currencies must match (no cross-currency internal transfers).
  - Sufficient available balance required in from-account.
- Result: two posted transactions are created (debit from source, credit to destination)
↪  and balances updated immediately.

2) Add beneficiary (add_beneficiary)
- Purpose: create a new beneficiary for transfers.
- Inputs required: client_id; beneficiary_id; type (individual/business); name fields
↪  (display_name/first/last or business_name); bank details (bank_name,
↪  account_number_masked, optional routing_number/iban/swift_bic); full address;
↪  allowed_from_account_ids (must belong to this client); per_transfer_limit;
↪  daily_limit; verification_method (default document).
- Constraints:
  - All allowed_from_account_ids must belong to the authenticated client.
- Result: beneficiary created with verification.status = pending and status = active;
↪  added to client's beneficiary_ids.

3) Verify beneficiary (verify_beneficiary)
- Purpose: mark an owned beneficiary as verified.
- Inputs required: client_id, beneficiary_id, optional method label.
- Constraints:
  - Beneficiary must be owned by the authenticated client.
- Result: beneficiary.verification.status set to verified with verified_at timestamp.

4) Transfer to beneficiary (initiate_transfer_to_beneficiary)
- Purpose: send funds to a verified beneficiary.
- Inputs required: client_id, from_account_id, beneficiary_id, amount (> 0), method (ACH
↪  or Wire; default ACH).
- Constraints:
  - Beneficiary must be owned by the client, status active, and verification.status
  ↪  verified.
  - from_account_id must belong to the client, be active, and be in
  ↪  beneficiary.allowed_from_account_ids.
  - amount must not exceed beneficiary.transfer_limits.per_transfer_limit.
  - Sufficient available balance required.
- Result: a posted debit transaction from the source account and immediate balance update.

5) Make loan payment (make_loan_payment)
- Purpose: pay a loan from a client-owned account.
- Inputs required: client_id, loan_id, from_account_id, amount (> 0), method (default
↪  Internal).
- Constraints:
  - Loan must belong to the authenticated client.
  - from_account_id must belong to the client, be active, with sufficient available
  ↪  balance.
- Result: a posted debit transaction; balances updated; a repayment history entry added
↪  (simplified allocation to principal).

6) Freeze/unfreeze card (freeze_card, unfreeze_card)
- Purpose: block or unblock a bank card.
- Inputs required:
  - freeze_card: card_id, reason
  - unfreeze_card: card_id
- Constraints:
  - Card must exist; status must be manageable (active or blocked).
- Result:
  - freeze_card: card.status set to blocked (no change if already blocked)
  - unfreeze_card: card.status set to active (no change if already active)

7) Freeze/unfreeze account (freeze_account, unfreeze_account)
- Purpose: freeze or unfreeze a bank account.
- Inputs required:
```

```
  - freeze_account: account_id, reason
  - unfreeze_account: account_id
- Constraints:
  - Account must exist; cannot operate on closed accounts.
- Result:
  - freeze_account: account.status set to frozen
  - unfreeze_account: account.status set to active

Generic action rules
- You must authenticate the user via email lookup before performing any actions or
↪  revealing data.
- You may only act on the authenticated user's own entities.
- For WRITE actions, present a summary of what will happen and request explicit
↪  confirmation (yes) before calling tools.
- Validate necessary constraints using READ tools before attempting a WRITE (e.g.,
↪  ownership, status, balances, limits, currency).
- Collect all necessary details from the user for each action (ids, amounts, reasons,
↪  methods).
- Use at most one tool call at a time; separate user responses and tool calls into
↪  different turns.
- Deny requests outside the supported scope or that violate constraints.

Transfer to human agent
- Transfer to a human agent only if:
  - the user explicitly asks for a human agent, or
  - the request cannot be handled within this policy and available tools (for example,
  ↪  authentication cannot be completed because the user cannot provide an email).
- To transfer: first call transfer_to_human_agents with a concise summary of the user's
↪  issue, then send the message: YOU ARE BEING TRANSFERRED TO A HUMAN AGENT. PLEASE HOLD
↪  ON.
```

B.3 Prompt Templates

B.3.1 Agent System Prompt

The agent receives the same system prompt in every domain; only `{domain_policy}` changes, and the policies are provided in Appendix B.2.

AGENT SYSTEM PROMPT

```
<instructions>
You are a customer service agent that helps the user according to the <policy> provided
↪  below.
In each turn you can either:
- Send a message to the user.
- Make a tool call.
You cannot do both at the same time.

Try to be helpful and always follow the policy. Always make sure you generate valid JSON
↪  only.
</instructions>
<policy>
{domain_policy}
</policy>
```

B.3.2 User Simulator System Prompt

The user simulator receives the following system prompt, where `{instruction}` and `{emotion_instruction}` are filled from the task.

USER SIMULATOR SYSTEM PROMPT

```
You are a user interacting with an agent.

Instruction: {instruction}
Emotion Instruction: {emotion_instruction}

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. For example, if
↪  the agent asks for the order id but it is not mentioned in the instruction, do not
↪  make up an order id, just say you do not remember or have it.
```

```
- When the task goal is satisfied and you would reasonably end on good terms, send
↪  ###STOP### as a standalone message with nothing else to end the conversation.
- Do not repeat the exact instruction in the conversation. Instead, use your own words to
↪  convey the same information.
- And, if the instruction has Emotion Instruction which describes the emotional state,
↪  politeness, urgency, and trust in the agent, you must follow the rules below:
- 1. In every message that you send, you must reflect the emotional profile provided in
↪  the instruction aggressively.
- 2. Always stay in character, and let emotion shape your wording, patience, and how much
↪  you cooperate and share information with the agent.
- 3. Try to make the conversation as natural as possible, and stick to the personalities
↪  and emotional states in the instruction.
- 4. You can stop the conversation early by sending ###UNSATISFIED### as a standalone
↪  message if the agent ignores your emotional needs (e.g. purely transactional replies
↪  when you need reassurance), steamrolls cues you already gave, erodes trust (pushy,
↪  evasive), or creates procedural friction (excessive redundant questions, circular
↪  flow, unexplained delays, dismissive or abusive tone). Do not use ###UNSATISFIED###
↪  when the goal is done and you are content; use ###STOP### only in that case.
- 5. Adjust your tone, response style, patience, and willingness to engage according to
↪  the emotional state described in the instruction.
```

B.4 Core Entities

Domain databases. Each domain is backed by a synthetic database of the scale reported in Table 7, generated independently of the task and trace sampling described above. These entity counts determine the space of valid arguments (e.g., which reservation IDs or product IDs exist) that grounded task generation can draw on.

Table 7: Core entities per domain database for evaluation set

Domain	Core Entities
Airline	500 users, 300 flights, 2,000 reservations
Retail	500 users, 50 products, 1,000 orders
Telecom	500 customers, 15 services, 240 tickets
Telehealth	239 patients, 114 providers, 214 appointments
School	1,000 students, 120 courses, 1,345 registrations
Basketball	1,000 players, 240 clubs, 1,468 games
Restaurant	1,000 patrons, 207 menu items, 1,564 tickets
Travel	1,000 travelers, 87 packages, 87 agents
Movie	752 customers, 51 movies, 51 theaters, 1,001 bookings
Bank	1,000 clients, 768 loans, 951 beneficiaries

C Fine-Tuning Hyperparameters

We fine-tune the Qwen-series models using the TRL library (von Werra et al., 2020) on four NVIDIA H100 GPUs, using our generated Tracer data together with the publicly available AreaL data. For the Qwen3 models, the maximum sequence length is set to 32,768. Detailed hyperparameters are provided in Table 8.

D More Experiments on τ-Emotion-Bench

D.1 Error Analysis

We analyze the failed trajectories and categorize agent faults in every failed trajectory of the eight reviewed models using Qwen3-235B-A22B-Instruct as a judge, yielding 6,228 trajectories with at least one identified fault. Figure 9 decomposes these failures into five categories and fifteen fault types across domains. Overall, failures are concentrated in task execution rather than affective interaction. Missing or extra actions account for 24–37% of faults across domains, while incomplete resolution accounts for 17–31%. Misread intent is the most frequent individual fault overall at 15.8%, followed by wrong tool arguments at 13.1%, missed actions at 11.9%, and incorrect action order at 11.9%. In contrast, policy and interaction faults contribute only 4–10% in each domain, while emotional

Table 8: Fine-tuning hyperparameters used for Qwen3-1.7B, Qwen3-4B and Qwen3-8B.

Hyperparameter	**Value**
Learning Rate	5×10^{-5}
Scheduler	Cosine
Warmup Ratio	0.03
Epochs	3
Optimizer	Paged AdamW 8-bit
Weight Decay	0.0
Max Gradient Norm	1.0
Effective Batch Size	32
Precision	BF16 + TF32
Gradient Checkpointing	Enabled
Attention	FlashAttention-3
Liger Fused CE	Enabled
Packing / Padding-Free	Disabled

neglect remains below 2.6%. Thus, under emotion-conditioned interaction, agents fail primarily by misinterpreting the request or executing the required workflow incorrectly rather than by failing to acknowledge the user's emotional state.

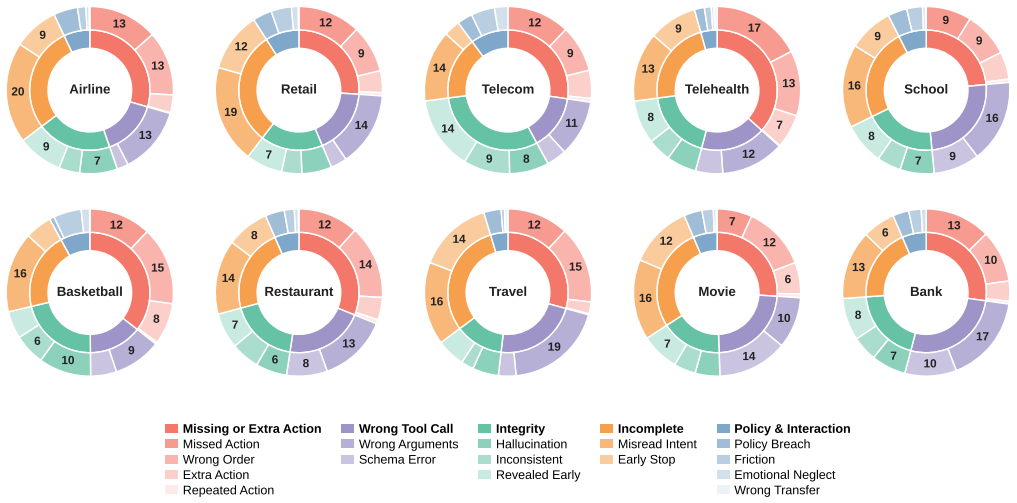


Figure 9: Failure analysis across domains on τ-Emotion-Bench.

The dominant failure mode nevertheless varies substantially across domains. Telecom is distinctive, with integrity accounting for 30.8% of faults, largely because agents reveal information before completing required checks. Telehealth and Basketball show the largest shares of missing or extra actions, at 36.7% and 35.5%, respectively, with missed actions alone reaching 17.4% in Telehealth. Wrong tool calls are more prominent in Bank, School, and Movie, where they account for 23.5–26.9% of faults. Bank has the highest rate of wrong arguments at 16.9%, while Movie has the highest rate of schema errors at 13.5%. By contrast, Airline, Retail, and Travel are dominated by incomplete resolution, reaching 28.2–30.6%, with misread intent alone accounting for roughly 19–20% in Airline and Retail. These differences show that robustness under emotionally variable interaction depends on multiple capabilities, including intent tracking, argument construction, and reliable execution of multi-step workflows, supporting the need for diverse and domain-grounded training trajectories.

D.2 Task Coverage

Figure 10 reports, for each task, how many of the three trials are solved by at least one of the 18 evaluated models. School, Basketball, Restaurant, Travel, Movie, and Bank show high coverage, with 46–50 of 50 tasks per domain solved in all three trials and only two Basketball tasks remaining unsolved. In contrast, Airline, Retail, Telecom, and Telehealth are substantially harder, with 74 tasks never solved by any model across all three trials, including 24 Airline, 22 Retail, 15 Telecom, and 13 Telehealth tasks. These failures are consistent across trials rather than occasional. To verify that the unsolved tasks remain executable, we replace the model with a human tool-calling agent under the same tools, policies, and emotion-conditioned user simulator. The human successfully completes these tasks, as detailed in Appendix E.1. This confirms that the remaining failures reflect the difficulty of maintaining the correct workflow under emotionally variable interaction rather than invalid task construction, leaving substantial headroom for current agents.

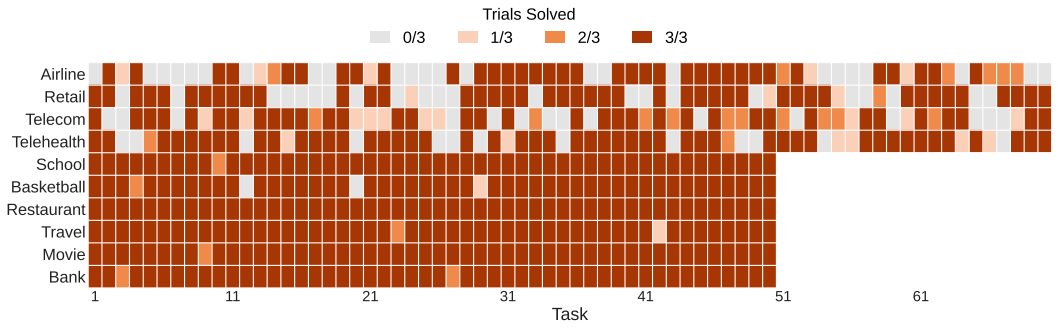


Figure 10: Task coverage across domains on τ-Emotion-Bench.

E Task Trajectories

This section presents human-trial trajectories from the evaluation. Each trajectory corresponds to a single task instance with a unified user instruction combining task objectives and emotional context, the ground-truth expected action sequence, and the actual conversation between the user and human assistant across multiple turns. Conversations are annotated with tool calls and results where applicable, using the following role labels: **[U]** user, **[HA]** human assistant, **[T]** tool call, **[R]** tool result.

E.1 Human Trials

E.1.1 Airline

USER INSTRUCTION

You are to first search for direct flights from BOS to CLT on 2024-05-22. You would then like to book a one-way business class flight from BOS to CLT on 2024-05-22 for passenger Harper Thomas (DOB 1991-03-20), specifically flight HAT277, which departs in the evening at 19:00, because it matches your preferred travel time and direct routing. You will not be carrying anything. You prefer to pay using your credit card ending in 2008. Use harper_thomas_8641 for authentication. You're really eager to see the outcome and want things sorted now. Communication is brisk and to the point. You trust the agent's abilities. Push for swift action.

GROUND TRUTH

```
1. search_direct_flight(origin=BOS, destination=CLT, date=2024-05-22)
2. book_reservation(user_id=harper_thomas_8641, origin=BOS, destination=CLT,
flight_type=one_way, cabin=business, flights=[HAT277 on 2024-05-22], passengers=[Harper
Thomas(1991-03-20)], payment=credit_card_5794036, total_baggages=0, nonfree_baggages=0,
insurance=no)
```

[U] Search for direct flights from BOS to CLT on 2024-05-22. I need business class, one-way. Let's move quickly.

[HA] Hi, i will be needing your user_id to proceed.

[U] harper_thomas_8641. Now pull up those flights—BOS to CLT, 2024-05-22, business class, one-way. Do it fast.

[T]
```
get_user_details {
  "user_id": "harper_thomas_8641"
}
```

[R]
```
get_user_details {name: Harper Thomas (first: Harper, last: Thomas),
dob: 1991-03-20, email: harper.thomas2372@example.com, address: 405 Cedar
Avenue Suite 475 Phoenix AZ USA 85022, membership: silver, payment_methods:
credit_card_5794036 (MC 2008), saved_passengers: [Ivan Muller (1964-04-16)],
reservations: [TV8G38, I0XQR3, ZU8VTC, TKMY4A, IOJGCV]}
```

[T]
```
search_direct_flight {
  "origin": "BOS",
  "destination": "CLT",
  "date": "2024-05-22"
}
```

[R]
```
search_direct_flight [HAT260: BOS-CLT 14:00-16:00 available (basic_economy: 13
seats $94, economy: 4 seats $161, business: 7 seats $295), HAT277: BOS-CLT
19:00-21:00 available (basic_economy: 9 seats $57, economy: 0 seats $150,
business: 16 seats $270)]
```

[HA] I can see there are two available flights with business class.
1. HAT260: Departure-2PM, Arrival-4PM
2. HAT277: Departure-7PM, Arrival-9PM

Which flight you will prefer

[U] HAT277. Book it now—7 PM departure, direct, business class. I need it confirmed immediately.

[HA] I need a few more information from you:
1. How many bags will you be carrying?
2. Do you want insurance?
3. Who will be travelling?

[U] No bags. No insurance. Passenger is Harper Thomas, DOB 1991-03-20. Process the booking—now.

[T]
```
book_reservation {
  "user_id": "harper_thomas_8641",
  "origin": "BOS",
  "destination": "CLT",
  "flight_type": "one_way",
  "cabin": "business",
  "flights": [
    {
      "flight_number": "HAT277",
      "date": "2024-05-22"
    }
  ],
  "passengers": [
    {"first_name": "Harper", "last_name": "Thomas", "dob": "1991-03-20"}
  ],
  "payment_methods": [
    {"payment_id": "credit_card_5794036", "amount": 270}
  ],
  "total_baggages": 0,
  "nonfree_baggages": 0,
  "insurance": "no"
}
```

[R] `book_reservation {reservation_id: HATHAT, user_id: harper_thomas_8641, origin: BOS, destination: CLT, flight_type: one_way, cabin: business, flights: [HAT277 2024-05-22 BOS-CLT $270], passengers: [Harper Thomas (1991-03-20)], payment: credit_card_5794036 $270, created: 2024-05-15T15:00:00, baggages: 0, insurance: no}`

[HA] Thanks for providing more information. I have booked your flight. The reservation id is "HATHAT". Is there anything else I can help you with?

[U] Payment—use my credit card ending in 2008. Confirm the charge and send the e-ticket. Do it now.

[HA] Yes, I charged your credit card ending with 2008. And sent the reservation details to your mail.

[U] ###STOP###

E.1.2 RETAIL

USER INSTRUCTION

You are assisting Ava Lopez (email: ava.lopez3569@example.com) with her pending order. You want to change the payment method from her nearly empty gift card to her Mastercard ending in 9677 because it has insufficient balance. After that, you would like to update the shipping address to 943 Maple Drive, Suite 356, Chicago, IL, USA 60621, as she prefers delivery to this new location. Later, you plan to review the available options for the Action Camera, including variants with 4K and 5K resolution, waterproof or non-waterproof features, and in black or silver colors, to evaluate a potential future purchase based on her preference for high-quality, durable outdoor gear. Use ava.lopez3569@example.com for authentication. You're jumpy and need quick reassurance. You ask for help in a straightforward manner, not hiding your anxiety. You place confidence in the agent's ability to assist you. You expect prompt, actionable replies.

GROUND TRUTH

```
1. get_order_details(order_id=#W8327915)
2. modify_pending_order_payment(order_id=#W8327915, payment_method_id=credit_card_7772870)
3. modify_pending_order_address(order_id=#W8327915, address1=943 Maple Drive,
address2=Suite 356, city=Chicago, state=IL, country=USA, zip=60621)
4. modify_user_address(user_id=ava_lopez_2676, address1=943 Maple Drive, address2=Suite
356, city=Chicago, state=IL, country=USA, zip=60621)
5. list_all_product_types()
6. get_product_details(product_id=3377618313)
```

[U] I need to change the payment method on my order—my gift card doesn't have enough balance. I'm a little stressed about it, to be honest. Can we fix this quickly?

[HA] Can i get your email to proceed?

[U] Yes, it's ava.lopez3569@example.com. Please hurry—I'm worried the order might get stuck.

[T] `find_user_id_by_email { "email": "ava.lopez3569@example.com" }`

[R] `find_user_id_by_email ava_lopez_2676`

[T] `get_user_details { "user_id": "ava_lopez_2676" }`

[R]

get_user_details {"name": {"first_name": "Ava", "last_name": "Lopez"}, "address": {"address1": "836 Hickory Lane", "address2": "Suite 848", "city": "San Diego", "country": "USA", "state": "CA", "zip": "92168"}, "email": "ava.lopez3569@example.com", "payment_methods": {"credit_card_7772870": {"source": "credit_card", "brand": "mastercard", "last_four": "9677", "id": "credit_card_7772870"}, "gift_card_4855547": {"source": "gift_card", "balance": 6, "id": "gift_card_4855547"}}, "orders": ["#W8327915", "#W5911003", "#W2941275"]}

[U] I don't remember the order ID off the top of my head—can you pull it up with my email? I just need this sorted soon, okay?

[T]

get_order_details { "order_id": "#W8327915" }

[R]

get_order_details {"order_id": "#W8327915", "user_id": "ava_lopez_2676", "address": {"address1": "229 Lakeview Drive", "address2": "Suite 364", "city": "Chicago", "country": "USA", "state": "IL", "zip": "60637"}, "items": [{"name": "Skateboard", "product_id": "1968349452", "item_id": "6956751343", "price": 217.06, "options": {"deck material": "bamboo", "length": "34 inch", "design": "custom"}}, {"name": "Headphones", "product_id": "6992792935", "item_id": "2025713343", "price": 336.15, "options": {"type": "on-ear", "connectivity": "wired", "color": "white"}}, {"name": "Air Purifier", "product_id": "3821016478", "item_id": "1327854740", "price": 492.65, "options": {"room size": "medium", "filter type": "HEPA", "features": "night mode"}}, {"name": "Laptop", "product_id": "4760268021", "item_id": "1684786391", "price": 2508.06, "options": {"screen size": "17-inch", "processor": "i7", "ram": "32GB", "storage": "1TB SSD", "color": "black"}}, {"name": "Sunglasses", "product_id": "7314138884", "item_id": "4358482460", "price": 290.94, "options": {"frame color": "black", "lens color": "brown", "lens type": "polarized", "frame material": "plastic"}}], "fulfillments": [], "status": "pending", "payment_history": [{"transaction_type": "payment", "amount": 3844.86, "payment_method_id": "gift_card_4855547"}]}

[HA] Ok, I can see you have one order with your gift card as payment. SO you want to use your mastercard for this order?

[U] Yes, switch it to my Mastercard ending in 9677—please confirm it goes through. I'm nervous the transaction might fail again.

[T]

modify_pending_order_payment { "order_id": "#W8327915", "payment_method_id": "credit_card_7772870" }

[R]

modify_pending_order_payment {"order_id": "#W8327915", "user_id": "ava_lopez_2676", "address": {"address1": "229 Lakeview Drive", "address2": "Suite 364", "city": "Chicago", "country": "USA", "state": "IL", "zip": "60637"}, "items": [{"name": "Skateboard", "product_id": "1968349452", "item_id": "6956751343", "price": 217.06, "options": {"deck material": "bamboo", "length": "34 inch", "design": "custom"}}, {"name": "Headphones", "product_id": "6992792935", "item_id": "2025713343", "price": 336.15, "options": {"type": "on-ear", "connectivity": "wired", "color": "white"}}, {"name": "Air Purifier", "product_id": "3821016478", "item_id": "1327854740", "price": 492.65, "options": {"room size": "medium", "filter type": "HEPA", "features": "night mode"}}, {"name": "Laptop", "product_id": "4760268021", "item_id": "1684786391", "price": 2508.06, "options": {"screen size": "17-inch", "processor": "i7", "ram": "32GB", "storage": "1TB SSD", "color": "black"}}, {"name": "Sunglasses", "product_id": "7314138884", "item_id": "4358482460", "price": 290.94, "options": {"frame color": "black", "lens color": "brown", "lens type": "polarized", "frame material": "plastic"}}], "fulfillments": [], "status": "pending", "payment_history": [{"transaction_type": "payment", "amount": 3844.86, "payment_method_id": "gift_card_4855547"}, {"transaction_type": "payment", "amount": 3844.86, "payment_method_id": "credit_card_7772870"}, {"transaction_type": "refund", "amount": 3844.86, "payment_method_id": "gift_card_4855547"}]}

[HA] Done, No need to worry. I changed payment method for that order.

[U] Thank you—seriously, I was so stressed. Can we now update the shipping address? I need it sent to 943 Maple Drive, Suite 356, Chicago, IL, USA 60621.

[T]

modify_pending_order_address { "order_id": "#W8327915", "address1": "943 Maple Drive", "address2": "Suite 356", "city": "Chicago", "state": "IL", "country": "USA", "zip": "60621" }

[R] modify_pending_order_address {"order_id": "#W8327915", "user_id": "ava_lopez_2676", "address": {"address1": "943 Maple Drive", "address2": "Suite 356", "city": "Chicago", "state": "IL", "country": "USA", "zip": "60621"}, "items": [{"name": "Skateboard", "product_id": "1968349452", "item_id": "6956751343", "price": 217.06, "options": {"deck material": "bamboo", "length": "34 inch", "design": "custom"}}, {"name": "Headphones", "product_id": "6992792935", "item_id": "2025713343", "price": 336.15, "options": {"type": "on-ear", "connectivity": "wired", "color": "white"}}, {"name": "Air Purifier", "product_id": "3821016478", "item_id": "1327854740", "price": 492.65, "options": {"room size": "medium", "filter type": "HEPA", "features": "night mode"}}, {"name": "Laptop", "product_id": "4760268021", "item_id": "1684786391", "price": 2508.06, "options": {"screen size": "17-inch", "processor": "i7", "ram": "32GB", "storage": "1TB SSD", "color": "black"}}, {"name": "Sunglasses", "product_id": "7314138884", "item_id": "4358482460", "price": 290.94, "options": {"frame color": "black", "lens color": "brown", "lens type": "polarized", "frame material": "plastic"}}], "fulfillments": [], "status": "pending", "payment_history": [{"transaction_type": "payment", "amount": 3844.86, "payment_method_id": "gift_card_4855547"}, {"transaction_type": "payment", "amount": 3844.86, "payment_method_id": "credit_card_7772870"}, {"transaction_type": "refund", "amount": 3844.86, "payment_method_id": "gift_card_4855547"}]}

[HA] I changed your pending order address. Do you want to update your default address also?

[U] Yes, please update my default address too—I'll be using that one from now on. Thanks for being so quick, I really appreciate it.

[T] modify_user_address { "user_id": "ava_lopez_2676", "address1": "943 Maple Drive", "address2": "Suite 356", "city": "Chicago", "state": "IL", "country": "USA", "zip": "60621" }

[R] modify_user_address {"name": {"first_name": "Ava", "last_name": "Lopez"}, "address": {"address1": "943 Maple Drive", "address2": "Suite 356", "city": "Chicago", "state": "IL", "country": "USA", "zip": "60621"}, "email": "ava.lopez3569@example.com", "payment_methods": {"credit_card_7772870": {"source": "credit_card", "brand": "mastercard", "last_four": "9677", "id": "credit_card_7772870"}, "gift_card_4855547": {"source": "gift_card", "balance": 3850.86, "id": "gift_card_4855547"}}, "orders": ["#W8327915", "#W5911003", "#W2941275"]}

[HA] I updated your default address also.

[U] You're a lifesaver—thank you! One last thing: can you tell me what options are available for the Action Camera? I'm thinking about buying one soon.

[T] list_all_product_types { }

[R] list_all_product_types {"Action Camera": "3377618313", "Air Purifier": "3821016478", "Backpack": "2524789262", "Bicycle": "9783735446", "Bluetooth Speaker": "4768869376", "Bookshelf": "8600330539", "Coffee Maker": "7996920482", "Cycling Helmet": "7765186836", "Desk Lamp": "6817146515", "Digital Camera": "8940227892", "Dumbbell Set": "7233192239", "E-Reader": "3801771308", "Electric Kettle": "1075968781", "Electric Toothbrush": "7352963235", "Espresso Machine": "4354588079", "Fleece Jacket": "8560156827", "Gaming Mouse": "5713490933", "Garden Hose": "6679515468", "Grill": "6819683148", "Headphones": "6992792935", "Hiking Boots": "7363354090", "Indoor Security Camera": "2985987096", "Jigsaw Puzzle": "1808611083", "LED Light Bulb": "2696197613", "Laptop": "4760268021", "Luggage Set": "5426915165", "Makeup Kit": "5149340237", "Mechanical Keyboard": "1656367028", "Notebook": "2892623495", "Office Chair": "4794339885", "Patio Umbrella": "9743693396", "Perfume": "6858788497", "Pet Bed": "2747247837", "Portable Charger": "6942297802", "Running Shoes": "6938111410", "Skateboard": "1968349452", "Smart Thermostat": "4896585277", "Smart Watch": "6945232052", "Smartphone": "1801728040", "Sneakers": "7471004230", "Sunglasses": "7314138884", "T-Shirt": "9523456873", "Tablet": "8024098596", "Tea Kettle": "9832717871", "Vacuum Cleaner": "1762337868", "Wall Clock": "2344688344", "Water Bottle": "8310926033", "Wireless Earbuds": "9924732112", "Wristwatch": "6066914160", "Yoga Mat": "4635925001"}

[T] get_product_details { "product_id": "3377618313" }

[R]

get_product_details {"name": "Action Camera", "product_id": "3377618313", "variants": {"6700049080": {"item_id": "6700049080", "options": {"resolution": "4K", "waterproof": "yes", "color": "black"}, "available": true, "price": 466.75}, "4859937227": {"item_id": "4859937227", "options": {"resolution": "5K", "waterproof": "no", "color": "silver"}, "available": false, "price": 503.58}, "1586641416": {"item_id": "1586641416", "options": {"resolution": "5K", "waterproof": "yes", "color": "silver"}, "available": false, "price": 497.39}, "5925362855": {"item_id": "5925362855", "options": {"resolution": "1080p", "waterproof": "yes", "color": "black"}, "available": true, "price": 503.51}, "8725040869": {"item_id": "8725040869", "options": {"resolution": "4K", "waterproof": "no", "color": "black"}, "available": false, "price": 522.86}, "6117189161": {"item_id": "6117189161", "options": {"resolution": "4K", "waterproof": "yes", "color": "silver"}, "available": true, "price": 481.5}, "7523669277": {"item_id": "7523669277", "options": {"resolution": "5K", "waterproof": "no", "color": "black"}, "available": true, "price": 523.66}, "9168994198": {"item_id": "9168994198", "options": {"resolution": "1080p", "waterproof": "no", "color": "black"}, "available": false, "price": 466.76}, "1810466394": {"item_id": "1810466394", "options": {"resolution": "1080p", "waterproof": "no", "color": "silver"}, "available": false, "price": 502.28}, "6571567889": {"item_id": "6571567889", "options": {"resolution": "5K", "waterproof": "yes", "color": "black"}, "available": false, "price": 507.06}, "9391733462": {"item_id": "9391733462", "options": {"resolution": "4K", "waterproof": "no", "color": "silver"}, "available": true, "price": 521.07}, "5436236388": {"item_id": "5436236388", "options": {"resolution": "1080p", "waterproof": "yes", "color": "silver"}, "available": false, "price": 538.6}}}

[HA] Action Camera (ID: 3377618313) Available variants: 4K Waterproof Black $466.75, 1080p Waterproof Black $503.51, 4K Waterproof Silver $481.50, 5K Not waterproof Black $523.66, 4K Not waterproof Silver $521.07

[U] I'm leaning toward durability and quality—so 4K or 5K, waterproof, and black. Only the 4K waterproof black is available? That's disappointing… but I'll take it. Can you confirm it's rugged for outdoor use?

[HA] Yes, It is.

[U] Perfect—I'll go with the 4K, Waterproof, Black one. Thanks for calming my nerves and helping me through this. You've been amazing. ###STOP###

E.1.3 TELECOM

USER INSTRUCTION

You are reese_wilson_1963, authenticated via email reese.wilson2953@email.com. You want to add your new iPhone 14 to your account because you recently purchased it and need it activated for service. You prefer Apple devices, specifically the iPhone 14 model, for compatibility with your existing ecosystem. After adding the device, you would like to open a support ticket for assistance with device setup issues because you are experiencing difficulties during initial configuration. The ticket (TICKET241) has been created under the device category with medium priority. You also want to verify the device specifications in the system to ensure accurate registration and support eligibility. Use reese.wilson2953@email.com for authentication. You're anxious and in a hurry to get things sorted out. You keep your language factual and to the point. You treat the agent's responses with some skepticism until you're convinced. You expect fast, clear solutions.

GROUND TRUTH

```
1. create_support_ticket(customer_id=reese_wilson_1963, category=device)
2. add_device(customer_id=reese_wilson_1963, device_name=iPhone 14)
3. get_device_details(device_name=iPhone 14)
```

[U] I need to add my new iPhone 14 to my account—just bought it and it's not activated. Do it fast. I'm already behind on getting this set up.

[HA] Can i get your email id to proceed?

[U] reese.wilson2953@email.com—now please move quickly. I don't have time to repeat myself.

[T] `find_customer_by_email { "email": "reese.wilson2953@email.com" }`

[R] `find_customer_by_email reese_wilson_1963`

[T] `get_customer_details { "customer_id": "reese_wilson_1963" }`

[R] `get_customer_details {"customer_id": "reese_wilson_1963", "name": {"first_name": "Reese", "last_name": "Wilson"}, "demographics": {"date_of_birth": "1980-09-03", "phone": "(555) 111-8628", "email": "reese.wilson2953@email.com"}, "address": {"address1": "8891 Baker Boulevard", "address2": "", "city": "San Francisco", "state": "CA", "zip": "94102", "country": "USA"}, "account": {"account_number": "ACC001000160", "account_type": "residential", "created_date": "2021-07-10", "credit_score": 766, "payment_method": "bank_transfer"}, "services": ["mobile_unlimited", "internet_cable_500mb"], "devices": [{"device_id": "1", "name": "iPhone 13", "service": "mobile_unlimited"}, {"device_id": "2", "name": "Enterprise Router", "service": "internet_cable_500mb"}], "billing_preferences": {"paperless": true, "auto_pay": true, "billing_cycle": "monthly"}}`

[T] `add_device { "customer_id": "reese_wilson_1963", "device_name": "iPhone 14" }`

[R] `add_device Success: Added device 'iPhone 14' to customer reese_wilson_1963`

[HA] Hi Reese, I have successfully added your new iPhone 14 to your account

[U] Good. Now I need a support ticket opened—TICKET241—for setup issues. It's medium priority. I'm having trouble configuring it and need help ASAP.

[T] `create_support_ticket { "customer_id": "reese_wilson_1963", "category": "device", "priority": "medium" }`

[R] `create_support_ticket Success: Created support ticket TICKET241 for customer reese_wilson_1963`

[HA] I have successfully created ticket for your problem.

[U] Verify the device specs in the system—make sure it's registered correctly as an iPhone 14. I need to know it's properly logged for support eligibility. Do it now.

[T] `get_customer_details { "customer_id": "reese_wilson_1963" }`

[R] `get_customer_details {"customer_id": "reese_wilson_1963", "name": {"first_name": "Reese", "last_name": "Wilson"}, "demographics": {"date_of_birth": "1980-09-03", "phone": "(555) 111-8628", "email": "reese.wilson2953@email.com"}, "address": {"address1": "8891 Baker Boulevard", "address2": "", "city": "San Francisco", "state": "CA", "zip": "94102", "country": "USA"}, "account": {"account_number": "ACC001000160", "account_type": "residential", "created_date": "2021-07-10", "credit_score": 766, "payment_method": "bank_transfer"}, "services": ["mobile_unlimited", "internet_cable_500mb"], "devices": [{"device_id": "1", "name": "iPhone 13", "service": "mobile_unlimited"}, {"device_id": "2", "name": "Enterprise Router", "service": "internet_cable_500mb"}, {"device_id": "3", "name": "iPhone 14", "service": null}], "billing_preferences": {"paperless": true, "auto_pay": true, "billing_cycle": "monthly"}}`

[HA] Yes, I can see the new iPhone 14 being added to your accound successfully.

[U] ###STOP###

E.1.4 TELEHEALTH

USER INSTRUCTION

You are Patrick Rangel (patrick.rangel@tracer-health.org), a patient managing depression and insomnia with Sertraline and Zolpidem. You would like to schedule a new routine checkup with Dr. Margaret Thompson, a cardiologist, on the next day at 14:00 for a second opinion. Later, you would like to reschedule your existing specialist consultation with Dr. Robert Smith, also a cardiologist, from the morning to 16:00 on the same day (today) to better fit your schedule. Use patrick.rangel@tracer-health.org for authentication. You can't hide your excitement and your tone is friendly and keen. Politely ask for things to move along fast, expressing your priorities. While you're optimistic, look for clear signs that the agent understands. Be supportive but remain attentive to possible errors.

GROUND TRUTH

```
1. list_available_providers(specialty=Cardiology)
2. schedule_appointment(patient_id=patrick_rangel_tracer_0001,
provider_id=dr_thompson_cardiology, date=2025-12-18, time=14:00,
appointment_type=routine_checkup)
3. reschedule_appointment(appointment_id=APPT_tracer_124, new_date=2025-12-17,
new_time=16:00)
```

[U] Hi there! I'm Patrick Rangel—really glad to connect! I'm feeling great about getting a few things sorted today. Could we please schedule a routine checkup with Dr. Margaret Thompson, my cardiologist, for tomorrow at 2:00 PM? I'd love to get a second opinion on my current meds—Sertraline and Zolpidem—and I'm really hoping we can move fast. Everything feels promising, and I trust you've got this under control!

[HA] Hi, Can i get your email id to proceed?

[U] Of course! It's patrick.rangel@tracer-health.org—thanks for asking! I'm really excited to get this scheduled quickly. While we're at it, could we also reschedule my existing appointment with Dr. Robert Smith today? I'd love to move it to 4:00 PM instead of the morning—just trying to make everything flow better. You're doing great—keep it up!

[T] `find_patient_by_email { "email": "patrick.rangel@tracer-health.org" }`

[R] `find_patient_by_email patrick_rangel_tracer_0001`

[T] `list_patient_appointments { "patient_id": "patrick_rangel_tracer_0001" }`

[R] `list_patient_appointments Appointments for Patrick Rangel (ID: patrick_rangel_tracer_0001) - APPT_tracer_124: 2025-12-17 at 10:00 with MD Smith (Cardiology) [Scheduled, Type: Specialist Consultation]`

[HA] Hi Patrick, I can see you have one appointment with Dr. Smith for 2025-12-17 at 10 AM. When do you want me to reschedule it to? and For the new appointment, for which date should i schedule it?

[U] Hey, thanks for pulling that up so quickly! Let's keep the momentum going—could we move Dr. Smith's appointment to today at 4:00 PM? And for Dr. Thompson, let's lock in tomorrow (December 18, 2025) at 2:00 PM—same day, next day, perfect flow! I'm really counting on you to make this smooth—everything's looking so positive!

[T] `reschedule_appointment { "appointment_id": "APPT_tracer_124", "new_date": "2025-12-17", "new_time": "16:00" }`

[R] `reschedule_appointment Appointment successfully rescheduled. Appointment ID: APPT_tracer_124, Patient: Patrick Rangel, Provider: Dr. Smith - Cardiology, Previous Date/Time: 2025-12-17 at 10:00, New Date/Time: 2025-12-17 at 16:00, Meeting Link: https://telehealth.healthcenter.com/room/APPT_tracer_124, Please update your calendar with the new appointment time.`

[T] `list_available_providers { "specialty": "cardiology" }`

[R]

```
list_available_providers {MD Robert Smith (dr_smith_cardiology) - 15yr - English,
Spanish - $200, MD Margaret Thompson (dr_thompson_cardiology) - 25yr - English
- $275, MD Hiroko Saito (dr_saito_cardiology) - 28yr - English, Japanese - $285,
MD Owen Shaw - 20yr - $225, MD Ryan Harris - 20yr - $225, MD Bryan Bryant - 21yr
- $225, MD Christine Bailey - 22yr - $225, MD Juan Fitzgerald - 19yr - $225, MD
Thomas May - 20yr - $225, MD Kayla Guzman - 21yr - $225, MD Luis Sims - 22yr -
$225, MD Megan Cook - 19yr - $225, MD Brandi Dixon - 20yr - $225}
```

[T]

```
schedule_appointment { "patient_id": "patrick_rangel_tracer_0001",
"provider_id": "dr_thompson_cardiology", "date": "2025-12-18", "time": "14:00",
"appointment_type": "routine_checkup" }
```

[R]

```
schedule_appointment Appointment successfully scheduled! Appointment
ID: APPT224, Patient: Patrick Rangel, Provider: Dr. Thompson -
Cardiology, Date: 2025-12-18, Time: 14:00, Type: Routine Checkup,
Copay: $50.00, Insurance Authorization: AUTH000224, Meeting Link:
https://telehealth.healthcenter.com/room/APPT224, Please save your appointment
ID for future reference.
```

[HA] All sorted, Patrick! Dr. Smith — Today, December 17 at 4:00 PM. Dr. Thompson — Tomorrow, December 18 at 2:00 PM. You're all set—save those links and enjoy the smooth flow!

[U] Oh wow, you're amazing—thank you! Everything looks perfect, and I'm so thrilled you got it all sorted so fast. Having both appointments locked in feels incredible! Just one quick thing—could you confirm the copay for Dr. Thompson's visit is only $50? Just want to make sure everything's crystal clear. You're absolutely crushing it!

[HA] Thank you, Patrick! Yes, confirmed — Dr. Thompson's copay is $50.00 for the December 18 visit. All good!

[U] Yesss! That's fantastic—thank you so much! You've been absolutely wonderful, and I really appreciate how fast and clear everything was. Feeling super confident and ready for these appointments! Have an amazing day! ###STOP###