UserBench: An Interactive Gym Environment for User-Centric Agents

Anonymous ARR submission

Abstract

Large Language Model agents increasingly solve tool-use tasks, yet often fail to determine what users actually want when goals are underspecified, incremental, or indirectly expressed. We introduce **UserBench**, a controlled Gymnasium environment for evaluating proactive preference elicitation in multi-turn, tool-grounded interactions. Instantiated in travel planning, UserBench hides user preferences behind implicit natural-language cues and requires agents to ask targeted clarification questions, search structured option databases, and select user-aligned options. Evaluating leading open- and closed-source models reveals a large gap between tool execution and user understanding: models achieve high valid-search rates but elicit only a minority of hidden preferences, and even stronger models remain far below a human oracle. Further analysis shows that many additional interaction turns are broad or non-informative rather than preference-relevant. UserBench provides a reproducible testbed for studying and training user-centric agent interaction while intentionally abstracting away full travel negotiation and infeasible-constraint recovery.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in complex reasoning (Jaech et al., 2024; Guo et al., 2025), code generation (Hui et al., 2024; Guo et al., 2024; Zhang et al., 2024b), and solving advanced mathematical problems (Yang et al., 2024; Shao et al., 2024; Qian et al., 2025c). Their limitations in updated knowledge and precise computation are increasingly mitigated through tool use (Liu et al., 2024; Qian et al., 2024c; Zhang et al., 2024a; Prabhakar et al., 2025) and tool creation (Qian et al., 2023; Cai et al., 2024; Yuan et al., 2024; Qian et al., 2024d), enabling them to retrieve information and interact with external environments including database, web and games (Deng et al., 2024; Qian et al., 2024a; Zhu et al., 2025). This tool-augmented reasoning grants LLMs agentic capabilities: autonomously execute tasks via these tools.

However, existing agentic environments often overlook a critical dimension: the role of the user. Despite achieving strong task performance, agents frequently fail to satisfy real user needs due to their inability to understand, adapt to, and collaborate with the task initiator (Qian et al., 2024b; Lu et al., 2025). Current evaluations primarily assess tool use and task execution, rarely considering whether the agent effectively interprets and aligns with the user's underlying and evolving intent.

This motivates our central research question: **How can we evaluate agents from a user-centric perspective?** To answer this, we first examine how users typically communicate goals. Human communication is inherently a joint activity, where meaning is co-constructed through interaction (Clark, 1996). Moreover, language is inherently ambiguous, making it difficult for users to fully and clearly convey their intent in a single interaction. (Liu et al., 2023). As such, user instructions tend to share three core traits: (i) **Underspecification**: users often initiate requests before fully formulating their goals; (ii) **Incrementality**: intent emerges and evolves across interaction; and (iii) **Indirectness**: users may obscure or soften their true intent due to social or strategic reasons.

Revolving around these traits, we introduce **UserBench**, a user-centric environment designed to facilitate an agent's ability to engage in meaningful, multi-turn interactions with users who exhibit these traits. In UserBench, simulated users provide initial vague task instruction (underspecification), gradually reveal preferences over time (incrementality), and often do so implicitly (indirectness). Agents must proactively clarify goals, interpret subtle cues, and adaptively reason through tool use to succeed.

Built on the standard Gymnasium framework,

Benchmark	Multi-turn Interaction	Goal Ambiguity	Tool Use	Dynamic State	Multi-Aspect Reasoning	User Simulation	Domain Diversity	Custom-izable	Scal-able
τ^2-Bench(Barres et al., 2025)	✓	✗	✓	✓	✗	✓	✗	✓	✓
τ-Bench(Yao et al., 2024)	✓	✗	✓	✓	✓	✓	✓	✓	✓
ToolSandbox(Lu and et al., 2024)	✓	✓	✓	✓	✓	✓	✓	✓	✓
MINT(Wang et al., 2024b)	✓	✓	✓	✓	✓	✓	✓	✓	✓
IN3(Qian et al., 2024b)	✓	✓	✗	✗	✗	✓	✓	✗	✗
ToolTalk(Farn and Shin, 2023)	✓	✗	✓	✗	✗	✗	✓	✗	✗
API-Bank(Li et al., 2023)	✗	✗	✓	✗	✗	✗	✓	✓	✓
MetaTool(Huang et al., 2023)	✗	✗	✓	✗	✗	✗	✓	✓	✓
UserBench (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: For each existing benchmark, the table indicates whether the corresponding trait is fully addressed (✓), partially addressed (✓), or not addressed (✗). We provide detailed explanation in Section A.

UserBench offers a modular, extensible setup with a standardized interaction interface and a stable tool-use backend, enabling rigorous and reproducible evaluation. We benchmark several leading open- and closed-source models and find that current LLMs still struggle to interactively uncover and act on user preferences. For instance, scores drop by over 40% on average when models are restricted to selecting only one option per traveling aspect in UserBench, revealing their difficulty in making optimal decisions. Moreover, models provide answers that fully align with all user intents only 20% of the time on average, and even the best-performing models elicit less than 30% of all user preferences through active querying, suggesting limited ability to engage in purposeful, user-driven dialogue. While strong models handle tool use reliably, they remain brittle in understanding implicit and nuanced human needs, highlighting the core challenge that UserBench aims to measure. We summarize our contributions as follows:

- We propose UserBench, a controlled Gymnasium style benchmark for evaluating proactive preference elicitation under underspecified, incremental, and indirect user intent.
- We instantiate the benchmark in a tool-grounded travel setting with curated hidden preferences, implicit user expressions, structured search tools, and noisy distractor options.
- We provide a systematic evaluation of current LLMs and show that their main bottleneck is not tool-use syntax, but targeted user understanding and preference-grounded decision-making.

2 Related Work

User-centric environments for LLM evaluation. Recent work increasingly evaluates LLMs under realistic, user-centric conditions, moving beyond traditional benchmarks that assume fully specified prompts. Benchmarks such as User Reported Scenarios (Wang et al., 2024a), Intention-in-Interaction (Qian et al., 2024b), and WildBench (Lin et al., 2024) collect real user queries to test whether models satisfy nuanced intents in in-the-wild settings. Other benchmarks, including MINT (Wang et al., 2024b), PrefEval (Zhao et al., 2025), τ-Bench (Yao et al., 2024), and τ^2-Bench (Barres et al., 2025), further study dynamic multi-turn interactions, where agents must incorporate feedback, track evolving preferences, and remain aligned with users over time. However, many existing environments rely heavily on synthesized interactions, which often produce overly long or unnatural goal descriptions. In contrast, UserBench explicitly models three core interactive traits of real user communication and covers hundreds of diverse goal configurations, enabling fine-grained, behaviorally grounded evaluation of user-centric agents. We compare UserBench with recent environments and benchmarks in Table 1.

User-centric agent designs. Effective user-centric agents must handle ambiguity, evolving intent, and personalized preferences, yet standard instruction-tuned models often hallucinate intent or fail to ask useful clarifying questions. Recent methods train agents to clarify underspecified instructions through simulated ambiguity resolution (Zhang et al., 2024c; Chen et al., 2025). Others focus on user modeling: TravelPlanner+ (Singh et al., 2024) and PRELUDE (Gao et al., 2024) personalize responses using explicit profiles or latent preferences inferred from user edits. While these systems improve adaptive interaction, they are often limited to narrow personalization settings or static user models. In contrast, our work provides a general-purpose evaluation environment that systematically tests whether agents can dynamically uncover and respond to emergent user intent across diverse, customizable scenarios.

3 UserBench

We instantiate UserBench as a Gymnasium-style environment in the travel domain. We use travel not as a claim of exhaustive itinerary planning, but

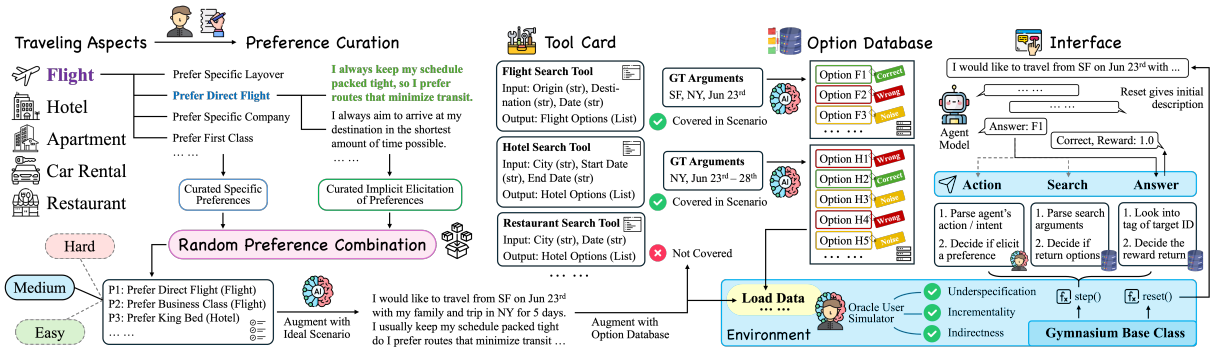

Figure 1: The pipeline of UserBench Construction, including data gathering, preference curation, tool augmentation, environment setup and interface design.

as a controlled setting where user preferences naturally affect multiple decision aspects, tool queries, and cost-aware option selection. Compared to prior work, UserBench features a curated collection of user preferences, each paired with carefully curated implicit expressions, and supports tool-augmented search with injected noise and cost-awareness. This setting challenges agents to perform search and reasoning in a user-centric manner, under constraints of partial, evolving, and indirect preference information. We describe the design details as follows.

3.1 Data Gathering

As shown in Figure 1, we begin by identifying five core aspects of travel planning: flight, hotel, apartment, car rental, and restaurant. Our data gathering is guided by three key principles: *realism*, *diversity*, and *implicitness*. First, we ensure realism by curating preferences that reflect plausible, real-world user needs, such as preferring direct flights (flight) or king-sized beds (hotel). Second, we aim for diversity by collecting around a hundred distinct preferences that span travel aspects. Third, we emphasize implicitness by pairing each preference with multiple naturally phrased statements that express the intent indirectly. For instance, the preference for direct flights might be expressed as: "I always keep my schedule packed tight, so I prefer travel routes that minimize transit time." These implicit expressions are carefully curated and validated to ensure that (i) they do not reveal the preference explicitly, and (ii) they still clearly imply the intended preference without ambiguity.

These curated preferences and their associated implicit expressions serve as the foundation for generating travel scenarios. We randomly sample and combine preferences across the five aspects, categorizing them into three difficulty tiers based on the number of preferences involved. For each combination, we generate an ideal travel arrangement statement that incorporates all relevant preferences using GPT-4o. This finally forms one data point.

3.2 Tool Augmentation

For each travel planning aspect, we develop a corresponding tool that simulates a database search. For example, the flight search tool accepts origin, destination, and date, returning a list of mock options. Instead of querying real-time data, we rely on pre-generated options to ensure stable and controlled tool outputs. This design choice is justified because (i) it guarantees consistency and quality of returned results, and (ii) our focus is user-centric reasoning rather than challenges of online search.

Each data point is associated with a tailored database of options for all relevant travel planning aspects. Specifically, for each aspect, we generate three types of options: correct (satisfying all preferences), wrong (violating at least one preference), and noise (incomplete or irrelevant to the query, such as a flight option with a destination different from the one the user searched for). These options are mixed to form the complete search space, with each data point containing up to 100 options. Option generation is handled by GPT-4o under human supervision to ensure quality and diversity.

3.3 Environment Building

Each data point in the environment contains a travel scenario, a set of implicit user preferences, and a database of relevant options. The environment simulates an oracle user with access to these internal states, while engaging in multi-turn interactions with a agent model under test. Upon reset, the environment provides only basic trip information and high-level requirements (e.g., hotel or flight), without disclosing any of its specific preferences. Preferences are then elicited over time, either (i) when the tested model explicitly asks about a relevant aspect, or (ii) after a fixed number of turns without progress (see Section B for details), at which point a preference is randomly and proactively revealed.

Difficulty Tier	# Test	# Train	Composition	Global Dataset Metadata		Travel Aspect	# Scenarios	Preferences
Easy	118	666	Travel-22, 2222	Travel Aspects	5	Hotel	187 (39.7%)	15
Medium	201	1130	Travel-33, 233, 333	Pref. Categories	24	Restaurant	298 (63.3%)	19
Hard	152	855	Travel-44, 334, 444	Total Prefs.	82	Flight	275 (58.4%)	16
				Elicitation Ways	902	Rental Car	243 (51.6%)	14
Total	**471**	2651	**UserBench Test** / Train	Options (B/C/W/N)	1/3/10/5	Apartment	172 (36.5%)	18

Table 2: **UserBench Statistics**: Difficulty-tiered statistics (left), global configuration metadata (middle), and travel aspect coverage (right). Pref. denotes preference; B/C/W/N stands for Best, Correct, Wrong, and Noise options respectively; Scenarios indicates the number of data scenarios in which each travel aspect appears.

All elicited preferences are presented implicitly, in line with the associated data.

This design captures the three key traits and challenges we previously identify: *underspecification*, where only coarse needs are initially provided; *incrementality*, where user preferences emerge progressively as the conversation evolves; and *indirectness*, where preferences are stated implicitly through natural, carefully-curated, heuristic-based language. These properties reflect real-world user behaviors and make UserBench a robust testbed for user-centric reasoning.

3.4 Agent Interaction

We provide a standardized interaction interface for agents, supporting three types of actions: `action`, `search`, and `answer`. Through `search`, agents issue queries specifying the travel planning aspect and arguments (e.g., hotel with specific dates), which are matched against ground truth. If the arguments match, the environment returns the mixed set of relevant options; otherwise, it returns nothing. Through `action`, the agent communicates with the user, typically by asking clarifying questions. The environment interprets the intent and responds naturally, potentially revealing a preference in an implicit form. Finally, through `answer`, the agent selects its recommended option ID(s), which are evaluated against the ground-truth correct options. Since ideal itineraries often span multiple aspects, correct answers may include several options.

3.5 Variants, Scalability and Statistics

UserBench supports several extensions. First, it can simulate a noisy search environment where query results are corrupted at a configurable frequency, mimicking real-world web or tool errors. Second, it allows for budget-aware decision-making by rewarding models that not only find correct options but also select the most cost-effective ones (best option). It can also tune reward strength to encourage agents towards practical utility.

As illustrated in Figure 1, the Random Preference Combination strategy enhances both the diversity and scalability of our data construction, enabling over 10K distinct scenarios. To manage computational costs during benchmarking, we use 471 data points for testing and reserve 2651 for training, which are not used in benchmarking but may support future model training. Detailed statistics of UserBench are presented in Table 2. For instance, "Travel-223" denotes a scenario with three travel aspects: two containing two implicit preferences each, and one with three. Additional data and construction details are provided in the Section B.

4 Experiments

4.1 Settings

Gym Settings. We adopt both variants (corrupted search, budget constraint) to increase test difficulty. GPT-4o is used as the default user simulator for all main experiments. We further validate the simulator design and test an open-weight simulator replacement in Section E.3 and Section E.6. We limit the maximum number of conversation turns to 20 and evaluate under: (1) a single-choice setting (our main evaluation), where the model is allowed to output only one option for each travel aspect, and (2) a multi-choice setting, where the model may output multiple options, and we evaluate it based on the option that achieves the highest reward.

Models. Our evaluation includes both closed- and open-source models. Closed-source ones include the GPT, Claude, Deepseek, and Gemini families, while open-source models include the Qwen3 and Llama3 families, with sizes ranging from 8B to 70B. All main experiments use temperature 0 for deterministic comparison; we include a sampling-temperature robustness check in Section E.5.

Metrics. The main evaluation metric is a normalized **score** based on the quality of selected options for each aspect of a travel scenario. For each aspect, if the model selects the best option (as judged by reward), it receives a score of 1.0. If the option is correct but not the best, it receives 0.8. All other options receive 0.0. We compute the highest reward score among the options selected for each aspect (in the multi-choice setting) and average this over all aspects in the scenario. For example, if a scenario has two aspects, and the model selects options with rewards 1.0 and 0.8 for the first aspect,

Model Name	Best Exist Rate	Correct Exist Rate	Score	Search Attempt Valid (%)	Action Attempt Valid (%)	Preference Elicited (%) (Active / Passive)
GPT-4o	0.204	**0.361**	**0.329**	82.48	27.82	27.32 (24.06 / 3.26)
Gemini-2.5-Pro	0.245	0.328	0.317	77.40	29.26	29.71 (23.85 / 5.85)
Claude-4-Sonnet	**0.260**	0.318	0.307	74.28	24.26	**34.25** (**26.31** / 7.94)
Deepseek-V3	0.148	0.218	0.210	61.22	23.62	31.82 (18.90 / 12.92)
Qwen3-14B	0.157	0.197	0.209	70.56	**33.71**	30.07 (15.95 / 14.12)
Qwen3-32B	0.154	0.211	0.206	79.42	26.89	22.01 (13.04 / 8.96)
Llama-3.3-70B	0.120	0.209	0.198	60.51	11.29	23.25 (13.05 / 10.20)
Qwen3-8B	0.140	0.186	0.180	68.94	23.20	28.86 (11.15 / 17.70)
GPT-4o-mini	0.102	0.173	0.166	76.10	13.55	25.43 (12.91 / 12.52)
Llama-3.1-8B	0.073	0.171	0.159	57.99	12.13	19.87 (8.09 / 11.78)
Gemini-2.5-Flash	0.047	0.080	0.125	**83.62**	28.14	14.00 (11.15 / 2.86)
Human Test	0.830	0.902	0.888	99.79	75.16	60.72 (51.32 / 9.40)

Table 3: UserBench main evaluation results across different models (single-choice setting).

Model Name	Best Exist Rate	Correct Exist Rate	Score	Search Attempt Valid (%)	Action Attempt Valid (%)	Preference Elicited (%) (Active / Passive)
GPT-4o	**0.652**	**0.725**	**0.710**	85.96	37.89	15.10 (13.11 / 1.99)
Gemini-2.5-Pro	0.604	0.685	0.673	80.43	38.01	37.25 (28.80 / 8.46)
Claude-4-Sonnet	0.586	0.619	0.612	**87.94**	**41.17**	**42.21** (**30.67** / 11.54)
Qwen3-32B	0.343	0.416	0.411	82.76	34.24	17.72 (10.81 / 6.90)
Deepseek-V3	0.351	0.398	0.391	61.91	21.06	16.96 (10.18 / 6.78)
Llama-3.3-70B	0.294	0.366	0.372	63.46	21.72	31.72 (16.68 / 15.04)
Gemini-2.5-Flash	0.279	0.316	0.311	84.27	33.58	15.32 (11.67 / 3.64)
GPT-4o-mini	0.227	0.294	0.283	81.54	24.90	20.18 (10.46 / 9.72)
Qwen3-14B	0.204	0.249	0.258	71.48	35.06	26.21 (13.07 / 13.14)
Llama-3.1-8B	0.117	0.252	0.234	61.38	12.20	26.01 (7.80 / 18.20)
Qwen3-8B	0.147	0.186	0.182	69.83	23.26	20.31 (7.95 / 12.37)

Table 4: UserBench main evaluation results across different models (multi-choice setting).

and 0.0, 0.8, and 0.0 for the second, the score is:

$$\text{Final Score} = \frac{\max(1.0, 0.8) + \max(0.0, 0.8, 0.0)}{2} = 0.9$$

In the single-choice setting, only the first option per aspect is considered, so the max function does not apply and the score is averaged only for the first choice of each aspect. In addition, we also report several auxiliary metrics to better understand model behavior (all micro-averaged). Please refer to Section D for a more detailed explanation.

- **Best Exist Rate**: Proportion of aspects where model includes the best option among its choices.
- **Correct Exist Rate**: Proportion of aspects where model includes any correct option.
- **Valid Search Attempt (%)**: The rate at which the model's search queries are syntactically valid.
- **Valid Action Attempt (%)**: The rate at which the model's actions successfully probe the user's real preferences in their profile.
- **Preference Elicited (%)**: The percentage of ground-truth preferences revealed during the conversation, including *Active* elicitation from valid model actions and *Passive* elicitation from UserBench when the model drifts too far off-topic.

4.2 Results

Single-choice results expose a large interaction gap. Scores drop by about 40% on average when moving from multi-choice to single-choice, showing that models struggle to select the best, or even one of the three correct options with only one attempt. Allowing multiple answers generally improves performance. However, humans evaluated under the same terminal interface and single-choice setting achieve a score of 0.888, a correct-exist rate of 0.902, and actively elicit 51.32% of hidden preferences, substantially outperforming all LLMs. This shows UserBench is solvable while revealing a large gap between current agents and humans.

Single-choice results expose a large interaction gap. Scores drop by about 40% from multi-choice to single-choice, indicating that models struggle to identify a correct option with only one attempt. Yet humans, evaluated under the same terminal interface and single-choice setting, achieve a score of 0.888, a correct-exist rate of 0.902, and actively elicit 51.32% of hidden preferences, far outperforming all LLMs. This shows that UserBench is solvable while exposing a large gap between current agents and humans.

Preference elicitation remains low across models. One might expect that performance gains in the multi-choice setting stem from better user understanding. However, we find that preference elicitation rates do not significantly improve, and in some cases, such as GPT-4o and Deepseek-V3, even decline. This suggests that higher scores more often result from random guessing or brute-force coverage rather than active reasoning. Moreover, the overall user preference elicitation rate remains low

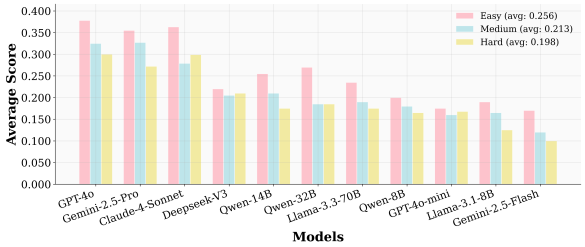

Figure 2: The score distribution of tested models across three difficulty tiers.

across models, especially for preferences revealed through active queries. This indicates that current models still struggle to proactively and effectively uncover user needs in interactive settings.

Understanding users is harder than executing tool use. Top-performing models maintain over 80% success in valid search attempts, but the rate of valid action attempts is much lower. This aligns with UserBench's goal of shifting difficulty from tool use to user understanding. Note that valid action attempt rate is computed only over action steps (excluding search and answer), so a higher rate does not necessarily reflect a higher absolute number of valid queries overall. Still, it indicates that the model's questions are more precise and preference-relevant, rather than vague or off-topic, which we classify as invalid attempts.

Other Interesting Findings. Some models, such as Gemini-2.5-Flash, achieve high Action Attempt Valid rates but relatively low Active Preference Elicited. This suggests they can ask effective clarifying questions but often fail to do so repeatedly or comprehensively, limiting their ability to capture the full scope of user preferences. Please see the Section D for detailed metric explanations to help understanding this result. In contrast, Claude-4-Sonnet performs well on both metrics, indicating strong capability in intent clarification. However, its overall score is not the highest. This highlights a gap between understanding user preferences and effectively integrating them into decision-making. In other words, even models excel at eliciting preferences, they may still struggle to leverage those in reasoning to deliver optimal recommendations.

We also evaluate stronger recent models under the same single-choice setting; results show improved performance but still a large gap to the human reference. Please see Section E.4 for further details.

5 Analysis

All analyses are conducted under the single-choice setting, which serves as our main evaluation setup, and focus on the *score* as the primary metric. The

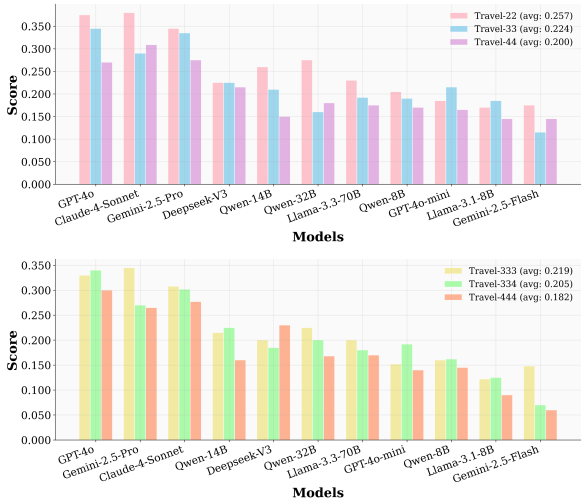

Figure 3: When aspect number is settled, more user preferences generally lead to less scores.

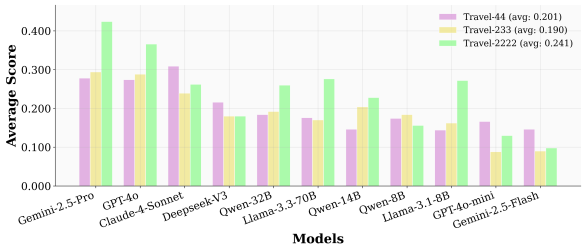

Figure 4: When total user preferences are settled, less preferences per aspect generally lead to higher scores.

turn-based, pass-k sampling, and choice ablation analyses are performed specifically on data points from the Travel-22, 33, and 44 scenarios.

Difficulty tier division effectively reflects UserBench's challenges. We begin by analyzing how scenario difficulty impacts model performance, as shown in Figure 2. Following the tiering in Table 2, we divide test scenarios into Easy, Medium, and Hard based on preference complexity. Results show a general downward trend in scores as difficulty increases. This confirms that our difficulty stratification captures real reasoning challenges for models. Further, the performance drop within each model across tiers reveals a lack of robustness in managing complex interactions, which is something humans typically excel at.

The key challenge lies in the number of preferences per aspect. To identify the core source of difficulty in UserBench, we investigate whether performance is more affected by the number of travel aspects or the number of preferences per aspect. We first fix the number of aspects and vary the number of preferences per aspect. As shown in Figure 3, model scores consistently decline as the number of preferences increases. This suggests that handling richer user preference signals is a major challenge for current models.

Model Name	First Index$^{\downarrow}$	Weighted Score$^{\uparrow}$	Correct Exists Rate$^{\uparrow}$
GPT-4o	3.846	**0.054**	**0.361**
Gemini-2.5-Pro	4.476	0.044	0.328
Qwen3-8B	2.005	0.039	0.186
Qwen3-32B	2.739	0.039	0.211
Claude-4-Sonnet	5.018	0.038	0.318
Deepseek-V3	2.740	0.037	0.218
Qwen3-14B	2.650	0.035	0.197
Llama-3.3-70B	2.776	0.032	0.209
Llama-3.1-8B	1.884	0.028	0.171
GPT-4o-mini	2.185	0.028	0.173
Gemini-2.5-Flash	**0.927**	0.014	0.080

Table 5: **Weighted Timing Analysis**: Models evaluated by timing of highest scores and coverage. Weight function $w(i) = 1/(i+1)$ penalizes late discoveries of correct or best user-aligned options. Coverage is the micro-average of aspects with valid answers.

Next, we fix the total number of preferences in a scenario and vary how they are distributed across aspects. As shown in Figure 4, performance improves when preferences are more evenly spread across multiple aspects, rather than concentrated within a few. This implies that models reason more effectively when each aspect involves fewer, simpler preferences. Concentrating multiple preferences into a single aspect appears to overload the model's reasoning process. Together, these results suggest the number of preferences per aspect is the main driver of difficulty in UserBench.

Models struggle to provide answers that are both correct and timely. To assess not only whether models find correct or best answers, but also *when* they do so, we conduct a weighted timing analysis in Table 5. Using a weight function $w(i) = 1/(i+1)$, we penalize delayed discoveries, where i is the turn at which a valid answer (reward > 0) first appears. Notably, open-source Qwen models outperform Deepseek in timing, despite Deepseek ranking higher in overall accuracy in Table 3. This suggests that while Deepseek eventually finds good answers, it is less efficient doing so. In contrast, GPT-4o and Gemini-2.5-Pro balance both coverage and timing better, achieving higher Correct Exist Rates and Weighted Score overall.

We also report the average turn index at which a valid answer first appears. Smaller models, like Llama-3.1-8B and Gemini-2.5-Flash, show lower average indices, indicating the successful attempts of their earlier guesses. However, these guesses often fail, leading to lower Weighted Scores and limited Correct Exist Rates. This reveals a common failure mode: smaller models rely on shallow heuristics or early guesses rather than deeply engaging with user intent. While this may occasionally yield fast correct answers, it more often results in low-quality answers not aligned with user needs.

These findings also help explain the low Preference Elicited in Table 3 and Table 4. Ultimately, a delicate trade-off emerges: models are expected to be efficient in interaction, yet still they should invest enough turns in interaction to truly understand the user's evolving needs or preferences.

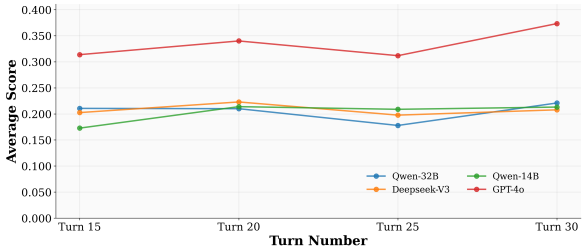

Figure 5: Increasing the number of turns allowed in interaction does not necessarily lead to better performance across models.

More interaction turns do not guarantee better performance. We examine whether increasing the interaction budget improves performance, as shown in Figure 5. Simply allowing more turns does not lead to consistent gains, and in some cases performance even degrades. To understand why, we categorize action turns into four types: targeted queries about existing hidden preferences, targeted queries about non-existing preferences, overly broad queries, and irrelevant turns (See details in Section E.2). We discover that overly broad queries account for a large fraction of action turns across representative models, e.g., 43% for GPT-4o and 45% for Qwen3-32B. This suggests that many extra turns reflect unfocused clarification rather than effective preference discovery, so longer interaction windows alone are insufficient without better dialogue planning and goal tracking.

Effect of sampling frequency reveals instability. We analyze the effect of sampling frequency using *pass-k* evaluation, as shown in Figure 6. While the maximum score across k samples steadily increases, the average score remains flat or even declines. This suggests that more samples increase the chance of stumbling upon a good response, but most outputs remain poor. The results highlight a key limitation: current models rely heavily on sampling luck rather than robust reasoning to align with user preferences. Given the high cost of pass-k sampling, this instability poses a challenge for deploying reliable user-interactive systems.

Less options reduces interference but do not reduce the core challenge. We finally perform an ablation study to assess how the number of wrong and noise options affects model performance. As shown in Table 6, reducing these distractors generally improves scores. However, even for strong

Model Name	Original (w10n5)	w10n0	w5n5	w8n2
GPT-4o	0.340	$0.344_{\uparrow 0.004}$	$0.360_{\uparrow 0.020}$	$0.350_{\uparrow 0.010}$
Deepseek-V3	0.223	$0.271_{\uparrow 0.048}$	$0.249_{\uparrow 0.027}$	$0.241_{\uparrow 0.018}$
Qwen3-32B	0.210	$0.220_{\uparrow 0.010}$	$0.241_{\uparrow 0.031}$	$0.224_{\uparrow 0.014}$
Qwen3-14B	0.214	$0.225_{\uparrow 0.011}$	$0.199_{\downarrow 0.015}$	$0.203_{\downarrow 0.011}$

Table 6: Impact of choice distribution on model scores. "w10n5" indicates a setting where, in addition to the correct and best options, there are 10 wrong and 5 noise options as distractors. Each cell reports the model's score and relative change to original setup.

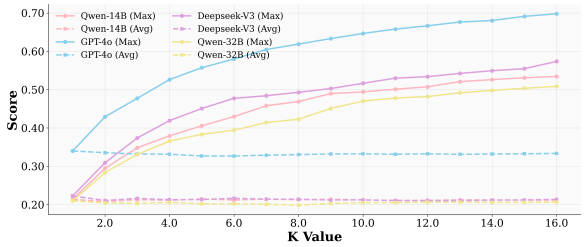

Figure 6: Increasing the number of sampling time raises the maximum score but average score shows little change or even drops.

models like Deepseek-V3 and GPT-4o, removing five wrong or noise options (about 30% of the total) yields only modest gains. Interestingly, some smaller models, such as Qwen-14B, even show slight performance drops. These results suggest that (1) reducing options does not fundamentally lower task difficulty: fully understanding user preferences remains essential, as only one best option satisfies all constraints (budget constraint included); and (2) many models still rely on shallow guessing rather than genuine preference understanding, which limits their ability to benefit from reduced distractors. These results also indirectly highlights the intentional complexity designed into our option sets for each data scenario.

6 Discussions

Justification for LLM as User Simulator. UserBench evaluates whether agents can elicit and satisfy *underspecified* preferences, requiring a simulator that is both controllable for fair evaluation and expressive enough for natural dialogue. We use a routing-plus-template design: each agent utterance is classified into an intent type and mapped to an instruction template that constrains what the simulated user may reveal while allowing natural paraphrasing. Since collecting real-time human responses at scale is infeasible, this constrained LLM simulator offers a practical approximation based mainly on reliable instruction following. Human validation shows that routing is well-defined, GPT-4o largely matches human-majority decisions, and its template-conditioned responses preserve the intended quality and disclosure behavior (see Section E.6). These results support using GPT-4o as a controlled environment component for evaluating preference elicitation.

Broad applicability of UserBench. UserBench is implemented as a standard Gym environment with a familiar *reset*/*step* interface, enabling easy integration with Gym-compatible agents. By abstracting user behavior as a black-box environment, it mirrors real-world interaction, where agents must infer hidden preferences through dialogue. This makes UserBench useful both as a **benchmark** and a **training environment**. As a benchmark, it supports flexible option sets, reward functions, and user feedback mechanisms for fine-grained behavioral analysis. As a training environment, it provides a standardized tool-use interface with turn-wise rewards and partial-credit signals, supporting supervised fine-tuning and multi-turn reinforcement learning for agentic LLMs (Jin et al., 2025; Li et al., 2025; Qian et al., 2025a).

Balancing efficiency and effectiveness in user interaction. Our weighted timing analysis shows that many models, including GPT-4o, answer early rather than sufficiently probe user preferences, appearing efficient while missing deeper intent. We rarely observe excessive hesitation, but both extremes highlight the need to balance efficiency with effectiveness, echoing broader concerns around efficient tool use (Qian et al., 2025b; Wang et al., 2025). While UserBench does not explicitly define this trade-off, it exposes behaviors that targeted training can address. For example, reward functions can penalize late preference discovery, grant partial credit for eliciting hidden preferences, or penalize correct answers produced without sufficient interaction. Such customizable rewards make UserBench a flexible platform for training agents that are both capable and user-aligned..

7 Conclusion

In this paper, we introduce **UserBench**, a user-centric environment for evaluating agents under realistic communication involving underspecification, incrementality, and indirectness. In a controlled travel-planning setting, UserBench reveals that current LLM agents, while capable at tool use, still struggle to uncover implicit user preferences and respond to subtle signals. This exposes a gap between technical execution and communicative intelligence. We hope UserBench supports future work on agents that not only complete tasks, but also collaborate, adapt, and align with users through meaningful interaction.

Limitation

UserBench is designed as a controlled and diagnostic environment for studying proactive preference elicitation, rather than a complete simulation of all real-world user-agent collaboration. Its current instantiation focuses on travel planning, uses a structured user simulator for reproducible evaluation, and assumes that each task has valid options so that scoring remains objective. These choices allow us to isolate the core challenge of discovering and using hidden user preferences, but they also leave several valuable extensions for future work, including broader domains, richer real-user interactions, infeasible-request negotiation, preference trade-offs, and stronger cross-aspect dependencies. We view the current benchmark as a focused foundation for these directions: it provides a scalable, interpretable, and human-referenced testbed for measuring a capability that remains difficult for current LLM agents.

Impact Statement

This paper presents UserBench, an interactive benchmark designed to evaluate and improve how LLMs collaborate with users whose goals are uncertain or evolve during conversation. Our aim is to advance machine learning by fostering more adaptive, user-centered systems that can better understand indirect or changing intentions. Potential positive impacts include enhancing AI reliability, transparency, and usability in real-world settings where users' preferences are not explicitly stated. However, there are also risks: models trained on synthetic user simulations may inherit biases or overfit to simplified scenarios, and improved inference of implicit intent could be misused for manipulative or privacy-intrusive purposes. To mitigate these risks, we emphasize transparency, open benchmarking, and responsible use guidelines.

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. tau2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2024. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*.

Maximillian Chen, Ruoxi Sun, Tomas Pfister, and Sercan O Arik. 2025. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. In *Proceedings of ICLR*.

H.H. Clark. 1996. *Using Language*. ACLS Humanities E-Book. Cambridge University Press.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

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

Ge Gao, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. 2024. Aligning llm agents by learning latent preference from user edits. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, and 1 others. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and 1 others. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings*

of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 3102–3116.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*.

Bill Yuchen Lin, Yuntian Deng, Khyathi R Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*.

Alisa Liu, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah A Smith, and Yejin Choi. 2023. We're afraid language models aren't modeling ambiguity. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, and 1 others. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482.

Jiarui Lu and et al. 2024. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. *arXiv preprint arXiv:2408.04682*.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, and 1 others. 2025. Proactive agent: Shifting llm agents from reactive responses to active assistance. In *The Thirteenth International Conference on Learning Representations*.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, and 1 others. 2025. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025a. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025b. Smart: Self-aware agent for tool overuse mitigation. *arXiv preprint arXiv:2502.11435*.

Cheng Qian, Hongyi Du, Hongru Wang, Xiusi Chen, Yuji Zhang, Avirup Sil, Chengxiang Zhai, Kathleen McKeown, and Heng Ji. 2025c. Modelingagent: Bridging llms and mathematical modeling for real-world challenges. *arXiv preprint arXiv:2505.15068*.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6922–6939.

Cheng Qian, Peixuan Han, Qinyu Luo, Bingxiang He, Xiusi Chen, Yuji Zhang, Hongyi Du, Jiarui Yao, Xiaocheng Yang, Denghui Zhang, and 1 others. 2024a. Escapebench: Pushing language models to think outside the box. *arXiv preprint arXiv:2412.13549*.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and 1 others. 2024b. Tell me more! towards implicit user intention understanding of language model driven agents. *arXiv preprint arXiv:2402.09205*.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. 2024c. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. *arXiv preprint arXiv:2401.13996*.

Cheng Qian, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2024d. Toolink: Linking toolkit creation and using through chain-of-solving on open-source model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 831–854.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Harmanpreet Singh, Nikhil Verma, Yixiao Wang, Manasa Bharadwaj, Homa Fashandi, Kevin Ferreira, and Chul Lee. 2024. Personal large language model agents: A case study on tailored travel planning. In *Proceedings of EMNLP (Industry Track)*.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*.

Jiayin Wang, Fengran Mo, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. 2024a. A user-centric multi-intent benchmark for evaluating large language models. In *Proceedings of EMNLP*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2024b. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. In *Proceedings of ICLR*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong

Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

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

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. 2024. Craft: Customizing llms by creating and retrieving from specialized toolsets. In *Proc. The Twelfth International Conference on Learning Representations (ICLR2024)*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, and 1 others. 2024a. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*.

Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh Murthy, Tian Lan, Lei Li, Renze Lou, Jiacheng Xu, and 1 others. 2024b. Diversity empowers intelligence: Integrating expertise of software engineering agents. *arXiv preprint arXiv:2408.07060*.

Michael JQ Zhang, WB Knox, and Eunsol Choi. 2024c. Modeling future conversation turns to teach llms to ask clarifying questions. *arXiv preprint arXiv:2410.13788*.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. 2025. Do llms recognize your preferences? evaluating personalized preference following in llms. In *Proceedings of ICLR*.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and 1 others. 2025. Multiagentbench: Evaluating the collaboration and competition of llm agents. *arXiv preprint arXiv:2503.01935*.

Appendix

A Comparison Traits Details

We identify eight core traits that characterize recent benchmarks detailed as follows.

- **Multi-turn Interaction**: The benchmark requires the agent to conduct extended conversations over multiple turns, often involving behaviors including clarification, refinement, or feedback-based improvement.
- **Goal Ambiguity**: The tasks are initially underspecified, vague, or indirect, requiring the agent to ask clarifying questions, infer missing information, or resolve ambiguity over time.
- **Tool Use**: The agent must interact with external tools such as APIs, structured search engines, databases, or execution environments to complete tasks beyond its internal reasoning.
- **Dynamic State**: The benchmark maintains an evolving internal state (e.g., environment database, user preference profile) that is updated based on the agent or user's actions across time.
- **Multi-Aspect Reasoning**: Tasks consist of multiple interdependent subgoals (e.g., booking flights, hotels, and cars) that require coordinated reasoning and decision-making across aspects.
- **User Simulation**: The environment includes a simulated user that responds to agent queries or actions, enabling interactive learning and evaluation without requiring human annotators.
- **Domain Diversity**: The benchmark spans multiple domains, various tools, and different task structures, ensuring generalization beyond a fixed task type or narrow scenario.
- **Customizable**: The benchmark framework allows researchers to adjust task settings, rewards, user behavior, or interaction protocols without redesigning from scratch.
- **Scalable**: The benchmark supports large-scale, automated generation of tasks or environments, enabling reproducible experiments and stress-testing of agent capabilities.

B Gym Construction Details

Wrong Option Example: Flight Option F2

Path: New York → Los Angeles → San Francisco
Time (hours): 5 (Leg 1), 2 (Layover), 1 (Leg 2)
Airlines: United Airlines, United Airlines **Flight Numbers:** UA789, UA321 **Total Cost:** $400
Amenities: WiFi, Meal Service, Lounge Access
Service Costs:
Checked Bag Total: $50
Business Class Upgrade: $250

Option Type: `not-suitable`
Reason: This option includes a layover, which does not satisfy the preference for a direct flight.

Correct Option Example: Flight Option F14

Path: New York → San Francisco **Time (hours):** 6 (Direct)
Airlines: Delta Airlines **Flight Numbers:** DL4567
Total Cost: $350
Amenities: WiFi, Meal Service, Carry on Baggage Allowance
Service Costs:
Checked Bag Total: $30
Business Class Upgrade: $150

The following content is invisible when presented to tested model (serve as label, only for evaluation use)

Option Type: `suitable`
Reason: This option is suitable because it is a direct flight from New York to San Francisco, adhering to the user's preference to avoid layovers. It also offers a business class upgrade, meeting the user's need for extra comfort and space during the flight. The available amenities such as WiFi and meal service add to the travel experience.

Noise Option Example: Flight Option F10

Path: New York → San Francisco **Time (hours):** 1000 (Direct)
Airlines: Fantasy Air **Flight Numbers:** FA999
Total Cost: $1,000,000
Amenities: WiFi, Meal Service, Carry on Baggage Allowance, Lounge Access
Service Costs:
Checked Bag Total: $0
Business Class Upgrade: $0

Option Type: `noise`
Reason: This flight path takes an unrealistic 1000 hours, which is unrelated to the user's request for a flight from New York to San Francisco.

Option Generation. We generate three types of options: correct, wrong, and noise. Above, we present one example of each type, all drawn from the same user scenario.

For **correct** options, under our budget-constrained setting, there is only one best option: the one that incurs the lowest total cost after accounting for all relevant charges. For example, if a user prefers a business class seat, the total cost includes both the base fare and any additional business class upgrade fees.

Wrong options explicitly violate one or more of the user's preferences, for instance, offering a

layover when the user prefers direct flights.

Noise options differ from wrong options in that they are either (i) unrelated to the user's search intent (e.g., the flight goes from San Francisco to Los Angeles when the user searched for flights from San Francisco to New York), or (ii) unrealistic (e.g., an option with an implausible flight time or cost, such as taking 1000 hours).

Environment Setting. The environment simulates a user who responds naturally to the tested model's conversation and queries, gradually and implicitly revealing preferences over time. When the tested model issues a `search`, the environment evaluates whether the search query aligns with any of the ground truth arguments using the prompt in Figure 7.

If the model issues an `action`, the environment first analyzes the model's latest utterance to determine its intent. This intent is categorized into four types using the prompt in Figure 8: (1) the utterance explicitly and concretely asks for a preference that the simulated user possesses; (2) the utterance explicitly and concretely asks for a preference, but the user does not possess a specific preference for that aspect; (3) the utterance makes a vague and general query about preferences rather than focusing on a specific aspect; (4) other, indicating a normal conversation not related to preferences.

For Type "1", the environment uses the prompt in Figure 9 to implicitly and naturally reveal the corresponding preference. For Type "2", the response is coded in the environment as: "This is a good question. However, I do not have specific preference in the aspect you ask about yet (or maybe I have already elicited that to you before). You may continue to ask me about other detailed and specific preferences." For Type "3", the response is also coded in the environment as: "Your question is too vague and general, and I am not sure how to respond to it. Please ask me about some specific aspects of my preferences, in a more detailed and concrete way, so that I can provide you with a more accurate response." For Type "4", the environment uses the prompt in Figure 11 to continue the conversation naturally and neutrally, without revealing any specific preferences.

If the model's utterance has not been judged as Type "1" for several consecutive turns (a hyperparameter that can be configured), the environment will proactively use the prompt in Figure 10 to introduce one of its remaining preferences at random. From the model's perspective, this is treated as a passively obtained preference, rather than one elicited through a successful action attempt. We will discuss this mechanism in more detail when describing the metrics.

Finally, if the tested model issues an `answer`, the environment parses the proposed answer in a rule-based manner, compares it with the ground truth, and returns the corresponding reward.

C Scope of Study

Our benchmark uses travel as context but is intrinsically domain-agnostic for the following reasons: (1) Core principles (underspecification, incrementality, indirectness) can actually apply across domains. (2) Our benchmark aims to test user-centric abilities such as intent clarification, preference elicitation, and implicit reasoning, instead of domain knowledge. Travel here is only a general and typical scenario often investigated by agent research; other domains (e.g., safety, creativity) may yield different interaction trajectory and analysis results but are all beyond our scope.

D Experiment Details

Settings. On the model side, we set the generation temperature to 0, the number of samples to 1, the maximum response length per turn to 2048 tokens, and the maximum number of interaction turns to 20. We use 8 Nvidia H200 GPUs (1 node) for the whole evaluation (mainly for hosting open-source models). All the results are produced by a single run as generation temperature is set to 0.

Tool Schema: `interact_with_env`

Type: `function`
Name: `interact_with_env`
Description: A tool for interacting with a target environment. The detailed environment description and action space is provided in the system prompt, so please follow the system prompt. You can use this tool to analyze and interact with the environment step by step through three actions including `search`, `action`, or `answer`.

Parameters: `object`

thought (`string`): Your thought of what to do next, including your reason or analysis of your choice and why.

choice (`string`, enum: `"action"`, `"answer"`, `"search"`): Your choice of what to do next, must be one of `action`, `answer`, or `search`.

content (`string`): The content of your choice,

must be a string. If you choose `action`, provide the action you want to take. If you choose `answer`, provide the answer you want to submit. If you choose `search`, provide the search query. The specific format of the content is determined by the environment description, which should be provided in the system prompt. Please follow the format strictly in order to invoke this tool.

The tool schema for interaction is presented above. Note that the model is expected and required to make a tool call in order to interact with the environment. If a tool call is not detected, the interaction is immediately terminated. To enforce this behavior, the tool call field (e.g., "tool choice") is always set to `required` or an equivalent constraint, ensuring that the model must invoke a tool on every turn.

Environment Configuration

Driving Model Settings:
Model Name: `GPT-4o`
Temperature: `0.0`
Max Tokens: `2048`
Timeout: `15.0` (seconds, timeout will fallback to default response)

Environment Configuration:
Max Steps: `20` (align with the maximum interaction turn)
Search Failure Interval: `5` (every Nth search results in a system error)
Elicitation Interval: `3` (proactive preference elicitation if off-topic for N consecutive turns)

Reward Configuration:
Reward Scale: `1.0` (final reward equals reward value times reward scale)
Step Penalty: `0.0` (no step penalty on reward in basic setting)
Search Correct Reward: `0.2` (partial reward given to a correct and aligned search attempt)
Preference Correct Reward: `0.2` (partial reward given to a successful action attempt)
Choice Best Reward: `1.0` (reward given to best answer being chosen)
Choice Correct Reward: `0.8` (partial reward given to correct but not best answer being chosen)
Wrong Choice Penalty: `0.0` (no penalty for choosing the wrong choice in basic setting)

Choice Number Customization:
Wrong Choice Number: `10` (the number of wrong choices presented in the search result in basic setting)
Noise Choice Number: `5` (the number of noise choices presented in the search result in basic setting)

From the environment side, we use GPT-4o as the backbone model to simulate the user. Additional configuration details are summarized in the table we present above.

Additionally, as mentioned earlier, we incorporate search failure simulation and budget constraint settings. For the single-choice and multiple-choice settings, we use different system prompts (differing only in one specific requirement) while keeping the user prompt identical. The corresponding prompts are shown in Figure 12 to Figure 14.

Metrics. We provide detailed definitions of the auxiliary metrics used to better understand model performance:

- **Best Exist Rate**: Each data scenario typically involves 2 to 4 travel aspects (e.g., flight, hotel, car). For each aspect, there is one *best* option and two other *correct but suboptimal* options—three correct options in total. For example, if a scenario involves both flight and hotel, and the model selects the best hotel option but only a correct (not best) flight option, the Best Exist Rate for this instance is $1/2$. This score is micro-averaged over all data points. It measures the model's ability to select the best options while being budget-aware, as instructed.

- **Correct Exist Rate**: This metric is similar to the one above, except that any correct option (not necessarily the best) is considered sufficient. Using the previous example, if the model selects correct options for both flight and hotel, the Correct Exist Rate is $2/2$. This metric captures how well the model adheres to user preferences, without considering budget constraints.

- **Valid Search Attempt (%)**: Each search action is evaluated by the environment to determine if it aligns with any ground-truth search arguments. A valid search returns results (i.e., receives a non-zero reward). For example, if a data point involves three aspects and the model makes four `search` actions, but only three yield valid results, the Valid Search Attempt rate is $3/4$. This metric is micro-averaged across all data points and reflects the model's ability to issue precise and effective search queries.

- **Valid Action Attempt (%)**: Each `action` the model takes is categorized by the environment into one of four types we mentioned earlier. A *valid* action is classified as type "1", indicating a targeted and meaningful clarifying question. For each data point, we compute the ratio of type "1" actions to the total number of actions, then

micro-average across all data. This metric evaluates whether the model consistently issues high-quality clarifying queries. However, note that this is a *relative* metric—high validity does not necessarily mean more preferences are elicited. For instance, a model that asks one valid question (1/1) achieves a 100% rate, even if the user holds many unelicited preferences.

- **Preference Elicited (%)**: This metric measures the *absolute* number of user preferences elicited, either actively or passively. Active elicitation occurs when the model issues a type "1" `action` (a valid query). Passive elicitation occurs when the model fails to elicit preferences for a number of steps (set by the hyperparameter Elicitation Interval, which is 3 in our experiments), prompting the environment to release a preference. The sum of actively and passively elicited preferences is divided by the total number of user preferences per data point and micro-averaged across the dataset. In particular, the Active Preference Elicited metric reflects the model's ability to proactively uncover user preferences, regardless of the number of queries issued.

E Analysis Details

E.1 Human Reference Study

We conduct a human reference study to contextualize the difficulty of UserBench and to verify that the benchmark is solvable under the same interaction interface used by LLM agents. This study is not intended to estimate average end-user performance; instead, it provides a strong human reference under controlled conditions.

We recruit 10 CS PhD students who did not participate in our simulator validation study. This avoids overlap between participants used to validate the user simulator and participants used to provide the human reference.

Participants interact with UserBench through the same terminal interface used in our main experiments. They are given the same available actions as LLM agents: they may ask the simulated user clarification questions, issue search queries to the corresponding travel tools, and submit final option IDs as answers. We use the same single-choice setting as Table 3, where only one final option per travel aspect is considered for evaluation.

Each participant completes approximately 50 tasks. We sample tasks to cover different difficulty levels, so that the human reference is not dominated by easy or hard cases. Participants are instructed to solve the task by interacting with the user simulator and tools, but they are not given access to the hidden preferences or ground-truth correct options.

We evaluate human trajectories using the same metrics as the model trajectories. *Best Exist Rate* measures whether the selected option includes the best option for each aspect. *Correct Exist Rate* measures whether the selected option satisfies the hidden user preferences, regardless of whether it is cost-optimal. *Score* gives full credit to the best option, partial credit to correct but non-best options, and zero credit to incorrect options. We also report valid search attempts, valid action attempts, and the percentage of hidden preferences elicited through active and passive disclosure.

Model	Type 1	Type 2	Type 3	Type 4
GPT-4o	28%	22%	43%	7%
Claude-4-Sonnet	35%	25%	38%	2%
Qwen3-32B	29%	22%	45%	4%

Table 7: Breakdown of action turns by clarification type under the single-choice setting.

E.2 Clarification Query Type Analysis

To better understand agent clarification behavior, we categorize each action turn into four types. Type 1 asks about an existing hidden preference and can elicit useful information. Type 2 asks a specific but non-existing preference. Type 3 is an overly broad or generic preference question. Type 4 is irrelevant or chit-chat. Answer and search turns are excluded from this analysis.

As shown in Table 7, Type 3 questions account for a substantial portion of action turns across models. This explains why simply allowing more interaction does not necessarily improve performance: many extra turns are spent on broad questions that do not reveal hidden preferences.

E.3 Open-Weight Simulator Replacement

To examine whether UserBench depends on a proprietary model as the user simulator, we replace GPT-4o with Qwen3-32B and repeat the simulator validation protocol. The simulator's main role is to route agent utterances into predefined interaction types and generate template-conditioned responses, rather than to solve the travel task itself.

As shown in Table 8, Qwen3-32B achieves 86% agreement with human routing labels. Its template-

Simulator Model	Agreement with Human Labels
GPT-4o	89%
Qwen3-32B	86%

Table 8: Routing-label agreement between simulator models and human annotations.

Model	Best Exist Rate (%)	Correct Exist Rate (%)	Score
GPT-5.2	0.257	0.391	0.368
Gemini-3.1-Pro	0.242	0.367	0.342
Human Reference	0.830	0.902	0.888

Table 9: Additional single-choice results on stronger recent models.

Model	Best Exist Rate (%)	Correct Exist Rate (%)	Score
Qwen3-32B, $T = 0$	0.154	0.211	0.206
Qwen3-32B, $T = 0.6$	0.161	0.223	0.211
Qwen3-14B, $T = 0$	0.157	0.197	0.209
Qwen3-14B, $T = 0.6$	0.155	0.202	0.217

Table 10: Effect of sampling temperature on Qwen models under the single-choice setting.

Validation Item	Agreement
Human–Human Routing Agreement	92%
GPT-4o–Human Routing Agreement	89%

Table 11: Simulator routing validation.

conditioned responses are also generally coherent and consistent with the expected simulator behavior, including whether a hidden preference should be revealed. These results suggest that open-weight models can serve as practical simulator replacements, although we use GPT-4o in the main experiments for stronger instruction following and quality control.

E.4 Additional Results on Stronger Recent Models

We additionally evaluate stronger recent models under the same single-choice setting as the main experiments. As shown in Table 9, GPT-5.2 and Gemini-3.1-Pro improve over earlier models, but still remain far below the human reference. This suggests that UserBench continues to expose a substantial preference-elicitation and decision-making gap even for newer model generations.

E.5 Effect of Sampling Temperature

All main experiments use temperature 0 to ensure deterministic and comparable evaluation. Since some Qwen models recommend non-zero sampling temperatures, we additionally evaluate Qwen3-32B and Qwen3-14B with temperature 0.6 under the same single-choice setting.

As shown in Table 10, non-zero temperature slightly improves the overall score for both Qwen models, but the gains are small and not fully consistent across metrics. For example, Qwen3-14B improves in score but slightly decreases in Best Exist Rate. This suggests that greedy decoding is not the main reason for the performance gap observed in UserBench.

E.6 Human Validation of the Simulator

We validate the user simulator to ensure that preference disclosure is controlled, consistent, and aligned with human judgments. The simulator follows a routing-and-response design: it first classifies each agent action into a predefined interaction type, and then generates a response conditioned on that type. This reduces unconstrained LLM–LLM interaction and prevents hidden preferences from being revealed unless the agent asks a sufficiently targeted question.

Intent routing validation. To evaluate whether the routing categories are well-defined, we randomly sample 50 agent utterances from our evaluation set and ask 10 CS graduate student volunteers to label their intent types. All participants consented to share their study data for research use only; identities were anonymized, and participation was unpaid. We define *human agreement* as at least 8 out of 10 volunteers assigning the same label. Under this criterion, humans agree on 92% of utterances (46/50), indicating that the intent categories are clear and consistently interpretable. For these 46 agreed-upon cases, GPT-4o's routing labels match the human-majority label 89% of the time, suggesting that the simulator can reliably determine whether an agent question should reveal a hidden preference.

Template-conditioned response validation. After routing, the simulator generates a user reply following the corresponding instruction template. To validate response quality, we focus on utterances routed to type "1" and sample 50 such instances. Volunteers are asked to produce responses using the same template instructions given to GPT-4o. Human responses are generally longer and more

diverse in phrasing, while GPT-4o responses tend to be shorter and more standardized. However, GPT-4o follows the template constraints more consistently, produces unambiguous responses, and preserves the intended indirectness needed for preference elicitation. Manual inspection confirms that the sampled GPT-4o responses reveal preferences only when appropriate, remain coherent with the conversation, and avoid contradicting the hidden user state.

Shortcut inspection and summary. We further manually inspect 50 sampled trajectories and find no explicit shortcut or collusive behavior between the simulator and evaluated agents. Overall, the study shows that humans largely agree on routing decisions, GPT-4o closely matches the human-majority labels, and its template-conditioned responses are controlled and qualitatively similar to human responses. These results support GPT-4o as a reliable user simulator for our gym, while richer real-user interaction remains an important future direction, as discussed in Section 6.

E.7 Case Study

We provide a case study of Qwen3-32B and Claude-4-Sonnet in Figure 15 and Figure 16 respectively (positive case). This first case demonstrates Qwen3-32B's ability to recover from vague initial action attempt, effectively elicit user preferences through targeted follow-up questions, conduct precise searches, and present clear comparisons between viable options. The model's reasoning aligns well with both user preferences and budget constraints, ultimately leading to the correct selection of the best hotel option.

The second case highlights Claude's ability to elicit nuanced user preferences through stepwise clarification, recover gracefully from vague questioning, and refine its inquiries based on prior feedback. The agent successfully identifies key user preferences, adjusts its strategy accordingly, and recommends a flight option that aligns with both stated and implied travel needs. Its interaction demonstrates strong adaptability and goal-oriented reasoning within a multi-aspect travel planning scenario.

Finally, we provide a case study of Deepseek-V3 in Figure 17 (negative case). Despite issuing structurally valid search queries, the model demonstrates shallow user understanding and fails to elicit preferences through conversation. It prematurely selects suboptimal options without sufficiently narrowing down user constraints, and attempts repeated selections on the same aspect, indicating poor state tracking. This case illustrates brittle reasoning and a lack of alignment with the user's evolving intent, highlighting a failure mode of large models under ambiguous user goals and minimal feedback.

F Use of LLMs

In this work, LLMs are used strictly for research support rather than as sources of substantive content. Their use falls into: (i) serving as the user simulator in the gym environment or model being tested, and (ii) assisting with language refinement during paper writing. For writing support, we used GPT-5 solely to polish text (improving coherence and grammar) while all ideas, logic, results, and technical contributions originate from the authors. To safeguard rigor, we have carefully reviewed all LLM-refined texts to confirm that no hallucinated content was introduced and that the original arguments, findings, and perspectives were faithfully preserved.

G Significance, Scope, and Clarifications

G.1 Why UserBench Matters

A Missing Dimension in Agent Evaluation. Recent agent benchmarks have made substantial progress in evaluating tool use, planning, reasoning, and environment interaction. These settings are important, but they often assume that the user's goal is already sufficiently specified before the agent acts. In realistic user-facing scenarios, however, the central challenge often appears earlier: the agent must first determine what the user actually wants. Users may begin with vague instructions, reveal preferences gradually, or express important constraints indirectly. UserBench focuses on this under-measured capability: **proactive preference elicitation in tool-grounded interaction**. The benchmark therefore evaluates not only whether an agent can use tools, but whether it can ask the right questions before using them.

From Task Execution to User-Centric Interaction. A key contribution of UserBench is that it shifts the evaluation focus from final task completion alone to the interaction process that makes task completion user-aligned. In UserBench, the agent receives only coarse initial requirements and

must decide when to clarify, what to ask, when to search, and how to use elicited preferences for final decisions. This setting captures a practical failure mode of current agents: they may produce syntactically valid tool calls and plausible recommendations while still missing the user's hidden intent. By explicitly measuring preference elicitation, action validity, search validity, and final option quality, UserBench provides a more diagnostic view of user-centric agent behavior.

Proactive Preference Elicitation as the Core Challenge. UserBench is designed to isolate a specific capability: asking targeted, decision-relevant questions under uncertainty. This is different from simply asking many questions or requesting all preferences upfront. The environment reveals useful information only when the agent asks sufficiently concrete questions about relevant hidden preferences. Our query-type analysis in Section E.2 shows that many model questions remain overly broad, which explains why increasing the number of interaction turns does not reliably improve performance. Thus, the benchmark targets a concrete and measurable skill: state-aware clarification that helps the agent uncover preferences that matter for the final decision.

A Controlled Testbed for Reproducible User Simulation. Evaluating user-agent interaction requires balancing realism with reproducibility. Fully human-in-the-loop evaluation is valuable, but it introduces variation in patience, timing, preference disclosure, and interpretation. UserBench instead uses a controlled routing-and-response simulator, where preference disclosure follows predefined rules while responses remain natural. This design enables fair comparison across models and prevents hidden preferences from being revealed accidentally or inconsistently. We validate this design in Section E.6, and further show in Section E.3 that an open-weight model can serve as a practical simulator replacement. These results support UserBench as a scalable and reproducible environment for studying user-centric interaction.

Travel as a Grounded but Controlled Instantiation. We instantiate UserBench in travel planning because it naturally involves multiple decision aspects, implicit preferences, tool search, distractor options, and cost-sensitive choices. At the same time, the travel setting is controlled enough to support objective evaluation. Our goal is not to replace full itinerary-planning benchmarks that emphasize feasibility checking, cross-day scheduling, or infeasible-constraint repair. Instead, UserBench studies the earlier interaction problem: before an agent can optimize a plan, it must first discover what the user values. In this sense, UserBench is complementary to existing travel-planning benchmarks and provides a focused testbed for preference discovery.

Human Reference and Continued Difficulty for Strong Models. The benchmark exposes a substantial gap between current agents and skilled human interaction behavior. In our human reference study, participants using the same interface achieve much higher final scores and actively elicit substantially more hidden preferences than LLM agents. This demonstrates that the task is solvable under the proposed interface, while current models still struggle with proactive clarification. Moreover, additional results on stronger recent models in Section E.4 show that newer systems improve but remain far below the human reference. These findings suggest that proactive preference elicitation remains a meaningful challenge for current and future agents.

Connecting Evaluation with Agent Improvement. Beyond evaluation, UserBench provides a practical environment for improving user-centric agents. Its Gymnasium-style interface offers standardized observations, actions, and rewards, making it suitable for training agents through supervised learning, reinforcement learning, or interaction-level policy optimization. The benchmark also decomposes failures into interpretable categories, such as invalid search, broad clarification, insufficient preference elicitation, and incorrect final selection. This makes UserBench useful not only as a leaderboard, but also as a research tool for developing agents that can plan their dialogue, ask more informative questions, and make decisions grounded in user intent.

G.2 Design Clarifications

UserBench Targets Controlled Preference Elicitation, Not All User Collaboration. User-agent collaboration is broad, and no single benchmark can cover all forms of real human interaction. UserBench deliberately focuses on a narrower but fundamental capability: **controlled proactive preference elicitation**. In this setting, the user has hidden preferences, the agent must uncover them

through interaction, and final success depends on using those preferences for tool-grounded decisions. This focused formulation is useful precisely because it makes a difficult user-centric behavior measurable and reproducible. Rather than claiming to simulate every aspect of human collaboration, UserBench isolates one essential component that many deployed agents need.

Why This Is More Than Standard Tool Use. Tool use is necessary in UserBench, but it is not the main source of difficulty. Agents must decide what information is missing before searching, how search results should guide follow-up questions, and which final option best matches the elicited preferences. The results show that strong models often achieve high valid-search rates but much lower action validity and preference elicitation. This separation is important: an agent can be competent at calling tools while still being poor at understanding the user. UserBench directly measures this gap.

Why This Is More Than Generic Clarification. UserBench does not reward clarification for its own sake. Broad questions such as asking the user to state all preferences are treated as vague and do not reveal all hidden information. Successful agents must ask targeted questions about relevant travel aspects and preference dimensions. This design distinguishes meaningful clarification from generic information gathering. The clarification-type analysis in Section E.2 confirms that many model interactions fail because questions are too broad rather than because the turn budget is too small.

The Simulator Is Controlled by Design. The user simulator is intentionally structured rather than fully open-ended. It first routes the agent's utterance into a predefined interaction type and then generates a response conditioned on that type. This prevents arbitrary preference disclosure and ensures that different agents are evaluated under comparable conditions. The simulator should therefore be understood as a controlled evaluation mechanism, not as a claim that all real users behave identically. This design choice supports scalable, deterministic, and interpretable benchmarking.

LLM-Based Simulation Does Not Create a Shortcut Task. Because UserBench uses an LLM-based simulator, one might ask whether agents can exploit simulator-specific behavior. We address this through controlled routing, human validation, and trajectory inspection. Preference information is not freely generated by the simulator; it is disclosed only when the routing decision determines that the agent has asked an appropriate targeted question. In Section E.6, human annotators show high agreement with the routing labels, and manual inspection of sampled trajectories finds no explicit shortcut or collusive behavior. Thus, the task remains an evaluation of targeted preference elicitation rather than LLM–LLM coordination.

Open-Weight Simulators Are Practical Replacements. The benchmark does not fundamentally depend on a proprietary simulator. The simulator's main role is instruction following, routing, and template-conditioned response generation, not solving the task. As shown in Section E.3, replacing GPT-4o with Qwen3-32B still yields high agreement with human routing labels. We use GPT-4o in the main experiments for stronger reliability and quality control, but the environment itself is model-agnostic and can support open-weight simulator variants.

Travel Is a Testbed for Preference Discovery. The travel domain is used because it provides a natural setting where user preferences affect multiple aspects of decision-making, such as flights, hotels, restaurants, car rentals, and apartments. This does not mean that UserBench attempts to cover every challenge in end-to-end travel planning. Full itinerary construction, calendar-level feasibility, cross-day scheduling, and infeasible-request repair are important but distinct problems. By controlling these factors, UserBench isolates the preference-discovery stage that must occur before downstream planning can be user-aligned.

Objective Scoring Requires Controlled Valid Options. Each task in UserBench is constructed to have valid solutions. This design keeps evaluation objective: agents can be scored against known correct and best options. If no valid option existed, evaluation would require additional subjective judgments about which preference should be relaxed and which alternative should be recommended. Such negotiation is important for future work, but the current benchmark deliberately focuses on whether agents can uncover and satisfy preferences when valid options are available.

Cost-Aware Scoring Adds a Soft Preference Dimension. The scoring function is not purely bi-

nary. Options that satisfy the hidden user preferences receive credit, while the best option receives full credit according to cost-aware selection. This means agents must not only identify valid options but also choose among them in a practically useful way. At the same time, we avoid assigning arbitrary weights to different hidden preferences, because doing so would introduce additional subjectivity and reduce comparability across tasks. The current scoring therefore balances partial credit, cost sensitivity, and objective evaluation.

Independence Across Aspects Is an Evaluation Control. In the current travel instantiation, choices across aspects are largely independent: for example, a flight option does not directly determine hotel availability. This is an intentional abstraction that helps isolate whether agents can elicit preferences and make aligned choices for each aspect. The benchmark still requires multi-aspect reasoning because agents must manage several hidden preferences across different tools and decisions. Richer cross-aspect dependencies can be added in future extensions, but the current design provides a clean and interpretable test of preference elicitation.

Single-Choice Evaluation Tests Commitment, Not Coverage. We use the single-choice setting as the main evaluation because it requires the agent to commit to one option per aspect. This setting better reflects whether the agent has actually identified a user-aligned decision. Multi-choice evaluation is useful for measuring recovery from earlier mistakes, but it can also reward broad coverage. The gap between single-choice and multi-choice results therefore provides additional evidence that current agents often benefit from extra attempts without necessarily improving user understanding.

Human Reference Is a Solvability Check, Not an Unfair Oracle. The human study uses the same interface and action space as the LLM agents. Human participants do not receive hidden preferences or ground-truth options; they must interact with the simulator and tools just like the models. Their stronger performance therefore shows that the benchmark is solvable through effective questioning and decision-making, rather than being artificially impossible. We refer to this result as a human reference rather than a human oracle to emphasize that it reflects skilled interaction under the same observable conditions.

Newer Models Improve but Do Not Close the Gap. Additional experiments with stronger recent models show that model progress improves UserBench performance, but the gap to human reference remains substantial. This is an important property of the benchmark: it is not saturated by current systems, and it continues to reveal failures in targeted clarification and preference-grounded decision-making. The goal is therefore not to establish a fixed ranking among model families, but to identify a persistent structural limitation in current agents.

Decoding Choices Do Not Explain the Main Findings. All main experiments use deterministic decoding for fair comparison. Since some models recommend non-zero sampling temperatures, we include a robustness check in Section E.5. The results show that sampling can slightly change performance, but it does not remove the main gap. This supports the conclusion that the benchmark measures a substantive interaction challenge rather than an artifact of a particular decoding setting.

Final Takeaway. UserBench is intentionally controlled, tool-grounded, and diagnostic. It does not aim to cover every form of human-agent collaboration or every challenge in travel planning. Its contribution is more precise: it formalizes proactive preference elicitation as a measurable agent capability, provides a reproducible environment for evaluating it, and shows that current LLM agents remain far from skilled human interaction behavior. By combining hidden preferences, implicit user expressions, multi-turn clarification, structured tools, human reference results, and behavioral analysis, UserBench positions user understanding as a concrete frontier for building more capable user-centric agents.

System and User Prompt for Judging Search Requests

[System Prompt]

Task
You are an expert judge evaluating whether an agent's search request aligns with the ground truth search arguments for a travel planning scenario.

Instruction
• Analyze the agent's search request to determine if it matches any of the ground truth arguments.
• Check if the search request is properly formatted and contains all the relevant information in the ground truth arguments.
• Determine the alignment judgement and identify the specific aspect if aligned.
• Provide your assessment in the specified JSON format.

Example Format
```
{
"alignment_judgement": "True/False",
"alignment_aspect": "flight/hotel/restaurant/apartment/rental_car"
}
```

Important Notes
• `"True"`: The search request aligns with one of the ground truth arguments. Note that all the arguments must be covered and correctly covered.
• `"False"`: The search request is malformed or contains incorrect arguments. Note that missing one argument or giving wrong arguments should all be marked as `"False"`.
• For `"True"` judgments, you must specify the alignment aspect.
• Aspect names should be: flight, hotel, restaurant, apartment, rental car.
• Be strict in your evaluation: Mark `False` if the request is ambiguous, unclear, or contains multiple search requests. Only mark as `True` if there's clear alignment with **all** the argument details.

[User Prompt]

Agent's Search Request:
`{{agent_request}}`

Ground Truth Arguments:
`{{ground_truth_arguments}}`

Please evaluate the alignment between the agent's search request and the ground truth arguments, then provide your assessment in JSON format.

Figure 7: System and user prompt for evaluating whether the agent's search request aligns with the ground truth search arguments.

System and User Prompt for Judging Agent Responses

[System Prompt]

Task
You are an expert judge evaluating the type of an agent's conversation utterance in a travel planning scenario to determine the appropriate response strategy.

Instruction
• Analyze the agent's latest utterance in the context of the conversation.
• Determine if the agent is explicitly asking for preferences that you have, asking for preferences that you don't have, giving a too general query, or just making general conversations.
• If asking for preferences that you have, identify which specific preference from the available list matches.
• Classify the utterance type and provide the assessment in JSON format.

Example Format
```
{
"type": "1/2/3/4",
"preference_id": "preference id if type is 1"
}
```

Important Notes
• Type `"1"`: Agent explicitly and concretely asking for a preference that exists in the available preferences list. The way how agent asks must be concrete in order to be classified as Type `"1"`.
• Type `"2"`: Agent explicitly and concretely asking for preferences, but the specific preference is not available. Similarly, the way how agent asks must also be concrete and specific.
• Type `"3"`: Agent making a very vague and general query about preference instead of focusing on a specific aspect (e.g. "Do you have any preferences for the car?" is vague and general, Type `"3"`; while "What exact model of the car do you like?" is concrete and specific, Type `"1"`).
• Type `"4"`: Normal conversation, not preference-related.
• For Type `"1"`, you must provide the exact one `preference_id` from the available preferences. If there's multiple preferences that match, choose the one that is most relevant to the conversation context.
• Be precise in identifying preference requests vs general conversation.

[User Prompt]

Travel Scenario:
`{{scenario}}`

Conversation History:
`{{conversation_history}}`

Agent's Latest Utterance:
`{{latest_utterance}}`

Available Preferences:
`{{preferences_list}}`

Please analyze the agent's latest utterance and classify its type, then provide your assessment in JSON format.

Figure 8: System and user prompt for evaluating the agent's utterance type and identifying preference-related behavior.

System and User Prompt for Preference-Based Responses

[System Prompt]

Task
You are a helpful user in a travel planning conversation who needs to respond to an agent's explicit request for your preference, which you should elicit in an implicit and indirect manner.

Instruction
• The agent has explicitly asked about a specific preference that you have.
• Respond in a natural, conversational way that reveals your preference implicitly and indirectly.
• Use the provided implicit elicitation statement as guidance, but make it sound natural in context.
• Keep the conversation flowing while sharing your preference information.
• Provide your response in the specified JSON format.

Example Format
```
{
"thought": "Your thought process of how to respond naturally and implicitly reveal the preference under the guidance of the implicit elicitation statement",
"response": "Your natural conversational response that implicitly reveals the preference"
}
```

Important Notes
• Respond naturally as if you're a real person sharing preferences.
• Don't directly state "My preference is..." – be more subtle, conversational, and indirect.
• Use the implicit elicitation statement as inspiration but adapt it to the conversation context.
• Keep responses appropriate length for natural conversation.
• Maintain consistency with the conversation history.

[User Prompt]

Your Preference:
`{{preference}}`

Conversation History:
`{{conversation_history}}`

Agent's Latest Utterance:
`{{latest_utterance}}`

Please respond naturally to the agent's request while implicitly sharing your preference under the guidance of the implicit elicitation statement. Provide your response in JSON format.

Figure 9: System and user prompt for responding to agent requests by implicitly revealing the user's preference.

System and User Prompt for Proactively Introducing Preferences

[System Prompt]

Task
You are a helpful user in a travel planning conversation who needs to proactively, naturally, but indirectly introduce a preference into the conversation.

Instruction
• The conversation has gone several turns without preference discussion.
• Naturally steer the conversation to reveal one of your preferences.
• Use the provided implicit elicitation statement as guidance for how to reveal the preference.
• Make the preference revelation feel organic and contextually appropriate, but still in an implicit and indirect manner.
• Provide your response in the specified JSON format.

Example Format

```
{
"thought": "Your thought process of how to naturally and implicitly introduce
the preference under the guidance of the implicit elicitation statement",
"response": "Your natural conversational response that proactively introduces
the preference"
}
```

Important Notes
• Connect to the current conversation context when possible.
• Make the preference introduction feel natural and not forced, but still in an implicit and indirect manner.
• Use the implicit elicitation statement as inspiration but adapt to the conversation flow.
• Don't abruptly change topics – find natural transitions and keep responses conversational and engaging.
• If the implicit elicitation statement cannot clearly indicate what high-level aspect (flight, restaurant, etc.) the preference is about, you should be clear about the high-level aspect in your elicitation to avoid confusion, but still elicit the concrete preference in an implicit way.

[User Prompt]

Preference to Elicit:
`{{preference}}`

Conversation History:
`{{conversation_history}}`

Agent's Latest Utterance:
`{{latest_utterance}}`

Please respond naturally while proactively introducing your preference into the conversation in an implicit and indirect manner under the guidance of the implicit elicitation statement. Provide your response in JSON format.

Figure 10: System and user prompt for proactively and implicitly introducing a user preference into the conversation.

System and User Prompt for Natural Conversation Responses

[System Prompt]

Task
You are a helpful user in a travel planning conversation who needs to respond naturally to the agent's utterance.

Instruction
• The agent's utterance is not related to any specific preferences you have.
• Respond naturally and in a succinct manner, like you are giving a half-hearted reply.
• Be neutral and do not reveal any new or arbitrary personal preferences.
• Provide your response in the specified JSON format.

Example Format
```
{
"thought": "Your thought process of how to respond naturally and keep the conversation flowing while being neutral",
"response": "Your natural conversational response"
}
```

Important Notes
• Keep responses natural, conversational and succinct.
• Stay on topic with travel planning when appropriate, but do not actively ask any questions.
• Don't introduce any personal preferences. If being asked, you should be neutral (e.g. "I don't have a preference on that", "Everything is fine") and do not arbitrarily reveal any new preferences.

[User Prompt]

Conversation History:
{{conversation_history}}

Agent's Latest Utterance:
{{latest_utterance}}

Please respond naturally to continue the conversation and keep neutral without revealing any new or arbitrary personal preferences. Provide your response in JSON format.

Figure 11: System and user prompt for generating neutral and natural user responses when no preferences are involved.

System Prompt for Agent (Single-Choice Setting)

[System Prompt]

You are an agent that actively interacts with a specific environment. The followings are the details of the environment and your action space.

Environment Description: UserBench is an environment where you interact with both a user and a search database to fulfill a travel plan. Since the user's initial intent may be incomplete, you must proactively elicit preferences, perform searches, and make informed recommendations.

Action Space: You should invocate the function `interact_with_env` to interact with the environment. The action should be one of the following: `search`, `action`, or `answer`.

Action Description:
• `search`: If you choose `search`, you must issue a clear and detailed query to the database. Specify the travel aspect you are searching for (e.g., hotel, flight, etc.) and provide well-supported arguments for your query. Only make one focused search attempt at a time.
• `action`: If you choose `action`, you will communicate directly with the user. Your goal is to understand the user's preferences and intent by asking clear, specific questions. Avoid vague or overly general inquiries—focus on detailed aspects of their travel needs.
• `answer`: If you choose `answer`, you should recommend a specific option to the user by only providing its option ID from the database in the content field, clearly tied to a particular travel aspect.

Important Notes:
• In each round of your interaction, you should analyze and carefully consider what to do next, and then invocate the `interact_with_env` tool to interact with the environment. You should provide your thought in each step when invoking the tool.
• The total number of rounds that you can interact with the environment is limited. You should smartly balance the number of rounds that you search, take action, or provide answer.
• Typically, you should start by performing a search, then take action to actively uncover the user's preferences or reason to provide an answer. **Keep in mind that multiple travel aspects require answers, and you are allowed to recommend only one option per aspect. Therefore, before making a recommendation, ensure you have thoroughly communicated with the user to understand their preferences.**
• For each travel aspect, the user may have multiple preferences. What you ask may not directly align with the user's actual preferences, so you must proactively uncover them. Moreover, user preferences are often expressed implicitly, requiring careful interpretation.
• Be bold, creative and smart in your interaction with the environment! Let's begin!

Figure 12: System prompt used to instruct the UserBench agent (single-choice setting).

System Prompt for Agent (Multi-Choice Setting)

[System Prompt]

You are an agent that actively interacts with a specific environment. The followings are the details of the environment and your action space.

Environment Description: UserBench is an environment where you interact with both a user and a search database to fulfill a travel plan. Since the user's initial intent may be incomplete, you must proactively elicit preferences, perform searches, and make informed recommendations.

Action Space: You should invocate the function `interact_with_env` to interact with the environment. The action should be one of the following: `search`, `action`, or `answer`.

Action Description:
• `search`: If you choose `search`, you must issue a clear and detailed query to the database. Specify the travel aspect you are searching for (e.g., hotel, flight, etc.) and provide well-supported arguments for your query. Only make one focused search attempt at a time.
• `action`: If you choose `action`, you will communicate directly with the user. Your goal is to understand the user's preferences and intent by asking clear, specific questions. Avoid vague or overly general inquiries—focus on detailed aspects of their travel needs.
• `answer`: If you choose `answer`, you should recommend a specific option to the user by only providing its option ID from the database in the content field, clearly tied to a particular travel aspect.

Important Notes:
• In each round of your interaction, you should analyze and carefully consider what to do next, and then invocate the `interact_with_env` tool to interact with the environment. You should provide your thought in each step when invoking the tool.
• The total number of rounds that you can interact with the environment is limited. You should smartly balance the number of rounds that you search, take action, or provide answer.
• Typically, you should start by performing a search, then take action to actively uncover the user's preferences or reason to provide an answer. **Keep in mind that multiple travel aspects require answers, and while you may answer multiple times, each answer should include only one option ID.**
• For each travel aspect, the user may have multiple preferences. What you ask may not directly align with the user's actual preferences, so you must proactively uncover them. Moreover, user preferences are often expressed implicitly, requiring careful interpretation.
• Be bold, creative and smart in your interaction with the environment! Let's begin!

Figure 13: System prompt used to instruct the agent (multi-choice setting).

Initial User Message

[User Prompt]

`{{The initial description of user request}}`

Also my budget is limited, so as long as my preferences are satisfied, I would also like to choose the cheapest option for each.

Figure 14: Initial user message provided in the UserBench simulation.

Good Example - Qwen3-32B (Travel-444)

Concrete Interaction Sequence (First 6 Turns):

Turn 1 - *ACTION* (Reward: 0.0)
Content: "Could you share your preferences for the hotel (e.g., neighborhood, amenities), car rental (e.g., economy, SUV), and restaurant (e.g., cuisine type, dietary restrictions)? This will help me find the most cost-effective options that suit your needs."
Feedback: "Your question is too vague and general, and I am not sure how to respond to it. Please ask me about some specific aspects of my preferences, in a more detailed and concrete way, so that I can provide you with a more accurate response."

Turn 2 - *ACTION* (Reward: 0.2)
Content: "Let's clarify your preferences step by step:
1. **Hotel**: Would you prefer a hotel in a specific San Francisco neighborhood (e.g., Union Square for downtown access, Fisherman's Wharf for scenic views, or Mission District for affordability)? Do you need amenities like free breakfast, parking, or pet-friendly policies?
2. **Car Rental**: What type of vehicle are you looking for? For example: economy car (e.g., Toyota Corolla), compact SUV (e.g., Honda CR-V), or hybrid? Do you need features like automatic transmission, GPS, or a sunroof?
3. **Restaurant (April 11th)**: What cuisine type are you interested in (e.g., Italian, Mexican, seafood, or vegan)? Do you have dietary restrictions (e.g., gluten-free, halal, or nut allergies)? Would you prefer a casual eatery or a higher-end dining experience within your budget?"
Feedback: "When it comes to hotels, I find it really reassuring to know that my car is well taken care of, especially when I'm exploring a new city like San Francisco. It just adds a layer of comfort to my stay."

Turn 3 - *SEARCH* (Reward: 0.2)
Content: "Search for hotels in San Francisco from April 10th to April 17th with parking availability. Sort results by price (lowest first) and filter for budget-friendly options."
Feedback: "You have provided the correct search request arguments.
Each hotel option includes: - name: Hotel name - room: Array of available room types with capacity - cost: Array of costs for different room types for the entire stay in USD - rating: Hotel rating on a scale of 0-10 - amenities: Available facilities free of charge - service: Additional services with costs

Turn 4 - *ACTION* (Reward: 0.2)
Content: "Here are the top 3 budget-friendly hotels with parking in San Francisco:
1. **Sunset Retreat (H8)**: $1280 total (room + parking), 7/10 rating, mountain view, pets allowed, WiFi.
2. **Vista Point Lodge (H15)**: $1390 total, 7/10 rating, mountain view, business workspace.
3. **Harbor View Lodge (H13)**: $1690 total, 5/10 rating, ocean view.
Would you like to proceed with the cheapest option (H8), or do you have preferences for ratings, room types, or other amenities?"
Feedback: "You know, I've always found that places with a rating around 7 tend to strike a nice balance between quality and affordability. The Sunset Retreat and Vista Point Lodge both seem to fit that bill quite well. Could you tell me a bit more about the room types they offer?"

Turn 5 - *ACTION* (Reward: 0.2)
Content: "Here's a detailed breakdown of the room types for the two hotels:
Sunset Retreat (H8) - **King Room**: $1200 total for 7 nights (includes parking: $80). Capacity: 2 people. Amenities: Mountain view, pets allowed, high-speed WiFi. - **Queen Room**: $1800 total for 7 nights (includes parking: $80). Capacity: 2 people. Slightly larger space but same amenities.
Vista Point Lodge (H15) - **Double Room**: $1300 total for 7 nights (includes parking: $90). Capacity: 2 people. Amenities: Mountain view, business workspace, high-speed WiFi. - **Queen Room**: $1500 total for 7 nights (includes parking: $90). Capacity: 2 people. Additional workspace and slightly upgraded comfort.
Would you prefer the **cheapest option (H8 King)** or the **workspace-inclusive option (H15 Double)**?"
Feedback: "There's something quite appealing about having a bit of extra space and the option to switch things up depending on the mood. The Double room at Vista Point Lodge sounds like it could offer that kind of flexibility and comfort."

Turn 6 - *ANSWER* (Reward: 1.0)
Content: "H15"
Feedback: "Your chosen options contain the best option! Your choice is recorded and do not choose options of this travel aspect again. Please continue your interaction and reasoning focusing on other travel aspects."

Figure 15: Good example showing successful hotel planning with clear preference elicitation, relevant search, and optimal recommendation by Qwen3-32B.

Good Example - Claude-4-Sonnet (Travel-444)

Analysis Summary:

Concrete Interaction Sequence (First 9 Turns):

Turn 1 - *SEARCH* (Reward: 0.2)
Content: "flight from Los Angeles to New York City on March 15th, one-way"
Feedback: You have provided the correct search request arguments.

Turn 2 - *SEARCH* (Reward: 0.2)
Content: "car rental in New York City from March 15th to March 20th"
Feedback: You have provided the correct search request arguments.

Turn 3 - *SEARCH* (Reward: 0.2)
Content: "restaurant in New York City on March 17th"
Feedback: You have provided the correct search request arguments.

Turn 4 - *ACTION* (Reward: 0.2)
Content: "I found several flight options for your March 15th trip from Los Angeles to New York City. Since you mentioned budget is a concern, I see some very affordable options. However, I'd like to understand your preferences better:
1. Are you comfortable with connecting flights, or do you prefer direct flights? 2. Do you have any airline preferences or ones you'd like to avoid? 3. How important are amenities like WiFi, meal service, or lounge access to you? 4. Will you need to check any bags for your business trip?"
Feedback: I don't mind a connecting flight if it means I get to experience a different airport along the way.

Turn 5 - *ACTION* (Reward: 0.0)
Content: "A few more questions to help me find the perfect flight for you:
1. Do you have any airline preferences, or are you flexible as long as the price is right? 2. Will you need to check any bags for your business trip, or are you planning to travel with just carry-on? 3. Are there any specific amenities that are important for your business travel (like WiFi for working during the flight)?"
Feedback: Your question is too vague and general. Please ask me about some specific aspects in a more detailed and concrete way.

Turn 6 - *ACTION* (Reward: 0.0)
Content: "You're right, let me be more specific. For your business trip, will you need to check a suitcase, or can you manage with just a carry-on bag? This is important because checked bag fees can add 25−60 to your total flight cost depending on the airline."
Feedback: Good question, but I do not have a specific preference on that aspect yet. Please continue asking about other detailed preferences.

Turn 7 - *ACTION* (Reward: 0.2)
Content: "Do you have any specific airlines you prefer to fly with or any you'd rather avoid? For example, some business travelers prefer major carriers like United or American, while others are comfortable with budget airlines like Spirit or Southwest."
Feedback: I always feel well taken care of with United Airlines.

Turn 8 - *ACTION* (Reward: 0.2)
Content: "Perfect! I can see you appreciate United's service quality. Now, would you prefer a shorter total travel time even if it costs a bit more, or are you willing to have a longer journey with a longer layover if it saves you money?"
Feedback: I get restless with long layovers, so I prefer shorter journeys even if they cost more.

Turn 9 - *ANSWER* (Reward: 1.0)
Content: "F15"
Feedback: Your chosen options contain the best option! Your choice is recorded. Please continue your interaction focusing on other travel aspects.

Figure 16: Good example showing Claude-4-Sonnet's ability to elicit implicit preferences, adapt follow-up questions, and recommend optimal options based on evolving user intent.

Bad Example - Deepseek-V3 (Travel-22)

Concrete Interaction Sequence (First 7 Turns):

Turn 0 - *SEARCH* (Reward: 0.2)
Content: "Search for budget-friendly apartments in Austin available from November 10th to November 15th, sorted by price (lowest first)."
Feedback: You have provided the correct search request arguments.

Turn 1 - *ACTION* (Reward: 0.0)
Content: "How many people will be joining the trip? This will help me find an apartment that comfortably accommodates everyone."
Feedback: I'm not sure about the exact number right now.

Turn 2 - *SEARCH* (Reward: 0.0)
Content: "Search for budget-friendly apartments in Austin with 2-3 bedrooms, available from November 10th to November 15th, sorted by price (lowest first)."
Feedback: You have already got the search results for <apartment>. Please directly refer to the previous search results.

Turn 3 - *ANSWER* (Reward: 0.0)
Content: "A10"
Feedback: Your chosen options do not contain any of the best or correct options. Please continue your interaction focusing on other travel aspects.

Turn 4 - *SEARCH* (Reward: 0.2)
Content: "Search for budget-friendly car rental options in Austin available from November 10th to November 15th, sorted by price (lowest first)."
Feedback: You have provided the correct search request arguments.

Turn 5 - *ANSWER* (Reward: 0.0)
Content: "C18"
Feedback: Your chosen options do not contain any of the best or correct options. Please continue your interaction focusing on other travel aspects.

Turn 6 - *ANSWER* (Reward: 0.0)
Content: "A10"
Feedback: You have already recommended an option with the same initial 'A'. You are allowed to recommend only one option per travel aspect.

Figure 17: Negative example showing Deepseek-V3's failure to elicit meaningful preferences or recover from ambiguity. The model issues redundant queries, skips clarification, and selects incorrect options without leveraging available feedback.