GROUPTRAVELBENCH: BENCHMARKING LLM AGENTS ON MULTI-USER, MULTI-TURN TRAVEL PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Travel planning in the real world is overwhelmingly a *group* activity, yet existing LLM travel-planning benchmarks reduce it to a single user, leaving a considerable gap relative to real-world needs. This single-user assumption sidesteps what makes group planning hard for an agent: discovering private preferences across multiple users, surfacing conflicts, and balancing utility against fairness. To bring the task back to its multi-user reality, we introduce ***GroupTravelBench***, the first benchmark for **multi-user, multi-turn** travel planning. Built from real user profiles, Point-Of-Interest (POI) data, and ticket prices, it comprises 650 tasks across three difficulty levels, each running in a controlled synchronous group-chat framework with cached tool data for reproducible offline evaluation. Beyond the multi-step reasoning and tool use that single-user benchmarks already test, GroupTravelBench probes three group-specific capabilities: *(i) elicitation* of private preferences through multi-turn dialogue; *(ii) coordination* of inter-user conflicts via compromise or subgrouping; and *(iii) planning* that balances group utility against fairness. We pair this with a complementary evaluation framework combining rule-based outcome metrics and LLM-judge process metrics. Across a wide range of frontier models, even the strongest agents fall short on all four rule-based outcome metrics, with plan validity at most 12%, suggesting that group-level planning quality is a key open challenge for LLM travel-planning agents.

1 INTRODUCTION

LLM agents have been widely studied for travel planning. In real-world travel planning, families, friends, and colleagues plan trips together, and their preferences are private, often partial, and frequently in conflict—where to go, how much to spend, when to wake up. Producing a satisfying itinerary therefore demands more than search and tool use: an agent must *elicit* the preferences each user holds privately, *coordinate* disagreements when they surface, and *plan* an itinerary that balances overall utility with fairness. These three capabilities are routine in human group coordination, yet they sit outside the scope of how travel planning is currently studied for LLM agents.

A growing body of work has framed travel planning as a benchmark for LLM agents, starting from TravelPlanner (Xie et al., 2024) and extending toward stricter constraint checking (Shao et al., 2026), finer scoring (Qu et al., 2025), multi-turn interaction (Qin et al., 2025; Cheng et al., 2026), and dynamic disturbances (Karmakar et al., 2025). Yet all of these reduce travel planning to a *single* user, and on this surface the field is approaching saturation: TravelPlanner has been largely solved by symbolic methods, with reported pass rates above 97% (Hao et al., 2025), and the strongest models on the most recent multi-turn travel-planning benchmark already approach 80% (Cheng et al., 2026). Group-side work, in contrast, has long existed in operations research (Sylejmani et al., 2017; Jouyandeh & Moradian Zadeh, 2023; Liu et al., 2025), where group planning is formulated as a combinatorial optimization problem and preferences are treated as fixed numerical vectors given upfront—skipping precisely the interactive parts that make group planning hard for LLM agents.

Bringing the planning task back to its multi-user reality changes the problem along two compounding axes. **(a)** The plan space grows combinatorially with group size, since every additional user introduces preference dimensions, schedule constraints, and conflict points that must be jointly satisfied—this

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

stresses the underlying planning capability beyond what current single-user benchmarks can probe. **(b)** Multi-user planning demands an interactive loop of *elicitation*, *coordination*, and group-aware *planning* that no single-user setting can test by construction, regardless of how complex its constraints. A benchmark targeting this regime therefore evaluates *harder* versions of what existing benchmarks cover, while opening up a class of capabilities they cannot reach at all.

To operationalize this, we introduce ***GroupTravelBench***: 650 multi-user travel-planning tasks built from real user profiles, real POI data, and real-world ticket prices, organized into three difficulty levels. Each user's preferences follow a four-tier hierarchy (*must* / *prefer* / *avoid* / *reject*) and are tracked in three coexisting tables—original, agent-inferred, and post-compromise—so that elicitation and conflict resolution can be scored deterministically. Each task runs in a synchronous group-chat sandbox where one agent interacts with multiple user simulators, supported by 10 travel tools and over 250K cached real-world tool records for stable, reproducible offline evaluation.

We pair the benchmark with a comprehensive evaluation framework that combines two complementary views: rule-based metrics for outcome quality (preference completeness, group utility, group fairness, plan validity) and an LLM judge for process quality (hallucination, tool use, interaction, conflict coordination, plan humanization). Across a wide range of frontier models, even the strongest agents fall short across all four rule-based outcome metrics—plan validity is at most 12% across all models—indicating that group-level outcome quality remains an important bottleneck.

Main contributions.

(1) ***GroupTravelBench***, the first benchmark for **multi-user, multi-turn** travel planning, jointly evaluating preference *elicitation*, conflict *coordination*, and group-aware *planning*.

(2) A **reproducible interactive framework** that simulates group-chat dynamics—@-mention scheduling, convergence checkpoints, and compromise or subgrouping—backed by a sandbox of 10 travel tools and over 250K cached real-world records for stable offline evaluation.

(3) An evaluation methodology that uses three preference tables to **deterministically score preference elicitation and conflict resolution**, complemented by an LLM judge for process quality.

2 Related Work

2.1 Group Travel Planning

Planning itineraries for groups with heterogeneous preferences has long been studied as combinatorial optimization in the operations research community. Early efforts extend the Tourist Trip Design Problem to multiple travelers by modeling individual preference profiles and inter-member relationships, and solve the resulting problem with metaheuristics such as tabu search and evolutionary algorithms (Sylejmani et al., 2017; Jouyandeh & Moradian Zadeh, 2023). Subsequent work enriches this formulation with sustainability objectives and fuzzy-set preference modeling (Ruiz-Meza et al., 2022) and with dynamic subgrouping or "joining-and-forking" strategies that let members temporarily separate when preferences diverge (Liao et al., 2022; Alatiyyah, 2025); more recent efforts cast the problem as a crowd-aware Markov decision process (Liu et al., 2025) or as multi-party multi-objective optimization with bargaining game theory (Kogoya et al., 2026).

Despite steady progress, these approaches encode preferences as pre-defined numerical vectors and restrict planning scope to POI sequencing. None of these approaches features a conversational agent that actively elicits, clarifies, and reconciles conflicting requirements through dialogue, limiting their applicability to interactive real-world settings.

2.2 LLM-Based Single-User Travel Planning

Building on TravelPlanner (Xie et al., 2024), which first framed travel planning as tool-grounded itinerary construction under hard and commonsense constraints, this line of work has evolved along four axes. The first pushes *constraint complexity*: ChinaTravel (Shao et al., 2026) adds DSL-based compositional constraints, while more recent benchmarks tighten cross-action coupling (Wang et al., 2026b) and demand long-horizon global feasibility (Zhang et al., 2026c; Yang et al., 2025b); on the solution side, solver-based methods (Hao et al., 2025; Shao et al., 2025) and decoupled or RL-trained

planners (Yuan et al., 2026; Wang et al., 2026a) largely close the feasibility gap on these benchmarks. The second moves *beyond feasibility*: TP-RAG (Ni et al., 2025), TripTailor (Wang et al., 2025a), and TripScore (Qu et al., 2025) shift evaluation toward spatiotemporal rationality, personalization, and unified plan quality. The third models *realistic interaction* with implicit or under-specified intent, where the agent must proactively clarify (Zhang et al., 2024), optimize user preferences across multi-turn dialogue (Qin et al., 2025; Deng et al., 2025), or jointly elicit implicit preferences and flag unsolvable requests (Cheng et al., 2026). The fourth probes *robustness and root causes*: TripTide (Karmakar et al., 2025) stress-tests itineraries under real-world disruptions, and a recent diagnostic study attributes residual failures to weak inference of implicit constraints and ineffective self-correction (Zhang et al., 2026a).

Across this body of work, however, the unit of evaluation remains *one* traveler's plan. No existing benchmark requires an agent to elicit preferences from multiple participants, surface inter-user conflicts, or balance group-level fairness—yet real travel planning is overwhelmingly a group activity, the gap our benchmark is designed to fill.

2.3 LLM Agents and Agentic RL in Travel

Building on ReAct-style tool use (Yao et al., 2022), a growing body of work trains LLM agents with reinforcement learning along three threads. The first establishes general recipes for tool-use RL, showing that outcome-based or hierarchically structured rewards suffice for learning strategic, multi-tool reasoning (Qian et al., 2025; Feng et al., 2025; Dong et al., 2025). The second targets practical bottlenecks of agentic RL—scaling to long-horizon trajectories (Wu et al., 2026) and stabilizing reward signals on open-ended tasks via tournament-based relative ranking (Zhang et al., 2026b). The third applies agentic RL directly to travel: DeepTravel (Ning et al., 2025) introduces hierarchical reward modeling and failure replay buffers, and TourPlanner (Wang et al., 2026a) adds constraint-gated optimization with multi-path consensus. All such advances, however, target *single-user, single-objective* settings: no current training framework addresses multi-user planning. Our benchmark identifies precisely this capability gap, indicating a clear direction for future training and providing an initial resource for such research.

3 GroupTravelBench

Figure 1 gives an overview of *GroupTravelBench*. Preference modeling is foundational to *GroupTravelBench*, as it directly shapes task synthesis, user simulator behavior, and how the evaluation protocol scores agent outputs. We therefore present this scheme first (§3.1), and then describe task synthesis (§3.2), multi-user interaction (§3.3), and the evaluation protocol (§4).

3.1 Preference Modeling

A key challenge in group travel planning is how to model user preferences and evaluate conflict resolution in a deterministic manner. We argue that real-world preferences are inherently hierarchical: users may have options they strongly desire, clearly dislike, or feel neutral about. Such structured preferences naturally test an agent's ability to discover conflicts and resolve them effectively. A strong agent should be able to infer preferences from natural language, recognize their intensity, and use this information to mediate conflicts and generate a feasible itinerary.

To enable rule-based evaluation of both preference elicitation and conflict resolution, we design a structured preference table to model user preferences based on realistic travel needs (see Appendix A.1 for details). Each preference is associated with one of four intensity levels: `must`, `prefer`, `avoid`, and `reject`. The final itinerary receives different rewards or penalties depending on whether these preferences are satisfied or violated. For example, satisfying a `must` preference yields +2 points, whereas violating an `avoid` preference incurs a penalty of 1 point. During interaction, users express their preferences according to this table, and accepted compromises are reflected through updates to the preference table.

Three pieces of information must coexist for evaluation to be deterministic: what each user *actually* prefers, what the agent *believes* they prefer, and what they *agree to* after compromise. We therefore maintain three coexisting preference tables for each user i: the **original preference table** $\mathbf{p}_i$, which

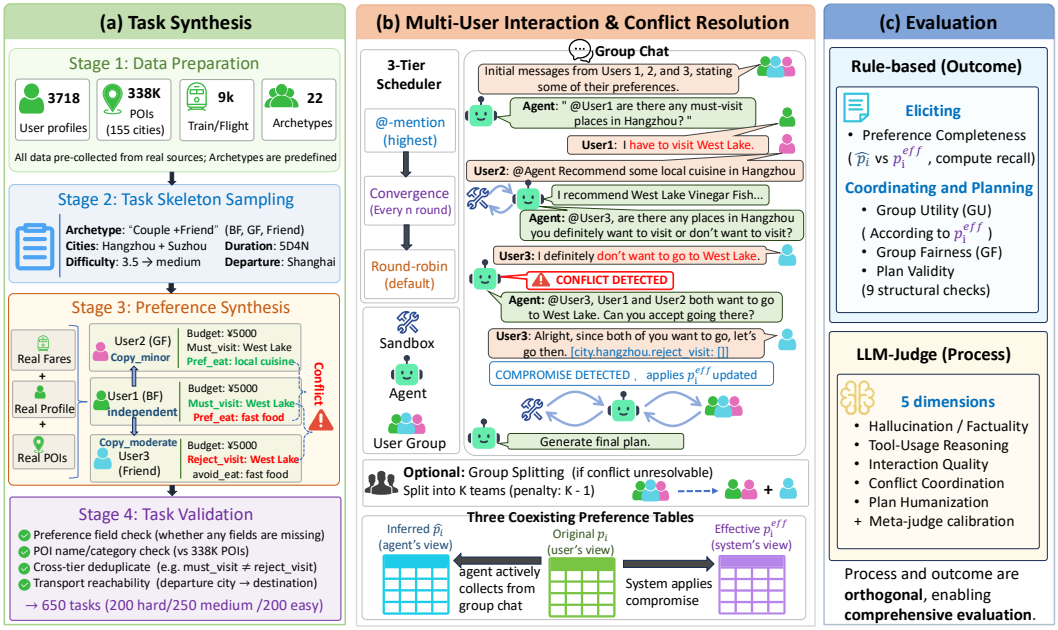


Figure 1: Overview of *GroupTravelBench*: (a) task synthesis grounded in real-world data, (b) controlled multi-user interaction, and (c) outcome- and process-level evaluation. The dialogue excerpt is abridged from an actual benchmark run.

contains the user's hidden ground-truth preferences; the **agent-inferred table** $\hat{\mathbf{p}}_i$, which records the agent's current belief about that user; and the **effective preference table** $\mathbf{p}_i^{\text{eff}}$, which applies all accepted compromises to the original preferences during natural-language interaction. When the agent identifies a conflict, it may explicitly ask a user to compromise. If the user is willing to compromise, the simulator responds in natural language and additionally emits a system-readable update. This update is validated by the framework and written into $\mathbf{p}_i^{\text{eff}}$. The update itself is not exposed to the agent. Instead, the agent must infer from the user's natural-language response whether a compromise has occurred and update its inferred preference table $\hat{\mathbf{p}}_i$ accordingly. This design allows us to evaluate not only conflict resolution, but also the agent's ability to recognize and track preference changes during interaction.

3.2 Task Synthesis

Real multi-user travel-planning data is hard to obtain at scale—it lives in private group chats and mixes implicit preferences with verbal hedging. Yet the underlying task template is highly regular: a fixed group of N people heading to one or more destinations over a fixed number of days. This regularity lets us synthesize a large, diverse benchmark *around* a real-world data backbone (POIs, ticket prices, user profiles), rather than fabricating content from scratch. We instantiate this idea via a four-stage pipeline—*preparing* real data, *sampling* task skeletons, *generating* user preferences, and *post-validating* the final tasks—with each stage controllable and grounded in real data, thereby balancing realism and distributional diversity.

Stage 1: Data preparation. We first prepare four types of resources for task construction: **POI data** contain about 338K entries collected from Chinese map services, including 100K attractions, 119K restaurants, and 119K hotels across 155 predefined cities, each annotated with categories and opening hours. **Transportation data** include pre-collected train and flight prices between city pairs. **User profiles** consist of 3,718 anonymized real user preferences covering demographics such as age, gender, marital status, and parenting status. **Group archetypes** cover 22 common travel settings for groups of size 2–6 (Appendix Table 5).

Stage 2: Task skeleton sampling. We then sample a task skeleton by jointly selecting a group archetype, a destination set (single city or 2–3 geographically adjacent cities), a departure city, a trip duration of 2–7 days, and a compromise configuration that specifies which users can make

compromises. Sampling is popularity-weighted to reflect realistic travel demand. Difficulty labels are then assigned based on group size, trip length, and number of destinations, yielding 200 easy, 250 medium, and 200 hard tasks (Appendix A.9.2).

Stage 3: Preference table generation. Real groups share some tastes but diverge on others; generating preferences fully independently erases this correlation, while copying them across users erases individual diversity. We therefore generate each group member's hierarchical preference table with controllable inheritance. We first sample a role-consistent user profile from the profile pool; for example, a "grandfather" role is sampled from profiles with matching age and family attributes. We then provide the LLM with real city-specific POI candidates and transportation prices as context, and ask it to generate user i's original structured preference table $\mathbf{p}_i$ grounded in those inputs.

Preference tables include both *global constraints*, such as budget and transportation preferences, and *city-level preferences*, such as attraction and food choices. To create realistic within-group diversity, we use three generation strategies based on the strength of user relationships: **copy_minor**, where broad preferences are inherited but specific POI-level choices are regenerated; **copy_moderate**, where only *global constraints* are inherited while city-level preferences are regenerated; and **independent**, where the entire table is generated from scratch. These strategies allow group members to share some tastes without becoming unrealistically identical (Appendix A.4). We also generate one opening utterance per user, revealing only 1–3 preferences as conversational seeds, with wording aligned to preference strength. The model used for preference generation is Gemini-3-Flash (Google, 2025).

Stage 4: Post-validation. LLM-based synthesis can hallucinate POIs or introduce internal inconsistencies, so we apply three validation checks—POI existence, cross-tier consistency, and transportation reachability (Appendix A.4)—before releasing each task. Invalid tasks trigger regeneration; passing tasks are grounded in the available resources and internally consistent at the preference-table level.

3.3 Multi-Party Interaction Framework

A central goal of *GroupTravelBench* is to faithfully reproduce how multi-user travel planning unfolds in real chat groups. In practice, users @-mention the agent to ask for information, the agent @-mentions individual users to elicit private preferences or request compromises, and the agent has to keep nudging the discussion toward consensus. To capture this loop within an evaluable framework, each task runs in a synchronous group-chat framework where one agent interacts with N LLM-based user simulators through a shared ordered conversation history.

Interaction protocol. The interaction begins with a broadcast of the pre-generated opening utterances and proceeds under a simple scheduler: when the agent @-mentions a user, that user gains immediate speaking priority; otherwise, turns follow round-robin order. Every κ rounds the agent must summarize the preferences and conflicts collected so far, providing periodic convergence signals. When two preferences cannot be reconciled, it may request a compromise from a specific user or split the group into subgroups with parallel sub-itineraries; subgrouping makes planning easier and therefore incurs a penalty, treating it as a costly meta-decision rather than a free escape from coordination. Tool calls are routed through a sandbox of 10 travel tools backed by over 250K cached real-world records (Appendix B), with retrieval-based fallback on cache misses for stable offline evaluation (Cheng et al., 2026; Guo et al., 2024). Per-difficulty round budgets, compromise and subgrouping mechanics, termination conditions, and further details appear in Appendix A.6.

User behavior constraints. To ensure the benchmark measures *the agent's* elicitation and coordination rather than user-side planning, simulators are forbidden from proactively revealing preferences; they speak only when @-mentioned or when their strong preferences are violated. This restriction deliberately trades open-ended conversational realism for controlled, reproducible evaluation. The complete behavior contract is in Appendix A.5.

3.4 Dataset Statistics

Appendix A.9 includes the main statistics of *GroupTravelBench*. The benchmark contains 650 tasks (200 easy, 250 medium, and 200 hard), covering 2,748 preference-bearing users in total and an average of 4.2 such users per task. Pre-school children (under 8) are listed as participants in the group but carry no preference table, and are therefore excluded from this count. It spans 22 group archetypes, 143 destination cities, and 32 departure cities. In total, the benchmark includes nearly

63K atomic preference items grounded in 3,718 real user profiles and approximately 338K real POIs. Conflict is common by construction: 77.4% of tasks contain at least one within-group preference conflict and the per-task mean is 7.52 (Appendix A.9.3), providing substantial coverage for evaluating conflict discovery and coordination.

4 Evaluation Protocol

We evaluate each run from two complementary perspectives: **outcome quality** and **process quality**. Rule-based metrics can verify the final itinerary against explicit constraints, but they cannot capture process quality or soft requirements such as accommodation for elderly travelers or children. We therefore additionally introduce an LLM judge (Hashemi et al., 2024) to evaluate process quality. The two views are complementary rather than redundant: the LLM judge is explicitly instructed not to score aspects already covered by deterministic checks, such as formatting, time consistency, or exact numerical matching.

4.1 Rule-Based Outcome Metrics

Preference Completeness (PC). PC measures how accurately the agent collects user preferences during dialogue. We compare the agent's final inferred table $\hat{\mathbf{p}}_i$ against the effective preference table $\mathbf{p}_i^{\text{eff}}$ of user i. For list-valued fields, we count matched items; for scalar fields, we require exact match. We report recall as the main metric:

$$\text{PC} = 100 \cdot \frac{\sum_i \text{collected}_i}{\sum_i \text{possible}_i}, \tag{1}$$

where i indexes users in the group.

Group Utility (GU). GU measures how well the final itinerary satisfies group preferences after compromise. For each user i, we compute a utility score u_i against $\mathbf{p}_i^{\text{eff}}$, where strong satisfaction and violation receive ± 2, and weak satisfaction and violation receive ± 1. The score covers attractions, food, transportation mode, hotel type, activity intensity, and budget. If the agent uses subgrouping, we apply a split penalty: each split event e that divides the group into K_e subgroups incurs a penalty of $K_e - 1$. We report:

$$\text{GU} = \frac{1}{N}\left(\sum_{i=1}^{N} u_i - \sum_e (K_e - 1)\right), \tag{2}$$

where N is the group size.

Group Fairness (GF). A plan with high overall utility may still neglect some users. GF measures the balance of utility across users. For groups in which every user receives positive utility, we report:

$$\text{GF} = 100 \cdot \frac{\min_i u_i}{\max_i u_i}. \tag{3}$$

When any user receives non-positive utility ($\min_i u_i \leq 0$), we set GF to 0.

Plan Validity (PV). PV measures whether the generated itinerary is physically feasible. We apply nine deterministic checks spanning transportation, lodging, scheduling, and cost completeness (full list in Appendix A.8). A plan is counted as valid only if it passes all checks:

$$\text{PV} = \mathbb{1}[\text{issues}(\hat{y}) = \emptyset], \tag{4}$$

where $\hat{y}$ is the agent's final plan and $\text{issues}(\cdot)$ returns the list of failed checks.

4.2 LLM-Based Process Evaluation

Rule-based metrics capture only verifiable outcomes. We therefore employ an LLM judge to evaluate process quality along five dimensions: **hallucination**, **tool-use reasoning**, **interaction quality**, **conflict coordination**, and **plan humanization**. Each dimension is scored on a 1–5 scale with evidence grounded in specific dialogue turns. More details are provided in Appendix A.7.

A single LLM judge can be biased or inconsistent across runs, so we add a **meta-evaluation** step that calibrates each judge verdict: a second LLM reviews the first judge's reasoning and produces a

calibration score $s^{\text{meta}} \in \{1, \ldots, 5\}$ that scales the final score. Calibration setup and rater-agreement statistics appear in Appendix C.11. The final LLM-judge score for a task is:

$$S^{\text{LJ}} = 100 \cdot \frac{1}{5} \sum_{d=1}^{5} \frac{r_d - 1}{4} \cdot \frac{s^{\text{meta}}}{5}, \tag{5}$$

where r_d is the score in the d-th dimension.

4.3 Reporting

We report all five metrics separately: PC, GU, GF, PV, and S^{LJ}. All five are first computed per task and then averaged over the evaluation set $\mathcal{D}$. The four rule-based outcome metrics align with the capabilities *GroupTravelBench* targets—PC measures *elicitation*, GU and GF measure *coordination* into a group-aware plan, and PV measures the feasibility floor that all of the above must respect—while S^{LJ} supplies process-level signals that no deterministic check can capture. Because GU cannot be normalized without access to the optimal achievable utility, we additionally report GU as a standalone metric and the average of the other four metrics.

5 Experiments

5.1 Experimental Setup

Models. We report results on nine LLMs spanning both frontier and lightweight models. The *frontier models* include GPT-5.1 (OpenAI, 2025), DeepSeek-V4-Pro (DeepSeek, 2026), and Qwen3.6-Max-Preview (abbreviated as Qwen3.6-Max in tables and figures) (Yang et al., 2025a), while the *lightweight models* mainly consist of Qwen reasoning models with different parameter sizes. We find that some models, especially smaller instruct models, struggle to complete this task. A detailed discussion is provided in Appendix C.5.

Other Details. Each model is run three times per task ($T = 3$) on all 650 tasks, using an agent sampling temperature of 0.7. We report the mean score across trials. DeepSeek-V4-Flash serves as the user simulator, GPT-4.1 as the tool simulator, and Gemini-3-Flash-Preview performs process evaluation and meta-judge calibration; all three use temperature 0 to improve reproducibility. We adopt difficulty-aware interaction budgets by scaling the maximum number of interaction rounds and convergence intervals according to task difficulty: Easy (15 rounds, $\kappa = 3$), Medium (20 rounds, $\kappa = 4$), and Hard (25 rounds, $\kappa = 5$). More details are provided in Appendix C.1.

5.2 Main Results

Table 1: Main results on *GroupTravelBench* across models and difficulty levels. Judge denotes the average score assigned by the LLM-judge, and Other denotes the average of PC, GF, PV, and Judge. For Qwen3 variants, Th denotes the thinking mode and It denotes the instruction-following variant. GPT-5.1 is evaluated in instruct mode, whereas the other models are used with reasoning enabled. Best results in each column are shown in **bold**.

Model	All (650)					Easy (200)		Medium (250)		Hard (200)	
	GU	PC	GF	PV	Judge	GU	Other	GU	Other	GU	Other
Frontier Models											
DeepSeek-V4-Pro	**10.47**	**64.0**	**54.2**	8.0	**92.3**	**9.46**	**62.2**	**11.35**	**53.5**	**10.36**	**48.1**
GPT-5.1	7.21	31.0	48.1	1.0	85.2	6.35	46.8	7.44	40.8	7.78	36.5
Qwen3.6-Max	9.66	31.0	52.1	7.0	57.3	8.54	43.4	10.03	36.7	10.33	31.5
Lightweight Models											
Qwen3.5-Plus	8.63	30.0	44.9	**12.0**	62.2	7.57	42.0	9.02	38.0	9.20	32.4
Qwen3.5-Flash	6.97	21.0	41.9	1.0	43.7	6.20	31.4	7.05	26.7	7.65	23.3
Qwen3-235B-Th	7.35	26.0	49.0	3.0	45.4	6.47	37.9	7.40	29.9	8.15	25.7
Qwen3-30B-Th	6.20	18.0	48.6	1.0	21.3	5.27	27.6	6.26	21.4	7.07	18.4
Qwen3-4B-Th	5.72	22.0	47.6	0.0	24.8	4.89	29.7	5.64	22.3	6.64	19.3
Qwen3-4B-It	6.12	18.0	41.7	1.0	22.9	5.22	26.1	6.05	20.1	7.10	16.9

Table 1 reports five evaluation dimensions on the full benchmark and per-difficulty breakdowns. We have some findings in this table:

(1) DeepSeek-V4-Pro outperforms all other models on almost every dimension, especially on Preference Completeness (64.0% vs. $\leq$31% for all others), indicating that the model tends to obtain preference information through multi-turn interactions.

(2) Group Fairness remains below 55% for *all* models, revealing a shared inability to distribute utility equitably across group members, likely because agents either fail to elicit or overlook some users' preferences. Plan Validity is alarmingly low ($\leq$12%), indicating that most models generate invalid or incomplete itineraries.

(3) GPT-5.1 achieves the second-highest LLM-Judge score (85.2%) while ranking only fifth on GU, suggesting that strong interaction quality does not necessarily translate into better outcomes and, to some extent, that the evaluation dimensions are orthogonal.

(4) The task is extremely challenging for smaller instruction-tuned models: most open-source instruction-tuned models we tested (Appendix C.5) fail to produce well-formed tool calls as the interaction sequence becomes longer, whereas GPT-5.1 succeeds despite also being an instruction-following model. This result suggests that sufficient model capacity is a prerequisite for reliable multi-turn agentic behavior in this setting.

LLM-Judge sub-dimension profile. Figure 2 compares four models across the five process-quality dimensions. DS-V4-Pro leads uniformly; GPT-5.1 nearly matches it on Interaction and Conflict but falls behind on Hallucination (77.3 vs. 86.8) and Humanization (76.9 vs. 90.0). This breakdown clearly highlights where each model excels and where it remains limited.

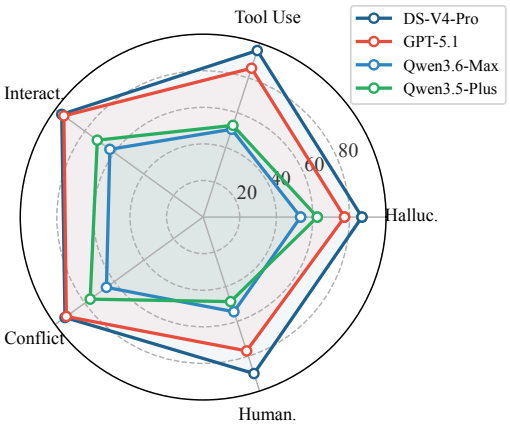


Figure 2: LLM-Judge sub-dimension profiles. DeepSeek-V4-Pro leads uniformly; GPT-5.1 excels on interaction/conflict but lags on hallucination and humanization.

Bottleneck localization. In the standard end-to-end evaluation, we require the agent to proactively query users to collect their preferences before planning the trip. However, many models score low on PC, i.e., they have weak elicitation ability, which may confound the assessment of their coordination and planning abilities. We therefore provide an oracle-preference experiment in Appendix C.2, where the agent plans with the complete preference tables given. The results show that providing the full preference tables indeed improves the other dimensions, but the overall scores remain relatively low. This suggests that elicitation is indeed part of the bottleneck, but the more important limitation lies in the models' own coordination and planning abilities.

5.3 Reliability

Cross-evaluation stability. We repeat the complete evaluation three times, with each repetition itself averaging $T = 3$ trajectories per task. Table 2 shows that results are stable across these repetitions: the standard deviation is at most 0.17 for Group Utility and at most 0.30 for Other. We attribute the remaining variation primarily to stochasticity introduced by the agent sampling temperature (0.7). Overall, these small deviations indicate that both our inference pipeline and evaluation framework are stable and reproducible.

Table 2: Cross-evaluation stability (mean $\pm$ std over three complete benchmark repetitions; each repetition uses $T = 3$ trajectories per task).

Model	GU	Other (%)
GPT-5.1	7.20 $\pm$ 0.02	41.04 $\pm$ 0.21
DeepSeek-V4-Pro	10.47 $\pm$ 0.05	54.82 $\pm$ 0.30
Qwen3.5-Flash	7.02 $\pm$ 0.05	27.01 $\pm$ 0.10
Qwen3.5-Plus	8.76 $\pm$ 0.17	37.31 $\pm$ 0.22

Table 3: Offline (isolated sandbox) vs. online (live API) comparison. *Hit* is cache hit rate and *Tools* the average number of tool calls. Aggregate scores and tool-use statistics remain close across modes.

Model	GU		Other (%)		Hit (%)		Tools	
	Off.	On.	Off.	On.	Off.	On.	Off.	On.
GPT-5.1	7.21	7.14	41.33	40.78	37.7	36.9	12.29	12.59
DeepSeek-V4-Pro	10.47	10.54	54.62	54.45	66.8	67.2	33.83	34.88
Qwen3.5-Flash	6.97	6.92	26.88	27.11	62.6	61.5	20.32	20.29
Qwen3.6-Max	9.66	9.69	36.84	37.26	61.7	62.3	20.00	19.93

Offline/online consistency. Table 3 compares `ISOLATED` (cache + LLM simulator) and `ONLINE` (live API) modes across four models. Score differences are small in absolute terms: $|\Delta\text{GU}| \leq 0.07$ and $|\Delta\text{Other}| \leq 0.55$. Cache hit rates and tool call counts also remain close between modes ($|\Delta\text{Hit}| \leq 1.1$ percentage points), showing that cache-miss simulation preserves aggregate behavior relative to live APIs.

LLM-Judge score distribution. Figure 3 shows the per-dimension rating distribution for four models spanning the performance spectrum. Top-performing models are concentrated at scores 4–5, with DS-V4-Pro achieving score-5 rates above 70% on Tool Use, Interaction, and Conflict. In contrast, lower-performing models show markedly different patterns: Qwen3.5-Flash peaks at score 2 on Hallucination, indicating pervasive fabrication, while Qwen3-30B clusters at scores 1–2 on most dimensions except Conflict. We further validate that the LLM-judge scores align with human preferences on 100 sampled trajectories (Appendix C.10).

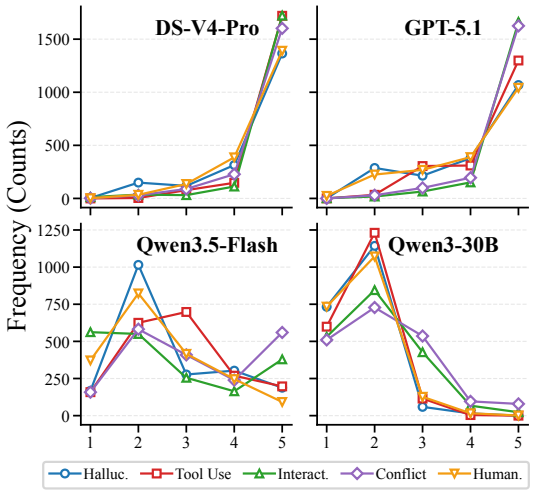


Figure 3: LLM-Judge score distribution per dimension for four models. Top models cluster at 4–5; bottom models show dimension-dependent failure patterns.

Simulator reliability. To verify the reliability of the user and tool simulators, we conduct a human audit and a simulator-replacement experiment. Three researchers independently audited 327 user responses from 20 trajectories, with majority vote used for each response; 98.4% complied with the assigned preference table and behavior contract. Replacing either the user or the tool simulator with another model changes GU by at most 0.08 and Other by at most 0.85 points. This indicates that the user and tool simulators are reliable within our interaction framework. Further details are provided in Appendix C.3 and C.4.

6 Conclusion

We introduce **GroupTravelBench**, the first benchmark for multi-user, multi-turn travel planning with LLM agents. The benchmark synthesizes 650 tasks across 22 group archetypes and three difficulty levels, grounded in 3,718 real user profiles, ∼338K real POIs, and real-world transportation prices; each task runs in a controlled group-chat framework backed by over 250K cached real-world tool records for stable offline evaluation. By coupling structured preference tables with this sandbox, GroupTravelBench evaluates three capabilities that single-user benchmarks cannot test: *elicitation* of private preferences, *coordination* of conflicts via compromise or subgrouping, and group-aware *planning* that balances utility and fairness. We pair the benchmark with a complementary evaluation framework combining four rule-based outcome metrics with a meta-judge-calibrated LLM judge for process quality. Across a wide range of experiments, even the strongest agents fall short across all four rule-based outcome metrics—plan validity is at most 12% across all models—indicating that robust multi-party travel planning remains far from solved and pointing to a clear training and evaluation surface for future LLM agents.

AI USE STATEMENT

Generative AI was used in this work for three purposes. (i) Data synthesis: generating the structured synthetic preference tables and the users' opening utterances from anonymized user-profile summaries and grounded travel data. (ii) Evaluation: serving as the user simulator, the tool simulator for cache-miss responses, and the LLM judge for process evaluation and meta-judge calibration. (iii) Implementation and writing: assisting with code development and debugging, and with language polishing. We did not use generative AI to develop theoretical models or conceptual frameworks, propose or refine hypotheses, design experiments or the evaluation protocol, clean or reformat the source data, formulate mathematical claims, or interpret results; proving mathematical claims is not applicable to this work.

All AI-assisted artifacts were reviewed or validated by the authors. Generated preference tables passed automatic checks for POI existence, cross-tier consistency, and transportation reachability. Three researchers independently audited 327 user-simulator responses, with majority vote determining correctness. Tool simulation was checked against live APIs, and LLM-judge scores were compared with ratings from three researchers on 100 trajectories. AI-assisted code was reviewed and tested by the authors, and all polished text was manually verified. The authors take full responsibility for the final text, claims, code, data, and other artifacts in this work.

ETHICS STATEMENT

The source profiles were accessed and processed under internal legal, privacy, and organizational compliance procedures. Benchmark construction uses anonymized profile summaries rather than direct identifiers. The preference representations prepared for release contain travel-related constraints and interests, not raw profiles or personally identifying information. Before release, all artifacts undergo a final privacy and license-compliance review; only data and derived artifacts that can be legally redistributed will be published.

Role-conditioned preference generation and group archetypes may reproduce demographic or social stereotypes present in source data or generative models. We mitigate this risk by restricting generated content to a predefined travel-preference schema, grounding named entities in real POI inventories, validating generated tables, and treating archetypes as evaluation templates rather than normative descriptions of social groups. Finally, the benchmark reveals that current agents often produce infeasible plans. Its outputs are research artifacts and should not be used for real travel decisions without independent verification of schedules, prices, accessibility, safety, and local requirements.

REPRODUCIBILITY STATEMENT

We document the task schema, preference-generation procedure, interaction and compromise protocols, deterministic metrics, tool-cache implementation, model settings, and complete prompts in Appendices A–D. We report three independent trials per task, cross-trial variation, live-API comparisons, human validation, and simulator-replacement checks. The anonymous repository link containing the complete source code and data is provided in the abstract.

REFERENCES

Mohammed Alatiyyah. A novel group tour trip recommender model for personalized travel systems. *PeerJ Computer Science*, 11:e2589, 2025.

Xiang Cheng, Yulan Hu, Xiangwen Zhang, Lu Xu, Lide Tan, Zheng Pan, Xin Li, and Yong Liu. Beyond itinerary planning—a real-world benchmark for multi-turn and tool-using travel tasks. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29200–29251, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1347. URL https://aclanthology.org/2026.acl-long.1347/.

DeepSeek. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf, Apr 2026. Accessed: 2026-04-24.

Bin Deng, Yizhe Feng, Zeming Liu, Qing Wei, Xiangrong Zhu, Shuai Chen, Yuanfang Guo, and Yunhong Wang. Retail: Towards real-world travel planning for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14881–14913, 2025.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*, 2025.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

Google. A New Era of Intelligence with Gemini 3. https://blog.google/products/gemini/gemini-3/, dec 2025. Accessed: 2025-12-18.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11143–11156, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.664. URL https://aclanthology.org/2024.findings-acl.664/.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. Large language models can solve real-world planning rigorously with formal verification tools. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3434–3483, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.176. URL https://aclanthology.org/2025.naacl-long.176/.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. LLM-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13806–13834, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.745. URL https://aclanthology.org/2024.acl-long.745/.

Farzaneh Jouyandeh and Pooya Moradian Zadeh. Personalized group itinerary recommendation using a knowledge-based evolutionary approach. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, pp. 1684–1692, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701207. doi: 10.1145/3583133.3596345. URL https://doi.org/10.1145/3583133.3596345.

Priyanshu Karmakar, Soumyabrata Chaudhuri, Shubhojit Mallick, Manish Gupta, Abhik Jana, and Shreya Ghosh. Triptide: A benchmark for adaptive travel planning under disruptions. *arXiv preprint arXiv:2510.21329*, 2025.

Krisdahim Kogoya, Javed Hossain, Yuetong Sun, and Peilan Xu. Automatic group itinerary planning: An evolutionary multi-party multi-objective approach from game perspective. In *2026 IEEE International Conference on Emerging Computing and Intelligent Technologies (ICoECIT)*, pp. 1–6. IEEE, 2026.

Zhixue Liao, Weimin Zheng, and Zhibin Lin. Time apart while together: A smart trip design for group travelers. *Annals of Tourism Research*, 93:103374, 2022. ISSN 0160-7383. doi: https://doi.org/10.1016/j.annals.2022.103374. URL https://www.sciencedirect.com/science/article/pii/S0160738322000251.

Junhua Liu, Aldy Gunawan, Kristin L. Wood, and Kwan Hui Lim. Optimizing group utility in itinerary planning: a strategic and crowd-aware approach. *Journal of Big Data*, 12(1):201, 2025. doi: 10.1186/s40537-025-01249-9. URL https://doi.org/10.1186/s40537-025-01249-9.

MiniMax. MiniMax M2 & Agent: Great Skill Concealed in Simplicity. https://www.minimaxi.com/en/news/minimax-m2, oct 2025. Accessed: 2025-10-27.

Hang Ni, Fan Liu, Xinyu Ma, Lixin Su, Shuaiqiang Wang, Dawei Yin, Hui Xiong, and Hao Liu. TP-RAG: Benchmarking retrieval-augmented large language model agents for spatiotemporal-aware travel planning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12403–12429, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.626. URL https://aclanthology.org/2025.emnlp-main.626/.

Yansong Ning, Rui Liu, Jun Wang, Kai Chen, Wei Li, Jun Fang, Kan Zheng, Naiqiang Tan, and Hao Liu. Deeptravel: An end-to-end agentic reinforcement learning framework for autonomous travel planning agents. *arXiv preprint arXiv:2509.21842*, 2025.

OpenAI. GPT-5.1 is Now Available: A Smarter and More Conversational ChatGPT. https://openai.com/index/gpt-5-1/, nov 2025. Accessed: 2025-11-12.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=eOLdGbXT6t.

Tian Qin, Felix Bai, Ting-Yao Hu, Raviteja Vemulapalli, Hema Swetha Koppula, Zhiyang Xu, Bowen Jin, Mert Cemri, Jiarui Lu, Zirui Wang, and Meng Cao. Compass: A multi-turn benchmark for tool-mediated planning & preference optimization. *ArXiv*, abs/2510.07043, 2025. URL https://api.semanticscholar.org/CorpusID:281892403.

Yincen Qu, Huan Xiao, Feng Li, Gregory Li, Hui Zhou, Xiangying Dai, and Xiaoru Dai. Tripscore: Benchmarking and rewarding real-world travel planning with fine-grained evaluation. *arXiv preprint arXiv:2510.09011*, 2025.

José Ruiz-Meza, Julio Brito, and Jairo R. Montoya-Torres. A grasp-vnd algorithm to solve the multi-objective fuzzy and sustainable tourist trip design problem for groups. *Applied Soft Computing*, 131: 109716, 2022. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2022.109716. URL https://www.sciencedirect.com/science/article/pii/S1568494622007657.

Jie-Jing Shao, Bo-Wen Zhang, Xiao-Wen Yang, Baizhi Chen, Siyu Han, Pang Jinghao, Wen-Da Wei, Guohao Cai, Zhenhua Dong, Lan-Zhe Guo, and Yu-Feng Li. Chinatravel: An open-ended travel planning benchmark with compositional constraint validation for language agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0YRVlxY9BH.

Zijian Shao, Jiancan Wu, Weijian Chen, and Xiang Wang. Personal travel solver: A preference-driven LLM-solver system for travel planning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27622–27642, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1339. URL https://aclanthology.org/2025.acl-long.1339/.

Kadri Sylejmani, Jürgen Dorn, and Nysret Musliu. Planning the trip itinerary for tourist groups. *Information Technology & Tourism*, 17(3):275–314, 2017. doi: 10.1007/s40558-017-0080-9. URL https://doi.org/10.1007/s40558-017-0080-9.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang

Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Kaimin Wang, Yuanzhe Shen, Changze Lv, Xiaoqing Zheng, and Xuanjing Huang. TripTailor: A real-world benchmark for personalized travel planning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9705–9723, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.503. URL https://aclanthology.org/2025.findings-acl.503/.

Yinuo Wang, Mining Tan, Wenxiang Jiao, Xiaoxi Li, Hao Wang, Xuanyu Zhang, Yuan Lu, and Weiming Dong. Tourplanner: A competitive consensus framework with constraint-gated reinforcement learning for travel planning. *arXiv preprint arXiv:2601.04698*, 2026a.

Zexuan Wang, Chenghao Yang, Yingqi Que, Zhenzhu Yang, Huaqing Yuan, Yiwen Wang, Zhengxuan Jiang, Shengjie Fang, Zhenhe Wu, Zhaohui Wang, et al. Worldtravel: A realistic multimodal travel-planning benchmark with tightly coupled constraints. *arXiv preprint arXiv:2602.08367*, 2026b.

Zhensheng Wang, Wenmian Yang, Kun Zhou, Yiquan Zhang, and Weijia Jia. Retqa: A large-scale open-domain tabular question answering dataset for real estate sector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25452–25460, 2025b.

Zhensheng Wang, ZhanTeng Lin, Wenmian Yang, Kun Zhou, Yiquan Zhang, and Weijia Jia. ODUTQA-MDC: A task for open-domain underspecified tabular QA with multi-turn dialogue-based clarification. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10907–10928, San Diego, California, United States, July 2026c. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.499. URL https://aclanthology.org/2026.acl-long.499/.

Zhensheng Wang, Xiaole Liu, Wenmian Yang, Kun Zhou, Yiquan Zhang, and Weijia Jia. ODTQA-FoRe: An open-domain tabular question answering dataset for future data forecasting and reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 6997–7022, San Diego, California, United States, July 2026d. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.347. URL https://aclanthology.org/2026.findings-acl.347/.

Xixi Wu, Qianguo Sun, Ruiyang Zhang, Chao Song, Junlong Wu, Yiyan Qi, and Hong Cheng. Demystifying reinforcement learning for long-horizon tool-using agents: A comprehensive recipe. *arXiv preprint arXiv:2603.21972*, 2026.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=l5XQzNkAOe.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Dongjie Yang, Chengqiang Lu, Qimeng Wang, Xinbei Ma, Yan Gao, Yao Hu, and hai zhao. Wide-horizon thinking and simulation-based evaluation for real-world LLM planning with multifaceted constraints. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=b50IW9yV2M.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Duanyang Yuan, Sihang Zhou, Yanning Hou, Xiaoshu Chen, Haoyuan Chen, Ke Liang, Jiyuan Liu, Chuan Ma, Xinwang Liu, and Jian Huang. Decoupled travel planning with behavior forest. *arXiv preprint arXiv:2604.21354*, 2026.

Bo-Wen Zhang, Jin Ye, Peng-Yu Hua, Jia-Wei Cao, Jie-Jing Shao, Yu-Feng Li, and Lan-Zhe Guo. Revisiting the travel planning capabilities of large language models. *arXiv preprint arXiv:2605.03308*, 2026a.

Qiang Zhang, Boli Chen, Fanrui Zhang, Ruixue Ding, Shihang Wang, Qiuchen Wang, Yinfeng Huang, Haonan Zhang, Rongxiang Zhu, Pengyong Wang, Ailin Ren, Xin Li, Pengjun Xie, Jiawei Liu, Ning Guo, Jingren Zhou, and Zheng-Jun Zha. Arenarl: Scaling rl for open-ended agents via tournament-based relative ranking, 2026b. URL https://arxiv.org/abs/2601.06487.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. Ask-before-plan: Proactive language agents for real-world planning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10836–10863, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.636. URL https://aclanthology.org/2024.findings-emnlp.636/.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. Deepplanning: Benchmarking long-horizon agentic planning with verifiable constraints. *arXiv preprint arXiv:2601.18137*, 2026c.

A BENCHMARK DETAILS

This appendix documents the components that define the benchmark itself: how user preferences are structured, how groups are organized, how user simulators behave during interaction, what dimensions the LLM judge scores, and how the 650 benchmark tasks are distributed across groups, cities, time, and difficulty.

A.1 PREFERENCE SCHEMA

Each user's preference table $\mathbf{p}_i$ follows a hierarchical structure with two levels: *global constraints* (trip-wide) and *city-specific preferences* (per destination city). Table 4 provides the complete field inventory.

Table 4: Complete preference schema. *Tier* indicates the strength level used in Group Utility scoring. Strong tiers (must/reject) carry ± 2 weight; weak tiers (prefer/avoid) carry ± 1 weight.

Level	Field	Description	Type	Tier
Global	`avg_budget`	Per-person budget cap (CNY)	scalar	strong (-2)
	`transport.must`	Transport modes the user insists on	list	strong ($+2$)
	`transport.reject`	Transport modes the user refuses	list	strong (-2)
	`transport.prefer`	Transport modes mildly preferred	list	weak ($+1$)
	`transport.avoid`	Transport modes mildly avoided	list	weak (-1)
	`intensity.max_poi_per_day`	Max POIs per day before penalty	scalar	strong (-2)
	`intensity.max_active_hours`	Max active hours per day	scalar	strong (-2)
	`hotel_preference.prefer`	Hotel categories mildly preferred	list	weak ($+1$)
	`hotel_preference.avoid`	Hotel categories mildly avoided	list	weak (-1)
City-specific	`attractions.must_visit`	POI names the user must visit	list	strong ($+2$)
	`attractions.reject_visit`	POI names the user refuses to visit	list	strong (-2)
	`attractions.category_pref.positive`	Attraction categories preferred	list	weak ($+1$)
	`attractions.category_pref.negative`	Attraction categories disliked	list	weak (-1)
	`food.must_eat`	Foods/restaurants the user must try	list	strong ($+2$)
	`food.reject_eat`	Foods/restaurants the user refuses	list	strong (-2)
	`food.prefer_eat`	Foods the user mildly prefers	list	weak ($+1$)
	`food.avoid_eat`	Foods the user mildly avoids	list	weak (-1)

Design rationale: specific instances vs. category-level preferences. The choice of *specific instances vs. categories* per field reflects how people actually voice travel preferences. For **attractions** and **food**, strong tiers (`must_visit` / `reject_visit`, `must_eat` / `reject_eat`) take specific POI or dish names—users usually have a small set of places or foods they actively want or refuse ("I must see the Forbidden City"; "I refuse to eat hotpot"). Weak tiers (`prefer` / `avoid`), in contrast, are stated as broader categories (e.g., *museum* vs. *theme park*; *Cantonese* vs. *fast food*), because casual preferences typically come at the category level. For **hotels**, the two supported tiers (`prefer` / `avoid`) are category-level (*economy* / *comfort* / *business* / *luxury*) since users rarely name specific hotels in conversation. For **transportation**, all four tiers are restricted to a fixed mode set (flight, train, high-speed rail, self-driving). Finally, **budget** and **activity intensity** are scalar caps that bound the entire trip rather than enumerated lists. Together, these per-field choices let a single schema cover both *deliberate* preferences (a specific must-see) and *categorical* taste (a general dislike of certain food types), giving a comprehensive yet realistic picture of each user.

Strength tiers and user-simulator tone mapping. The four-tier system serves a dual purpose. In *evaluation*, it determines the scoring weight: strong-tier items (must, reject, budget cap, intensity cap) carry ± 2 points, while weak-tier items (prefer, avoid, category preferences, hotel preferences) carry ± 1 points. The asymmetry between positive and negative tiers is deliberate: for positive tiers (must/prefer), satisfying the preference earns credit but *not* satisfying it incurs no penalty; for negative tiers (reject/avoid), violating the preference incurs a penalty but *not* violating it earns no credit. This reflects the real-world semantics that fulfilling a wish is a bonus, while violating a taboo is a failure.

In *interaction*, the tier directly controls the user simulator's conversational tone. When expressing a `must` preference, the user speaks with a firm, non-negotiable tone (e.g., "I must take a flight"); when expressing a `prefer`, the tone is mild and flexible (e.g., "If possible, I would prefer to fly"). This creates a natural signal that the agent must learn to interpret: the strength of a preference is encoded in *how* the user says it, not in an explicit label. The user simulator is strictly forbidden from using field names (e.g., `must_visit`) in conversation.

A.2 Group Archetypes

We curate 22 social archetypes covering group sizes $N \in \{2, \ldots, 6\}$. Table 5 lists each archetype with its group size, member roles, social tags, and number of benchmark tasks. Each archetype is additionally annotated with:

- **Role slots**: named roles for each member (e.g., "boyfriend", "grandmother", "roommate A"), used to condition user-profile sampling and preference generation.
- **Social dynamics description**: a free-text paragraph (in Chinese) describing the typical decision-making dynamics, common sources of conflict, and who is most likely to compromise. These descriptions guide the LLM during preference synthesis and initial-message generation.
- **Compromise patterns**: an exhaustive enumeration of all plausible subsets of members who might agree to compromise in a given task. At synthesis time, one pattern is sampled per task to set each user's `compromisable` flag.

A.3 Data Field Definitions

Each task in `test.jsonl` is a JSON object containing the fields listed in Table 6.

A.4 Preference Generation Strategies

Design rationale: who can disagree on what. We generate within-group preferences under a single principle: **global constraints—budget, transport mode, activity intensity, and hotel class—are never regenerated within a related group**. This reflects how real groups actually disagree. Close-knit groups—couples, families, regular friend circles—almost never clash on whether to fly or take the train, or whether to book an economy or a comfort hotel; if they did, they typically would not be traveling together in the first place. They do, however, routinely disagree on which museum to visit or which restaurant to try. To realize this principle, we use three strategies—*Independent*, *Copy_minor*, and *Copy_moderate*—assigned by a per-archetype generation plan: tight-knit archetypes

Table 5: Distribution of the 22 social archetypes across group size in the 650-task benchmark. The planned release includes all role slots, tag annotations, and admissible compromise patterns.

N	**Archetype**	**Roles (abbreviated)**	**Tags**	**#Tasks**
2	Couple	M, F	romance	17
	Married couple	husband, wife	ritual	13
	Female friends	F-A, F-B	food, photo	24
	Male friends	M-A, M-B	outdoor, adventure	20
3	Couple + female friend	BF, GF, single-F	mixed	21
	Couple + male friend	BF, GF, single-M	mixed	18
	Three female friends	F-A, F-B, F-C	democratic	25
	Three male friends	M-A, M-B, M-C	free-form	23
	Family w/ toddler	father, mother, child (0–8)	parenting	16
	Family w/ school-age	father, mother, child (8–18)	parenting	18
4	Two couples	BF-A, GF-A, BF-B, GF-B	dual-CP	21
	Couple + two friends	BF, GF, F-A, F-B	mixed	17
	Four female friends	F-A–D	democratic	28
	Four male friends	M-A–D	adventure	19
	Family w/ two children	father, mother, teen, child	age-gap	21
	Couple + parents	husband, wife, father-in-law, mother-in-law	two-gen	20
5	Five female friends	F-A–E	large group	37
	Five male friends	M-A–E	large group	46
	Family + grandparents	father, mother, child, grandpa, grandma	three-gen	41
6	Three-gen large family	grandpa, grandma, father, mother, teen, child	complex	80
	College dorm	roommate A–F	familiar but diverse	53
	Three couples	BF/GF $\times$ 3	triple-CP	72

(e.g., couples, parent–child families) admit only city-level conflicts, while looser archetypes (e.g., college roommates, three couples) contain more *Independent* users and can additionally exhibit top-level disagreements. The resulting conflict distribution mirrors the kinds of disagreements observed in real group travel.

The three strategies (plus a *Skip* option for members without independent preferences) are detailed below:

1. **Independent**: A fresh LLM call generates the complete preference table, conditioned on the user's sampled real profile. Used for the first user in each group, producing a fully original preference set.
2. **Copy_minor**: The global constraints and city-level category preferences are inherited from a reference user (typically the partner or close companion). Only the fine-grained name-level preferences (must_visit, reject_visit, must_eat, reject_eat) are regenerated by the LLM. This simulates couples or close friends who share broad preferences but differ on specific POIs and restaurants.
3. **Copy_moderate**: Only the top-level global constraints (budget, transport mode, intensity, hotel class) are inherited. All city-specific preferences and meso-level category preferences are regenerated. This simulates friends who share a similar travel style but have distinct interests.

Table 6: Field definitions for a *GroupTravelBench* task instance.

Field	Description
task_id	A unique identifier for each task (e.g., `task_004329`).
query	A coarse natural-language travel request specifying destination cities, trip duration, and group size. This is the only information visible to the agent at the start of the conversation.
time	The departure date, sampled uniformly between 2025-09-01 and 2026-05-01.
context	Background context including departure city, travel dates, and group type description. Provided to both the agent and user simulators.
metadata	A structured object containing: departure_city, cities (list of 1–3 destinations), date (trip duration), group_id, group_name, group_tags, compromise_pattern (list of compromisable user IDs), and child_members.
user_preferences	A dictionary mapping each user ID to their profile: role (e.g., "boyfriend", "grandmother"), trace_id (anonymized real profile ID), preference (hierarchical table; see §A.1), and compromisable (boolean flag).
initial_messages	A list of pre-generated opening statements, one per user. Each entry contains role (user ID) and content (the statement text). Broadcast as Phase 1 of the interaction.
difficulty_score	A real-valued score: $0.5 \times f_N + 0.3 \times f_D + 0.2 \times f_C$, each dimension mapped to a 1–5 ordinal scale.
difficulty_type	One of `easy` ($s \leq 2.8$), `medium` ($2.8 < s < 4.2$), or `hard` ($s \geq 4.2$).

4. **Skip**: For members who do not carry independent preferences (e.g., toddlers aged 0–8), no preference table is generated. These members still appear in the group and affect the `participants` field of the plan but do not contribute to utility scoring.

After generation, each preference table undergoes three validation passes:

- **POI name validation**: Every must_visit and must_eat item is checked against the city's real POI inventory ($\sim$338K entries). Invalid names trigger regeneration (up to 3 retries).
- **Cross-tier contradiction removal**: Items appearing in mutually exclusive tiers (e.g., the same POI in both must_visit and reject_visit) are deduplicated by keeping the stronger tier.
- **Transport reachability check**: If any user's must/prefer transport mode has no available service on the sampled route, the user is force-flagged as compromisable ($\kappa_i = 1$).

A.5 User Simulator

Recent benchmarks are shifting from static question answering to dynamic multi-turn interaction: fully specified tabular queries (Wang et al., 2025b; 2026d) give way to clarification against a user simulator (Wang et al., 2026c), and we likewise withhold each user's preferences so that they surface only through multi-turn dialogue.

Each user simulator is an LLM instance with a role-conditioned system prompt that embeds:

1. The user's **role** within the group (e.g., "You are Boyfriend A").
2. The user's **complete preference table** in structured format.
3. A **compromise block** that varies based on the `compromisable` flag: compromisable users are instructed to "tend to agree when the agent asks for a compromise, and append a machine-readable marker"; non-compromisable users are instructed to "politely but firmly decline compromise requests on strong-tier preferences."
4. **Five behavioral scenarios** defining when the user should speak: (A) when @-mentioned by the agent, (B) when a conflict with their strong preferences is detected, (C) to ask the agent factual questions, (D) to respond to a compromise request, (E) to acknowledge the agent's answer to a previous question. In all other situations, the user emits `[pass]`.

5. **Strict prohibitions**: no spontaneous preference disclosure, no planning suggestions, no @-mentioning other users, no pretending to have tool access, no repetition of previously stated preferences, no idle chat, and no answering on behalf of others.
6. **Tone mapping rules**: must = firm and non-negotiable, reject = firm refusal, prefer = mild suggestion, avoid = mild discomfort. Detailed guidelines distinguish avoid ("I'd rather not, but I can live with it") from reject ("absolutely not").

The user simulator is deployed at temperature 0 with a strong instruction-following LLM to minimize behavioral variance. It sees only *outward* messages (other participants' visible messages); the agent's internal reasoning, tool calls, and tool responses are hidden, just as a real user in a group chat would only see the messages, not the agent's thought process.

A.6 Interaction Protocol Details

Max turn and convergence settings by difficulty.

	Easy	Medium	Hard
Max turns	15	20	25
Convergence interval (κ)	3	4	5

Compromise protocol mechanics.

1. The agent identifies a conflict and @-mentions the relevant user (e.g., `@User2 Would you accept switching to high-speed rail?`).
2. The scheduler grants the @-mentioned user immediate speaking priority.
3. If the user is compromisable ($\kappa_i > 0$), the simulator replies with natural-language agreement plus a trailing machine-readable marker: `[transport.must : ["high-speed rail"]]`.
4. The framework validates that the preceding message was indeed the agent @-mentioning this user.
5. On validation success: the marker is parsed, the modification is applied to $\mathbf{p}_i^{\text{eff}}$ via dotted-path traversal, the marker is stripped from the visible message, κ_i is decremented, and if κ_i reaches 0, the user's system prompt is rebuilt to mark them as non-compromisable.
6. On validation failure (e.g., the marker refers to a nonexistent field path, or the preceding message was not from the agent): the user's response is regenerated (up to 3 retries).

Maximum compromise quota. Each user can agree to at most $K = 2$ compromises. After 2 accepted compromises, the user becomes non-compromisable for the remainder of the conversation, regardless of the original `compromisable` flag. This cap prevents degenerate strategies where the agent resolves all conflicts by asking the same amenable user to yield on everything.

Termination conditions. The conversation ends when any of the following conditions is met:

1. The agent emits a structurally valid travel plan JSON (immediate termination, no revision loop).
2. The maximum number of turns is reached (the framework force-injects a final-plan instruction and the agent must generate a plan within 3 attempts).
3. A safety guard is triggered: (a) a user persistently emits `[pass]` after being @-mentioned (mention exhaustion), (b) the same (tool_name, arguments) tuple is called ≥ 3 times across iterations or appears in two consecutive agent turns (repetitive tool-call termination), or (c) the per-round event cap ($5 \times |\text{polling_order}|$) is breached (runaway prevention).

Subgrouping mechanics. When the agent decides to subgroup, the group is partitioned into K_e subgroups with disjoint membership at split event e. Subgrouping is restricted to intra-city activities—attractions, dining, and local transportation—while shared resources remain unsplit: hotels (by night) and inter-city transportation are required to be common across all members. This restriction reflects practical group travel: members who diverge during the day still share a hotel and travel together to the next city. Each split event incurs a penalty of $K_e - 1$ in the GU computation (§4).

A.7 LLM-JUDGE EVALUATION DIMENSIONS

Table 7 defines the five process-quality dimensions scored by the LLM judge. Each dimension is rated on a 1–5 ordinal scale with explicit rubric anchors. The judge is instructed to quote specific dialogue turns as evidence for each rating.

Table 7: LLM-Judge evaluation dimensions. Each dimension targets a distinct aspect of process quality that is not covered by rule-based metrics. The judge is explicitly forbidden from scoring format, time arithmetic, or numeric preference matching (handled by rules).

Dimension	**Evaluation Criteria**
Hallucination	Is every price, place name, and time traceable to a tool return? Does the agent fabricate information or present uncertain facts as definitive? Does it appropriately express uncertainty?
Tool-Use Reasoning	Is the tool-call pattern staged and purposeful? Are calls redundant or missing? Are tool results correctly utilized in subsequent reasoning and responses?
Interaction Quality	Are @-mentions used correctly and precisely? Are questions targeted, natural, and free of field-name leakage? Is dialogue pacing appropriate (not too aggressive, not too passive)?
Conflict Coordination	Does the agent proactively identify preference conflicts? Is the mediation strategy explicit and equitable? Are quieter or weaker-voiced users given fair consideration?
Plan Humanization	Does the daily rhythm respect meals and rest? Is there diversity across days? Are special group needs accommodated (elderly mobility, child-friendly activities)? Is the route efficient with local flavor?

A.8 PLAN VALIDITY CHECK LIST

The PV metric is computed by passing each generated plan through nine deterministic checks. A plan is counted as valid (PV $= 1$) only if it passes all of them; otherwise PV $= 0$.

1. **Inter-city transportation**: every required inter-city leg is covered by a flight or train segment with a feasible schedule.
2. **Hotel coverage**: every overnight stay in a destination city is matched by a hotel booking; no night is left uncovered or double-booked.
3. **Temporal consistency**: every event has a valid interval, with its start time preceding its end time.
4. **Activity overlap**: no scheduled activity (attraction visit, meal, lodging) overlaps in time with another for the same participant.
5. **Opening-hour compliance**: every attraction or restaurant is visited within its declared opening hours.
6. **Local transportation continuity**: between consecutive intra-city activities, a feasible local-transport segment exists (walking, taxi, metro, etc.).
7. **Day-level temporal monotonicity**: events within each day are in non-decreasing temporal order.
8. **Cost completeness**: each scheduled item carries a cost field consistent with the corresponding tool return.
9. **Participant validity**: every `participants` field references group members declared at task start; preschool members are not assigned tasks beyond physically reasonable activities.

Table 8: Cross-tabulation of key task dimensions by difficulty level.

	Easy (200)	Medium (250)	Hard (200)
Group size (N)			
2	65	9	0
3	77	44	0
4	42	84	0
5	16	33	75
6	0	80	125
City count			
1	116	43	0
2	82	131	48
3	2	76	152
Duration			
2–3 days	144	66	0
4–5 days	56	134	109
6–7 days	0	50	91

A.9 Dataset Statistics

The remaining subsections summarize the geographic, group-size, preference, and temporal distributions of the 650 benchmark tasks.

A.9.1 Geographic Distribution

The benchmark covers 143 destination cities and 32 departure cities across China. Cities are organized into a three-tier weighting system for sampling:

- **Tier-1 cities** ($\times 3$ weight, 19 cities): Beijing, Shanghai, Guangzhou, Shenzhen, Hangzhou, Chengdu, Chongqing, Wuhan, Suzhou, Xi'an, Nanjing, Changsha, Tianjin, Zhengzhou, Dongguan, Qingdao, Kunming, Ningbo, Foshan.
- **Popular tourist cities** ($\times 2$ weight, $\sim$55 cities): e.g., Xiamen, Guilin, Lijiang, Sanya, Huangshan, Dali, Luoyang.
- **General tourist cities** ($\times 1$ weight, $\sim$80 cities): remaining cities with notable tourist resources.

Destination cities are further organized into 175 geographically coherent multi-city clusters (derived from 10 regional tourism clusters: North China, Yangtze River Delta, Pearl River Delta, Southwest, etc.), ensuring that multi-city trips follow realistic travel routes rather than arbitrary city combinations.

The 32 departure cities correspond to provincial capitals and major transportation hubs, ensuring realistic intercity connectivity. The most frequent departure cities include Guiyang (31), Hefei (31), Shanghai (28), Fuzhou (26), and Changsha (26). Figure 4 shows the top-20 most frequent destination cities, reflecting the tier-weighted sampling strategy: Tier-1 cities dominate due to their $\times 3$ weight, while popular tourist cities form the long tail.

A.9.2 Group Size and Difficulty Distribution

Table 8 shows the cross-tabulation of group size, city count, and trip duration against difficulty level. The difficulty score is computed as a weighted combination:

$$s = 0.5 \times f_N(N) + 0.3 \times f_D(D) + 0.2 \times f_C(C), \tag{6}$$

where f_N, f_D, f_C map group size N, duration D (in days), and city count C to $\{1, \ldots, 5\}$ ordinal scales respectively.

As shown, easy tasks are dominated by 2–3 person groups visiting a single city for 2–3 days, while hard tasks exclusively feature 5–6 person groups across 2–3 cities over 4–7 days. This design ensures that difficulty scales primarily along the *coordination axis*: larger groups produce more pairwise preference conflicts, more cities increase planning complexity, and longer durations expand the constraint space.

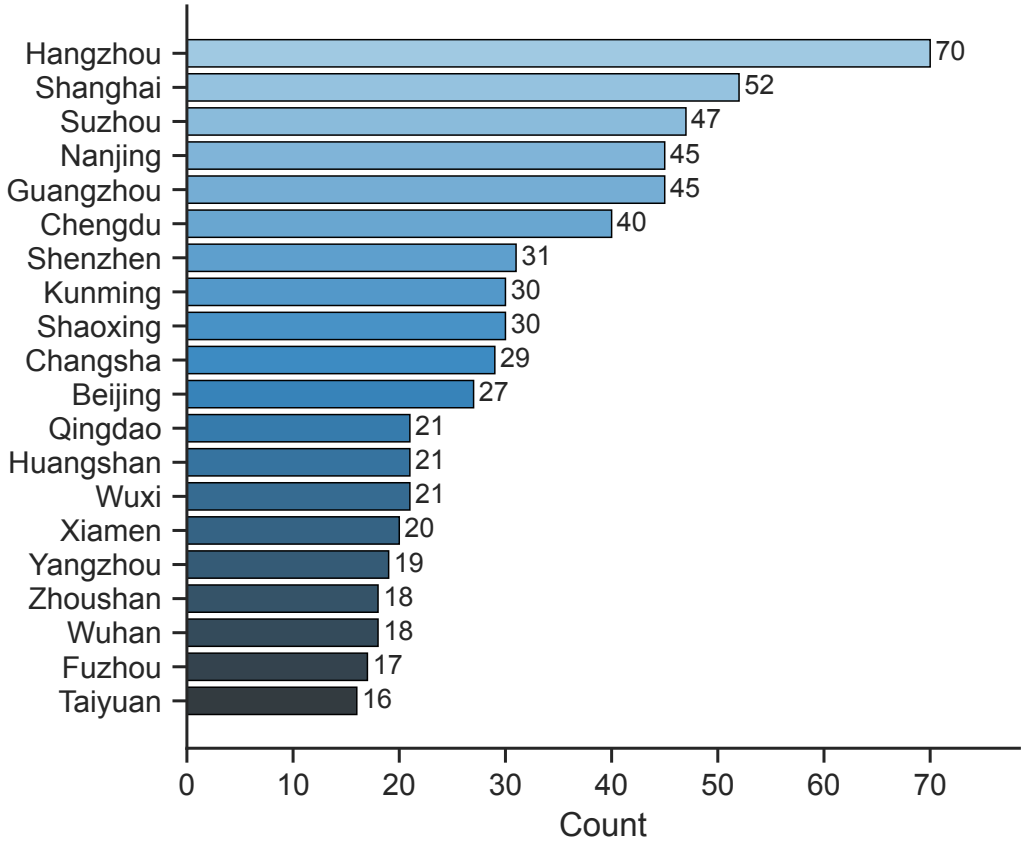


Figure 4: Top-20 destination city frequency in the 650-task benchmark.

A.9.3 Preference Distribution

Table 9 summarizes the distribution of preference items across the 2,748 users in the benchmark.

Every user carries both a budget cap and an intensity cap, ensuring that every task has quantitative constraints the agent must discover and satisfy. Transport preferences are roughly balanced between positive (must: 2,184, prefer: 2,230) and avoidance (avoid: 2,172) tiers, with hard rejections (reject: 182) being deliberately rare to avoid trivially infeasible tasks. City-specific preferences are richer: an average user has $\sim$2.8 must-visit attractions, $\sim$0.9 reject-visit attractions, $\sim$2.6 positive categories, and $\sim$1.6 negative categories per city, plus $\sim$2.2 must-eat and $\sim$2.2 prefer-eat food items.

Intra-group preference conflicts. We measure conflict density at the *all-tier* level: a conflict is recorded whenever the same item appears in one user's positive tier (`must` / `prefer` / `category_pref.positive`) and another user's negative tier (`reject` / `avoid` / `category_pref.negative`), within the same task and the same city when applicable. Conflicts are counted at the item level across all directed user pairs. Under this definition, 503 / 650 tasks (77.4%) contain at least one preference conflict, with an overall mean of 7.52 conflicts per task (median 4, max 54). Broken down by conflict source:

- **Food**: 365 / 650 tasks (56.2%) – the most common source, reflecting the breadth of must/prefer/avoid/reject options in the food schema.
- **Attraction category**: 282 / 650 tasks (43.4%) – positive vs. negative category-level disagreements.
- **Attraction name**: 215 / 650 tasks (33.1%) – direct `must_visit` vs. `reject_visit` clashes on specific POIs.
- **Transport**: 165 / 650 tasks (25.4%) – positive vs. negative tier disagreements on transport mode.

Table 9: Preference field statistics across all 2,748 users.

Statistic	Value
Global constraints	
Budget constraint coverage	100%
Intensity constraint coverage	100%
Transport must / prefer / avoid / reject	2,184 / 2,230 / 2,172 / 182
Hotel prefer / avoid	5,469 / 2,741
City-specific preferences	
Attractions: must_visit / reject_visit	7,699 / 2,501
Attractions: category pos. / neg.	7,195 / 4,460
Food: must / prefer / avoid / reject	5,945 / 5,940 / 3,896 / 2,140
Aggregates	
Total atomic preference points	62,998
Average per user	22.9
Compromisable users	54.2% (1,490 / 2,748)
Budget range (per person)	550–10,500 CNY
Budget mean / median	3,265 / 3,000 CNY

- **Hotel**: 151 / 650 tasks (23.2%) – `prefer` vs. `avoid` clashes on hotel categories.

Conflict density scales sharply with difficulty: easy tasks average 1.82 conflicts (48.5% with any conflict), medium tasks average 6.52 (84.0%), and hard tasks average 14.46 (98.0%). The left panel of Figure 7 shows the full per-task distribution: most tasks fall below 10 conflicts but a heavy right tail extends past 20, ensuring substantial coverage for evaluating conflict-discovery and coordination abilities. Hard-tier clashes (`must` vs. `reject`) remain deliberately rare on transport (5 / 650, 0.8%) so as not to render tasks infeasible.

A.9.4 Temporal Distribution

Departure dates span the period from September 1, 2025 to May 1, 2026, covering:

- National Day holiday (Oct 1–7, 2025)
- Winter holiday and Spring Festival season (Jan–Feb 2026)
- Qingming, Labor Day (Apr–May 2026)
- Regular weekdays and weekends throughout the period

This temporal range ensures that opening-hours checks, weather queries, and seasonal pricing in the sandbox reflect realistic variation, providing additional test surface beyond static preference matching.

B Sandbox Tools and Cache

To support stable, reproducible offline evaluation, the agent interacts with a sandboxed tool environment rather than live APIs. This appendix documents the tool library, the cache that backs it, the fallback mechanism used on cache misses, the call schema seen by the model, and the empirical distribution of cached entries across tools.

B.1 Tool Library

The agent has access to 10 real, production-grade travel tools organized into four domains. Table 10 provides an overview, and detailed descriptions follow.

Table 10: Overview of the tool library used in our benchmark sandbox, grouped by domain.

Domain	Tool name	Function
Maps & navigation	`search_poi`	Retrieve POIs by keyword, coordinates, or area
	`get_poi_detail`	Retrieve detailed POI information by ID (hours, reviews, pricing)
	`maps_geo`	Geocode an address or place name to coordinates
	`plan_route`	Plan routes across modes from origin to destination
	`compare_routes`	Compare route options across different transport modes
	`search_along_route`	Search POIs within a corridor along a given route
Transportation	`travel_search_flights`	Search China domestic flights with flexible dates
	`travel_search_trains`	Search China train and high-speed rail trips with flexible dates
Weather	`maps_weather`	Return current weather and multi-day forecasts
General information	`web_search`	Perform open-domain web search

Maps & Navigation Tools.

1. `search_poi`: A large-coverage POI retrieval tool supporting nationwide search across China. It accepts keywords, categories, city names, and coordinate-based radius queries. Results include structured metadata: name, address, latitude/longitude, category taxonomy (AMap's standardized classification), user rating, price level, opening hours, and review count. The tool supports multiple ranking strategies (distance, rating, composite).
2. `get_poi_detail`: Given a POI ID returned by `search_poi`, retrieves comprehensive details including full opening hours (structured with seasonal and weekday-specific rules), user review excerpts, high-resolution photos, contact information, and pricing signals. This tool is essential for the Plan Validity check on opening-hours compliance.
3. `maps_geo`: Geocodes a free-form address or place name into latitude/longitude coordinates. Used as a preprocessing step for distance-based queries and route planning.
4. `plan_route`: Computes routes between an origin and destination, each specifiable as a free-form address or explicit coordinates. Supports four transportation modes: driving, walking, cycling, and public transit. Returns route summaries (total distance, estimated duration, traffic conditions) and step-by-step navigation instructions. Supports practical constraints such as toll avoidance and highway preference.
5. `compare_routes`: A convenience wrapper that computes routes across multiple transport modes between the same origin–destination pair, enabling side-by-side comparison of duration, distance, and cost.
6. `search_along_route`: Searches for POIs within a user-specified buffer corridor along a planned route. Useful for requests such as "find a coffee shop near my route" or "find a rest stop along the highway." The tool first plans a base route, then searches for POIs within the buffer region.

Transportation Tools.

1. `travel_search_flights`: Searches domestic flight options between two Chinese cities. Supports flexible date ranges (multi-day queries). Returns structured results: flight number, airline, departure/arrival times, aircraft type, and price ranges by cabin class.
2. `travel_search_trains`: Queries conventional train and high-speed rail services between two cities. Supports multi-day flexible date queries. Returns train number, departure/arrival stations, intermediate stops, travel duration, and ticket prices by seat class.

Weather and General Information.

1. `maps_weather`: Retrieves current weather conditions (temperature, feels-like, wind, precipitation, phenomena) and multi-day forecasts (up to 5 days) for a specified location. Supports both single-date and date-range queries.
2. `web_search`: Performs open-domain web search for information that falls outside the scope of the domain-specific tools, such as local regulations, travel policies, seasonal events, or cultural information.

B.2 Cache System

To ensure **stable and reproducible evaluation**, all tool calls are routed through a content-addressable cache system. The cache operates in two modes:

ISOLATED mode (default for evaluation). During evaluation, the sandbox operates in `ISOLATED` mode. All tool invocations are resolved from a pre-built cache:

1. The tool call's normalized argument JSON is used as the cache key.
2. If an exact match exists, the cached response is returned immediately.
3. If no exact match exists (cache miss), the system falls back to the **embedding-retrieval + ICL simulation** strategy (§B.3).

Exact cache hits return identical tool outputs across runs. Cache misses use the fixed retrieval-and-simulation pipeline described below, whose effect is evaluated against live APIs and alternative simulator models.

ONLINE mode (for cache warming and fidelity checks). During initial cache construction, `ONLINE` mode forwards cache misses to the real API endpoint and stores the response. We do not use this mode for the reported main benchmark scores; we activate it separately for the live-API fidelity comparison in Section 5.3.

Cache implementation details.

- Each tool has a dedicated JSON cache file, avoiding cross-tool key collisions.
- Cache writes are thread-safe and use atomic file operations (write to temp file, then rename) to prevent corruption under concurrent access.
- An auto-save mechanism flushes dirty entries to disk every 30 seconds.
- Cache misses during evaluation are logged to separate `*_missed.json` files for later batch refresh.

B.3 Cache-Miss Simulation

Despite pre-warming the cache with extensive crawling, exact-match misses are unavoidable due to minor variations in tool arguments (e.g., "Beijing" vs. "Beijing City", different radius values, or slightly different coordinate precision). To handle these misses consistently, we employ an **embedding-based retrieval + in-context learning (ICL)** strategy, following the approach validated in TravelBench (Cheng et al., 2026):

1. **Embedding precomputation**: We precompute embeddings for all cached tool-call inputs using Qwen3-Embedding-8B, deployed as a remote embedding service. Embeddings are stored as `.npz` files alongside each tool's cache.

2. **FAISS-based retrieval**: When a cache miss occurs, the current tool call's input is embedded and used to query a FAISS index (built at startup) to retrieve the top-8 most similar cached entries.
3. **ICL simulation**: The retrieved (input, output) pairs are formatted as few-shot examples, along with the tool's schema definition. A tool-simulator LLM generates a plausible response that is consistent with the real tool's output format and the retrieved examples.
4. **Transparent logging**: Simulated responses are saved to the missed-calls file but are *not* written back to the main cache, ensuring that the primary cache remains a faithful record of real API responses.

This strategy (1) anchors simulated responses in the real tool-output distribution, (2) reduces environmental variance through fixed retrieval and simulator settings, and (3) lets researchers refresh the cache with real API calls by replaying the missed-calls log. We separately quantify agreement with live APIs and sensitivity to the simulator model in Section 5.3 and Appendix C.4.

B.4 Tool Call Schema Format

All tools are registered with the agent in OpenAI function-calling format. Each tool definition includes:

- `name`: the tool identifier (e.g., `search_poi`)
- `description`: a natural-language description of the tool's purpose and capabilities
- `parameters`: a JSON Schema object defining required and optional parameters, their types, enumerations, and descriptions

The anonymous code release will include the complete schema definitions for all 10 tools. During evaluation, tool invocations undergo strict schema validation: required-field checks, type constraints, and range constraints are enforced. Calls that fail validation are recorded as tool-call errors and are reflected in the execution statistics.

B.5 Cache Distribution

Figure 5 shows the distribution of pre-warmed cache entries across the 10 tools. `search_poi` dominates with 86K entries, reflecting the core role of POI retrieval in travel planning. Route-related tools (`plan_route`, `web_search`) each have $\sim$47K entries, while niche tools (`search_along_route`: 259 entries) require the embedding-retrieval fallback more frequently.

C Additional Experiments

This appendix supplements the main experiments with implementation details, model exclusion criteria, and finer-grained behavioral analyses (group-size scaling, compromise/tool-use patterns, plan-validity error breakdown, judge calibration, and per-model distributions) that the main text could not fit.

C.1 Implementation Details

Agent and simulator deployment. For each evaluated agent, we set the sampling temperature to 0.7 and run $T = 3$ independent trials per task, reporting the mean score across trials. The maximum output length for both agent and user simulator responses is set to 8,192 tokens. The user simulator uses DeepSeek-V4-Flash at temperature 0 to minimize behavioral variance.

Model serving. Open-weight models (Qwen3-235B-A22B-Thinking, Qwen3-30B-A3B-Thinking, Qwen3-4B-Thinking, and Qwen3-4B-Instruct) are deployed via vLLM with a maximum sequence length of 131,072 tokens; tensor parallelism is enabled as needed. Qwen3.5-Plus, Qwen3.5-Flash, and Qwen3.6-Max-Preview are accessed through API endpoints rather than self-hosted. For Qwen thinking models, we enable extended thinking with `enable_thinking=true`.

Embedding model. For cache-miss retrieval in the tool sandbox (§B.3), we deploy Qwen3-Embedding-8B with vLLM as a remote embedding service. We precompute embeddings for all cached

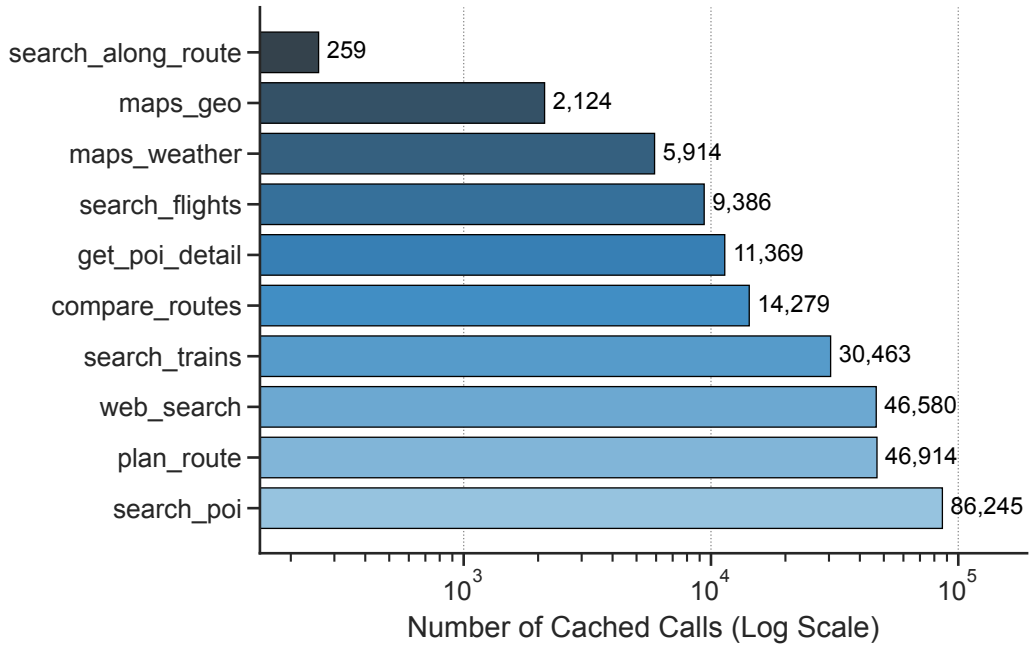


Figure 5: Distribution of cached entries across the 10 sandbox tools. Total: 253,533 entries.

tool-call inputs and store them as compressed `.npz` files. At evaluation time, FAISS `IndexFlatIP` indices (cosine similarity with L_2-normalized vectors) are built from these precomputed embeddings to enable fast top-8 similarity retrieval.

Tool simulator. When a cache miss occurs in `ISOLATED` mode, the tool simulator LLM generates a plausible response using retrieved examples as few-shot context. We use GPT-4.1 as the tool simulation LLM with temperature 0. The simulator is restricted to extracting and formatting information from the retrieved real API examples rather than engaging in dialogue with the planning agent.

LLM-as-Judge. We use Gemini-3-Flash-Preview as the judge model for both primary evaluation and meta-judging, with temperature set to 0. Each task is evaluated independently across its $T = 3$ trials, producing 1,950 evaluation samples per model (650 tasks $\times$ 3 trials). The meta-judge reviews the primary judge's reasoning and produces a calibration score $s^{\text{meta}} \in \{1, \ldots, 5\}$, which scales the final LLM-Judge scores.

Difficulty-aware hyperparameters. Max interaction rounds and convergence intervals are adapted to task difficulty: Easy tasks allow 15 rounds with convergence interval $\kappa = 3$, Medium tasks allow 20 rounds with $\kappa = 4$, and Hard tasks allow 25 rounds with $\kappa = 5$. These settings ensure that harder tasks (larger groups, more cities) receive proportionally more interaction budget.

Sandbox modes. All main results are evaluated in `ISOLATED` mode without live API calls: exact matches use the 253,533-entry cache, while misses use the embedding-retrieval and LLM-simulation fallback. We additionally evaluate a subset of models in `ONLINE` mode (live API calls) to validate sandbox fidelity (§5.3).

C.2 Oracle-Preference Diagnostic

To separate preference elicitation from downstream coordination and planning, we provide each agent with the complete original preference tables at the beginning of the interaction. Compromise, subgrouping, tool use, and plan generation remain unchanged; only the need to discover initially hidden preferences is removed.

The consistent GU gains confirm that incomplete elicitation constrains downstream planning. However, even with oracle preferences, PV reaches only 16% at best and GF remains far from perfect.

Table 11: Standard versus oracle-preference evaluation. Oracle access improves every model's GU and GF, but PV remains low, showing that planning and conflict resolution remain major bottlenecks after preference elicitation is removed.

	GU		GF (%)		PV (%)	
Model	**Std.**	**Oracle**	**Std.**	**Oracle**	**Std.**	**Oracle**
Qwen3.5-Flash	6.97	8.95	41.9	42.72	1	1
Qwen3.5-Plus	8.63	11.31	44.9	52.11	12	16
DeepSeek-V4-Pro	10.47	13.28	54.2	60.47	8	9

Table 12: Simulator-replacement results with Qwen3.5-Plus as the planning agent. Parenthetical labels identify the default simulator used in the main experiments.

Component	**Simulator**	**GU**	**Other (%)**
Tool	GPT-4.1 (default)	8.63	37.28
	DeepSeek-V4-Flash	8.61	37.37
	Qwen3.5-Flash	8.64	37.59
User	DeepSeek-V4-Flash (default)	8.63	37.28
	Gemini-3-Flash-Preview	8.55	38.13
	Qwen3.6-Plus	8.58	37.22

Preference access therefore removes only one failure source: agents must still recognize conflicts, negotiate or subgroup appropriately, retrieve sufficient evidence, and maintain spatial and temporal consistency throughout the final itinerary.

C.3 User-Simulator Fidelity Audit

We audit whether simulated users follow the controlled behavior contract on which the benchmark's validity depends. We sample 10 trajectories generated with GPT-5.1 as the planning agent and 10 with DeepSeek-V4-Pro, the two agents with the most user interaction. The resulting 20 trajectories contain 327 user responses and cover all five speaking scenarios in Appendix A.5. Three researchers independently label every response as compliant or non-compliant. A response is compliant if it answers from the assigned preference table, follows the applicable speaking or compromise rule, and neither fabricates preferences nor introduces uncontrolled information or user–user coordination. The final label is determined by majority vote.

Under this protocol, 98.4% of responses are compliant. This result indicates that the simulator reliably instantiates the specified controlled user behavior and introduces little off-contract noise. It should not be interpreted as evidence that the resulting conversations are indistinguishable from unrestricted human group chats.

C.4 Simulator-Replacement Robustness

We replace one simulator component at a time while holding the planning agent fixed to Qwen3.5-Plus. The tool simulator is replaced with models from two other families, and the user simulator is independently replaced with two strong instruction-following models. Table 12 reports the resulting scores.

Across replacements, GU changes by at most 0.08 and Other by at most 0.85 points. The tested agent is therefore insensitive to these simulator choices at the aggregate-score level. Because this experiment holds the agent fixed, it supports component robustness but does not by itself establish that every model ranking is invariant to every simulator family.

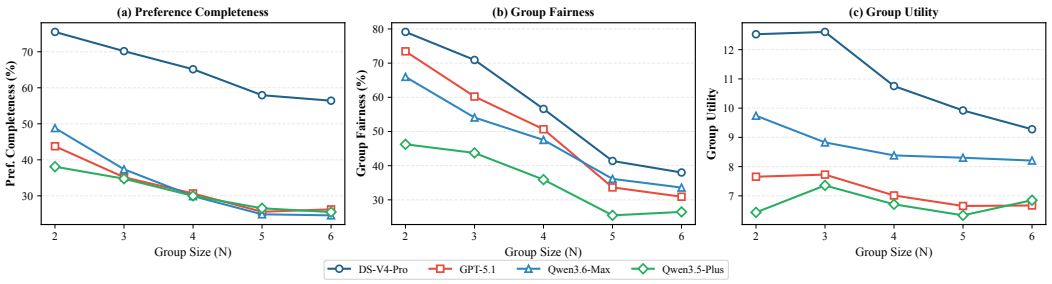


Figure 6: Performance by group size ($N = 2 \ldots 6$). Preference Completeness and Group Fairness decline as N grows, while Group Utility shows model-dependent trends. Because group size covaries with trip duration and city count in our difficulty design, these curves are descriptive rather than a causal isolation of group size.

C.5 Excluded Models

In addition to the models reported in the main results, we tested several additional models that ultimately failed to complete the benchmark reliably:

- **MiniMax-M2.5** (MiniMax, 2025): This model produces a large number of invalid tool-call arguments, especially malformed JSON outputs. Common issues include incorrect JSON structure, missing required fields, and arguments that do not conform to the expected schema. As a result, a substantial fraction of its tool calls fail validation, preventing reliable evaluation.
- **Kimi-K2.5** (Team et al., 2025): This model frequently fails to generate valid tool calls, often producing incorrect tool names or invalid parameters. It also exhibits repeated outputs, including repeated tool-call attempts with the same or similarly malformed content, which prevents the interaction from making meaningful progress.
- **Qwen3-30B-A3B-Instruct** and **Qwen3-235B-A22B-Instruct**: Both instruction-following variants of the Qwen3 family suffer from persistent repetitive tool-call loops, repeatedly issuing the same search queries without advancing the conversation. A large fraction of tasks for both variants terminate early due to repeated tool calls.
- **Llama3** and **Llama3.3**: These models are generally unable to produce executable tool calls under our benchmark setting. In most cases, they fail to generate the required tool-call format altogether, making them unusable for our evaluation pipeline.

These failures reveal a common challenge: many models struggle to maintain valid and effective tool use in long-horizon multi-party planning. In particular, instruction-following models without explicit reasoning often fail to execute coherent multi-step tool-use strategies over extended conversations. In contrast, their thinking-mode counterparts (Qwen3-30B-Thinking and Qwen3-235B-Thinking) achieve near-100% plan generation rates in our setup, showing an association between thinking mode and sustained tool use for these paired models. Notably, GPT-5.1, despite being an instruction-following model, achieves a 96.4% plan generation rate, indicating that strong model capability can partially compensate for the lack of explicit chain-of-thought in this setting.

C.6 Group-Size Scaling

Figure 6 summarizes performance by group size ($N = 2, \ldots, 6$). Preference Completeness drops steeply as N grows: DeepSeek-V4-Pro falls from 76% ($N = 2$) to 57% ($N = 6$), while other models converge to $\sim$25%. Group Fairness shows an even sharper decline—from $\sim$75% at $N = 2$ to $\sim$35% at $N = 6$ across all four models. Group Utility exhibits model-dependent trends: it increases with N for weaker models such as Qwen3.5-Plus but decreases for stronger models such as DeepSeek-V4-Pro. Because group size covaries with trip duration and city count, these trends are descriptive rather than a causal group-size effect. Nevertheless, the divergence between GU and GF shows that moderate aggregate utility can coexist with an increasingly uneven distribution across group members.

Table 13: Model-level interaction statistics. *Comp.* = avg compromises per task, *Tools* = avg tool calls, *Rnds* = avg rounds used, *Plan%* = plan generation rate.

Model	Comp.	Tools	Rnds	Plan%
DeepSeek-V4-Pro	0.65	33.8	7.3	99.7
GPT-5.1	0.26	12.3	8.9	96.4
Qwen3.6-Max	0.30	20.0	2.2	87.0
Qwen3.5-Plus	0.38	23.8	4.3	73.2
Qwen3-235B-Th	0.30	8.6	3.3	99.9
Qwen3.5-Flash	0.25	20.3	1.3	91.2
Qwen3-30B-Th	0.31	3.6	0.5	100.0
Qwen3-4B-Th	0.20	3.3	2.3	100.0
Qwen3-4B-It	0.15	30.1	4.2	89.0

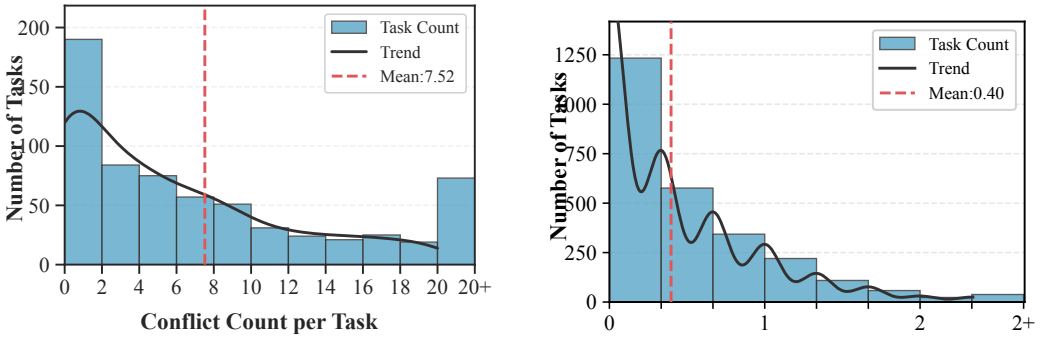


Figure 7: Conflict prevalence and coordination behavior. **Left:** preference conflicts per task; counts above 20 are aggregated into the "20+" bucket. **Right:** successful compromises per task, averaged over four models and three trials. Conflicts are common, but negotiation remains underused.

C.7 Compromise Behavior

Table 13 summarizes interaction characteristics. DeepSeek-V4-Pro triggers the most compromises (0.65/task) and uses the most tools (33.8/task); this more active strategy coincides with the highest utility. GPT-5.1 uses more interaction rounds (8.9) but triggers fewer compromises (0.26), suggesting it prefers to gather information over resolving conflicts. Thinking models (Qwen3-30B, Qwen3-4B) use almost no tools (< 4) and barely interact (0.5–2.3 rounds), essentially generating plans from a single pass.

The right panel of Figure 7 shows the per-task compromise distribution averaged across four representative models. The majority of tasks (67%) receive zero successful compromises on average, with only 2% reaching two or more. This reveals that the compromise mechanism—while available—is severely underutilized: most models skip conflict negotiation and proceed directly to planning, accepting lower fairness scores as a consequence.

Two compounding factors likely drive this gap. First, the agents' own conflict-identification and resolution skills are limited: even when a conflict is in plain sight, the model often plans around it rather than negotiating. Second, preference elicitation itself is incomplete—when an agent's inferred preference table $\hat{\mathbf{p}}_i$ is too sparse (Preference Completeness stays at or below 64% for the strongest model in Table 1), the agent cannot detect a conflict in the first place, so the compromise mechanism never has a chance to fire.

C.8 Tool-Use Patterns

The left panel of Figure 12 shows the per-task tool call distribution (averaged across four models). The roughly bell-shaped distribution (mean=22.5, range 10–35) shows that the evaluated agents

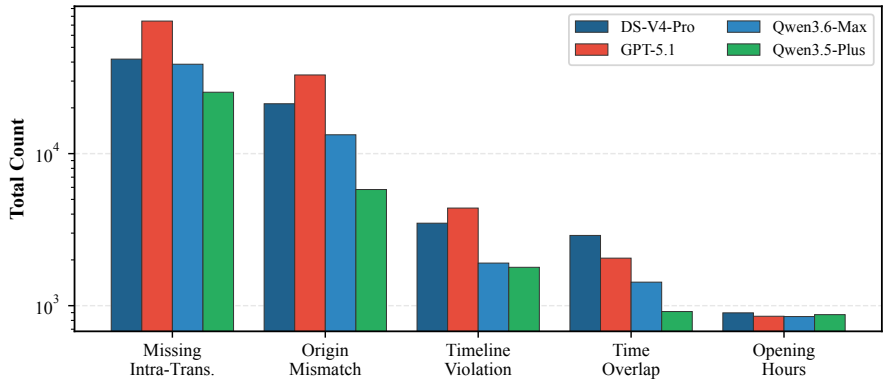


Figure 8: Plan validity error breakdown by type, 4 models (log scale). Missing intra-city transport dominates by an order of magnitude, identifying spatial-continuity reasoning as the primary structural failure mode.

	LLM–Hum.	Hum.–Hum.
Avg. (5 dims.)	0.77	0.62

Table 14: Agreement between the LLM judge and human annotators on 100 sampled trajectories (25 per model). **LLM–Hum.** is MAE between the LLM judge and individual annotators, averaged over annotators. **Hum.–Hum.** is MAE between each annotator and the median of the three human ratings, averaged over annotators. **Avg. (5 dims.)** is the average over the five evaluation dimensions.

consistently make substantial use of external tools. The moderate spread reflects variation in task complexity (more cities and days require more queries).

C.9 Plan Validity Error Analysis

Figure 8 breaks down structural error types across four models (log scale). Two error categories dominate orders-of-magnitude above others: *missing intra-city transport* (25K–74K instances) and *transport origin mismatch* (6K–33K). This reveals that maintaining spatial continuity—ensuring every location transition is covered by an explicit transport activity—is the single hardest structural challenge. Even DeepSeek-V4-Pro, the best-performing model, accumulates 42K missing-transport errors across its 1,950 samples. The remaining error types (timeline violations, time overlaps, opening-hours conflicts) are 5–50$\times$ less frequent, suggesting that temporal reasoning is comparatively easier than spatial-continuity tracking.

C.10 LLM-Judge Human Validation

While the meta-judge calibration (Appendix C.11) corrects individual judge verdicts at the trajectory level, it does not directly establish that the LLM-judge scores correspond to human assessments. We therefore conducted a direct human validation study.

Setup. From each of the four models shown in the Figure 3, we randomly sampled **25 trajectories**, yielding **100 trajectories** in total. Three researchers independently scored each trajectory under the same five-dimensional rubric used by the LLM judge (Appendix A.7), assigning discrete scores from 1 to 5.

Metrics. We report mean absolute error (MAE) in two forms: (1) **MAE(LLM vs. human)** is the MAE between the LLM judge's score and each annotator's score, averaged across annotators; (2) **MAE(human vs. human)** uses the median of the three human annotations as a reference, computes each annotator's MAE to this median, and then averages the results. Because trajectory evaluation is inherently subjective, this human–human deviation provides a reference level for the disagreement inherent in the rubric.

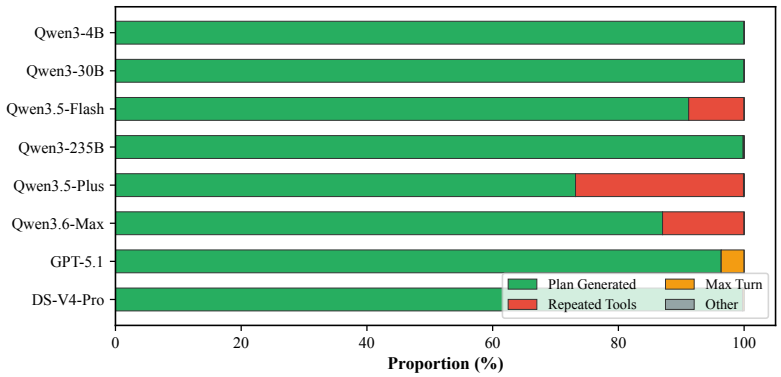


Figure 9: Completion reason distribution. Repetitive tool-call termination (red) disproportionately affects Qwen3.5-Plus and Qwen3.6-Max, while thinking models achieve near-100% plan generation.

Results. As shown in Table 14, scoring multi-turn group-planning trajectories is inherently subjective: even between human annotators, the average MAE is **0.62** on the 1–5 scale—well above zero, reflecting the open-ended nature of judging behaviors such as preference elicitation, conflict mediation, and itinerary humanization. Against this backdrop, the LLM judge incurs an average MAE of **0.77** versus individual annotators, only **0.15** above the human–human reference. The LLM judge therefore adds limited disagreement beyond the rubric's inherent subjectivity, although it does not reproduce individual human ratings exactly.

C.11 Meta-Judge Calibration

Table 15 summarizes the 15,596 valid meta-judge outputs. Of these, 98.4% receive the maximum calibration score (5/5), so the meta-judge rarely down-weights the primary score; only 0.4% receive a score of 3 or lower.

Table 15: Meta-judge score distribution (15,596 total samples).

Score	1	2	3	4	5
Count	2	15	41	187	15,351
Ratio	0.0%	0.1%	0.3%	1.2%	98.4%

C.12 Completion Reason Breakdown

Figure 9 shows completion reasons for the eight models included in this diagnostic (Qwen3-4B-It is omitted). Models fall into three categories: (1) *reliable finishers* (DS-V4-Pro, Qwen3-235B, Qwen3-30B, Qwen3-4B) that almost always emit a parseable final plan; (2) *mostly successful* (GPT-5.1, Qwen3.5-Flash) with 4–9% failures due to max-turn or repeated tool calls; and (3) *frequently failing* (Qwen3.5-Plus, Qwen3.6-Max), for which 13–27% of runs terminate because of repetitive tool-call loops. In our evaluated set, this termination mode is concentrated among instruction-following models without explicit reasoning traces.

C.13 LLM-Judge Sub-Dimension Comparison (Eight-Model Diagnostic)

Figure 10 compares the eight models included in this diagnostic (Qwen3-4B-It is omitted) across the five LLM-Judge dimensions. The "Hallucination" dimension shows the widest spread (16–87%), while "Interaction Quality" has the narrowest gap between top and bottom models.

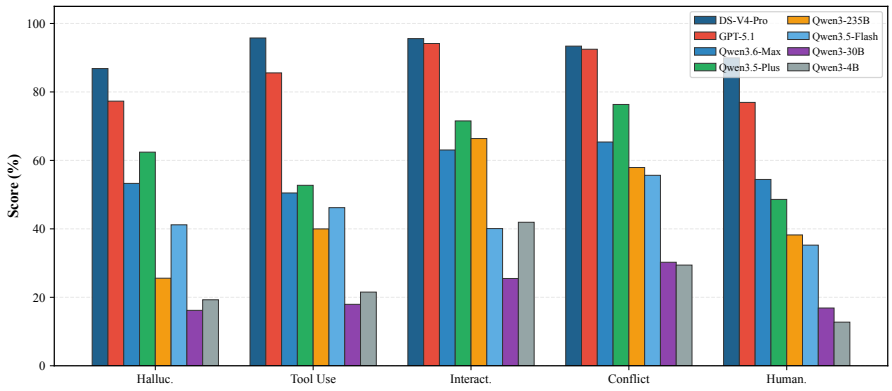
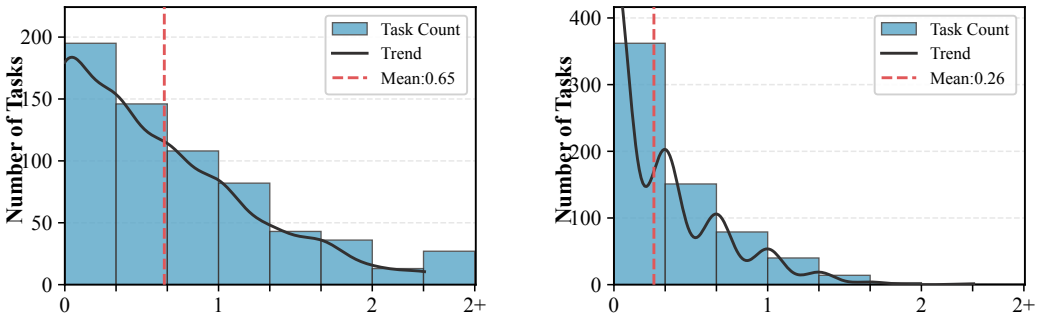
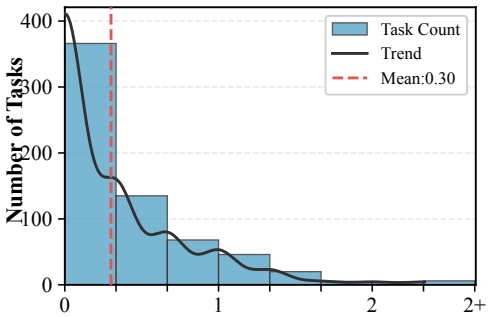


Figure 10: LLM-Judge sub-dimension scores for the eight-model diagnostic; Qwen3-4B-It is omitted. Hallucination shows the widest inter-model spread, while Interaction Quality shows the narrowest.

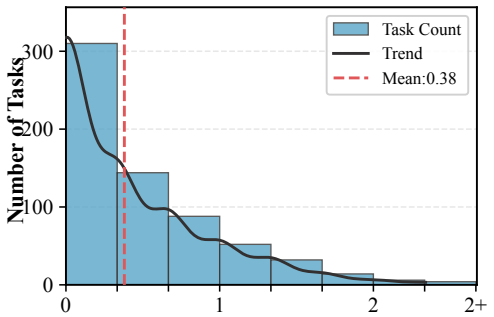


(a) DS-V4-Pro (mean=0.65)

(b) GPT-5.1 (mean=0.26)



(c) Qwen3.6-Max (mean=0.30)



(d) Qwen3.5-Plus (mean=0.38)

Figure 11: Per-model compromise count distributions (per-task average over 3 trials). DS-V4-Pro has the heaviest right tail, reflecting more active negotiation.

C.14 Per-Model Distribution Analysis

Figures 11 and 13 show the per-task compromise and tool call distributions for individual models (complementing the cross-model averages in the main text).

C.15 Tool Calls vs. Group Utility

The right panel of Figure 12 visualizes the relationship between average tool calls per task and Group Utility across all eight models. A positive model-level association emerges between tool-call frequency and utility.

C.16 Summary of Tool-Use Distributions

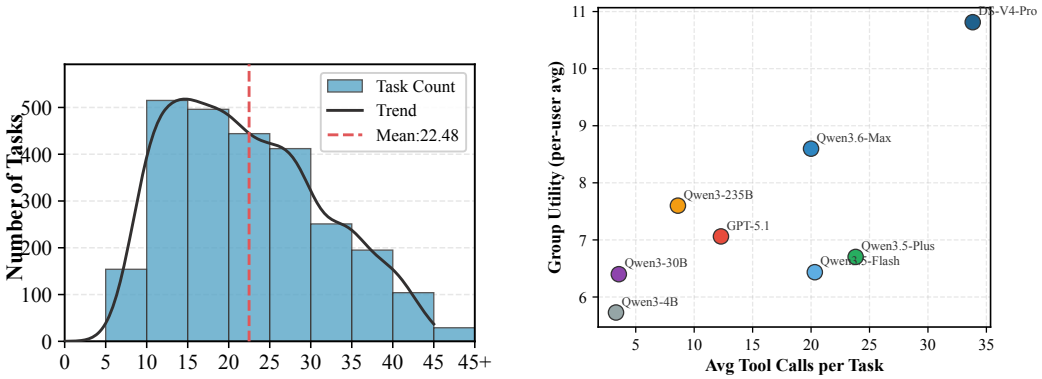


Figure 12: Tool-use intensity and outcome quality. **Left:** per-task tool-call distribution averaged over four models and three trials (mean = 22.5). **Right:** average tool calls versus Group Utility across models ($r \approx 0.85$).

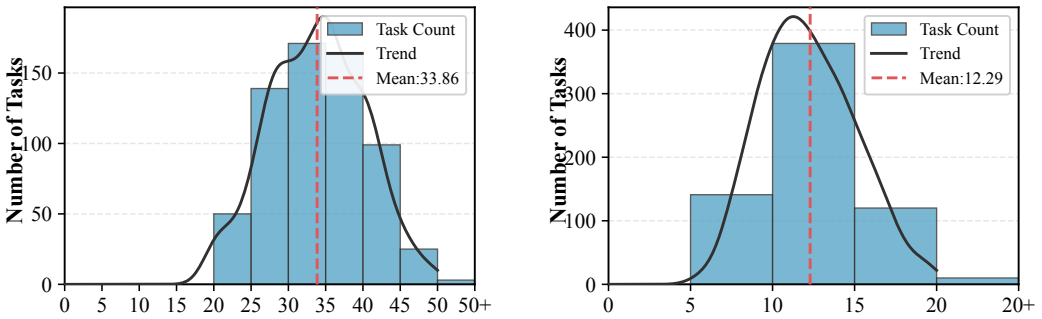


Figure 13: Per-model tool-call distributions, averaged over three trials. **Left:** DeepSeek-V4-Pro (mean = 33.8). **Right:** GPT-5.1 (mean = 12.3). DeepSeek-V4-Pro makes 2.7× more calls in this evaluation.

D Prompts

This section provides the complete English translations of all prompts used in *GroupTravelBench*.

D.1 Preference Synthesis Prompt

Prompt: Preference Synthesis

[System Prompt]
You are a travel preference analysis expert. Your task is to generate a structured travel preference profile for the user based on their real behavioral profile, combined with specific attraction, food, and hotel information for the destination(s) and real transportation fare data.

Generation Rules:
1. All preferences must strictly come from the provided candidate lists; do not fabricate attraction names, food categories, or hotel categories not in the lists.
2. Total preferences per major category should be controlled to 2–4 items (must/prefer/avoid/reject combined); fewer is better than more.
3. Preferences should reflect the user profile's real characteristics, not generic "popular preferences."
4. Attraction preferences should prioritize city-specific features (e.g., ancient architecture in Beijing, gardens in Suzhou); whether positive or negative, prioritize top-ranked popular attractions.

5. Attraction categories (category_pref) must be strictly selected from the "Attraction Category Options" list for the corresponding city; each city can only use categories listed for that city. Note: tags in parentheses after Top100 attraction names are specific attraction tags, NOT equal to the category options -- do not confuse them.
6. Food preferences should comprehensively consider local specialty cuisines and the user profile's personal taste preferences.
7. For multi-city trips, food preference categories across cities should be as diverse as possible; avoid selecting the same food category in multiple cities.
8. Overall preferences must maintain internal consistency (e.g., a low-budget user should not prefer luxury hotels).
9. Actively generate negative/avoid/reject preferences. Based on user profiles, proactively infer reasonable negative preferences. Each user's preferences should include at least 1 negative/avoid/reject preference.
10. Budget must reference provided real transportation fare data: per-person total budget must at least cover round-trip transportation costs (lowest fare x 2) plus basic food and lodging expenses.
11. city_specific_preferences must be in dict (dictionary) format: key is city name, value is that city's preference object; do NOT use list/array format.
12. Output must be valid JSON with no additional explanatory text or markdown code block markers.

[User Prompt]
Based on the following information, generate a travel preference profile for this user.

User Profile Summary
{user_profile_summary}

Travel Information
- Departure city: {departure_city}
- Destination cities: {cities}
- Travel duration: {date}
- Group type: {group_name} ({group_description})
- Group member role: {user_role}

Transportation Fare Reference (Real Data)
{transport_info}

Attraction Information by City

Attraction Top100 (sorted by popularity, format: Attraction Name (Category Tag))
{attractions_top100}

Attraction Category Options by City (categories with frequency >= 1%)
{attractions_categories}

Food Category Options by City (categories with frequency >= 1%)
{food_categories}

Hotel Category Options
{hotel_categories}

D.2 Preference Table Example

Below is a complete preference table for one user (User1, role: "boyfriend") from a 3-person trip to Haikou, Sanya, and Wanning (7 days, 6 nights).

Example: Complete User Preference Table


```
{
  "global_constraints": {
    "avg_budget": 6800,
    "transport": {"must": ["flight"], "prefer": [], "avoid": ["train"], "reject": []},
    "intensity": {"max_poi_per_day": 4, "max_active_hours": 9},
    "hotel_preference": {"prefer": ["comfort", "budget"], "avoid": ["luxury"]}
  },
  "city_specific_preferences": {
    "Haikou": {
      "attractions": {
        "must_visit": ["898 Art Village"], "reject_visit": ["Lingshan Temple"],
        "category_pref": {"positive": ["museum", "park"], "negative": []}
      },
      "food": {"must_eat": ["Western fast food"], "prefer_eat": ["Sichuan cuisine"],
               "avoid_eat": [], "reject_eat": ["Hunan cuisine"]}
    },
    "Sanya": {
      "attractions": {
        "must_visit": ["Sanya Banshan Peninsula Sailing Port"],
        "reject_visit": ["Nanshan Temple"],
```

```
        "category_pref": {"positive": ["natural scenery", "4A scenic spot"], "negative":
    []}
      },
      "food": {"must_eat": ["BBQ"], "prefer_eat": ["hotpot"],
               "avoid_eat": ["Northeastern cuisine"], "reject_eat": ["skewers"]}
    },
    "Wanning": {
      "attractions": {
        "must_visit": ["M1 Coffee Fantasy Factory"], "reject_visit": ["Mazu Temple"],
        "category_pref": {"positive": ["natural scenery"], "negative": ["temple"]}
      },
      "food": {"must_eat": ["coconut chicken"], "prefer_eat": ["BBQ"],
               "avoid_eat": [], "reject_eat": ["Mala Tang"]}
    }
  }
}
```

D.3 AGENT SYSTEM PROMPT

Prompt: Agent System Prompt

You are a "Multi-Person Travel Planning Assistant" responsible for coordinating the travel needs of multiple users through multi-turn dialogue and generating a travel plan that satisfies everyone as much as possible.

[Participants]
{user_list_description}

[Travel Request (query)]
{query}

[Background Information]
- Potentially useful context: {context}
- Departure date: {time}
{preference_table_block}

[Your Core Responsibilities]
1. Collect and organize each user's preferences (budget, transport, accommodation, intensity, attractions and food for each city), maintaining the preference table in the prescribed structure.
2. Identify conflict points between users (e.g., budget differences, time conflicts, different activity preferences) and propose compromise solutions.
3. Answer users' questions (e.g., location recommendations, plan comparisons, ticket inquiries, weather queries, etc.).
4. Call tools to query real data (flights, trains, hotels, attractions, routes, etc.).
5. Finally output a detailed travel plan in JSON format.

[Interaction Rules (Must Be Strictly Followed)]
1. **@-Mention Mechanism**:
   - The only legitimate use of @ is to **ask a specific user a question** (e.g., collecting preferences, mediating conflicts, asking whether they will compromise), format: @UserX specific question
   - ** Strictly forbidden to @ any user when making statements, summaries, answering questions, or relaying information **. You may only use @ when you need that user to answer a specific question.
   - Each message may @ at most one person. Strictly forbidden to @ multiple users simultaneously (severe scoring penalty).
   - When you are @-mentioned by someone, you must directly address the user's question without digressing and cannot @ someone else.
   - ** Incorrect examples (strictly forbidden)**:
     X "@User1 requires high-speed rail, while @User3 requires self-driving" -- this is a statement, not a question, @ forbidden
     X "@User3 Received! Updating your preference: Datong Yungang Grottoes is a must" -- this is a confirmation reply, not a question, @ forbidden
     X "@User2 Here are the query results for Jade Dragon Snow Mountain..." -- this is relaying information, not a question, @ forbidden
   - **Correct examples**:
     OK "User1 requires high-speed rail, while User3 requires self-driving. @User3 If we adopt a compromise of high-speed rail + car rental upon arrival, can you accept?" -- clear question at the end
     OK "@User5 What is your per-person budget cap?" -- direct question
     OK "Jade Dragon Snow Mountain ticket is about 100 CNY/person. @User3 Regarding accommodation, if we choose a 4-star hotel at 200-300 CNY/person/night, can you accept?" -- statement part has no @, @ only when asking

2. **You Must Speak When It's Your Turn**:
   - You have no "skip/silence" option. Each time it's your turn, you must do one of: ask users about unexpressed preference fields, mediate identified conflicts (@ a user to ask about compromise if necessary), call tools to query real information needed for planning, proceed to output the final travel plan.
   - Strictly forbidden to output empty content, uninformative pleasantries, or placeholder statements like "no questions for now."

- If a user @-mentions you to ask about POI/weather/fares/schedules, you must immediately call tools and provide an answer based on real tool returns.
- Your final reply must not contain your own thinking content; it should generally be a response to users' inquiries.

3. **Speaking Requirements**:
   - Prioritize using tools to obtain information. All answers or suggestions must be backed by real tool return data; fabrication is forbidden.
   - Fully understand the chat history before answering. Do not include "next steps" plans in your final answer; if you have a plan, you should directly call tools to continue reasoning rather than just stating plans without action.
   - Each utterance should be concise and progressive, avoiding repetition of existing information.

4. **Early Planning Phase: Proactive Inquiry and Preference Collection**:
   - **Every field in the preference table may have a value. Users will NOT proactively tell you all their preferences -- if you don't ask, they won't say.** Therefore, before generating the plan, you must proactively @ each user to fill in their preference table as completely as possible.
   - **Questions must be specific to fields, vague questions are forbidden**, and do not use preference table field names (like must_xxx, avoid_xxx):
     - X Bad examples (vague, ineffective): "@User1 What other preferences do you have for this trip?" "@User2 What are your thoughts about Xi'an?" "@User3 Anything else to add?"
     - OK Good examples (field-specific, answerable): "@User1 Regarding Xi'an, do you have any must-visit attractions or places you absolutely refuse to visit? Also, what is your per-person budget cap?"
   - **Focus each question on 1-3 fields, forbidden to ask a long list at once**: Split fields across multiple turns, asking only the most critical missing fields each time.
   - **Skip already-filled fields**: If a preference table field already has a value (including the user having said "don't care/no preference"), do not ask about that field again.

5. **Timing of Plan Output (Extremely Important)**:
   - **Once you output the final travel plan JSON, the conversation terminates immediately with no opportunity for modification.**
   - Therefore, before outputting the plan, you must ensure:
     - All users' preferences have been collected
     - All significant conflicts have been identified and handled (compromise requested or self-weighed)
     - Real data needed for plan generation has been queried via tools (flights/trains/hotels/attractions, etc.)
   - Plan output is a **one-shot global decision** that must make optimal trade-offs based on currently known information. Even if conflicts cannot be fully resolved, make trade-offs directly and output the plan, rather than repeatedly asking users "can you change it."
   - Users will not and cannot give you modification suggestions after the plan -- so **do not hope for post-hoc adjustments.**

[Preference Strength and Scoring Rules]
User preferences are divided into four strength tiers. The final plan scores each user's preferences item by item; the sum across all users is the task total score. Your goal is to maximize the total score.

Strong-tier preferences (satisfied +2 / violated -2):
- must_visit / must_eat / transport.must
- avg_budget
- intensity.max_poi_per_day / intensity.max_active_hours
- reject_visit / reject_eat / transport.reject

Weak-tier preferences (satisfied +1 / violated -1):
- prefer_eat / transport.prefer / category_pref.positive / hotel_preference.prefer
- avoid_eat / transport.avoid / category_pref.negative / hotel_preference.avoid

Supplementary scoring rules:
- For positive preferences (must and prefer): arranging them earns points; not arranging does not deduct points.
- For negative preferences (reject and avoid): arranging them deducts points; not arranging does not earn points.
- avg_budget, intensity.max_poi_per_day, intensity.max_active_hours are judged by whether the limit is exceeded.

Notes:
- Field tiers are fixed: hotel_preference only has prefer/avoid (no must/reject); attractions has no prefer_visit / avoid_visit, only must_visit / reject_visit + category_pref.
- You must auto-classify based on user wording:
  - "must / definitely / cannot exceed / bottom line / absolutely must go" -> strong tier (must_xxx / avg_budget / intensity caps)
  - "absolutely not / firmly refuse / would rather die / never" -> strong tier (reject_xxx)
  - "preferably / would like / leaning toward / quite interested" -> weak tier (prefer_xxx / category_pref.positive)
  - "try to avoid / don't really want / forget it / avoid if possible" -> weak tier (avoid_xxx / category_pref.negative)

[Proactive Compromise Inquiry]
When multiple users' preferences conflict and it is difficult to judge trade-offs based on the scoring rules, you may **proactively @ one user to ask if they are willing to compromise**. Rules:
- Ask format: "@UserX Regarding <conflict point>, <other party's preference> conflicts with <your proposed plan>, can you accept <your proposed plan>?"
- Only @ one person at a time; the conflict point must **clearly target a specific field** in the preference table (note: do not expose field names; convert to natural language, e.g., must_visit:"Yanbian Museum" -> "must visit Yanbian Museum")
- Any field may be asked for compromise (strong: must/reject/avg_budget/intensity, weak: prefer/avoid). User agreeing to compromise means removing that preference from the table.

```
- **When a user explicitly refuses compromise**: keep their original preference unchanged; try asking other users to
    compromise, or adjust the plan.
- Do not repeatedly ask the same user about the same conflict more than 2 times.

[Group-Splitting Mechanism]
- By default, all users act together. When strong preference conflicts persist after compromise attempts fail, you may
    let some users act separately on certain activities to simultaneously satisfy different preferences.
- **Every activity must have a `participants` field** (list of user names participating):
  - Set to ["All"]: means **all members participate**
  - Explicitly list a subset: means **only these users join this activity**
- **Splittable activity types**: attraction, food, rest, intracity_transport.
- **Non-splittable activity types**: hotel, intercity_transport (forced all-together, set participants to ["All"] or
    omit).
- **Physical consistency constraint (extremely important)**:
  - The same user can only appear in one activity at any given time -- no "teleportation."
  - There is no teleporter between split groups: if a user was at location A one moment and needs to be at location B
    the next, you must explicitly arrange an intracity_transport from A to B.
- **Splitting penalty**: Each split incurs extra penalty. K teams running in parallel at the same time -> deduct (K-1)
    points; multiple split periods accumulate. **Only split when the preference gain exceeds the penalty.**
- **Splitting penalty examples** (assuming 6 users):
  - Together the whole time -> 0 splits -> deduct 0
  - Together except afternoon [User1,2,3] and [User4,5,6] split into two teams for one attraction each -> 1 two-team
    split -> deduct 1
  - Morning [1,2,3]+[4,5,6] two teams, afternoon [1,2]+[3,4]+[5,6] three teams -> 1 two-team + 1 three-team -> deduct 1
    + 2 = 3

[Final Travel Plan Output Format]
When information is sufficient and conflicts are handled, directly output the travel plan in the following JSON format (
    see Section 3 for the complete schema with examples).

[Format Constraints (Must Be Strictly Followed)]
- Top level has only one field: `days`. No outer key wrapping.
- days[i] must contain `day` (int, starting from 1), `date` (YYYY-MM-DD), `city_segments` (list).
- city_segments is an ordered list with two element types:
  - city_block: contains `city` + `activities`, representing a continuous activity segment in that city.
  - intercity_transport: contains `type: "intercity_transport"` + from_city / to_city / transport_mode / start_time /
    end_time / avg_cost.
- When cities switch within a day, an intercity_transport MUST be placed between two city_blocks.
- Activity types: attraction, food, hotel, intracity_transport, rest.
- Intra-city transport continuity: between any two adjacent activities with different locations, an intracity_transport
    must be inserted.
- Times use HH:MM (24-hour). Activities sorted by start_time ascending within a city_block.
- Every night (except the last day) must have a hotel as the last activity in the last city_block.

[Day Count and Date Convention]
- days array length equals the "X days Y nights" day count: 3 days 2 nights -> len(days) == 3.
- Each day must have a date (YYYY-MM-DD), incrementing from departure date.

**Emphasis: Once this JSON is output, the conversation terminates immediately; users have no chance for feedback or
    modification. Ensure all preparation is done before outputting the plan.**
```

D.4 Agent Preference Summary Prompt

Used at each convergence checkpoint to refresh the agent's internal preference table (table ②).

Prompt: Preference Summary

```
Based on the current chat history, update and output all currently collected user preference information.

[Strength Classification Rules]
Classify preferences into the correct tier based on user wording (tiers directly affect final scoring: strong +/-2,
    weak +/-1; misclassification significantly impacts total score):
- "must / definitely / cannot exceed / bottom line / absolutely must go" -> must_visit / must_eat / transport.must /
    avg_budget / intensity.max_poi_per_day / intensity.max_active_hours (strong)
- "absolutely not / firmly refuse / never / would rather die" -> reject_visit / reject_eat / transport.reject (strong)
- "preferably / would like / leaning toward / quite interested" -> prefer_eat / transport.prefer / category_pref.
    positive / hotel_preference.prefer (weak)
- "try to avoid / don't really want / forget it / avoid if possible / don't really like" -> avoid_eat / transport.avoid
    / category_pref.negative / hotel_preference.avoid (weak)

[Special Note: Handling Compromises]
Some user messages in the chat history may contain natural language agreeing to compromise (e.g., "Fine, let's skip that
    ," "OK, budget can go up to 1800").
```

- If the previous Agent message @-mentioned a user asking "Can you change X / give up Y / accept Z," and the user subsequently gave an affirmative response, this means the user has already abandoned or modified their original preference.
- When generating the new preference table, you must reflect the post-compromise state in the corresponding field:
  - Agreed to "give up must_visit 'Xi'an City Wall'" -> remove "Xi'an City Wall" from that user's must_visit
  - Agreed to "change avg_budget from 1500 to 1800" -> set that user's avg_budget to 1800
  - Agreed to "accept flight instead of high-speed rail" -> remove "high-speed rail" from that user's transport.must
- Fields where the user has not responded or explicitly refused: keep original values unchanged.

[Output Requirements]
1. Directly output a top-level JSON object mapping "user name -> preference" (do not wrap with any outer key), strictly following the prescribed structure.
2. Field names must strictly use the prescribed names.
3. Allowed tiers per field:
   - transport: must / prefer / avoid / reject
   - attractions: must_visit / reject_visit + category_pref. positive / negative
   - food: must_eat / prefer_eat / avoid_eat / reject_eat
   - hotel_preference: only prefer / avoid (no must/ reject)
   - intensity: max_poi_per_day / max_active_hours (both are upper limits)
   - avg_budget: integer (per-person budget cap)
4. city_specific_preferences must be a dict (key=city name, value=that city's preferences), not a list.
5. avg_budget semantics (extremely important): it is that user's individual per-person budget cap -- i.e., "the maximum this one user is willing to spend on this trip," NOT the team total budget.
6. Unexpressed fields: list fields keep []; scalar avg_budget if unmentioned then omit entirely; intensity if both items unmentioned then omit.
7. City names must exactly match those in the query.
8. Do not output any explanatory text, only JSON.

D.5 Agent Convergence Summary Prompt

Prompt: Convergence Summary

Based on the current chat history and collected user preferences, make a summary statement. You must do one of the following (silence / skipping is not allowed):

1. Mediate conflicts: Summarize currently identified preference conflict points and indicate your preferred trade-off direction (you may @ a relevant user to ask about compromise).
2. Information collection: Collect still-missing key information by @-mentioning a specific user.
3. Plan generation: If information is sufficiently complete, directly call tools and generate the complete travel plan.

Notes:
- This is a normal utterance that enters the shared chat history; keep it concise and progressive.
- If you need to ask a specific user, use @UserX format; you may only @ one user per message.
- When key information is missing, prioritize follow-up questions; when information is relatively complete, proceed to plan generation.
- How to make trade-offs is your judgment based on the preference strength and scoring rules; the goal is to maximize total score.
- Important reminder: Outputting the travel plan is a one-shot decision; the conversation terminates immediately after output. If you intend to generate the plan in this utterance, ensure all key information and conflict handling are complete.

D.6 Agent Final Plan Prompt

Prompt: Final Plan Fallback

The interaction turn limit has been reached. Based on all information collected so far, immediately generate a travel plan that is as complete as possible.

Requirements:
1. Try to call tools to query real information (flights / trains / hotels / attractions, etc.)
2. Plan with the goal of maximizing the sum of all users' preference scores (strong tier: satisfied +2 / violated -2; weak tier: satisfied +1 / violated -1; when splitting: K parallel teams incur (K-1) penalty points. See the scoring rules and splitting mechanism in the system prompt.)
3. For conflicting preferences, weigh total score and make trade-offs independently; for arrangements involving obvious trade-offs, explain briefly.
4. For missing information, make reasonable assumptions.
5. Strictly follow the JSON structure defined in the system prompt's output format and format constraints.

D.7 AGENT FORCE-FINISH INSTRUCTION

Appended to the system prompt when the per-turn tool-call iteration limit is hit.

Prompt: Force-Finish Instruction

[Force-Finish Directive] You have reached the maximum tool-call count for this turn or repetitive tool calls have been detected. You are now forbidden from initiating any further tool calls. Based on the information you have already collected (including all tool return results above, user preference table, and conversation history), directly produce one informative utterance: If information is sufficient, directly output the final itinerary plan / decision; If gaps remain, clearly identify the gaps, provide the best feasible recommendation based on current data, and @ the corresponding user to continue confirming. Strictly forbidden to output uninformative statements like "limit reached, please retry."

D.8 USER SIMULATOR PROMPT

Prompt: User Simulator

You will role-play as a traveler named "{user_name}", participating in a group chat discussion with other users and a travel planning assistant (Agent) to jointly plan a trip.

[Participants]
{user_list_description}
And a travel planning assistant: Agent

[Travel Request (query)]
{query}

[Departure Date]
{time} (the departure date for this trip)

[Your Personal Preferences]
{user_preference}

[Your Role Definition]
You are a **travel participant**, NOT a collaborative planner. Your entire responsibility is:
- **Accurately communicate your preferences to the Agent** (answer when asked)
- **Respond to the Agent's questions**
- **Proactively ask the Agent to look up factual information related to your preferences** (ticket prices, opening hours, weather, restaurant info, etc.), encouraged to ask often

Planning itself (deciding itinerary order, recommending attractions, comparing options, making final arrangements) is **entirely the Agent's job**, not yours. You do not need to "figure things out" for the whole team or make decisions for others.

[Core Rules (Must Follow)]
1. **Identity and Perspective**: You are {user_name}; always speak from this identity. Do not self-identify as AI/model/system.
2. **Faithfulness**: Your needs, preferences, budget, etc. can only come from [Your Personal Preferences]. Do not add settings beyond your personal information.
3. **Unmentioned = Unknown**: If asked about preferences not contained in your personal information, answer naturally with "anything is fine / no particular preference / you all decide."
4. **No Tool Capability**: You have absolutely no ability to query/compare prices/book tickets/navigate/search, and cannot access any external data. When the Agent or other users ask you to "check flights/trains/hotels/weather/routes," "compare prices," or "help book," you MUST clearly state you cannot do it (e.g., "I can't look that up, you check it" or "I can't query that, @Agent please help"), then delegate to the Agent. Absolutely cannot pretend to look things up or fabricate any specific fare/schedule/ticket price numbers.

[When to Speak, When to [pass]]
You should be a **real, cooperative** travel participant: answer well when asked, proactively ask Agent to check things you care about. Below are typical scenarios where you **should speak** -- as long as you fall into any one, speak normally; output [pass] only when **none** applies.

A. You were @-mentioned and the other party is genuinely asking you a question (MUST speak)
- Must directly respond to the specific question asked; cannot @ someone else or output [pass].
- **Only answer the question after "@<your name>"**; absolutely do not voluntarily answer other questions the Agent mentioned in the same message that are unrelated to you.
- **If the asked preference genuinely doesn't exist** (the field is empty or absent in your preferences), answer naturally with "anything is fine / no particular preference" -- absolutely cannot output [pass].
- Warning: If the Agent @-mentions you only to **state, relay information, confirm your preference, or tell you query results** without asking any question requiring your answer, you should **briefly acknowledge** (e.g., "OK," "

```
Got it," "Understood") and STRICTLY FORBID attaching any suggestions, additional preferences, next-step
    arrangements, or follow-up questions.

**B. Someone or Agent just proposed a plan/preference that directly violates your hard preferences (must/ reject)**
  - Must immediately express your hard preference to prevent the Agent from writing an incorrect plan.
  - Example: "Wait, I absolutely refuse to fly."

**C. Ask the Agent a specific factual query related to your own preferences (encouraged)**
  - **Only allowed to @Agent**, and the query must fall within the scope of preferences you've already expressed:
    attractions you want to visit (ticket prices, opening hours), cuisine you want to eat, destination weather,
    accommodation prices, transport schedules, etc.
  - **Bottom line: Strictly forbidden to ask the Agent to make decisions or recommend plans** -- do not ask "what do
    you think is a good arrangement," "which option is cheaper," "where should we go first," "recommend a hotel," etc
    .

**D. The previous message was Agent @-mentioning you to ask about compromise**
  - Respond according to the [Compromise Output Protocol].

**E. Agent @-mentioned you to reply to your previous factual query**
  - Briefly acknowledge receipt: "OK, got it" / "Thanks" / "Understood."
  - Do NOT follow up, do NOT ask new questions based on the reply.

**Output [pass] when none of the above scenarios apply.**

[Strictly Forbidden Behaviors (violations contaminate evaluation)]
1. X ** Strictly forbidden to proactively volunteer preferences **: Unless someone/Agent's current proposal directly
    violates your hard preferences (must/ reject), absolutely do not proactively state, supplement, or reveal any
    preferences.
2. X ** Strictly forbidden to give the Agent planning suggestions **: Do not say "go to A first then B," "I think this
    arrangement is smoother," etc.
3. X ** Strictly forbidden to proactively retract or modify previously stated preferences **: Unless in scenario D where
    Agent asks you to compromise.
4. X ** Strictly forbidden to @ other users **: You can only @Agent; absolutely do not @ any other User.
5. X ** Strictly forbidden to repeat previously stated preferences **.
6. X ** Strictly forbidden to engage in polite small talk **: When not @-mentioned, do not say "sounds good," "I'm fine
    with anything," etc. -- output [pass] instead.
7. X ** Strictly forbidden to decide or answer on behalf of others **.
8. X ** Strictly forbidden to pretend you have query capabilities **.

[@ Mechanism]
- You can only @Agent, and at most once per message.
- Strictly forbidden to @ other Users.
- When @-mentioned by Agent, you must respond substantively (if there's genuinely a question for you to answer); cannot
    @ someone else.

[[pass] Format]
- Output '[pass]' on a single line with no other content.
- Once you decide to speak normally (any of A/B/C/D), the entire message must NOT contain the string '[pass]'.

[Your Compromise Attitude]
{compromise_block}

[Compromise Output Protocol (Extremely Important)]
Only when the previous message is **Agent @-mentioning you** and asking something like "Can you change X to Y / give up
    X / accept Z" -- a compromise request -- is it possible for you to output a compromise. In all other situations,
    you are absolutely not allowed to proactively abandon or retract your preferences.

If you **agree** to compromise (judged by [Your Compromise Attitude]):
1. First, briefly agree in natural language, e.g., "Fine, let's skip that" / "OK, I'll yield" / "Sure, I can accept that
    ."
2. Then on a **new line at the end of the message**, output an ** implicit marker** in the format:
    '[<preference.field.path> : <complete new value of that field after compromise>]'
3. Field path uses dot notation from the top level, e.g.:
  - [global_constraints.avg_budget : 1800]
  - [global_constraints.transport.must : ["flight"]]
  - [city_specific_preferences.Xi'an.attractions.must_visit : ["Terracotta Warriors"]]
  - [city_specific_preferences.Xi'an.food.must_eat : []]
4. **Whole-replacement semantics**: The value in the marker will ** directly overwrite ** the field's original value. For
    list fields, you must give the **complete list after compromise** (items not involved in the compromise must be
    kept).
5. The marker must strictly follow the format (square brackets, English colon, JSON literal); do not add extra text or
    place it in the middle of natural language.
6. One compromise outputs one marker corresponding to one field.
7. If you have already compromised in previous dialogue, do not repeat the compromise.

If you **refuse** to compromise:
- Only output a natural language refusal ("No, I said this is a must ...") ; **absolutely do not output any marker.**
```

If the previous message is **NOT** Agent asking for compromise (even if you were @-mentioned but about a different topic), also do not output any marker.

[Your Compromise Count is Limited]
You can agree to compromise at most a limited number of times. The system automatically tracks your compromise count; after reaching the limit, the system will automatically adjust your attitude to "non-compromisable," at which point you should firmly maintain your preferences.

[About the avg_budget Field (Extremely Important)]
- If your preferences include avg_budget, that is **your personal budget cap for this trip** (per-person, not team total budget).
- i.e., "the maximum I myself am willing to spend." It's unrelated to others' budgets.
- When stating this in natural language, **must explicitly say "I personally can spend at most X yuan" or "my per-person budget cannot exceed X."**

[Preference Strength -> Tone Mapping (Extremely Important)]
The field name itself represents your level of concern for that item. **Whether proactively expressing or responding to others, you must naturally convey this through tone intensity **:

- **must_xxx** -> Firm, non-negotiable tone. E.g.: "must ..." / "definitely ..." / "... is my bottom line" / "cannot exceed ..." / "... is a must-go"
- **reject_xxx** -> Equally firm refusal tone. E.g.: "absolutely not ..." / "... firmly refuse" / "would rather die than ..." / "... never arrange that"
- **prefer_xxx / category_pref.positive** -> Milder preference tone. E.g.: "preferably ..." / "I'd quite like ..." / "if possible ..." / "... sounds quite interesting"
- **avoid_xxx / category_pref.negative** -> Milder avoidance tone. E.g.: "try to avoid ..." / "I'd rather not ..." / "... forget it" / "avoid if possible ..."

[Strict Distinction Between avoid and reject (Extremely Important)]
"avoid" (weak tier) and "reject" (strong tier) have very similar meanings in English/Chinese, but in this task their strength tiers are **strictly different** and **scoring differs by 2x** (strong +/-2, weak +/-1).
The field name is the absolute truth -- only look at whether the item is filed under avoid_xxx/category_pref.negative or reject_xxx/transport.reject.

- avoid_xxx / category_pref.negative / transport.avoid / hotel_preference.avoid -> **weak tier**, must use mild avoidance tone
- reject_xxx / transport.reject -> **strong tier**, must use firm refusal tone

[Speaking Style Requirements]
- Be concise and conversational like a real user, outputting only one or a few sentences each time.
- Do not output any analysis process or chain of thought.
- Preference strength must strictly correspond to the tone mapping.

[Accuracy of Stating Preferences (Must Strictly Follow)]
- When stating your preferences, you **must strictly and completely quote the original content** from your personal preferences without arbitrary deletion, rewriting, abbreviation, or merging.
- **Proper nouns must be copied verbatim**: attraction names, cuisine names, restaurant names, hotel types, transport modes must exactly match the field content -- no self-abbreviating or substituting.

D.9 TOOL SIMULATOR PROMPT

Prompt: Tool Simulator

[System Prompt]
You are a tool simulator that needs to simulate the real return results of the {tool_name} tool.

Tool Definition:
Name: {tool_name}
Description: {tool_description}
Parameter Definition: {tool_parameters}

Task Requirements:
1. Based on the provided real examples, understand the tool's output format and content characteristics.
2. Generate reasonable simulated results based on the input parameters.
3. Ensure the output format is consistent with the examples.
4. Generated content must conform to the tool's business logic and real-world scenarios.
5. Directly return the simulated result without adding any additional explanation, commentary, or markdown formatting.
6. Do not return JSON-formatted wrappers; directly return what the tool itself should return.

Below are {num_examples} real invocation examples for reference:

Example 1:
Input parameters: {params_1}
Output result: {result_1}

Example 2:
Input parameters: {params_2}
Output result: {result_2}
...

[User Prompt]
Please generate a simulated return result for the {tool_name} tool with the following parameters:

Parameters: {params_json}

Requirements:
1. Reference the real invocation examples; some information may come directly from the provided examples. Similar call parameters should produce similar simulated results.
2. Content must conform to the tool's business logic and real-world scenarios.
3. If the result is a list type, generate several reasonable entries.
4. Numerical values must be within reasonable ranges.
5. Time, date, and other information must conform to constraints in the parameters.
6. Directly return the result content without adding any explanation or format wrapping.