MAESTRO: Adapting GUIs and Guiding Navigation with User Preferences in Conversational Agents with GUIs

Sangwook Lee
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
sangwooklee@vt.edu

Adnan Abbas
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
adnana99@vt.edu

Sang Won Lee
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
NAVER AI Lab
Seongnam, Republic of Korea
sangwonlee@vt.edu

Young-Ho Kim
NAVER AI Lab
Seongnam, Republic of Korea
yghokim@younghokim.net

Yan Chen
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
ych@vt.edu

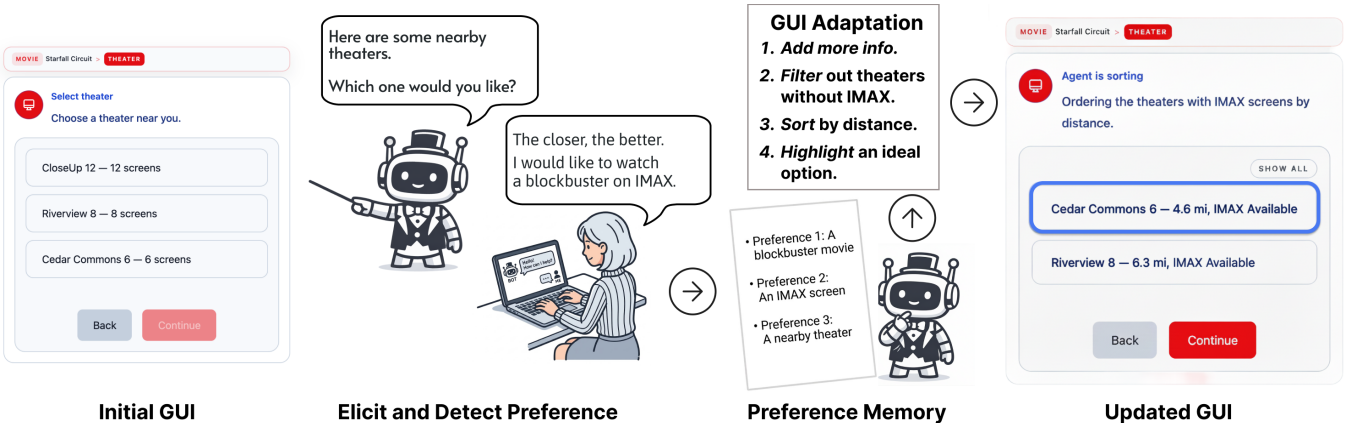

Figure 1: Overview of MAESTRO at the theater selection stage. Initial GUI: The default GUI presents three theater options with names and screen counts. Elicit and Detect Preference: The agent asks about theater choice, and the user replies with preferences. Preference Memory: MAESTRO extracts three preferences and plans four GUI adaptation operators. Updated GUI: The adapted GUI shows only IMAX-available theaters sorted by distance, enabling the user to compare and confirm directly.

Abstract

Modern task-oriented chatbots present GUI elements alongside natural-language dialogue, yet the agent's role has largely been limited to interpreting natural-language input as GUI actions and following a linear workflow. In preference-driven, multi-step tasks such as booking a flight or reserving a restaurant, earlier choices constrain later options and may force users to restart from scratch. User preferences serve as the key criteria for these decisions, yet existing agents do not systematically leverage them. We present MAESTRO, which extends the agent's role from execution to decision support. MAESTRO maintains a shared preference memory that extracts hard and soft preferences from natural-language utterances and provides two mechanisms. *Preference-Grounded GUI Adaptation* applies in-place operators (augment, sort, filter, and highlight) to the existing GUI according to preference strength, supporting comparison among options. *Preference-Guided Workflow Navigation* detects conflicts between preferences and available options, proposes backtracking, and records failed paths to avoid revisiting dead ends. Through a controlled experiment ($N = 33$), we demonstrated that MAESTRO improved decision quality in movie ticketing: final bookings left fewer hard preferences unmet, and users made fewer selections that violated their stated preferences during the process than in the baseline condition, although task success rate and completion time did not differ significantly. In addition, we showed that using MAESTRO in voice mode can increase users' active engagement as well as their mental burden, revealing the nuanced tension in agentic interaction design for conversational agents with a GUI.

UIST '26, Detroit, MI, USA

ACM ISBN 979-8-4007-2856-3/2026/11
https://doi.org/10.1145/3830398.3830672

CCS Concepts

• **Human-centered computing** → **Natural language interfaces**; **Graphical user interfaces**; *User interface management systems*; *Empirical studies in HCI*.

Keywords

conversational agents with GUIs, adaptive user interfaces

ACM Reference Format:
Sangwook Lee, Adnan Abbas, Sang Won Lee, Young-Ho Kim, and Yan Chen. 2026. MAESTRO: Adapting GUIs and Guiding Navigation with User Preferences in Conversational Agents with GUIs. In *The 39th Annual ACM Symposium on User Interface Software and Technology (UIST '26), November 02–05, 2026, Detroit, MI, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3830398.3830672

1 Introduction

Conversational Agents with GUIs (CAGs) are chatbots that interleave structured GUI widgets, such as buttons or drop-down menus, with natural-language dialogue. They are proliferating across domains: personal banking (Erica [3]), customer support (Xfinity [3]), and recommendations (Amazon Rufus [9], Booking.com [4]). GUIs accelerate input, organize options for visual comparison, and guide intent through predefined choices [32]. Yet interacting with a GUI through natural language does not automatically unlock its full expressive potential.

In current CAGs, the agent's role is largely limited to event handling, mapping GUI actions to system behaviors, or translating users' natural-language utterances into GUI actions (e.g., typing "Yes" is equivalent to pressing the "Yes" button). This mapping overlooks the potential to adapt GUIs or streamline workflows based on contextual information gathered during the conversation [7, 28]. For example, a user who told a CAG they were purchasing two tickets for a date night could have those tickets pre-selected during the ticket selection stage, based on the previous conversation. As a further example, consider a CAG that supports troubleshooting a Wi-Fi connectivity problem: a user who says they restarted their router may skip a binary Yes/No button press asking whether they did so, which is typically the first question of the fixed GUI workflow. This potential is particularly pronounced in preference-driven, multi-step tasks like booking a movie ticket: earlier choices (theater, date) constrain later options (IMAX availability), often forcing users to backtrack [5, 13]. The CAG can provide agentic decision support [11], proactively reflecting user preferences in the current GUI and steering subsequent workflow steps.

To address these gaps, we present **MAESTRO** (**M**ultimodal **A**gent **E**mpowering **S**election by **T**ailoring GUIs, **R**ecalling preferences, and **O**rchestrating exploration), which supports users' decision making through GUI adaptation and workflow navigation assistance. MAESTRO maintains a *Preference Memory* that structurally extracts preferences from users' natural-language utterances and provides two capabilities grounded in this memory. **Preference-Grounded GUI Adaptation** applies in-place operators (augment, sort, filter, highlight) to manipulate information representation within the existing GUI without altering its structure, so that users can be informed about their available choices within the context of presented GUIs, making it easy to select and compare options. **Preference-Guided Workflow Navigation** detects conflicts between preferences and available options, proposes a specific step to return to, and logs the current path to avoid revisiting the failed paths. While recent systems externalize user intent into structured representations that generate new interfaces [6, 25], MAESTRO uses its Preference Memory to adapt an existing GUI in place, preserving the familiar context of a structured workflow.

We investigate two research questions:

- **RQ1** How do preference-based GUI adaptation and navigation guidance in CAGs affect decision-making performance?
- **RQ2** How do users perceive and experience such agentic interventions by CAGs?

While it is not our central research question, we also investigate how interaction modality (text vs. speech) influences decision-making performance and users' perceptions of the agent as a subquestion of RQ1 and RQ2.

To answer these research questions, we evaluated MAESTRO through a 2×2 within-subjects study ($N = 33$) using a movie-ticketing CAG, crossing Condition (Baseline vs. MAESTRO) with Mode (Text vs. Voice). Mode is a secondary factor: prior work shows that interacting with an agent by voice rather than through a screen changes how actively users engage with agent-assisted tasks [37], and speech may amplify the effect of MAESTRO by making preferences easier to express, particularly in hands-free contexts such as automotive UI, smart displays, or AR glasses. In the Baseline condition, the agent delivers information as textual advice within a single chat-stream layout with GUIs and without Preference Memory. In the MAESTRO condition, the agent additionally adapts a persistent GUI panel through in-place operators and steers exploration via Preference Memory. Each task involved a set of requirements that a participant needed to satisfy (e.g., a kid-friendly movie on Saturday for three at the closest theater). We evaluated performance through task success, preference violations, unpreferred selection count, task completion time, utterance pattern analysis, and subjective measures, including perceived usefulness, perceived burden, and preference rankings.

Our study yields three key insights. First, supporting preference-aware interaction through GUI adaptation yields measurable reductions in preference violations, both in the final outcome and during the decision process. Second, while it did not accelerate task completion, the system promoted more engaged decision-making, shifting user behavior toward expressing preferences and actively navigating the workflow. Third, interaction modality plays a critical role: while voice facilitates preference expression to a greater extent, it also introduces additional burden due to delays and turn-taking constraints. These findings demonstrate that integrating preference-aware adaptation and workflow guidance can be a new avenue for agentic behaviors of conversational agents with GUIs, highlighting both its benefits and design trade-offs. The research contributions of this paper include:

(1) Design and development of MAESTRO, a novel conversational agent that facilitates decision making in CAGs, equipped with GUI adaptation operators (augment, sort, filter, highlight) and preference-guided workflow navigation.

(2) Empirical findings from a controlled within-subjects study that evaluates the effects of agentic decision support in CAGs.

2 Related Work

2.1 Agents in GUI-Based Conversational Systems

Agents have been used as autonomous operators that manipulate GUI on natural language input on the user's behalf [46, 51]. Yet these agents share some limitations: they do not involve users at important decision points, and hallucinations remain a significant concern on complex, multi-step tasks [52, 54]. To restore user control, some systems introduce pause-and-override mechanisms. MIWA [8] supports step-through debugging and refinement in web automation, CowPilot [19] lets users pause or override agent actions during web navigation, and Morae [35] pauses execution at detected decision points. While these systems help support user control, they remain automation-centric: agent execution is the default mode, and user intervention occurs only at identified breakpoints rather than in a collaboration-centric manner.

A parallel stream grounds conversational interaction in the GUI itself. Weidele et al. [47] study conversational control of GUIs in semantic automation, META-GUI [41] links GUI elements with dialogue context, and MALACHITE [38] provides a GUI-aware natural-language interface for complex applications. Traditional task-oriented dialogue systems similarly rely on intent detection and slot filling to guide task completion [27, 48]. Although these systems connect language to interface elements, the agent still mainly executes user intent within a fixed GUI. Recent work has also used LLMs to generate interface components or whole interfaces dynamically [1, 6, 7, 17, 31]. Such generation is increasingly guided by structured representations of user intent, including evolving task-driven data models [6] and just-in-time objectives [25], which define the task at hand and seed the synthesis of a new interface. However, generating *new* interfaces rather than adjusting an existing GUI in place disconnects the user from the familiar context of the workflow.

A growing body of work instead adapts *existing* interfaces through natural language rather than generating new ones at each turn. Stylette [22] maps styling goals to CSS edits, DynaVis [44] creates manipulable widgets for visualization editing, and DirectGPT [30] supports in-place modification of selected objects. These systems show that natural language can support in-situ GUI changes, but each interaction is largely self-contained. Recent works such as IRF [35] and CARE [34] explored sustained interaction by updating interface content as users refine preferences over time. Still, these systems largely position the agent as the primary executor of user intent.

2.2 Decision Support with Preference Management

Conversational recommender systems (CRSs) have long studied how to elicit and refine user preferences through dialogue [10, 20, 43]. Recent LLM-based systems extend this work with stronger preference elicitation, explanation, and recommendation [14, 16, 24]. For example, RecLLM [15] builds user profiles from conversation history to personalize recommendations. However, these systems mainly operate within a single recommendation stage, where one round of elicitation leads to one set of results. In multi-step tasks such as travel booking, earlier selections constrain later options, and users must compare alternatives, revisit previous choices, and balance competing preferences across stages [5, 13]. Qin et al. framed this complexity as a constrained optimization problem and found that LLMs are good at following hard constraints, but perform poorly in following soft preferences [36]. This motivates further research in designing CUIs for preference-driven multi-step tasks.

For cross-stage preference management, the system must track preferences as they change over longer conversations. Recent evaluations suggest this remains difficult. PrefEval [53] shows that preference-following accuracy drops below 10% by 10 turns in zero-shot settings. PERSONAMEM [21] finds that frontier models reach only around 50% accuracy in recognizing dynamic profile changes, and CUPID [23] reports under 50% precision in inferring contextual preferences from multi-turn histories. These results motivate structured external representations instead of relying only on the LLM context window. Recent works have used natural-language records, reflection-driven summaries, and confidence-weighted propositions about user behavior [33, 39]. These systems show that external representations support more reliable capture and retrieval of user information, guiding the way for dedicated preference stores for sustained, preference-aware interaction in multi-step settings.

The remaining challenge is how to apply stored preferences during the task, both for interface *adaptation* and workflow *navigation*. Within a stage, option presentation affects decision quality: structuring choices around a preference model and showing trade-offs improves spoken-dialog performance [11, 12], while in visual interfaces, format, sorting, and filtering affect search time and decision confidence [18, 42, 45]. Nav Nudge [49] combines voice input with an LLM to highlight relevant options on a mobile GUI. Across stages, accumulated preferences may conflict with available options, forcing the system to decide how to proceed. The Frames corpus [13] shows that users naturally revisit prior options and compare alternatives, making frame tracking a core dialogue capability. However, these adaptations are driven by fixed rules or direct manipulation rather than by an agent interpreting ongoing dialogue.

2.3 Summary

Across the approaches reviewed in Section 2.1 (autonomous execution, mixed-initiative pausing, GUI-aware dialogue, generative interfaces, and language-driven adaptation), agents either act on behalf of the user within an existing GUI, generate entirely new interfaces, or adapt a single interface state in response to one-shot commands; none adaptively modify an existing GUI based on preferences accumulated over a multi-turn conversation to support ongoing decision making. Unlike intent representations that define a task and seed the generation of its interface [6, 25], MAESTRO's Preference Memory augments and curates an existing GUI and steers navigation within its fixed workflow. Research on preference management (Section 2.2) provides elicitation loops, memory mechanisms, and local conflict-handling strategies, but these remain confined to single-stage settings or isolated queries,

and LLMs alone cannot reliably sustain preference tracking over extended conversations. MAESTRO bridges these two streams: it interprets preferences expressed in dialogue to apply in-place GUI adaptation operators (augment, sort, filter, highlight), and it structurally tracks preferences across workflow stages to detect conflicts, propose backtracking, and remember failed paths.

3 System Description: MAESTRO

MAESTRO comprises three modules built upon a Conversational Agent with GUI (CAG): an agent architecture that serves as the base system; a Preference Memory that extracts user preferences as structured records and maintains them throughout the session (Section 3.3); and two modules informed by Preference Memory, Preference-Grounded GUI Adaptation (Section 3.4) and Preference-Guided Workflow Navigation (Section 3.5). The agent architecture alone constitutes a fully functional CAG; MAESTRO extends it with the preference memory and both modules. All inference tasks described in this section, including generating responses, extracting preferences from conversations, deciding to adapt GUIs, and guiding navigation steps, are powered by OpenAI GPT-5.4.

3.1 Target Domain: Movie Ticketing Assistant

While the approach is applicable to domains where a CAG can be used, we focus on a specific domain—a movie ticketing agent—to demonstrate it. We introduce the target domain here before describing the system, as the examples will be used in the system description.

The system follows a linear multi-stage workflow (movie, theater, date, time, seat, and confirmation), where a user has to select one option at each stage (at the seat stage, one or more seats). Each stage is associated with a dedicated GUI component: button groups for movie, theater, and time selection; a calendar for date selection; and a seat map for choosing seats. This workflow is modeled after the information structure of major movie-ticketing services such as Fandango or AMC; stages that co-occur on a single screen in production services (e.g., theater, date, and showtime) are separated into individual stages so that each decision point can be addressed independently. Users may navigate backward to revise earlier choices, and skipping or reordering stages is not permitted.

3.2 Interface Layout

MAESTRO divides the screen into a GUI panel on the left and a conversation panel on the right (Figure 2). The GUI panel persistently displays the current stage's options and the agent's adaptation actions with smooth animation. (see Section A.5) While typical CAG designs present a linear stream of chat and GUI elements within a scrollable window, we found that this approach can make it difficult for users to comprehend the interface. In particular, GUI elements may be pushed out of view as users and the agent add new chat messages, or the GUI adaptation module may append entirely new GUI containers, making the intermixed history of GUI and chat unnecessarily lengthy. Placing the GUI panel on one side ensures users can immediately see what has changed. Scrolling the conversation panel synchronizes the GUI panel, allowing users to review previous GUI states associated with each chat message. This preserves the ability to inspect conversation history that users

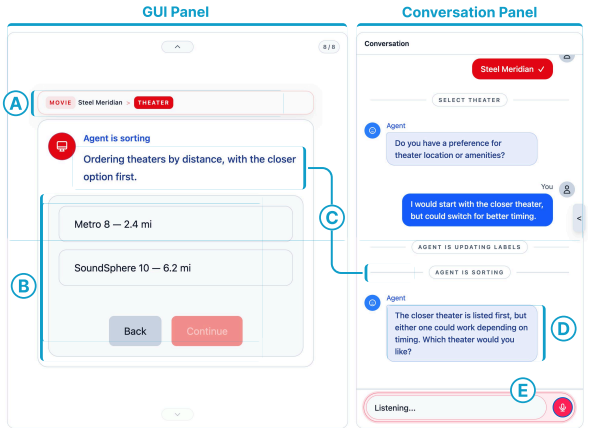

Figure 2: MAESTRO layout at the theater selection stage. (A) Workflow breadcrumb showing current progress. (B) The GUI shows distance augmentation and proximity-based sorting. (C) Adaptation actions shown in the chat panel. (D) Agent follow-up summarizing its actions and information, prompting the next decision. (E) Voice input interface.

expect from typical chat interfaces, where prior messages and associated GUI states remain accessible.

3.3 Preference Memory

Preference Memory serves as the shared state for Preference-Grounded GUI Adaptation (Section 3.4) and Preference-Guided Workflow Navigation (Section 3.5). Preferences are extracted from users' natural-language utterances as structured records and maintained throughout the session, so that the current screen's representation and the navigation path are consistently grounded in the same preference state.

Extraction and update. Preference extraction is invoked on every user message. The underlying system prompt of the agent is designed to elicit users' preferences (e.g., "Do you have a preference for theater location or amenities?", See Figure 2). While merely selecting an option is not recognized as a preference, explicit expressions of intent, such as constraints and desires, are stored as preferences. When a new preference overlaps or conflicts with an existing one, the record's preference is updated.

Preference Record Structure. Each preference is represented as a record with three properties. The `description` field captures the preference as a natural-language description. The `strength` field is either `hard` (must be satisfied; cued by words such as "must," "only," "need") or `soft` (satisfy if possible; cued by "prefer," "ideally," "want"); ambiguous cases default to soft. The strength property is later used to determine which adaptation policy to use. The `relevantStages` field lists the workflow stages to which the preference applies, assigned by an LLM agent. An example, included in Section A.1, illustrates a representative set of preference records.

3.4 Preference-Grounded GUI Adaptation

Preference-Grounded GUI Adaptation dynamically changes the presentation of GUI elements based on the preferences stored in Preference Memory.

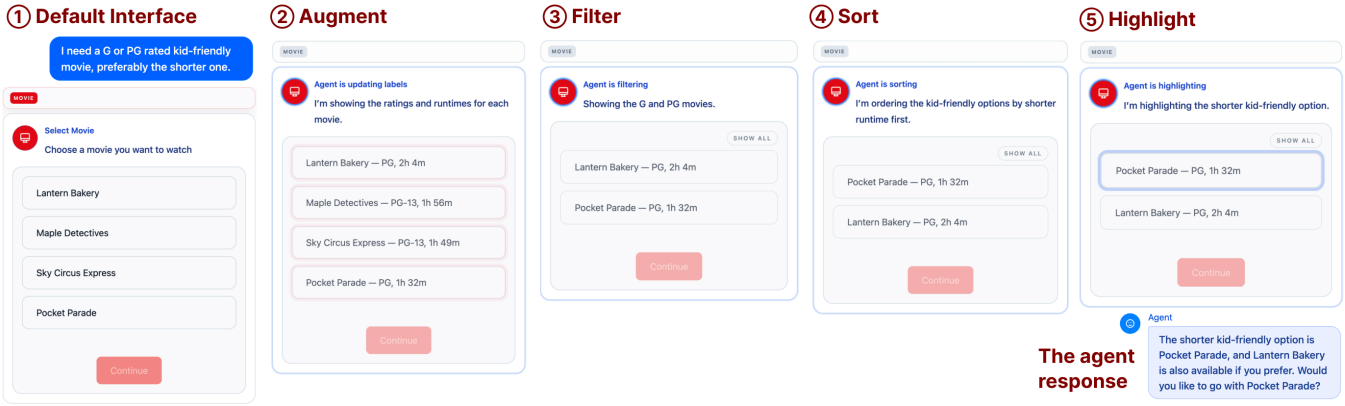

Figure 3: Four-step adaptation policy illustrated on a movie-selection example. ① Default interface ② Augment: Ratings and runtimes appended to each button. ③ Filter: PG movies filtered out ④ Sort: Filtered movies are sorted by runtime. ⑤ Highlight: Shortest option highlighted with agent confirmation.

3.4.1 Adaptation Operators. To support decision-making, MAESTRO reorganizes the presentation of information based on the current GUI structure and the user's preferences so far. Instead of generating GUI elements from scratch, we developed four GUI adaptation patterns that update GUI elements in place.

Augment operator appends additional attributes to the text displayed on each button. As described in Section 3.1, each button shows only a single key attribute by default; the Augment operator augments text with metadata available for an item (e.g., audience ratings next to movie titles, or screening end times alongside showtimes). Surfacing all decision-relevant attributes at once, rather than requiring users to request them incrementally, improves task completion and satisfaction [12].

The *Filter* operator curates information by filtering out items that do not meet a user's requirements. For example, it hides movies that are not suitable for all ages when a user specifies their preference for a family movie night with a 6-year-old kid. Filtering is commonly used and validated in dynamic query and faceted search research [42].

Sort operator reorders the same set of items according to a selected attribute, especially when the selected attributes are numeric. An example is reordering the movie list by audience rating when a user asks for ratings of the movies on screen. According to the literature, sorting facilitates comparative judgment [18].

Highlight assigns visual emphasis by adding a colored border to one or more elements, directing attention without adding or removing information. Visually emphasizing relevant options has been shown to help users locate target features more quickly [49, 50]. Examples include highlighting a recommended showtime or four adjacent seats in a seatmap.

These four adaptation operators span the space of in-place information transformations that preserve GUI structure (layout, components, and navigation remain intact). All four are implemented as lightweight frontend in-place modifications.

3.4.2 Applying GUI Adaptation. In this section, we explain how MAESTRO applies GUI adaptation operators through an example shown in Figure 3.

At each stage, MAESTRO determines which adaptation operators to apply to the current GUI elements based on the relevant preferences logged in Preference Memory. In Figure 3-①, the user states two preferences: "G or PG rated kid-friendly movie" (`hard`) and "preferably the shorter one" (`soft`). Once such preferences are identified, MAESTRO will first *augment* the GUI elements with metadata relevant to the preferences; each movie option will now have ratings and runtimes appended (Figure 3-②).

Because the first preference's strength is hard, MAESTRO applies the *filter* operator to hide options that do not satisfy it, such as R-rated movies or showtimes that start after 5 PM. At stages where filtering is not applicable (Calendar and Seat-map stages), highlight is used instead to mark all matching items. The user can always restore filtered-out items using the "Show All" button at the top-right of the UI container. (③).

For the `soft` preference, which reflects a user's preference for shorter movies, MAESTRO applies the *Sort* operator and uses metadata to reorder items (④). The *Sort* operator is effective when the metadata type is ordinal or numeric, such as ratings, price, and distance. The optimal option is then highlighted (⑤). Categorical metadata, such as genre or screen type (e.g., IMAX), cannot be used with the Sort operator; instead, the *Highlight* operator visually emphasizes options that satisfy a user's soft preference.

Lastly, once GUI adaptation is complete, the MAESTRO agent will pose a confirmation question in a follow-up message (e.g., "Would you like to go with this option?"). Rather than silently waiting for a selection, the agent proactively proposes the outcome of its adaptation and invites the user to accept, reject, or revise it. Not all four operators apply in every scenario; however, when any of the other three operators is used, the Augment operator always accompanies it, surfacing the metadata that justifies the adaptation.

The Preference Memory enables reapplying the GUI adaptation. For example, when a user wants to watch a movie on an IMAX screen and later selects another movie, MAESTRO filters out showtimes that are not on IMAX screens during the showtime stage.

3.5 Preference-Guided Workflow Navigation

When MAESTRO guides users through a workflow based on their preferences, conflicts can arise when those preferences (e.g., two adjacent seats) cannot be met due to real-world constraints (e.g., all available seats are single seats). In such cases, users must seek alternatives and choose different paths. However, they may forget which step to return to in order to find alternatives that satisfy other preferences, or attempt to revisit a dead end they have already encountered. To address this issue, the Preference-Guided Workflow Navigation module assesses the viability of the current navigation path and recommends a step the user should return to when conflicts arise.

3.5.1 Backtracking Suggestions based on Alternative Counts. MAESTRO tracks the number of remaining alternatives at each stage along the navigation path. When computing these counts, it leverages the results of Preference-Grounded GUI Adaptation. For example, if three theaters are available and filtering by a free-parking preference narrows the set to two, the one remaining theater (excluding the user's current selection) is stored as the alternative count for the theater stage. This calculation applies when the agent performs a filter action; for highlight actions, it applies only when the highlight was used to reduce choices rather than for visual comparison. For instance, at the date stage, only the highlighted dates are counted as viable alternatives.

These per-stage alternative counts are injected into the agent's input when a conflict is detected, so the agent can suggest an appropriate step to backtrack to. For example, if the agent perceives that four adjacent seats are unavailable at the current seat-selection stage, it receives the alternative counts from all preceding stages. If the user had indicated that only a specific date was acceptable (highlighting that date as the sole option), the date stage would have zero alternatives. If the theater stage preceding it had at least one alternative, the agent identifies the theater stage as the appropriate backtrack target and proposes returning to it, rather than going back to the date stage, which has no other alternatives.

3.5.2 Dead-End Recording. When the user accepts the backtrack suggestion due to the lack of a viable option, MAESTRO records the current path as a dead end. This record is then injected into the agent's input on subsequent turns, so the agent can avoid re-exploring previously failed paths. Each dead end is scoped to the specific combination of preceding selections that produced it. For example, if selecting Movie A (Stage 1) and Theater B (Stage 2) yields no viable showtimes (Stage 3), Theater B is excluded on that path given Movie A, but it remains a valid option when a different movie is selected on Stage 1.

3.5.3 Updating Alternative Counts and Dead-Ends When a Preference Changes. Because alternative counts and dead-end records are each linked to a specific entry in Preference Memory, MAESTRO can automatically update them when a preference changes. For example, if several paths were recorded as dead ends because no IMAX showtime was available, and the user later relaxes that constraint, the dead-end records tied to that preference (i.e., IMAX screen) are removed, and those paths become available for exploration again. Filter and highlight adaptations linked to the changed preference are likewise cleared, and alternative counts are recalculated accordingly. The agent then replans based on the updated alternative counts and dead-end records, without requiring the user to manually retrace their steps.

3.6 Agent Architecture

At each turn, MAESTRO observes the current GUI state, including displayed options and any applied adaptations, together with underlying data attributes not visible in the interface (e.g., ratings, distance), the user's utterance, and prior selections. Based on this, it selects actions via LLM tool calling from a context-dependent toolset that includes standard GUI interactions (selection, navigation) and the adaptation operators described in Section 3.4. Preferences expressed early are mapped to relevant stages via the `relevantStages` field in Preference Memory, so they can be evaluated when those stages are reached.

When a preference changes, related dead-end records and adaptations are automatically updated or removed, enabling the agent to replan without requiring the user to retrace steps. Because the Preference Memory carries the core decision context, the conversation history is truncated to recent turns to mitigate the lost-in-the-middle problem [26], where language models miss intermediate context as conversations grow; the current stage's GUI state is supplied separately as a representative snapshot.

3.7 Implementation

MAESTRO consists of a React frontend that renders the GUI and conversation panels, a Fastify backend that serves the scenario database and session state, and an agent runtime connected to the frontend through a WebSocket relay. We implemented the agent's two LLM components, the preference extractor and the action planner, as structured prompts over OpenAI GPT-5.4. The extractor runs only in the MAESTRO condition, and the planner's prompt is assembled from per-condition rule blocks. Each component calls the Responses API with temperature set to 0 and requests structured JSON output; the full system prompts appear in Section A.6. Responses that fail schema parsing are salvaged when possible and otherwise discarded, leaving the interface state unchanged for that turn. Voice mode uses OpenAI gpt-4o-mini-transcribe for speech recognition and gpt-4o-mini-tts for spoken output. Our implementation is publicly available.[1]

4 Evaluation Method

To evaluate whether MAESTRO improves users' decision-making compared to the current CAG paradigm, we conducted a within-subjects study in the movie-ticketing domain.

4.1 Study Design

4.1.1 2×2 Within-Subjects Experimental Design. All participants experienced four combinations of two factors: **Condition** (Baseline vs. MAESTRO) and **Mode** (Text vs. Voice). We denote each combination using the following labels: Baseline in Text mode (**BT**), Baseline in Voice mode (**BV**), MAESTRO in Text mode (**MT**), and MAESTRO in Voice mode (**MV**).

[1] https://github.com/wooogler/maestro-movie

The primary comparison is Condition (Baseline vs. MAESTRO), capturing the integrated effect of the agent's decision-support approach. The secondary comparison is Mode (Text vs. Voice), examining how input/output modality affects the decision process. The Condition × Mode interaction is explored to determine whether Voice amplifies or modulates the effect of MAESTRO. Because the same participant experiences all four cells, individual differences are controlled.

4.1.2 Condition: Baseline vs. MAESTRO. In the *Baseline condition* (Figure 4), participants used a single chat-stream layout common to most existing CAGs. The GUI component for the current stage was persistently displayed below the most recent message, and GUI components from previous stages remained visible in the chat history but were locked. Because no GUI adaptation tools were available, the screen did not change dynamically. In the *MAESTRO condition* (Figure 2), the screen was divided into a persistent GUI panel and a conversation panel, as described in Section 3.1.

The baseline agent still had many useful features. Both conditions provide the agent with access to the same backend metadata; the difference lies solely in how the agent supports the user's decision process. Both conditions perform proactive preference elicitation—the agent actively asks about preferences upon entering a stage. In the Baseline, elicited preferences are used only within the conversation history; in MAESTRO, they are stored as structured preference records. The baseline agent thus represents a modern, GUI-integrated, task-oriented chatbot: its tools are limited to standard GUI interactions, namely selecting an option and moving to the next or previous stage, and in place of a structured memory module such as Preference Memory, it uses the conversation history as its only memory. That history covers the full session, whereas the MAESTRO condition truncates it to recent turns (Section 3.6); the system prompts of both conditions are reproduced in Section A.6. Agents that persist structured records across sessions (e.g., ChatGPT's Memory) target long-term personalization and lie outside this comparison.

4.1.3 Mode: Text vs. Voice. The Mode factor varies the natural-language input/output channel: **Text** mode uses text typing plus GUI clicks with text-based agent responses, while **Voice** mode uses speech input plus GUI clicks with spoken agent responses and a text transcript. In Voice mode, the agent's spoken output was queued and played to completion before the input turn returned to the participant; once the participant activated the microphone, automatic turn-taking detected the end of the participant's utterance and triggered transcription. In both modes, participants can make simple selections via GUI clicks; the difference lies only in the natural-language channel. We used a commercial speech-to-text and text-to-speech engine to implement the Voice mode.

4.2 Domain and Task Design

4.2.1 Domain: Movie Ticketing. We chose movie ticketing as the study domain based on a content analysis of four major platforms (Fandango, AMC, Regal, Cinemark). All four share a common structure: Movie → Date/Theater/Time (compressed onto one screen) → Seat → Payment. We decomposed this into the six-stage linear workflow described in Section 3.1. Because each stage's available options depend on selections made in earlier stages, this sequential structure forms the basis for cross-stage conflicts that MAESTRO's Workflow Navigation is designed to address.

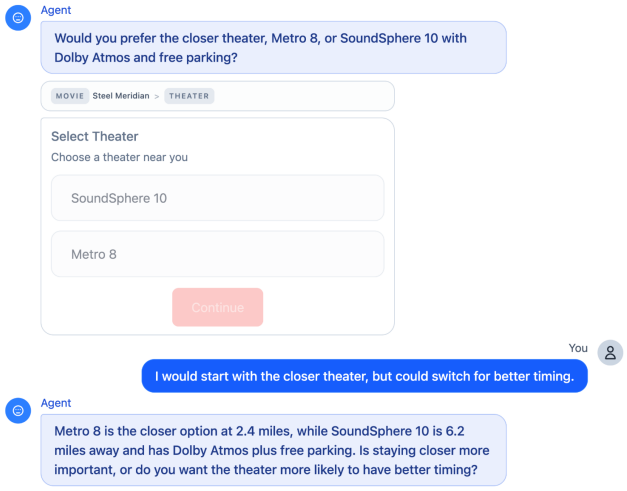

Figure 4: Baseline condition interface at the theater selection stage, designed after a typical CAG. A single chat-stream layout where GUI components and agent messages are mixed in a linear chat history. The baseline agent also has access to all metadata.

4.2.2 Task Design. Eight tasks are organized into four task sets (T1–T4) to avoid learning effects from repeating the same task, with each set containing one warm-up and one main task. Warm-up tasks require 8 stage visits and 1 backtrack on the success path; main tasks require 18 visits and 2–3 backtracks. Warm-up tasks familiarise participants with the current condition and are excluded from the main analysis.

All tasks share the same interaction skeleton (stage order and UI types) but use different scenario backgrounds and scenario-specific databases. Each task is presented as a scenario that includes a title, a brief background story, and a set of per-stage preference descriptions. Preferences are linguistically divided into *hard* and *soft*: hard preferences use obligatory language (e.g., "I need a G or PG-rated movie"), while soft preferences use hedged language (e.g., "I prefer the closest theater" or "preferably the shorter one"). Each scenario and paired data set has a single path that satisfies all the specified hard preferences. Example task prompts are shared in Section A.2. We typically used soft preferences to guide users toward a dead end and allowed them to use the system to return to previous steps to find a viable solution, indicating that no path satisfied both hard and soft preferences.

4.3 Procedure

Each participant completed one task set (T1–T4) under each condition. Task–condition pairings and presentation order were fully counterbalanced using a Latin-square design (Table 1), requiring N to be a multiple of 16 (4×4) to control for learning, order, and task–condition pairing effects.

Table 1: Latin-square Assignment of Task-Condition Group to Order for the first four participants.

Group	Round 1	Round 2	Round 3	Round 4
Group 1	T1 · BT	T2 · MV	T3 · BV	T4 · MT
Group 2	T2 · MT	T3 · BV	T4 · MV	T1 · BT
Group 3	T3 · BV	T4 · MT	T1 · BT	T2 · MV
Group 4	T4 · MV	T1 · BT	T2 · MT	T3 · BV

Participants joined remotely from personal laptops. The protocol proceeded as follows: (1) consent and pre-study survey; (2) mandatory screen recording setup (webcam optional); (3) tutorial via embedded video; (4–7) four rounds, each consisting of a warm-up task followed by a main task and a per-cell survey; (8) post-study survey with ranking and per-feature evaluation; and (9) a semi-structured interview (~10 minutes).

In each round, the warm-up task familiarised the participant with the current condition. During the warm-up, the researcher observed the session with their microphone and camera on and was available to answer questions about the system's use. The warm-up was occasionally skipped when the researcher judged the participant to be already sufficiently familiar with the condition. For the main task, the researcher turned off their microphone and camera and provided no assistance, allowing participants to work entirely on their own. Per-cell surveys were administered after each main task (4 times in total). The estimated session duration was approximately 90 minutes, and participants were compensated $30.

Each task provides a scenario background and preference descriptions. Constraints are not stated in the brief but discovered through exploration. The brief remains on screen throughout.

4.4 Dependent Variables

4.4.1 RQ1: Performance Measures. To assess the performance of MAESTRO, we developed objective measures as proxies for decision quality and efficiency. For decision-making quality, we compared their final submitted answer to the single solution that satisfies all the hard preferences in two ways: *Task Success Rate* (DV1)—whether the final booking matches the unique correct workflow, 0 if failed, 1 if successful—and *Violation Count* (DV2)—the number of hard preferences unmet; a lower value is better. We also measured *Unpreferred Selection Count* (DV3) — the total number of selected options that violate hard preferences across all stages within a task, which assesses the effectiveness of GUI adaptation during the process. For example, Unpreferred Selection Count can exceed Violation Count if a user repeats the same mistakes, as it captures errors made during the process, even if they are corrected later. Task Success Rate and Violation Count both assess the final outcome, with Violation Count grading the severity of failure that the binary success measure cannot capture, whereas Unpreferred Selection Count assesses the decision process leading to that outcome. For Efficiency, we measured *Task Completion Time* (DV4).

Another key factor in the success of CAGs is how users express their preferences. If users do not share their preferences and rely solely on GUI interactions, MAESTRO cannot be effective. As we implemented proactive and encouraging language to prompt users to articulate their preferences, we measured how much users interacted in natural language and expressed preferences that cannot be inferred from GUI interactions alone. We classified all user utterances, i.e., each message, into three functional categories using LLM-assisted coding validated against manual annotations: *Preference Statement Utterances* (DV5) — expressing a want, need, or priority (e.g., "I prefer the closest theater"); *Information-Seeking Utterances* (DV6) — requesting factual information not visible on screen (e.g., "How long is the movie?"); *Action Request Utterances* (DV7) — directing navigation or system actions (e.g., "Go back", "March 11th"). We also report *Total Utterances* (DV8).

4.4.2 RQ2: Perceived and Experiential Values. We used a standard questionnaire to evaluate the perceived value of MAESTRO compared to the baseline. We measured Ease of Use using five items drawn from the Difficulty of Use and Mental Burden subscales of the *User Burden Scale* (UBS) [40], averaging the five-point scale responses across these items. To measure the Perceived Usefulness of MAESTRO, we used four items from the *Perceived Usefulness* (PU) subscale of the Technology Acceptance Model [29], using a 7-point Likert scale (1 = Strongly disagree, 7 = Strongly agree).

Post-study survey was administered once after all four tasks. Participants ranked the four conditions (BT, BV, MT, MV) in order of preference and provided a rationale in open-ended responses.

Exit interview was administered at the end of the study and lasted approximately 10 minutes. The questions covered the most helpful features, confusing moments, relative preference between the two interface conditions, and suggestions for improvement.

4.5 Sample Size Justification & Recruitment

We needed $N = 32$ participants. The power analysis indicated that we need N=24 for a 2×2 within-subjects study with a medium effect size ($f = 0.25$), $\alpha = .05$, and power = .80. However, because the task-pair combinations and order groups were counterbalanced, we needed the number of participants to be a multiple of 16. Considering attrition, we recruited 36 participants in total. Two participants' data were excluded because technical issues (e.g., OpenAI API errors, system errors) disrupted their sessions, and one participant who failed all four main tasks was excluded from the analysis as an outlier, yielding $N = 33$.

We recruited participants through various mailing lists involving students at the authors' university. All participants (13 female, 20 male) were undergraduate or graduate students. Participants' ages were distributed as follows: 18–24 (n = 17, 51.52%), 25–34 (n = 13, 39.39%), and 35–44 (n = 3, 9.09%). The study lasted approximately 1.5 hours. All participants provided informed consent and were compensated $30 for their participation. The experimental protocol was reviewed and approved by the Institutional Review Board at our university.

4.6 Analysis

The primary analysis uses mixed-effects models with participant as a random intercept and Condition, Mode, and their interaction as fixed effects. The model family is chosen based on outcome type: binomial GLMM for binary outcomes (DV1), Poisson GLMM for count outcomes (DV2, DV3, DV5–DV8), and Gaussian LMM applied to log-transformed time values (DV4). For the composite scores of survey responses (UBS and PU), we analyzed them using a linear

mixed-effects model. Open-ended responses and semi-structured interview transcripts were analyzed using open coding; two researchers independently coded responses and consolidated codes through discussion until they agreed.

5 Results

All analyses use main-task data only (N = 33, 132 trials; one participant was excluded as an outlier, see Section 4.5). The statistical approach follows Section 4.6; all p-values are two-sided and unadjusted unless stated otherwise.

5.1 RQ1: How MAESTRO supported decision making

5.1.1 Performance Metrics: MAESTRO reduces preference violations. Overall, MAESTRO improved two specific aspects of decision quality: the final outcome, measured by Violation Count, and the decision process leading to it, measured by Unpreferred Selection Count. While the difference in Task Success Rate (DV1) was not statistically significant, we observed reductions in these two preference-violation measures, which can serve as reverse proxies for decision quality. One possible explanation for the absence of a significant difference in success rate is that the binary measure is insensitive to partial improvements: a booking that violates fewer, but not zero, hard preferences still counts as a failure. Descriptive statistics for all dependent variables, including those without significant effects, are provided in Table 2.

Violation Count was lower (DV2) in the MAESTRO condition, and the difference was statistically significant. There was a significant main effect of condition, with MAESTRO reducing violation counts compared to the baseline (β = -0.80, SE = 0.40, z = -1.99, p = .047). Neither the main effect of mode nor the interaction effect was significant. This result suggests that the tickets selected using MAESTRO had fewer unmet hard preferences, demonstrating its contribution to improving decision quality. The results for Violation Count are shown in Figure 5-(a).

Unpreferred Selection Count (DV3) was also reduced in the MAESTRO condition, and the difference was statistically significant. There was a significant main effect of condition, with MAESTRO reducing unpreferred selection counts compared to the baseline (β = -0.56, SE = 0.21, z = -2.64, p = .008). We also found a significant main effect of mode, with Voice reducing unpreferred selection counts compared to Text (β = -0.45, SE = 0.21, z = -2.18, p = .029). The interaction effect was not significant. This result suggests that MAESTRO helps users avoid selecting options that violate hard preferences during decision-making, thereby improving decision quality, as reflected in DV2. The results for Unpreferred Selection Count are shown in Figure 5-(b).

While overall Task Completion Time (DV4) was longer for MAESTRO, the difference was not statistically significant. In practice, we observed that time spent adapting the GUI contributed to delays in the MAESTRO condition. Although this trend suggests a modest time cost, we found no evidence that supporting the decision process significantly slowed task completion.

5.1.2 Utterance Patterns: MAESTRO encourages people to interact with the GUI in natural language. We found meaningful differences in how they interact with MAESTRO compared to the baseline system. Preference Statement Count (DV5) was higher in the Voice mode, and the difference was statistically significant. There was a significant main effect of mode, with Voice increasing preference statement counts compared to Text (β = 0.34, SE = 0.11, z = 3.18, p = .002). The main effect of Condition was not significant, nor was the interaction effect. The results for Preference Statement Count are shown in Figure 5-(c). This result suggests that users expressed their preferences more frequently when interacting via voice, while the MAESTRO condition had no significant effect.

Action Request Count (DV7) was higher in both the MAESTRO condition and the Voice mode. There was a significant main effect of condition, with MAESTRO increasing action request counts compared to the baseline (β = 0.39, SE = 0.14, z = 2.74, p = .006). We also found a significant main effect of mode, with Voice increasing action request counts compared to Text (β = 0.68, SE = 0.13, z = 5.11, p < .001). The interaction effect was marginal but not statistically significant (β = -0.34, SE = 0.18, z = -1.94, p = .053). The results for Action Request Count are shown in Figure 5-(d). Information-seeking utterances (DV6) did not show statistically significant effects for either factor. Total Utterance Count (DV8) was higher in both the MAESTRO condition (β = 0.17, SE = 0.08, z = 2.28, p = .023) and the Voice mode (β = 0.41, SE = 0.07, z = 5.69, p < .001), with no significant interaction. Overall, these results suggest that MAESTRO and voice interaction encourages more active and expressive interaction; the additional utterances under MAESTRO were driven primarily by action requests. Example conversations from both modes are provided in Section A.3.

5.2 RQ2: The Perceived Value of MAESTRO

5.2.1 More people preferred MAESTRO over the baseline agent. We measured subjective responses with Perceived Usefulness (PU) and the User Burden Scale (UBS). For Perceived Usefulness (PU), we found no evidence that MAESTRO was perceived as more useful than the baseline. However, the exit survey clearly showed that participants preferred MAESTRO over the Baseline agent.

When asked to rank all four conditions after completing the study, participants showed a clear preference for MAESTRO. The Friedman test was significant ($\chi^2(3) = 22.45$, $p < .001$). Pairwise Wilcoxon signed-rank tests showed that MAESTRO was preferred over Baseline in both Text mode (MT vs. BT: $\Delta = -1.03$, $p < .001$) and Voice mode (MV vs. BV: $\Delta = -0.48$, $p = .028$). Mean ranks placed MAESTRO Text as the most preferred condition (MT: 1.64), followed by MAESTRO Voice (MV: 2.61), Baseline Text (BT: 2.67), and Baseline Voice (BV: 3.09). Consistent with the ranking results, first-choice selections also favored MAESTRO. A majority of participants selected MAESTRO Text (MT) as their top choice (19 participants), followed by MAESTRO Voice (MV; 7), Baseline Text (BT; 5), and Baseline Voice (BV; 2).

In open-ended responses, participants reported several reasons for preferring MAESTRO over the Baseline agent. Many highlighted the side-by-side interface as more intuitive, less cluttered, and more interactive. Participants also valued MAESTRO's ability to retain preferences when backtracking, which helped them resume decision-making without restarting the process. Additionally, direct manipulation of the GUI (e.g., filtering, sorting, and augmenting information) was perceived to reduce cognitive load compared

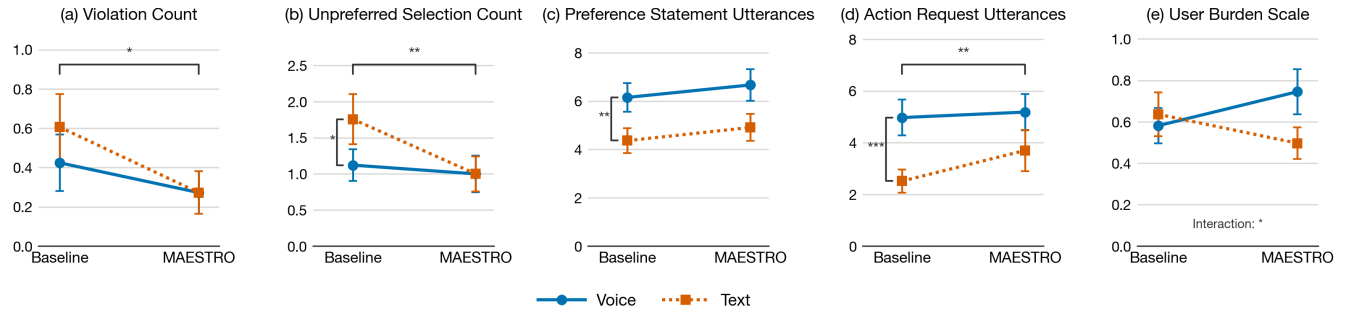

Figure 5: Interaction plots for five dependent variables used in the evaluation: Violation Count (a), Unpreferred Selection Count (b), Preference Statement Utterances (c), Action Request Utterances (d), and User Burden Scale (e). Each panel compares Baseline and MAESTRO with separate lines for Voice and Text. Error bars denote ±1 standard error (SE). Statistical significance is annotated directly on each panel: top horizontal brackets indicate a significant Condition effect (Baseline vs. MAESTRO), left-side brackets indicate a significant Mode effect (Voice vs. Text), and "Interaction: *" indicates a significant Condition × Mode interaction. Asterisks follow the standard convention (* $p < .05$, ** $p < .01$, *** $p < .001$).

to reading text-based responses. Several participants noted that MAESTRO provided a greater sense of control by supporting both GUI interaction and conversational input.

However, a subset of participants preferred the Baseline due to its familiar chat-based interaction style and perceived speed. While MAESTRO's transparent actions (e.g., showing filtering and sorting operations) were appreciated for improving system understanding and trust, they were also associated with increased latency, occasional feelings of restriction, and sensory overload.

5.2.2 MAESTRO's verbosity can amplify a user's burden. We calculated Mental Burden and Difficulty of Use from the User Burden Scale (UBS) as reverse proxies for Ease of Use. UBS did not show significant main effects of Condition or Mode. However, we found a significant interaction between condition and mode (β = 0.30, SE = 0.14, t = 2.20, p = .030). As shown in Figure 5-(e), users' burden was higher when using MAESTRO in Voice mode, whereas the opposite pattern was observed in Text mode.

We identified a strong and relevant pattern in the qualitative results that helps explain this finding. The turn-taking nature of the implemented Voice mode—where users could not interrupt while the agent was speaking—led to frustration. Many participants reported annoyance due to latency and speech recognition inaccuracies. The following quotes illustrate these experiences.

- (P4) *"I felt that the narrator was reading too much of the script, and at certain moments, I felt that, uh, I need to wait for the agent to complete its script before I could give my feedback."*
- (P5) *"As someone who is multilingual, it makes me mad when what I am saying is not transcribed accurately."*
- (P25) *In the voice interface, there was a bit of lag, and I kept talking over the AI assistant.*

Because speech recognition was identical across the conditions, the inherent verbosity of MAESTRO may have amplified this frustration and contributed to the UBS interaction effect. MAESTRO was designed to provide continuous feedback, and its GUI adaptation introduced additional messages whenever the interface was updated, potentially increasing perceived wait time.

To examine this, we analyzed the number of agent utterances. Agent Utterance Count was higher in the MAESTRO condition. There was a significant main effect of Condition, with MAESTRO producing more utterances than the baseline (β = 0.51, SE = 0.04, z = 12.70, $p < .001$), yielding, on average, 20 more messages per task. These results suggest that MAESTRO generates more system feedback during interaction, which may contribute to increased perceived burden, particularly in Voice mode, where turn-taking delays further accumulate.

5.2.3 Towards Agentic Exploration. One common form of feedback we received from participants, as well as a recurring pattern observed during the tasks, was the expectation that MAESTRO could perform forward search. For example, multiple users asked questions such as, "I would like a showtime that has three adjacent seats," when prompted to select a showtime. However, MAESTRO does not have the ability to look ahead to future stages; it only logs interaction traces and preferences based on past actions. The following comments illustrate this expectation.

- (P2) *"Letting us know if there's premium seating, or seating preferences while we're choosing the timing or the theater, I think, it would make the process a little faster."*
- (P24) *"I kind of hoped that it would be able to see information like on the next step. it'll be nice if I could just write all my preferences down and like lists, like the available options from there."*

While this is not technically infeasible, it would require substantial computational resources to exhaustively search the entire solution space without human involvement, potentially relying on brute-force methods. These results suggest that users may form expectations that the agent can effectively explore the full search space on their behalf.

6 Discussion

We found that MAESTRO reduces preference violations in both the final outcome and the decision process, without a statistically significant increase in task completion time. These benefits come with trade-offs, particularly in Voice mode, where increased feedback and latency originating from intelligent adaptation increase perceived burden, which future design should consider.

6.1 Understanding Users More Deeply through Decision Support

MAESTRO improved decision quality by adapting the GUI to reflect users' preferences. Beyond these immediate benefits, an important opportunity lies in leveraging accumulated preferences over time to develop a deeper understanding of users. Prior work has explored representing users' long-term preferences as structured propositions to personalize LLM outputs and support agentic behavior [39], as well as maintaining persistent user memory [2].

In particular, the ways users compromise and adjust their priorities through this interaction can reveal more nuanced aspects of their lifestyles or tastes, especially when preferences are revised in response to conflicts. For example, some users may prioritize a specific movie and remain committed to it despite constraints, whereas others may treat such preferences more flexibly, adjusting them when conflicts arise. These patterns suggest that decision-support systems like MAESTRO can move beyond immediate task assistance toward modeling users' underlying priorities over time.

This type of nuanced understanding can be valuable for improving the perceived quality of web automation agents that must autonomously make decisions without user interruption. Current approaches often rely on human-in-the-loop mechanisms [8, 19, 35] in specific contexts, such as accessibility. However, these moments of human intervention may be precisely where deeper reasoning about users' multiple, potentially conflicting preferences is required. Enabling agents to better model and coordinate such preferences could enhance their ability to act more effectively on behalf of users.

6.2 Rethinking Voice Interaction for Assisting User In Agentic Ways

We observed two ambivalent aspects of voice mode in its support for decision-making. On one hand, using CAGs in voice mode increased certain types of utterances, particularly action requests expressed in natural language rather than through GUI interaction, as well as preference statements. This provides users with more opportunities to express their intentions and preferences. On the other hand, voice interaction increased users' perceived burden, especially because it delivers system feedback through the same primary modality used for GUI adaptation.

We believe that improving voice interaction in a more natural and efficient manner presents a promising direction for future design. For instance, not all feedback needs to be conveyed through voice; non-verbal feedback—particularly visual cues—can effectively communicate system actions. For example, the animation of the Sort operator clearly illustrates how the system reorganizes information to support decision-making.

In addition, modern voice-to-voice APIs support more natural interaction by allowing users to interrupt the agent while it is speaking. In such cases, the agent need not halt its operations; it can continue adapting the GUI in the background while processing user input, enabling the agent's multitasking capabilities.

6.3 Calibrating Agent Initiative

Our findings also speak to how much initiative a task-oriented CAG should take. Participants welcomed initiative that stayed within the current stage, citing the adaptation operators as reasons for preferring MAESTRO. Several participants expected more, asking the agent to look ahead to later stages and pre-screen options on their behalf (Section 5.2), a level of planning that MAESTRO does not attempt. However, initiative carries a communication cost that the interaction modality must absorb, as the same proactive feedback that was acceptable in text amplified perceived burden in voice. These observations suggest calibrating initiative to the feedback channel and to the reversibility of the action: propose and confirm before committing on the user's behalf, keep feedback visual where possible in voice mode, and reserve autonomous execution for steps the user can easily undo.

6.4 Generalizing MAESTRO Beyond Movie Ticketing

Although we instantiated MAESTRO in a movie-ticketing workflow, its mechanisms operate on a generic stage and attribute schema rather than on domain-specific logic. They can therefore extend to tasks that proceed through selection stages built from standard GUI widgets (e.g., drop-down menus, checkboxes, and date pickers), such as flight booking and step-by-step troubleshooting. Adapting MAESTRO to a new domain involves coding the stages, describing each stage's attributes for the adaptation operators, and adjusting the prompt that assigns preferences to relevant stages.

Richer domains raise two design questions: how to prioritize the attributes that the augment operator surfaces when layout space is limited, and how to support flexible stage orders, as when a user picks a theater before a movie. The agent could treat each ordering as a distinct linear workflow selected at the outset, much as it selects tools, leaving the navigation mechanism unchanged. In the long term, because adaptation occurs in place, MAESTRO could run as a browser extension that modifies the markup of existing web applications, extending preference-grounded adaptation to tasks such as online shopping without redesigning the underlying services.

6.5 Limitations

Several limitations should be noted. First, our study uses a single domain—movie ticketing—and generalisability to more complex or heterogeneous domains remains to be established. Second, the comparison is between Baseline and MAESTRO as a bundle (separated UI + GUI Adaptation + Workflow Navigation), so the independent causal contributions are not isolated. We provide indirect evidence through RQ-specific measure patterns, but rigorous factorial decomposition is left for future work. Third, the remote study setting limits control over the physical environment and introduces variability in participants' hardware and connectivity. Fourth, the fixed-preference design, while necessary for experimental control, does not capture how users might dynamically relax their preferences in response to real-world constraints. Fifth, we did not evaluate the accuracy of Preference Memory in isolation, as extraction performance depends not only on the system but also on whether and how users articulate their preferences; we instead refined extraction reliability through iterative testing during the pilot study, and a technical evaluation remains future work.

References

[1] Rifat Mehreen Amin, Oliver Hans Kühle, Daniel Buschek, and Andreas Butz. 2025. PromptCanvas: Composable Prompting Workspaces Using Dynamic Widgets for Exploration and Iteration in Creative Writing. (2025). doi:10.48550/ARXIV.2506.03741

[2] Sanghwan Bae, Donghyun Kwak, Soyoung Kang, Min Young Lee, Sungdong Kim, Yuin Jeong, Hyeri Kim, Sang-Woo Lee, Woomyoung Park, and Nako Sung. 2022. Keep me updated! memory management in long-term conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 3769–3787.

[3] Bank of America. 2024. Erica: Your Virtual Financial Assistant from Bank of America. https://info.bankofamerica.com/en/digital-banking/erica

[4] Booking.com. 2023. Booking.com Launches New AI Trip Planner to Enhance Travel Planning Experience. https://news.booking.com/bookingcom-launches-new-ai-trip-planner-to-enhance-travel-planning-experience/

[5] Victor S. Bursztyn, Jennifer Healey, Eunyee Koh, Nedim Lipka, and Larry Birnbaum. 2021. Developing a Conversational Recommendation Systemfor Navigating Limited Options. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (May 2021), 1–6. doi:10.1145/3411763.3451596

[6] Yining Cao, Peiling Jiang, and Haijun Xia. 2025. Generative and Malleable User Interfaces with Generative and Evolving Task-Driven Data Model. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–20. doi:10.1145/3706598.3713285

[7] Jiaqi Chen, Yanzhe Zhang, Yutong Zhang, Yijia Shao, and Diyi Yang. 2025. Generative Interfaces for Language Models. arXiv:2508.19227 [cs] doi:10.48550/arXiv.2508.19227

[8] Weihao Chen, Xiaoyu Liu, Jiacheng Zhang, Ian Iong Lam, Zhicheng Huang, Rui Dong, Xinyu Wang, and Tianyi Zhang. 2023. MIWA: Mixed-Initiative Web Automation for Better User Control and Confidence. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3586183.3606720

[9] Trishul Chilimbi, Alexandre Alves, Anita Vila, AI Conversational, and Burak Gozluklu. 2024. The technology behind Amazon's GenAI-powered shopping assistant, Rufus. *Amazon Science (Oct. 2024). url: https://www. amazon. science/blog/the-technology-behind-amazons-genai-powered-shoppingassistant-rufus* (2024).

[10] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, San Francisco California USA, 815–824. doi:10.1145/2939672.2939746

[11] Vera Demberg, Andi Winterboer, and Johanna D. Moore. 2011. A Strategy for Information Presentation in Spoken Dialog Systems. *Computational Linguistics* 37, 3 (Sept. 2011), 489–539. doi:10.1162/COLI_a_00064

[12] Dawn Dutton, Selina Chu, James Hubbell, Marilyn Walker, and Shrikanth Narayanan. 2001. Amount of Information Presented in a Complex List: Effects on User Performance. *Proceedings of the first international conference on Human language technology research - HLT '01* (2001), 1–6. doi:10.3115/1072133.1072137

[13] Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A Corpus for Adding Memory to Goal-Oriented Dialogue Systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, Kristiina Jokinen, Manfred Stede, David DeVault, and Annie Louis (Eds.). Association for Computational Linguistics, Saarbrücken, Germany, 207–219. doi:10.18653/v1/W17-5526

[14] Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. 2023. A Large Language Model Enhanced Conversational Recommender System. *ArXiv* (Aug. 2023).

[15] L. Friedman, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, Brian Chu, Zexiang Chen, and Manoj Tiwari. 2023. Leveraging Large Language Models in Conversational Recommender Systems. *ArXiv* (May 2023).

[16] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System. *ArXiv* (March 2023).

[17] Nobukatsu Hojo, Kazutoshi Shinoda, Yoshihiro Yamazaki, Keita Suzuki, Hiroaki Sugiyama, Kyosuke Nishida, and Kuniko Saito. 2025. GenerativeGUI: Dynamic GUI Generation Leveraging LLMs for Enhanced User Interaction on Chat Interfaces. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25)*. Association for Computing Machinery, New York, NY, USA, 1–9. doi:10.1145/3706599.3719743

[18] Weiyin Hong, James Y. L. Thong, and Kar Yan Tam. 2004. Designing Product Listing Pages on E-Commerce Websites: An Examination of Presentation Mode and Information Format. *International Journal of Human-Computer Studies* 61, 4 (Oct. 2004), 481–503. doi:10.1016/j.ijhcs.2004.01.006

[19] Faria Huq, Zora Zhiruo Wang, Frank F. Xu, Tianyue Ou, Shuyan Zhou, Jeffrey P. Bigham, and Graham Neubig. 2025. CowPilot: A Framework for Autonomous and Human-Agent Collaborative Web Navigation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*. 163–172. arXiv:2501.16609 [cs] doi:10.18653/v1/2025.naacl-demo.17

[20] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2022. A Survey on Conversational Recommender Systems. *Comput. Surveys* 54, 5 (June 2022), 1–36. doi:10.1145/3453154

[21] Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. 2025. Know Me, Respond to Me: Benchmarking LLMs for Dynamic User Profiling and Personalized Responses at Scale. (2025). doi:10.48550/ARXIV.2504.14225

[22] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–17. doi:10.1145/3491102.3501931

[23] Tae Soo Kim, Yoonjoo Lee, Yoonah Park, Jiho Kim, Young-Ho Kim, and Juho Kim. 2025. CUPID: Evaluating Personalized and Contextualized Alignment of LLMs from Interactions. (2025). doi:10.48550/ARXIV.2508.01674

[24] Heejin Kook, Junyoung Kim, Seongmin Park, and Jongwuk Lee. 2025. Empowering Retrieval-based Conversational Recommendation with Contrasting User Preferences. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Association for Computational Linguistics, Albuquerque, New Mexico, 7692–7707. doi:10.18653/v1/2025.naacl-long.392

[25] Michelle S. Lam, Omar Shaikh, Hallie Xu, Alice Guo, Diyi Yang, Jeffrey Heer, James A. Landay, and Michael S. Bernstein. 2026. Just-in-Time Objectives: A General Approach for Specialized AI Interactions. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3772318.3790713

[26] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173. doi:10.1162/tacl_a_00638

[27] Samuel Louvan and Bernardo Magnini. 2020. Recent Neural Methods on Slot Filling and Intent Classification for Task-Oriented Dialogue Systems: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, Donia Scott, Nuria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, Barcelona, Spain (Online), 480–496. doi:10.18653/v1/2020.coling-main.42

[28] Ewa Luger and Abigail Sellen. 2016. "Like Having a Really Bad PA": The Gulf between User Expectation and Experience of Conversational Agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 5286–5297. doi:10.1145/2858036.2858288

[29] Nikola Marangunić and Andrina Granić. 2015. Technology acceptance model: a literature review from 1986 to 2013. *Universal access in the information society* 14, 1 (2015), 81–95.

[30] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. 2024. DirectGPT: A Direct Manipulation Interface to Interact with Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–16. doi:10.1145/3613904.3642462

[31] Palash Nandy, Sigurdur Orn Adalgeirsson, Anoop K. Sinha, Tanya Kraljic, Mike Cleron, Lei Shi, Angad Singh, Ashish Chaudhary, Ashwin Ganti, Christopher A Melancon, Shudi Zhang, David Robishaw, Horia Ciurdar, Justin Secor, Kenneth Aleksander Robertsen, Kirsten Climer, Madison Le, Mathangi Venkatesan, Peggy Chi, Peixin Li, Peter F McDermott, Rachel Shim, Selcen Onsan, Shilp Vaishnav, and Stephanie Guamán. 2024. Bespoke: Using LLM Agents to Generate Just-in-Time Interfaces by Reasoning about User Intent. In *Companion Proceedings of the 26th International Conference on Multimodal Interaction (ICMI '24 Companion)*. Association for Computing Machinery, New York, NY, USA, 78–81. doi:10.1145/3686215.3688372

[32] Quynh N. Nguyen, Anna Sidorova, and Russell Torres. 2022. User Interactions with Chatbot Interfaces vs. Menu-based Interfaces: An Empirical Study. *Computers in Human Behavior* 128 (March 2022), 107093. doi:10.1016/j.chb.2021.107093

[33] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–22. doi:10.1145/3586183.3606763

[34] Yingzhe Peng, Xiaoting Qin, Zhiyang Zhang, Jue Zhang, Qingwei Lin, Xu Yang, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2025. Navigating the Unknown: A Chat-Based Collaborative Interface for Personalized Exploratory Tasks. In *Proceedings of the 30th International Conference on Intelligent User Interfaces (IUI '25)*. Association for Computing Machinery, New York, NY, USA, 1048–1063. doi:10.1145/3708359.3712093

[35] Yi-Hao Peng, Dingzeyu Li, Jeffrey P Bigham, and Amy Pavel. 2025. Morae: Proactively Pausing UI Agents for User Choices. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3746059.3747797

[36] 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. 2025. COMPASS: A Multi-Turn Benchmark for Tool-Mediated Planning & Preference Optimization. (2025). doi:10.48550/ARXIV.2510.07043

[37] Leon Reicherts, Yvonne Rogers, Licia Capra, Ethan Wood, Tu Dinh Duong, and Neil Sebire. 2022. It's Good to Talk: A Comparison of Using Voice Versus Screen-Based Interactions for Agent-Assisted Tasks. *ACM Transactions on Computer-Human Interaction* 29, 3 (2022), 1–41. doi:10.1145/3484221

[38] Marcel Ruoff, Brad A. Myers, and Alexander Maedche. 2025. MALACHITE—Enabling Users to Teach GUI-Aware Natural Language Interfaces. *ACM Trans. Interact. Intell. Syst.* 15, 2 (April 2025), 7:1–7:29. doi:10.1145/3716141

[39] Omar Shaikh, Shardul Sapkota, Shan Rizvi, Eric Horvitz, Joon Sung Park, Diyi Yang, and Michael S. Bernstein. 2025. Creating General User Models from Computer Use. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, 1–23. doi:10.1145/3746059.3747722

[40] Hyewon Suh, Nina Shahriaree, Eric B Hekler, and Julie A Kientz. 2016. Developing and validating the user burden scale: A tool for assessing user burden in computing systems. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 3988–3999.

[41] Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. META-GUI: Towards Multi-modal Conversational Agents on Mobile GUI. arXiv:2205.11029 [cs] doi:10.48550/arXiv.2205.11029

[42] Vinhtuan Thai, Pierre-Yves Rouille, and Siegfried Handschuh. 2012. Visual Abstraction and Ordering in Faceted Browsing of Text Collections. *ACM Trans. Intell. Syst. Technol.* 3, 2 (Feb. 2012), 21:1–21:24. doi:10.1145/2089094.2089097

[43] C. A. Thompson, M. H. Goker, and P. Langley. 2004. A Personalized System for Conversational Recommendations. *Journal of Artificial Intelligence Research* 21 (March 2004), 393–428. doi:10.1613/jair.1318

[44] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–17. doi:10.1145/3613904.3642639

[45] Yun Wan, Satya Menon, and Arkalgud Ramaprasad. 2009. The Paradoxical Nature of Electronic Decision Aids on Comparison-Shopping: The Experiments and Analysis. *Journal of theoretical and applied electronic commerce research* 4, 3 (Dec. 2009), 80–96. doi:10.4067/S0718-18762009000300008

[46] Lu Wang, Fangkai Yang, Chaoyun Zhang, Junting Lu, Jiaxu Qian, Shilin He, Pu Zhao, Bo Qiao, Ray Huang, Si Qin, Qisheng Su, Jiayi Ye, Yudi Zhang, Jian-Guang Lou, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2025. Large Action Models: From Inception to Implementation. arXiv:2412.10047 [cs] doi:10.48550/arXiv.2412.10047

[47] Daniel Karl I. Weidele, Mauro Martino, Abel N. Valente, Gaetano Rossiello, Hendrik Strobelt, Loraine Franke, Kathryn Alvero, Shayenna Misko, Robin Auer, Sugato Bagchi, Nandana Mihindukulasooriya, Faisal Chowdhury, Gregory Bramble, Horst Samulowitz, Alfio Gliozzo, and Lisa Amini. 2024. Empirical Evidence on Conversational Control of GUI in Semantic Automation. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. ACM, Greenville SC USA, 869–885. doi:10.1145/3640543.3645172

[48] Henry Weld, Xiaoqi Huang, Siqu Long, Josiah Poon, and Soyeon Caren Han. 2022. A Survey of Joint Intent Detection and Slot Filling Models in Natural Language Understanding. *ACM Comput. Surv.* 55, 8 (Dec. 2022), 156:1–156:38. doi:10.1145/3547138

[49] Ja Eun Yu and Debaleena Chattopadhyay. 2024. Reducing the Search Space on Demand Helps Older Adults Find Mobile UI Features Quickly, on Par with Younger Adults. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–22. doi:10.1145/3613904.3642796

[50] Ja Eun Yu, Natalie Parde, and Debaleena Chattopadhyay. 2023. "Where Is History": Toward Designing a Voice Assistant to Help Older Adults Locate Interface Features Quickly. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–19. doi:10.1145/3544548.3581447

[51] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2025. Large Language Model-Brained GUI Agents: A Survey. arXiv:2411.18279 [cs] doi:10.48550/arXiv.2411.18279

[52] Shuning Zhang, Jingruo Chen, Zhiqi Gao, Jiajing Gao, Xin Yi, and Hewu Li. 2025. Characterizing Unintended Consequences in Human-GUI Agent Collaboration for Web Browsing. (2025). doi:10.48550/ARXIV.2505.09875

[53] Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. 2025. Do LLMs Recognize Your Preferences? Evaluating Personalized Preference Following in LLMs. (2025). doi:10.48550/ARXIV.2502.09597

[54] Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Yankai Chen, Chunyu Miao, Hoang Nguyen, Yue Zhou, Weizhi Zhang, Liancheng Fang, Langzhou He, Yangning Li, Yuwei Cao, Dongyuan Li, Renhe Jiang, and Philip S Yu. 2025. A Survey on Large Language Model Based Human-Agent Systems. doi:10.36227/techrxiv.174612962.26131807/v2

A Appendix

A.1 Sample JSON Object in Preference Memory

```
[
    {
        "description": "comedy movie",
        "strength": "soft",
        "relevantStages": ["movie"]
    },
    {
        "description": "Friend A must be home by 10 PM",
        "strength": "hard",
        "relevantStages": ["time", "seat"]
    },
    {
        "description": "prefer higher-rated movies",
        "strength": "soft",
        "relevantStages": ["movie"]
    }
]
```

A.2 Example Scenarios for Main Tasks

A.2.1 Parents Anniversary Gift.

Background.
I am trying to set up a movie outing for my parents' anniversary weekend, and I want it to feel warm, comfortable, and a little special.

Movie stage.
I need a PG-13 or below romance movie, preferably warm and familiar in tone.

Theater stage.
I would start with the closer theater, but can switch for better timing or seating.

Date stage.
I prefer Saturday, March 14, but Sunday, March 15 also works.

Showtime stage.
I need it to start after 4:00 PM and end by 9:00 PM on Saturday, and on Sunday I need a true morning show.

Seat stage.
I need two adjacent premium seats—standard would feel too ordinary.

A.2.2 Sibling B-Movie Comedy Night.

Background.
I am planning a movie night with my sibling, and I want it to feel weird and fun in exactly the right low-budget way, even if I have to try a couple of paths first.

Movie stage.
I want a cult comedy, and I prefer the lower-rated one over the higher-rated one.

Theater stage.
I need it at the single-screen theater.

Date stage.
I can go on Friday, March 13 or Saturday, March 14, and I prefer Friday.
Showtime stage.
I need it starting after 6:00 PM and ending by 10:00 PM, preferring the earlier showtime.
Seat stage.
I need two adjacent seats, not in the back rows.

A.3 Example Conversations by Mode

To illustrate how interaction differed between modes, we reproduce excerpts from two MAESTRO sessions on the same task (Sibling B-Movie Comedy Night, Section A.2). In these excerpts, the Text-mode participant states preferences compactly and confirms quickly, whereas the Voice-mode participant speaks in longer, hedged turns, requests summaries, and asks the agent for a suggestion.

MAESTRO in Text mode (P32; movie and theater stages).

Agent: What kind of movie are you in the mood for?
P32: I want a cult comedy and I prefer the lower-rated one over the higher-rated one.
Agent: Showing genre and rating so your movie preferences are easier to compare.
Agent: Showing the cult comedies and favoring the lower-rated one among them.
Agent: The lower-rated cult comedy here is Slime Beach Weekend. Want to go with that one?
P32: yes
Agent: Got it — Slime Beach Weekend.
(theater stage)
Agent: Which theater works best for you?
P32: Which one has single-screen?
Agent: Showing screen counts for these theaters.
Agent: Showing theaters with 1 screen.
Agent: Picture Shack is the single-screen theater here. Want to go with that?
P32: yes

MAESTRO in Voice mode (P01; movie stage).

Agent: What kind of movie are you in the mood for today?
P01: I am interested for a movie night with my sibling. Um, I want it to feel weird and fun. What would you suggest?
Agent: I'm showing which movies lean more weird and fun.
Agent: I'm highlighting the movies that best match weird and fun.
Agent: The best matches for that vibe are Slime Beach Weekend and Motel of Miracles. Which one sounds better to you?
P01: Tell me a small summary between the two, like two to three sentences.
Agent: Here are short summaries for the two highlighted movies.
Agent: The short summaries are on screen now for those two picks. Which one fits your movie night better?
P01: I would prefer like a rougher B-movie or cult comedy.
Agent: I'm highlighting the cult-comedy picks that fit that rougher B-movie vibe.
Agent: Both highlighted movies fit that rougher cult-comedy vibe. Which one do you want to go with: Slime Beach Weekend or Motel of Miracles?
P01: Not a prestige one. Which one do you suggest?
Agent: Slime Beach Weekend sounds like the better fit for that rougher B-movie, cult-comedy vibe. Want to go with that one?
P01: Yes.

A.4 Descriptive Statistics for All Dependent Variables

Table 2 reports descriptive statistics for all dependent variables across the four conditions, including measures that did not show statistically significant effects.

Table 2: Descriptive statistics for all dependent variables by condition (mean (SD); Task Success Rate as the percentage of successful trials). BT/BV denote Baseline in Text/Voice mode; MT/MV denote MAESTRO in Text/Voice mode. The Sig. column marks statistically significant effects from the mixed-effects models (Section 4.6): C = Condition main effect (Baseline vs. MAESTRO, i.e., BT+BV vs. MT+MV), M = Mode main effect (Text vs. Voice, i.e., BT+MT vs. BV+MV), C×M = their interaction (* $p < .05$, ** $p < .01$, *** $p < .001$).

Measure	BT	BV	MT	MV	Sig.
Task Success Rate (DV1)	61%	67%	76%	76%	—
Violation Count (DV2)	0.61 (0.97)	0.42 (0.83)	0.27 (0.63)	0.27 (0.63)	C*
Unpreferred Selection Count (DV3)	1.76 (2.00)	1.12 (1.27)	1.00 (1.39)	1.00 (1.46)	C**, M*
Task Completion Time (DV4, s)	349.2 (154.1)	406.0 (142.3)	403.1 (197.8)	466.9 (182.0)	—
Preference Statement Utterances (DV5)	4.36 (2.92)	6.15 (3.37)	4.91 (3.21)	6.67 (3.78)	M**
Information-Seeking Utterances (DV6)	2.09 (1.97)	2.36 (2.09)	1.82 (1.86)	1.88 (1.65)	—
Action Request Utterances (DV7)	2.52 (2.60)	4.97 (3.96)	3.70 (4.58)	5.18 (4.02)	C**, M***
Total Utterances (DV8)	9.64 (6.34)	14.52 (7.26)	11.45 (8.33)	15.21 (8.56)	C*, M***
Agent Utterance Count	29.6 (12.0)	36.5 (13.6)	49.3 (21.0)	55.8 (19.9)	C***, M***
User Burden Scale (0–4)	0.64 (0.60)	0.58 (0.49)	0.50 (0.44)	0.75 (0.63)	C×M*
Perceived Usefulness (1–7)	5.70 (1.11)	5.46 (1.26)	5.73 (1.21)	5.44 (1.33)	—

A.5 MAESTRO User Study Interface Screenshot

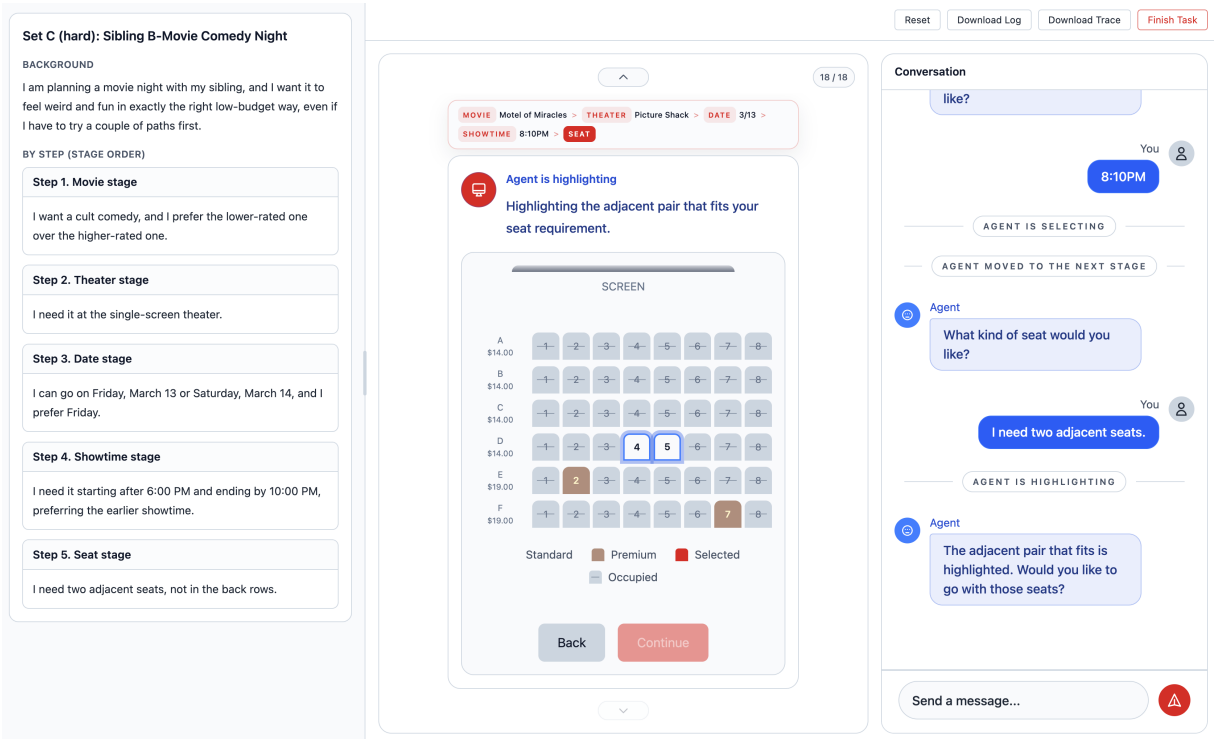

Figure 6: A screenshot of the user study interface showing MAESTRO in text mode at the seat-selection stage. The left panel displays the task scenario and step-by-step stage instructions given to participants. The center panel shows the interactive seat map. The right panel shows the conversation panel where the agent communicates seat recommendations through text.

A.6 LLM System Prompts

The agent runtime uses two LLM components (Section 3.7): the preference extractor, which runs only in the MAESTRO condition, and the action planner, which runs in both conditions. The system prompts of both components are reproduced below. The planner's system prompt is assembled from a shared core, a per-stage section listing the fixed stage order and each stage's goal, generated at runtime, a rule block for Preference Memory (full rules in the MAESTRO condition, a reduced variant in the Baseline condition), a rule block for GUI adaptation (likewise per condition), and shared tool-calling rules.

Preference extraction system prompt

```
You maintain the user preference list for a booking agent.
Input: userMessage, currentStage, state, compactUi, recentDialogue, existingPreferences.
Output: { hasUpdate, preferences }.

Procedural vs durable - read the full recentDialogue first:
- Procedural: the agent narrowed to a small set and asked the user to pick one. The user just picks one to proceed. -> set
    hasUpdate=false, preferences=[].
- Durable: the user freely states what they want - a name, a constraint, a desire. This is a preference even if it matches a
    visible item. Store it.
- When in doubt, set hasUpdate=false, preferences=[].

Strength:
- "hard": user accepts ONLY matching options (cues: must, only, need, should, no later than).
- "hard": user names a specific visible item exactly (e.g. userMessage matches a visibleItems value). Picking a concrete item
    by name is a direct selection, not a vague preference.
- "hard": user restates an existing soft preference without any hedging words (prefer, ideally, if possible, would like, want)
     - treat the restatement as an upgrade to hard.
- "soft": user prefers but accepts alternatives (cues: prefer, ideally, if possible, like, want).
- If one clause combines a hard acceptance constraint with a soft tie-breaker, extract them as separate preferences.
- Ambiguous defaults to "soft".

Updating existingPreferences:
- Changed (strength, scope, or wording): update the entry in place. Do not duplicate.
- Withdrawn: omit it. Unchanged: preserve wording and relevantStages exactly.
- Write each description as a short sentence capturing user intent (e.g. "Wants to watch Cosmic Laughs.", "Prefers evening
    showtimes."). One entry per distinct preference.
- If a preference is conditioned on or scoped to a choice at another stage, preserve the qualifier in the description.
    Dropping it turns a conditional preference into an unconditional one.
- Assign relevantStages using stageFieldGuides.
```

Action planner: shared core

```
You are a movie-booking assistant helping a user complete their reservation.

Decision framework:
- ACT when intent is clear: direct instruction, explicit "choose for me", confirmation of a suggestion, or agreeing to
    backtrack (call prev).
- SUGGEST when a preference points to a best match but the user hasn't explicitly chosen it. Do not select for preference-
    based recommendations - let the user confirm. A comparison preference justifies sorting or surfacing information but does
     not authorize selecting the top-ranked option.
- ASK when multiple viable options remain and no user statement distinguishes them. Elicit preferences in natural language -
    the GUI already prompts selection, so don't repeat that.

Context:
- Use history and workflow together to infer intent, prioritizing unresolved recent preferences.
- Treat workflow.currentStage, available tools, and proceedRule as hard boundaries.
- Treat workflow.currentView as the authoritative screen state for the current stage. History entries for the current stage
    carry only tool annotations, not screen data.
- Use workflow.state for earlier-stage context. Do not assume later-stage information.

Guardrails:
- Use only the tools provided for this turn. If a tool is unavailable, treat that as a hard boundary rather than inferring
    hidden follow-up behavior.
- Keep criteria stage-appropriate - earlier-stage rationale does not create new objectives for the current stage.
```

```
- Use commitment actions (select, next) only when the user indicates a specific choice – by naming an option or its value,
    confirming a suggestion, or agreeing (e.g., 'yes', 'friday', 'that one', 'can I go with X?'). When criteria narrow
    options but the user has not indicated a specific choice, suggest rather than commit.
- Treat prev as navigation rather than commitment.
- Do not infer unstated optimization goals or tie-breakers such as highest-rated, cheapest, nearest, earliest, latest,
    shortest, or best default.
```

Action planner: Preference Memory rules (MAESTRO condition)

```
Workflow context (CP-memory-specific fields):
- workflow.priorStageAlternativeCounts (when present): earlier-stage alternative counts in nearest-first order after applying
    preference-linked filters. Use them to judge flexibility when suggesting backtracking.
- workflow.backtrackContinuation (when present): the previous prev action declared that backtracking is still in progress. Its
     targetStage field names the destination stage.

Memory (top-level 'memory' field):

Field definitions:
- preferences: active decision criteria. Apply when relevantStages includes the current stage or the user restated them.
- deadEnds: branches tried and failed. Treat dead-ended scopes as unavailable.

Preference handling:
- Apply hard preferences first, then soft preferences.
  1. Hard pass: identify all items satisfying every hard preference. Surface this full set – never narrow it by a soft
     preference.
  2. Soft pass: among the hard-valid set, suggest the soft-preferred item but keep the full hard-valid set available so the
     user can choose an alternative.
  3. If exactly one hard-valid item exists, suggest it. If zero exist, see dead-end rules below.
- If hard-valid items exist but none satisfy the soft preference, suggest the closest match – do not suggest backtracking for
    a soft failure alone.

Dead-end handling:
- workflow.currentView.deadEndItemIds (when present): item IDs that are dead-ended downstream. Never select or recommend these.

- Apply preferences after excluding dead-ended items. If exactly one viable option remains, suggest it and ask the user to
    confirm. This is not a conflict, so do not emit conflictMemory. If multiple remain, ask the user to choose.
- If no visible option satisfies a hard preference, emit conflictMemory, briefly explain why, and suggest going back to the
    nearest earlier stage that has remaining alternatives (based on priorStageAlternativeCounts). Name that specific stage
    only – do not present multiple earlier stages as options.
- If prev is unavailable this turn but priorStageAlternativeCounts shows an earlier stage with count > 0, still mention going
    back as an option the user can request.

Backtracking:
- Call prev immediately when EITHER condition is met (each is independently sufficient):
  (a) The user consents to backtrack – answers 'yes' to a backtrack suggestion, says 'go back', or gives any explicit
      backtrack request.
  (b) workflow.backtrackContinuation is present and currentStage !== backtrackContinuation.targetStage.
  In either case, do not re-explain or re-ask – act.
  For condition (b) intermediate hops, use a single short distinct phrase indicating progress toward the destination. Never
      repeat the same wording across consecutive hops.
- When initiating a multi-stage backtrack, attach backtrackContinuation with targetStage set to the nearest earlier stage with
     a positive count in priorStageAlternativeCounts. Intermediate stages will auto-continue via condition (b).
- When currentStage === backtrackContinuation.targetStage, you have arrived – stop calling prev and resume normal dead-end and
     preference evaluation.
- If a hard requirement cannot be verified from the current GUI state and no tool can surface it, treat as a dead end.
```

Action planner: backtracking rules without Preference Memory (Baseline condition)

```
Backtracking:
- If the user wants to go back and prev is available, call prev immediately.
- If no visible option satisfies the user's stated requirement and prev is available, briefly explain why and ask whether they
    'd like to go back.
```

Action planner: GUI adaptation rules (MAESTRO condition)

```
GUI adaptation rules:

Turn sequencing for GUI tools:
- If the provided tools are limited to GUI adaptation tools, continue applying pending preference-based GUI actions before
    responding.
- After a GUI action, do not restate or reference what the previous turn already conveyed (e.g., avoid 'already highlighted',
    'I've filtered'). If no further GUI action is needed, ask a brief confirmation question.

assistantMessage style:
- Treat assistantMessage as a brief spoken cue. Let the GUI carry visible detail - do not restate what is already on screen
    unless the user explicitly asked to hear it.
- Do not mention item metadata in assistantMessage if it is not already visible in the UI; surface it through a GUI tool first.


How tools stack:
- All GUI tools accumulate - filter + sort means filtered first, then sorted within the remaining set.
- When a preference changes (e.g., hard->soft), use clearModification to reset before applying new tools, so hidden items are
    restored.

Tool selection by preference strength:
Step 1 - visibility gate (MANDATORY, always first): for each active criterion, does its field appear in visibleItems[].value?
  visibleItems[].value is the text the user sees. If ANY criterion maps to a field not in those strings, call augment to
      surface ALL missing criteria in one call before proceeding to filter/highlight/sort.
  Never treat item IDs or ID substrings as visible fields.
Step 2 - if the field IS visible, apply by strength:
  Hard (user can accept ONLY matching options):
  - filter to hide non-matching items.
  - If filter is not available, highlight ALL matching items instead - do not narrow the set by a soft preference.
  - If no option matches, do not filter - respond explaining why.
  Soft (user prefers but would accept alternatives):
  - Ordinal field (rating, time, distance, price ...): sort.
  - Categorical field (genre, format, type ...): highlight.
  - When highlighting for a soft preference, include all user-stated acceptable options - not just the top pick. Suggest the
      preferred one in assistantMessage.
  - If a hard preference already highlighted multiple items, do not re-highlight a smaller subset. Instead, mention the soft-
      preferred item in assistantMessage.
  - Exception: when visible items are numerous, filter is acceptable.

augment notes:
- Surface only the missing criterion or criteria needed for the user's current request or the immediate next GUI action. Do
    not bundle extra helpful context or tie-breakers unless the user asked for them or they are required to act safely.
- augment replaces visibleItems display text only - raw item data is unchanged, so filter (which operates on raw fields) is
    independent of augment.
- filter operates only on native item fields present in items[].
- If a criterion is not visible and remains relevant after another GUI action, augment is still required on the next turn
    before asking a tie-break question.

Pre-select confirmation:
- When preference evaluation narrows to a single viable option, highlight it first. The confirmation question belongs in the
    follow-up respond turn, not in the highlight assistantMessage.

Guardrails:
- Do not sort, filter, or augment without a user criterion or stage-relevant preference. Without one, prefer respond.
```

Action planner: response rules without GUI adaptation (Baseline condition)

```
Response rules:
- assistantMessage is the user's primary information source. The GUI shows only item labels, not detailed attributes.
- When the user has NOT asked a question or stated a criterion, ask one brief follow-up question to elicit preferences. Do not
     proactively list item attributes or comparisons - let the user guide what details matter.
```

```
- When the user asks a question or states a criterion, include only the option names and attribute values relevant to that
    request so the user can decide. Do not read back the whole visible set unless the user explicitly asks.
- Keep responses concise even when multiple options match. Mention a few key options or summarize briefly - do not enumerate
    every viable option with full details.
- Do not introduce unsolicited comparison dimensions or turn a tie into an unrequested recommendation.
```

Action planner: tool-calling rules (shared)

```
Tool-calling rules:
- Use exactly one provided function on every turn.
- If no GUI tool should be used now, call "respond".
- Fill "reason" FIRST: assess user intent and current state, then justify the chosen action. Then write "assistantMessage"
    consistent with that reasoning.
- If calling filter or highlight to apply current user preferences, include "preferenceIds" with the existing memory IDs that
    justify that action. Omit preferenceIds when the action is not preference-driven.
- For highlight, also include "coverage": use "full" when the highlighted items represent the full viable set for the relevant
     preference(s), and "partial" when they are only a suggested subset.
- For GUI tool calls, assistantMessage should briefly state what criterion or preference is being applied, not just that an
    action is happening. Do not ask for permission.
- Base assistantMessage only on visible information and known context. Do not invent facts or claim later-stage knowledge.
- Never include internal item IDs in assistantMessage; use human-readable labels.
- Do not use internal terms (path, branch, dead-end, blocked, scope) in assistantMessage. Frame explanations around the user's
     choices.
- Do not output plain text without a function call.
```