TUNING: Adaptive GUI Modification as Agent Tools in Task-Oriented Chatbots

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

Sang Won Lee
Virginia Tech
Blacksburg, Virginia, USA
sangwonlee@vt.edu

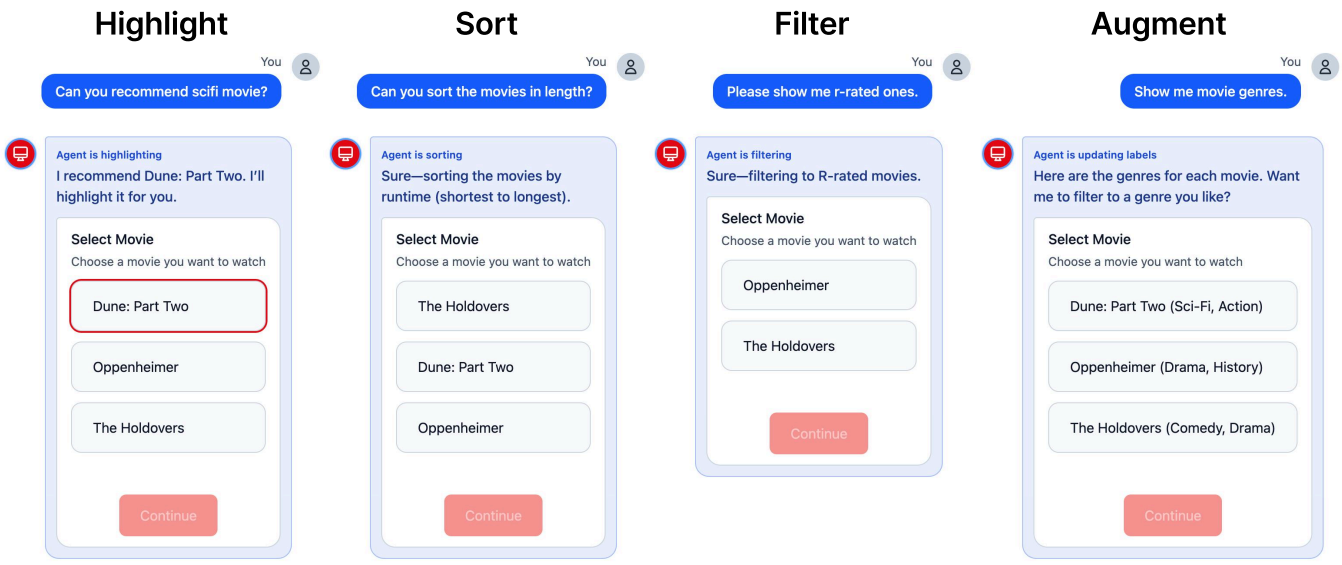

Figure 1: TUNING's four GUI adaptation tools applied to a movie selection screen. From left: *highlight* emphasizes a recommended sci-fi movie, *sort* reorders movies by runtime, *filter* shows only R-rated movies, and *augment* appends genre labels to each title. Each adaptation modifies the existing interface in response to a user's natural language request without replacing or regenerating the GUI.

Abstract

Task-oriented chatbots can provide graphical user interface (GUI) components, such as button groups, calendars, and seat maps, alongside conversational interaction at each step of a structured workflow to facilitate user interaction. GUIs reduce cognitive load and increase efficiency by visually presenting available options and supporting structured choices that are difficult through conversation alone. In such GUI-augmented chatbots, when users express preferences or requests in natural language, a design question arises: how should the system bridge the gap between conversational user interaction and structured GUI interaction when they are available at the same time? Current approaches either fully automate UI operations on behalf of the user, removing user agency; or generate entirely new interfaces via LLMs, discarding the carefully designed original context. In this position paper, we propose an alternative approach: an agent layer that adaptively modifies existing GUIs to support decision-making when preferences are ambiguous or subjective, rather than making selections for the user, while performing standard GUI interactions on the user's behalf when intent is clear. To demonstrate the approach, we present TUNING (Task-oriented UI Notation for Informed Nudging and Guidance), an LLM agent equipped with four lightweight GUI adaptation tools (*highlight*, *sort*, *filter*, and *augment*) that modify the visual presentation of existing UI components without requiring backend access. Through a movie ticket booking scenario, we illustrate how these adaptations provide visual cues that help users make informed decisions while maintaining their agency over the task. We argue that this "adapt before automate" approach occupies a middle ground that preserves user agency in the design space of GUI agents.

CCS Concepts

• **Human-centered computing** → **Natural language interfaces**; **Graphical user interfaces**.

Keywords

GUI agents, task-oriented chatbots, adaptive user interfaces, natural language interaction, human-agent collaboration, LLM agents

1 Introduction

Task-oriented chatbots are increasingly augmented with graphical user interfaces (GUIs) to support structured, multi-step workflows [?]. Rather than relying solely on text-based dialogue, these systems

present visual components such as button groups for selection, calendars for date picking, and seat maps for reservation at each stage of a task, such as booking a movie ticket or scheduling an appointment. This hybrid design leverages the complementary strengths of conversation and direct manipulation: the chatbot guides users through sequential steps while GUIs reduce cognitive load by presenting available options visually [**?** **?**]. Yet studies have shown that purely conversational interfaces can lead to lower perceived autonomy and higher cognitive load compared to structured interfaces [**?**].

A fundamental design question remains underexplored in these hybrid systems: when users provide natural language input within a GUI-augmented chatbot, how should the system respond? Considering that a persistent gap may exist between user expectations and the actual capabilities of conversational agents [**?**], users in GUI-augmented chatbots are likely to make natural language requests that go beyond what the GUI affords. A user might type "I want something funny" while viewing a grid of movie options, or ask "which seats have the best view?" while a seat map is displayed. Such requests go beyond simple slot-filling: they express preferences, seek recommendations, or request information that the current GUI does not foreground. The challenge is to bridge user intent expressed in natural language with the structured GUI interaction that the chatbot provides. This challenge has deep roots: Cohen [**?**] argued that natural language excels at describing objects and specifying relations, while direct manipulation excels at establishing context and resolving reference, a complementarity that remains relevant in the LLM era. Recent work on conversational control of GUIs [**?** **?**], GUI-aware natural language interfaces [**?**], and voice-based UI assistance [**?**] has further explored this intersection, yet the emergence of LLM-based agents has fundamentally expanded the design space.

The rise of LLM-based GUI agents offers one answer: automate the GUI interaction entirely. Systems such as OpenAI Operator and Anthropic Computer Use demonstrate agents that interpret natural language instructions and autonomously navigate browsers and desktops through direct UI manipulation [**?** **?**]. In principle, such an agent could operate a movie booking chatbot on the user's behalf, clicking through each step to complete the reservation. However, current UI agents tend to execute tasks end-to-end without involving users at critical decision points or informing them of relevant contextual information [**?**]. Fully autonomous agents also face fundamental limitations of restricted reliability due to hallucination and difficulty handling complex tasks [**?**]. When an agent selects a movie, a showtime, or a seat without user involvement, it may optimize for efficiency but miss the user's true intent, particularly when preferences are ambiguous, contextual, or evolving.

Recognizing this limitation, recent work has explored approaches that preserve user involvement. Mixed-initiative systems such as CowPilot [**?**] allow users to pause, reject, or override agent-proposed actions during web navigation. Morae [**?**] proactively identifies decision points during task execution and pauses to present users with enriched, agent-generated selection interfaces that include additional contextual information. While these systems represent important progress, they remain fundamentally automation-centric: the default mode is agent execution, with user involvement triggered at identified decision points. A different line of work eschews automation entirely, instead generating new interfaces dynamically. GenerativeGUI [**?**] uses LLMs to produce HTML-based GUIs tailored to ongoing conversations, while Generative Interfaces [**?**] and Malleable UIs [**?**] propose that LLMs should respond with task-specific interfaces that evolve with user goals. While promising for open-ended tasks, generating entirely new interfaces in a structured workflow introduces discontinuity, discarding the familiar, step-by-step GUI that the user has been navigating.

In this position paper, we propose an alternative that occupies a distinct position in this design space. Rather than automating GUI interactions or generating new interfaces, we argue that agents should *adaptively modify existing GUIs* to support user decision-making. We present TUNING (Task-oriented UI Notation for Informed Nudging and Guidance), an agent layer for task-oriented chatbots that equips an LLM agent with four lightweight GUI adaptation tools (*highlight*, *sort*, *filter*, and *augment*) that modify the visual presentation of existing UI components without requiring backend access (Figure **??**). The agent performs standard GUI interactions on the user's behalf when intent is clearly expressed, such as selecting an item or advancing to the next step, but when preferences are ambiguous or subjective, it adaptively modifies the GUI to support decision-making rather than making selections for the user. This approach preserves the original interface context and the user's spatial familiarity while providing visual cues that facilitate informed decisions.

2 TUNING

2.1 Domain: Task-Oriented GUI Chatbots

We focus on a specific class of chatbot systems that combine conversational interaction with structured GUIs in a task-oriented workflow. In these systems, a task consists of a fixed sequence of steps (e.g., *movie* → *theater* → *date* → *time* → *seat* → *ticket type* → *confirm* for movie ticket booking), and each step presents a corresponding GUI component: a button group for movie selection, a calendar for date picking, and a seat map for choosing seats. Users progress through the workflow by interacting with these GUI components, and each interaction is recorded as part of the conversation history. Users can navigate backward to revise earlier choices, but the sequence of steps is fixed: no steps can be skipped or reordered within a given task configuration.

This domain is a simplified abstraction of multi-step interactions common in web and mobile apps. We focus on it to isolate the effects of adaptive GUI modification from broader navigation overhead (e.g., locating relevant controls across screens or pages). In this environment, the agent knows at any point which step the user is on, what GUI component is displayed, and what choices are available. Because all interactions are logged as chat messages, the agent also has full visibility into the user's prior selections and navigation history.

2.2 Agent Layer with GUI Adaptation Tools

TUNING introduces an agent layer that operates on top of the existing chatbot system, following a perception–planning–action–memory cycle (Figure **??**). In the *perception* phase, the agent observes the current step, the visible options rendered in the GUI, the full GUI state (including current selections, active adaptations,

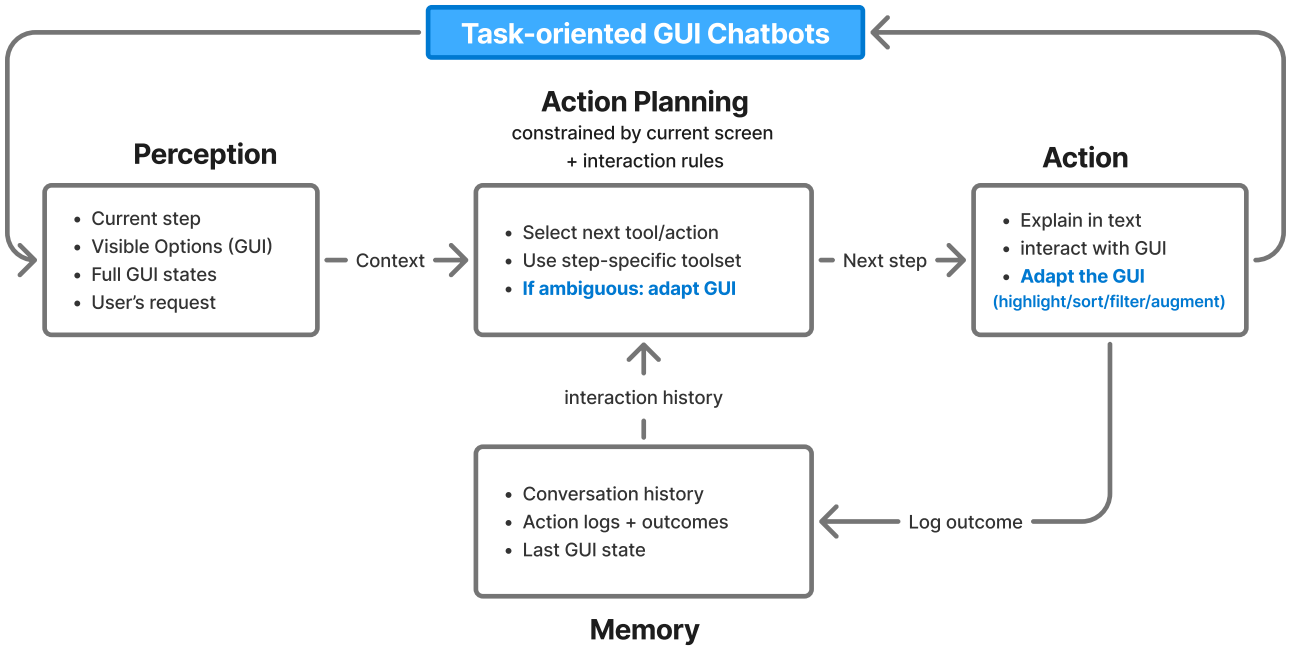

Figure 2: TUNING's agent layer follows a perception–planning–action–memory cycle on top of a task-oriented GUI chatbot. The agent perceives the current step and GUI state, plans an action from a step-specific toolset, executes standard interactions or GUI adaptations, and logs outcomes to memory for subsequent turns.

and step-level metadata), and the user's natural language request. During *action planning*, the agent selects the next tool from a *step-specific toolset*, constrained by the current screen and interaction rules. This toolset is dynamically scoped by both step and state (e.g., "next" is only available after required selections; "back" is absent at the first step; seat selection may be interpreted as toggle-based multi-select), and tools may be omitted when inapplicable. While the four adaptation tools keep the same names across steps, the agent instantiates their parameters from the current GUI (e.g., choosing the attribute/operator/value for filtering, the attribute/order for sorting, or the per-item content to inject for augmentation). When the user's intent is ambiguous, the agent plans a GUI adaptation rather than a definitive selection. The agent then executes the planned *action*: performing a standard GUI interaction (e.g., selecting an item or advancing to the next step), adapting the GUI via the four adaptation tools, or both, always pairing visual changes with a natural language explanation. Finally, the outcome is recorded in *memory* (the conversation history, action logs with outcomes, and the last GUI state), which feeds back into action planning on subsequent turns.

The toolset falls into two categories. First, *standard interaction tools* let the agent select GUI elements, advance to the next step, return to a previous step, or perform step-specific actions such as setting quantities or paging a calendar. When a user's intent is clear (e.g., "I'll go with this one"), the agent executes the corresponding interaction to keep the workflow moving; similarly, when a user wants to revise an earlier decision, the agent navigates back.

Second, four *GUI adaptation tools* form the core of our contribution. When preferences are ambiguous or subjective, the agent does not make task selections on the user's behalf; instead, it modifies the presentation of available options to support the user's own decision-making.

We define the four GUI adaptation tools as a design space of lightweight, frontend-only modifications, analogous to edits one might make through a browser's developer tools by modifying the rendered HTML and CSS:

Highlight. Visually emphasizes one or more UI elements to draw the user's attention. For example, when a user asks "what's popular right now?", the agent can highlight trending movies while explaining its recommendations. Highlighting preserves the full set of options while creating a visual hierarchy that guides attention.

Sort. Reorders elements within a GUI component based on a metadata attribute. When a user says "show me the cheapest options," the agent can sort items by price. Sorting leverages existing metadata and changes only the presentation order, not the available set.

Filter. Temporarily hides elements that do not match a specified condition, showing only a relevant subset. If a user says "I only want evening shows," the agent filters to display only matching options.

Augment. Injects additional information into existing UI elements by modifying their displayed content. The agent can append genre and runtime next to movie titles, attach remaining-seat indicators to showtimes, add view/price tags to seats on a seat map, or

add price indicators to calendar dates. Augmentation enriches the information density of the GUI without altering its structure.

These four tools can be organized along a spectrum of modification intensity: *highlight* modifies only visual emphasis; *sort* and *filter* restructure the presentation without changing content; and *augment* introduces new information into existing elements. Together, they span a design space from minimal visual cues to content enrichment, all while preserving the original interface structure (Figure **??**). Adaptations can be composed and are fully reversible (e.g., via a reset tool that clears active adaptations and restores the original GUI).

3 Scenario: Movie Ticket Booking

We illustrate TUNING's operation through a movie ticket booking scenario. A user opens the chatbot and reaches the movie selection step, where a button group displays 12 available movies with poster thumbnails and titles.

Movie Selection. The user types: "I'm in the mood for something funny, maybe an animation." Rather than selecting a movie automatically, the agent *filters* to show only comedies and animations (4 of 12) and *highlights* the two animated comedies among them. In the chat, the agent responds: "I found 4 comedies and animations for you. I've highlighted two animated comedies, [Movie A] and [Movie B]." The user browses all four options and clicks their choice.

Date Selection. A calendar GUI appears. The user asks: "When is it cheapest to go?" The agent *augments* the calendar by adding price indicators to each date ("$" for discount days, "$$$" for peak pricing) and *highlights* the lowest-price dates. It responds: "Ticket prices vary by day. I've marked the pricing on each date. Tuesdays and Wednesdays are the cheapest."

Seat Selection. The user reaches the seat map and types: "I want a good view but not too expensive." The agent *augments* each available seat with a view-rating and price-tier tag (e.g., "Great view / $$"), *filters* out premium-priced seats, and *highlights* the best value options. It explains: "I've highlighted seats with great views at moderate prices. Rows F and G look like the best value." If the user wants to reconsider, they can type "show me all seats" to reverse the filter.

Throughout this workflow, the agent provides paired output: a natural language explanation in the chat and a visual adaptation on the GUI. When a user's intent is explicit, the agent can carry out the corresponding GUI interaction (e.g., selecting an item and advancing). When preferences are subjective, it adapts the GUI and leaves the final choice to the user. The user retains full control over every selection, and all adaptations are recorded in the conversation history, maintaining a transparent and reversible interaction trail.

4 Discussion and Future Work

Positioning in the design space. TUNING's approach is complementary to existing GUI agent paradigms rather than a replacement. Full automation is appropriate when tasks are routine and preferences are unambiguous; generative UIs excel in open-ended, exploratory contexts. Adaptive modification fills the gap for structured workflows where user preferences are subjective, evolving, or ambiguous, precisely the situations where autonomous agents are most likely to make suboptimal decisions on the user's behalf. By modifying existing GUIs rather than operating or replacing them, the agent shifts from a decision-maker to a decision-support tool.

Constrained state spaces as an advantage. Unlike general-purpose GUI agents that must handle arbitrary web pages, TUNING operates within a well-defined task structure with known steps, GUI components, and metadata. This constrained state space enables more reliable agent behavior: the agent's tool selection is bounded by the current step, and each tool's effects are predictable and reversible. We argue that this constraint is not a limitation but a strength: by narrowing the scope of agent action, we achieve more dependable and trustworthy human-agent-UI interaction.

Open questions. Several questions remain for future investigation. First, while we present four adaptation tools, this set may need expansion as we explore additional task domains. Second, our design assumes that each GUI component's metadata is accessible to the agent; in practice, the degree of available metadata may vary across chatbot platforms. Third, the perceptual and cognitive effects of these adaptations, such as whether they reduce decision difficulty or introduce new biases like anchoring, require empirical validation. We plan to conduct user studies comparing TUNING's adaptive approach against text-only responses and full-automation baselines, measuring task performance, decision satisfaction, and perceived agency. Finally, the interplay between multiple simultaneous adaptations (e.g., filtering and augmenting at the same time) and their cumulative effect on interface clarity warrants careful study.

Toward a broader vision. As GUI agents become more capable, the default increasingly shifts toward full automation of user tasks. We argue that the field should equally invest in agents that make GUIs more useful, more informative, and more responsive to user needs: agents that help users *see* better, not agents that see *for* them.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.