CAN AGENT WORK FOR EVERYONE? CROSS-USER RELIABILITY FOR MOBILE GUI AGENTS IN PERSONALIZED USER INTERFACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Mobile GUI agents increasingly operate on interfaces influenced by users' histories and preferences, but their reliability across different users remains underexplored. We introduce PAIR (Personalized Application-state Instantiation and Rendering), a pipeline for constructing user-conditioned application states that enables controlled evaluation of the same task across different users. We further introduce RePAIR (Reinforcement learning with Personalization-Aware Interaction Rewards), a training approach that learns from cross-user differences in subgoal outcomes to improve reliability across user-conditioned mobile environments. Across six agents, we find substantial variation in task success across users and consistently lower subgoal achievement in user-conditioned UI contexts (6.98 to 15.4 pp). This gap further increases for personal targets drawn from each user's own content (8.77 to 22.0 pp). Failures in these contexts frequently involve selecting another item instead of the intended target, particularly before target exposure. Finally, RePAIR improves user-conditioned SAR (+5.87 pp), all-success (+7.50 pp), and overall Task SR (+9.42 pp) over its supervised fine-tuning parent on unseen users, providing initial evidence that explicitly learning from cross-user variation can improve GUI-agent reliability.

1 INTRODUCTION

Recent advances in mobile GUI agents have enabled them to interpret natural-language requests, interpret graphical user interfaces, and perform actions in a human-like manner (Zhou et al., 2025; Qin et al., 2025; Liu et al., 2024; Zhao et al., 2025a;a). However, GUI agents' action predictions are sensitive to interface presentation, including layout and element positioning (Yu et al., 2026; Wang et al., 2026b; Chen et al., 2025; Yang et al., 2026c). In deployed mobile GUI agents, interface variations naturally arise from users' prior interactions and activity histories.

Such user-dependent interface differences can lead to cross-user reliability failures, where an agent that succeeds for one user may fail on the same request for another. Figure 1 shows how user-dependent interfaces can lead an agent to different types of failure, for instance, selecting a visually similar but incorrect target or prematurely concluding that the requested target is unavailable. Our analysis of users' real mobile interactions from FingerTip 20K (Yang et al., 2026b) likewise indicates a 6.41% (SD 2.38) lower success rate (24.71% vs. 31.12%) on labeled target-bearing screen segments containing user-conditioned elements. However, because FingerTip 20k varies both tasks and screen states, it does not support controlled evaluation of the same request across user-conditioned interfaces.

To address this problem, we introduce PAIR (Personalized Application-state Instantiation and Rendering), a pipeline for constructing user-conditioned application states from user profiles. PAIR extracts application-relevant user preferences, matches them to candidate content for each application based on semantic similarity, selects the highest-ranked content, and renders it in each application's UI. By varying user-conditioned content while keeping the application structure, PAIR enables controlled cross-user evaluation of the same task. Additionally, we introduce RePAIR (Reinforcement learning with Personalization-Aware Interaction Rewards), a training-based approach to improve cross-user reliability by explicitly learning from differences in agent performance across users. RePAIR compares the outcomes of subgoals, goals associated with benchmark-defined UI contexts,

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



Figure 1: The same request can require navigating different user-conditioned interfaces. We evaluate whether GUI agents remain reliable across users and trace how user-conditioned content disrupts task progress and target selection.

across different users' environments under the same task condition and reinforces successful behavior in user-conditioned contexts, where the visible interface content varies across users. By leveraging these cross-user differences as a training signal, RePAIR aims to improve consistent task execution across user-conditioned application states.

Using the user-conditioned application states generated by PAIR, we evaluate whether agents complete the same task consistently across users and where cross-user reliability breaks down. Our evaluation reveals four main findings. First, task success varies substantially across users, with a maximum gap of 9.31 pp between the best- and worst-performing users. Second, subgoal achievement is consistently lower in user-conditioned contexts than in user-agnostic contexts, where the visible UI does not depend on the user (+12.0 pp). This gap becomes larger for personal targets where the agent must identify a target drawn from the user's own content (+14.4 pp). Third, in user-conditioned contexts, subgoal failures involve selecting another item instead of the intended target. These errors occur more frequently for personal targets, particularly before target exposure, when additional navigation is required to reach the intended target. Finally, RePAIR improves user-conditioned SAR by 5.87 points, the all-success rate by 7.50 points, and overall Task SR by 9.42 points over its SFT parent on unseen users, suggesting that explicitly training on cross-user differences in user-conditioned subgoals can improve reliability across unseen users.

We summarize our key contributions as follows:

- We introduce PAIR, a user profile-based pipeline for constructing user-conditioned GUI environments that enables controlled evaluation of the same task across different users.
- We provide an empirical study of cross-user reliability, showing (i) substantial variation in task success across users, (ii) lower subgoal achievement in user-conditioned contexts, and (iii) more target-selection errors for personal targets, especially before target exposure.
- We introduce RePAIR, a training approach that leverages cross-user differences in subgoal outcomes to improve reliable task execution across user-conditioned environments.

2 Related Work

2.1 GUI Agent Robustness under Interface Variation

GUI agents have evolved from grounding instructions in structured GUI representations to perceiving and acting on rendered interfaces (Deng et al., 2023; Cheng et al., 2024; Wu et al., 2025). Recent systems integrate visual grounding, reasoning, and action prediction in a unified interaction loop (Zheng et al., 2024; Qin et al., 2025). This shift highlights the need for robustness to variations

Table 1: Comparison of GUI-agent evaluation settings. The four dimensions are UI Variation, whether UI variations are explicitly evaluated; User Context, whether user preferences or histories are used; User-Conditioned GUI, whether personal records shape the app state; and Cross-User Comparison, whether the same task is evaluated across user-conditioned application states.

Evaluation setting	UI Variation	User Context	User-Conditioned GUI	Cross-User Comparison
GUI-Perturbed (Wang et al., 2026b)	✓	✗	✗	✗
GUI-Robust (Yang et al., 2026a)	✓	✗	✗	✗
D-GARA (Chen et al., 2026a)	✓	✗	✗	✗
PSPA-Bench (Nie et al., 2026)	✗	✓	✗	✗
Persona2Web (Kim et al., 2026)	✗	✓	✗	✗
KnowU-Bench (Chen et al., 2026b)	✗	✓	✗	✗
FingerTip 20K (Yang et al., 2026b)	✗	✓	✓	✗
iOSWorld (Jang et al., 2026)	✗	✓	✓	✗
PAIR (Ours)	✓	✓	✓	✓

in interfaces. Visual attributes affect web-agent selections (Yu et al., 2026), while zoom perturbations reduce grounding accuracy (Wang et al., 2026b). Prior methods address GUI complexity through context simplification (Chen et al., 2025) and adaptive visual focus (Ge et al., 2025). Meanwhile, benchmarks evaluate robustness using static anomaly datasets (Yang et al., 2026a) and dynamically injected disruptions (Chen et al., 2026a). Together, these studies show that reliable GUI interaction requires robustness to dynamic conditions, not just target recognition on fixed screens.

Personalization, however, creates persistent, user-conditioned variation beyond temporary or synthetic perturbations. HCI research shows that usage history and personalized content shape interface presentation (Findlater & McGrenere, 2004; Eslami et al., 2015), while user-specific records shape the content and context of modern applications (Jang et al., 2026). Existing robustness studies nevertheless focus on synthetic or transient changes, leaving reliability across persistent, personalized application states underexplored.

2.2 Personalization in Mobile GUI Agents

Existing personalization methods for mobile GUI agents use user preferences, routines, and histories to infer intended actions or provide proactive assistance (Wang et al., 2026a; Lyu et al., 2026). In contrast, we study whether an agent can reliably execute a fixed instruction when the mobile interface itself varies with user-specific content and state. Prior mobile evaluations examine either robustness to runtime interface disruptions (Chen et al., 2026a) or personalized interaction based on user context (Nie et al., 2026; Yang et al., 2026b; Jang et al., 2026; Chen et al., 2026b). However, they do not directly test whether the same instruction and target remain reliable across different user-conditioned mobile application states. We address this gap through controlled cross-user evaluation. As summarized in Table 1.

3 Motivation: Real-World Mobile GUI Agents Encounter User-Conditioned Interfaces

To investigate how user-conditioned content influences mobile GUI agents in everyday interfaces, we evaluate six widely used open-source agent models on Fingertip 20K (Yang et al., 2026b), which captures screenshots and action sequences from users' real mobile interactions. We analyze 103,199 screenshot–action steps spanning 12,016 episodes from 102 apps, collected from 83 users. We group consecutive interactions on the same screen into 60,383 screen segments, and success requires every scored action in a segment to be correct. Across all evaluated agents, segments containing user-conditioned elements, labels unique to a given user within each application, show a mean decrease in success of 4.88 pp (SD 1.50) relative to those without them (Figure 2a).

Figures 2b and 2c illustrate two types of errors influenced by user-conditioned elements. In the first case, the user asks to check the delivery status of the first order. The user intends to select the order's



(a) Success with/without user-conditioned elements.

(b) Mistaking user-conditioned order content for the task target.

(c) Overlooking the visible user-conditioned unread chat.

Figure 2: GUI-agent behavior on FingerTip 20K. (a) Success by the presence of user-conditioned content. (b,c) Example failures: the agent (b) selects a user-specific shop name instead of Track package and (c) overlooks a user-specific unread chat in favor of Discover group chats.

Track package button, but the agent selects the shop name instead, incorrectly assuming it leads to the order details, where delivery information can be found. In the second case, the user asks to view new messages in a group chat. The intended action is to open the visible group conversation with an unread badge, but the agent selects the shared *Discover group chats* entry instead, incorrectly assuming that it provides access to new messages in the user's group chats. These examples highlight the need to interpret user-conditioned content in relation to the instruction and identify the appropriate target in the current application state. See more details in Appendix A.

These observations suggest that user-conditioned content can interfere with reliable action selection. However, because FingerTip varies tasks and screen states together, it cannot determine whether the same request is completed consistently across different users' application states. We refer to this consistency as cross-user reliability. We therefore suggest PAIR to control for the instruction and target while varying user-conditioned application states and study the following questions:

- **RQ1:** How robustly do mobile GUI agents complete the same request across user-conditioned interface states?
- **RQ2:** Where does task progress break down in user-conditioned interfaces, and how do failure patterns vary with the semantic alignment between the target and the interface?

4 PAIR: Personalized Application Environment Generation

We introduce PAIR (Personalized Application-state Instantiation and Rendering), a pipeline that turns user profiles into personalized, executable GUI environments. PAIR provides a controlled cross-user evaluation environment by pairing user-conditioned application states with matched task templates and success criteria. This section describes the environment generation pipeline, task design, and evaluation protocol. Figure 3 illustrates the pipeline overview.

4.1 Cross-User Reliability Formulation

Our goal is to assess cross-user reliability, defined as the ability of a mobile GUI agent to execute the same request consistently across different users' interfaces, and to identify where task progress breaks down during execution. Given an instruction q and user p's application state, the agent observes a screenshot o_t and selects an action a_t at each interaction step t:

$$a_t = \pi(q, h_t), \qquad s_t = (o_t, a_t), \tag{1}$$

where π denotes the GUI agent, $h_t = (o_0, a_0, \ldots, o_t)$ is the interaction history available to the agent, and s_t denotes the interaction step. Each action updates the application state, and repeating this process yields the interaction trajectory $\tau_p = (s_0, s_1, \ldots, s_{T-1})$.

For each task, let $\mathbf{g}(q) = (g_1, \ldots, g_K)$ denote the ordered subgoals along a reference execution path. Each subgoal g_k specifies an objective associated with a predefined UI context in which that objective

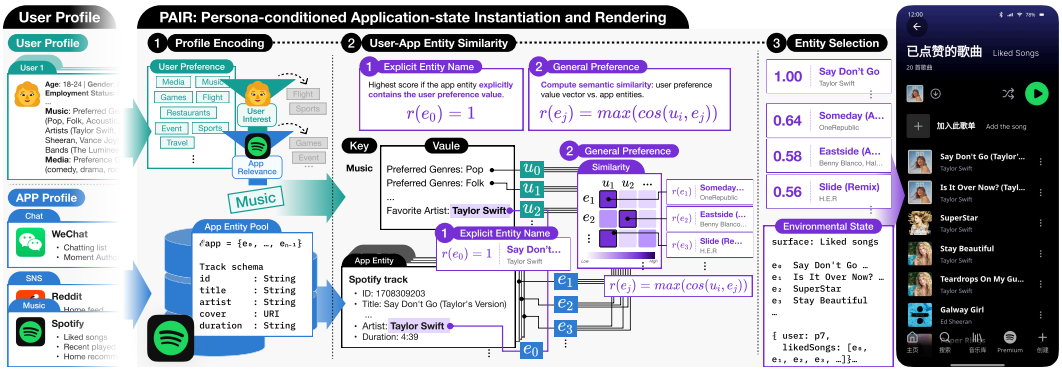


Figure 3: **Overview of PAIR.** (i) Given a user and app profile, PAIR extracts user preferences and filters them according to their relevance to each application. (ii) It then scores candidate application entities using exact matching for explicit entity preferences and semantic similarity for general preferences. (iii) The highest-ranked entities are selected and instantiated in the corresponding application state, which is rendered as a user-conditioned interface.

can be completed, which we call its *subgoal context*. For user p and subgoal g_k, let $I_{p,k}$ denote the set of step indices, ordered by time, that occur in the corresponding subgoal context. We define the context-specific interaction sequence as $\tau_{p,k} = (s_t)_{t \in I_{p,k}}$. Because an agent may revisit the same context multiple times, $I_{p,k}$ may contain non-consecutive indices, and $\tau_{p,k}$ therefore aggregates interactions across repeated visits to the same subgoal context.

In our setting, the visible interface state varies across users while the underlying application structure and interaction logic remain fixed. We distinguish user-conditioned (UC) contexts, where relevant visible content varies across users, from user-agnostic (UA) contexts, where the relevant interface content is shared across users, such as common navigation controls. Accordingly, each subgoal context receives a label $c_{p,k} \in \{\text{UC}, \text{UA}\}$: UC when it contains relevant visible user-conditioned elements, and UA otherwise. This labeling groups the corresponding subtrajectories by context type.

4.2 Environment Generation Pipeline

PAIR constructs user-conditioned application states from interests and attributes derived from user profiles, as illustrated in Figure 3. The framework proceeds in three stages: (i) *profile encoding*, which extracts application-relevant preferences from the user profile and candidate entities, content items that can appear in the application state, from the application profile; (ii) *user-app entity similarity*, which measures their correspondence through exact matching or semantic similarity; and (iii) *entity selection*, which selects the highest-ranked entities to instantiate the application state.

We first apply user interest filtering to retain preferences associated with user p's active interests, followed by app relevance filtering to retain those relevant to application A. Let $\mathcal{P}_{p,A} = \{u_i\}$ denote the filtered preference values for user p and application A, and let $\mathcal{E}_A = \{e_j\}$ denote the candidate entity pool for application A. For each candidate application entity e_j, we define its relevance as

$$r(e_j) = \begin{cases} 1, & \text{if } e_j \text{ directly matches any preference value } u_i \in \mathcal{P}_{p,A}, \\ \max_i \cos(f(u_i), f(e_j)), & \text{otherwise.} \end{cases} \tag{2}$$

where $f(\cdot)$ is a multilingual sentence encoder. We rank candidate entities by $r(e_j)$ and use the highest-ranked entities to populate application-specific collections. To instantiate executable environments, we build on MobileGym Wu et al. (2026), injecting the selected entities into user-specific runtime states according to each application's native schema. Structured records are created in their respective schema. The resulting states are rendered by the original application UI. (See Appendix B.1)

4.3 Task Design

PAIR constructs task instances by pairing generated user-conditioned states with task templates. Each task template defines the required operation, target UI context, and success criterion, while the target

and instruction are varied across conditions. For each user and template, we instantiate tasks along two axes: *target type* and *instruction type*. For more information, see Appendix B.2, B.3.

Target type. Targets are either common or personal. A common target is shared across users; its instruction wording is fixed. This enables comparison of the same task across user-conditioned states. In contrast, a personal target is selected from the user-conditioned content generated by PAIR, testing whether the agent can resolve a target specific to that user.

Instruction type. An explicit instruction refers to the task target by its visible label, whereas an implicit instruction refers to the same target using a contextual cue. For example, an explicit instruction requests, "Find the post 'Looking for interesting flowers for a fashion project!'", whereas an implicit instruction describes the same target as "a gardening post seeking flower ideas."

4.4 Evaluation Metrics

Task success rate (Task SR) measures end-to-end task completion as $\mathrm{Task\,SR} = E_s/E$, where E is the total number of evaluated tasks and E_s is the number of successfully completed tasks.

All-success rate measures cross-user consistency for each template-condition pair, defined as a task template under a fixed target type and instruction type. The all-success rate is defined as $\mathrm{AllSuccess} = M_a/M$, where M denotes the number of template-condition pairs and M_a denotes the number of pairs that achieve all-success; the agent succeeds for every user.

Subgoal achievement rate (SAR). measures the proportion of these subgoals that are successfully completed. We define it as $\mathrm{SAR} = S_s/S$, where S is the number of evaluated subgoals and S_s is the number successfully completed. SAR enables trajectory-level localization of failures and comparison of subgoal performance between user-conditioned and user-agnostic contexts.

5 RePAIR: Personalization-Aware Interaction Rewards

To investigate whether rewarding subgoal success in user-conditioned contexts can mitigate performance degradation in mobile GUI agents, we introduce RePAIR (Reinforcement learning with Personalization-Aware Interaction Rewards), which uses relative subgoal success across user-conditioned rollouts as the policy optimization signal; see Appendix C for more detail.

Rollout Group Construction. For each combination of task template, target type, and instruction type, we pair training users p and q with compatible subgoal structures. We collect two full-task rollouts from each user's task instance, forming the group $\mathcal{G}_{p,q} = (\tau_1, \tau_2, \tau_3, \tau_4)$, where τ_1 and τ_2 come from p and τ_3 and τ_4 from q. Each rollout starts from the corresponding initial state.

Personalization-Aware Rewards. For rollout group $\mathcal{G}_{p,q}$, let $\mathcal{P}$ denote its shared set of subgoals in user-conditioned contexts; subgoals in user-agnostic contexts are excluded. For rollout τ_j and subgoal $k \in \mathcal{P}$, we assign a binary success reward:

$$r_{jk} = \begin{cases} 1, & \text{if } \tau_j \text{ satisfies the criteria for subgoal } k, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

We determine subgoal success from recorded trajectory evidence, including entry-screen observations, visited screens, clicks, and task-judge outcomes.

Group-Relative Advantages. For each subgoal, we normalize its rewards across the four rollouts and then average the normalized rewards across user-conditioned subgoals to obtain one rollout-level advantage. Let μ_k and σ_k be the mean and population standard deviation of the rewards for subgoal k. For nonempty $\mathcal{P}$, we compute

$$A_j = \frac{1}{|\mathcal{P}|} \sum_{k \in \mathcal{P}:\, \sigma_k > 0} \frac{r_{jk} - \mu_k}{\sigma_k}. \tag{4}$$

For example, suppose $\mathcal{P}$ contains two subgoals. Across the four rollouts, the first subgoal has reward vector $(1, 1, 0, 0)$ and thus normalized reward vector $(1, 1, -1, -1)$. The second has reward vector $(1, 1, 1, 1)$ and thus contributes zero to every rollout. Averaging the two subgoal-level values for each rollout gives $(A_1, A_2, A_3, A_4) = (0.5, 0.5, -0.5, -0.5)$. If $\mathcal{P}$ is empty, we set $A_j = 0$.

Table 2: Overall performance of base mobile GUI agents and RePAIR on PAIR. Task SR measures task success, all-success measures the rate of tasks completed successfully by all user profiles, and SAR measures subgoal achievement over reached contexts. Parentheses show RePAIR's percentage-point gains over SFT.

Model	Params.	Task SR (%)	All-success (%)	User-agnostic SAR (%)	User-conditioned SAR (%)
UI-TARS-1.5 (Qin et al., 2025)	7B	48.5	22.5	79.7	65.4
Qwen3-VL-Instruct (Bai et al., 2025)	30B-A3B	47.6	19.2	79.3	67.2
Qwen3-VL-Instruct (Bai et al., 2025)	8B	47.5	22.5	74.6	67.6
AutoGLM-Phone (Xu et al., 2026b)	9B	53.7	21.7	85.3	69.9
MAI-UI (Zhou et al., 2025)	8B	53.4	25.8	86.2	71.0
GUI-Owl-1.5-Think (Xu et al., 2026a)	8B	49.5	16.7	77.4	69.1
Qwen3-VL-Instruct	8B	47.5	22.5	74.6	67.6
+ SFT	8B + 15.3M	55.1	20.0	92.3	71.9
+ SFT + RePAIR (ours)	8B + 15.3M	**64.5 (+9.4)**	**27.5 (+7.5)**	**94.3 (+2.0)**	**77.7 (+5.9)**

Policy Optimization. We optimize an SFT-initialized policy π_θ with GRPO (Shao et al., 2024) using the rollout groups and advantages defined above, while keeping the SFT policy fixed as the reference π_{ref}. Let π_{old} denote the rollout-generating policy. For rollout τ_j, let N_j be the number of generated output tokens. At each token position $t \in \{1, \ldots, N_j\}$, w_{jt} denotes the generated token and h_{jt} the instruction and observation–action history available before generating it. We maximize

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{G}_{p,q} \sim \pi_{\text{old}}} \frac{1}{4} \sum_{j=1}^{4} \frac{1}{N_j} \sum_{t=1}^{N_j} \left[\min \left(\frac{\pi_\theta(w_{jt} \mid h_{jt})}{\pi_{\text{old}}(w_{jt} \mid h_{jt})} A_j, \text{clip}\left(\frac{\pi_\theta(w_{jt} \mid h_{jt})}{\pi_{\text{old}}(w_{jt} \mid h_{jt})}, 1-\varepsilon, 1+\varepsilon \right) A_j \right) - \beta D_{\text{KL},jt} \right]. \tag{5}$$

Here ε sets the clipping range, while β controls the KL penalty from the SFT reference. For each sampled token, the penalty is

$$D_{\text{KL},jt} = \frac{\pi_{\text{ref}}(w_{jt} \mid h_{jt})}{\pi_\theta(w_{jt} \mid h_{jt})} - \log \frac{\pi_{\text{ref}}(w_{jt} \mid h_{jt})}{\pi_\theta(w_{jt} \mid h_{jt})} - 1. \tag{6}$$

Each rollout is weighted equally, and its advantage applies to all response tokens.

6 Experiments

We evaluate cross-user reliability at both the task and trajectory levels using PAIR. Specifically, we examine (1) how task success and (2) how subgoal achievement are affected across user-conditioned application states, (3) when incorrect-item selections occur relative to target visibility, and (4) whether RePAIR improves performance by rewarding subgoal success in user-conditioned contexts.

6.1 Experimental Setup

Using user profiles from PersonaLens (Zhao et al., 2025b), we evaluate PAIR by instantiating ten profiles (p0 to p9) with PAIR as the test set for our main evaluation. We then instantiate ten additional profiles (p10 to p19) for the RePAIR study, using eight for training and two for validation. For each user profile, PAIR instantiates three task templates across ten applications under two target types and two instruction types, yielding $3 \times 10 \times 2 \times 2 = 120$ instantiated tasks per persona. This results in 1,200 test tasks, 960 training tasks, and 240 validation instances.

Models. We evaluate Qwen3-VL-8B, GUI-Owl-1.5-8B-Think (Xu et al., 2026a), MAI-UI-8B (Zhou et al., 2025), AutoGLM-Phone-9B-Multilingual (Liu et al., 2024; Xu et al., 2026b), and UI-TARS-1.5-7B (Qin et al., 2025), which fall within a deployment-relevant model scale for mobile settings, together with and Qwen3-VL-30B-A3B (Bai et al., 2025) as a larger-scale comparison, on the same

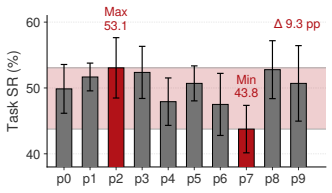
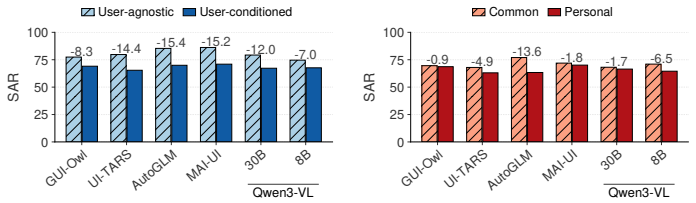


Figure 4: User-level task success rate(Task SR). Error bars show standard deviation across agents.



(a) UA vs. UC contexts. (b) Common vs. personal targets.

Figure 5: Subgoal achievement rate(SAR) across agents by (a) context type and (b) target type within user-conditioned contexts.

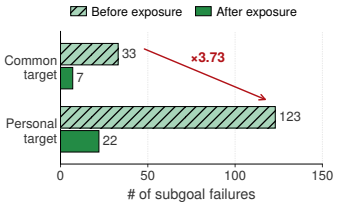
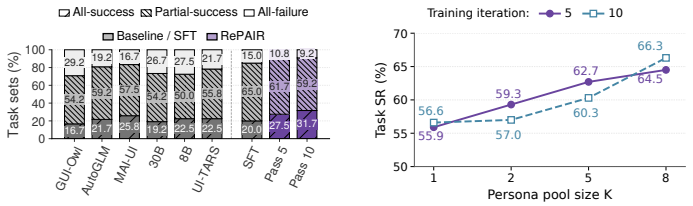


Figure 6: Incorrect item selections among subgoal failures by target exposure and target type.



(a) Task SR composition. (b) Task SR by user pool size.

Figure 7: RePAIR results. (a) Task success consistency across users. (b) Task SR by training user pool size and training iteration.

frozen test set. For the RePAIR, we first fine-tune Qwen3-VL-8B with LoRA (Hu et al., 2021) on explicit-instruction demonstrations from the training personas. We then apply RePAIR to tasks selected based on the preceding cross-persona analysis. More details are provided in Appendix D.

6.2 Result

To assess overall task completion, cross-user consistency, and subgoal progress, we examine the task success rate (Task SR), the all-success rate, and the subgoal achievement rate (SAR) (see Table 2). Across the six agents, mean Task SR is 50.0% (SD 2.82). As we expected, task success decreases when the instruction is implicit rather than explicit (40.7% vs. 59.3%; $p < .001$). Personal targets also show lower task success than common targets (46.6% vs. 53.5%). Neither this difference ($p = .0642$) nor the instruction–target interaction ($p = .283$) reached statistical significance.

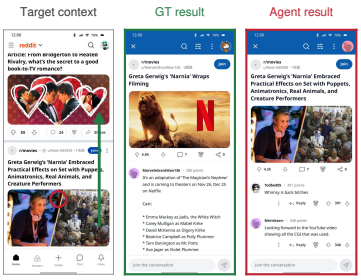
Task success varies substantially across personas. We therefore examine how task outcomes vary across personas. Across the evaluated agents, as illustrated in Figure 4, persona-level Task SR ranges from 43.8% to 53.1%, with a gap of up to 9.31 pp between the best- and worst-performing personas. This cross-persona variation occurs for both common targets with identical instructions and targets across users (6.94 pp) and personal targets drawn from each user's own records (13.6 pp). Thus, the same task can produce different outcomes depending on the persona-conditioned interface in which it is executed.

Goal achievement drops in user-conditioned contexts, with difficulty for personal targets. Across all six complete agents, the SAR is consistently lower in user-conditioned than in user-agnostic contexts, with an average drop of 12.1 (SD 3.64) pp (Figure 5(a)). Within user-conditioned contexts, personal targets yield lower SAR than common targets across all six agents (Figure 5(b)). These results suggest task progress is more difficult in interface regions containing user-conditioned content, particularly when the target is resolved from semantically related items.

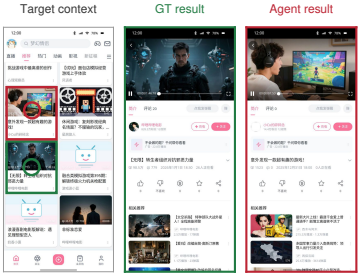
Beyond the aggregate SAR gap, the trajectory-level analysis reveals localized peaks in failure at specific subgoals (see Figure 11 in Appendix E). High failure rates frequently occur at subgoal conduct in user-conditioned contexts. This pattern appears across multiple applications and task types, suggesting that the difficulty is not confined to a single interface or operation. From a trajectory-level perspective, user-conditioned contexts can therefore act as major bottlenecks, disrupting task progress even when other stages are completed successfully.

Personal targets increase competing-item selection errors. To better understand failures in user-conditioned contexts, we further analyze the error patterns that arise when agents resolve targets. For both common and personal targets, incorrect-item selection failures occur more frequently before target exposure, when further navigation is required, than after exposure, with personal targets exhibiting more failures overall (Figure 6).

A closer inspection of these failures reveals two recurring patterns depending on whether the intended target has been exposed. Figure 8 illustrates representative personal target failures under explicit instructions. In the first case, the agent opens a similarly titled user-specific post before the intended target appears, rather than continuing to swipe. In the second case, the intended video is already visible, yet the agent selects a different video from the same personalized list. These examples indicate that agents may commit to a competing item before reaching the target, or fail to distinguish it from surrounding items even after it becomes visible. Notably, both failures occur despite the target being named explicitly, suggesting that exact target specification alone does not eliminate interference from competing user-specific context (Appendix F).



(a) Before target exposure.



(b) After target exposure.

Figure 8: Personal-target failures in user-conditioned contexts: the agent selects an incorrect item (a) before target exposure and (b) after the target becomes visible. Red: agent action; green: ground truth.

RePAIR improves cross-user reliability on unseen personas. We first adapt the base model to the benchmark tasks with supervised fine-tuning. This substantially improves SAR in user-agnostic contexts (+17.7 pp) but yields smaller gains in user-conditioned contexts (+4.24 pp) and no improvement in all-success, leaving difficulties in user-conditioned contexts and cross-user inconsistency unresolved. We then apply RePAIR to examine whether learning from cross-user differences improves reliability in user-conditioned contexts. RePAIR improves user-conditioned SAR (+5.87 pp), the all-success rate (+7.50 pp), and overall Task SR (+9.42 pp) on unseen users. The gain in user-conditioned SAR indicates greater robustness to user-conditioned interface variation, while the increase in the all-success rate reflects more consistent task completion across users (see Figure 7(a)).

The concurrent improvement in task success further shows that these gains translate to better end-to-end performance. To this end, we vary the training user pool size n_{p} and the number of training iterations. Increasing n_{p} from 1 to 8 improves overall Task SR after both 5 (+8.6 pp) and 10 (+9.7 pp) iterations (Figure 7(b)). However, extending training from 5 to 10 iterations decreases Task SR for $n_{\mathrm{p}} = 2$ (-2.3 pp) and $n_{\mathrm{p}} = 5$ (-2.4 pp), while improving it for $n_{\mathrm{p}} = 8$ (+1.8 pp). These results suggest that the benefit of additional RePAIR training depends on user-pool diversity, with repeated comparisons improving unseen-user performance only when training spans a broad set of users.

7 Conclusion and Future Work

We systematically study cross-user reliability in mobile GUI agents: whether the same request is completed consistently across user-conditioned application states. We introduce PAIR, which constructs such states for controlled evaluation, and RePAIR, which trains from cross-user differences in user-conditioned subgoal outcomes. Experiments reveal cross-user performance variation, lower subgoal achievement in user-conditioned contexts, especially for personal targets, and incorrect selections before and after target exposure; RePAIR improves performance and consistency on unseen personas, providing initial evidence that cross-user differences can serve as a useful training signal.

These findings suggest three directions for future work. 1) Develop training methods that leverage cross-user differences in subgoal outcomes to improve reliability in user-conditioned contexts. 2) Strengthen target grounding through uncertainty-aware search, target verification, and recovery from incorrect-item selections before and after target exposure. 3) Determine the optimal balance between persona pool size and the number of training iterations to improve generalization to unseen users.

AI USE STATEMENT

We used generative AI tools to assist with method implementation, debugging, translation, and manuscript editing. The authors developed the research hypotheses and experimental design without generative AI assistance. The authors take full responsibility for all content and artifacts in this work.

REPRODUCIBILITY STATEMENT

We provide detailed specifications of the PAIR environment-generation pipeline, task construction, evaluation protocol, and RePAIR training procedure in Appendices B.1–B.3 and C–D, including all task templates, reference subgoals, optimization settings, and training pseudocode. Code for PAIR and RePAIR, together with the benchmark generation and evaluation artifacts, will be released upon acceptance.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Gongwei Chen, Xurui Zhou, Rui Shao, Yibo Lyu, Kaiwen Zhou, Shuai Wang, Wentao Li, Yinchuan Li, Zhongang Qi, and Liqiang Nie. Less is more: Empowering gui agent with context-aware simplification. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5901–5911. IEEE, 2025.

Sen Chen, Tong Zhao, Yi Bin, Fei Ma, Wenqi Shao, and Zheng Wang. D-gara: A dynamic benchmarking framework for gui agent robustness in real-world anomalies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17419–17426, 2026a.

Tongbo Chen, Zhengxi Lu, Zhan Xu, Guocheng Shao, Shaohan Zhao, Fei Tang, Yong Du, Kaitao Song, Yizhou Liu, Yuchen Yan, et al. Knowu-bench: Towards interactive, proactive, and personalized mobile agent evaluation. *arXiv preprint arXiv:2604.08455*, 2026b.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.505.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Motahhare Eslami, Aimee Rickman, Kristen Vaccaro, Amirhossein Aleyasen, Andy Vuong, Karrie Karahalios, Kevin Hamilton, and Christian Sandvig. " i always assumed that i wasn't really that close to [her]" reasoning about invisible algorithms in news feeds. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 153–162, 2015.

Leah Findlater and Joanna McGrenere. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 89–96, 2004.

Zhiqi Ge, Juncheng Li, Xinglei Pang, Minghe Gao, Kaihang Pan, Wang Lin, Hao Fei, Wenqiao Zhang, Siliang Tang, and Yueting Zhuang. Iris: Breaking gui complexity with adaptive focus and self-refining. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–10. IEEE, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Lawrence Keunho Jang, Mareks Woodside, Geronimo Carom, Andrew Keunwoo Jang, Jing Yu Koh, and Ruslan Salakhutdinov. iosworld: A benchmark for personally intelligent phone agents. *arXiv preprint arXiv:2606.09764*, 2026.

Serin Kim, Sangam Lee, and Dongha Lee. Persona2web: Benchmarking personalized web agents for contextual reasoning with user history. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qvvD9hgHoX.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yibo Lyu, Gongwei Chen, Rui Shao, Weili Guan, and Liqiang Nie. Personalalign: Hierarchical implicit intent alignment for personalized gui agent with long-term user-centric records. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36074–36089, 2026.

Hongyi Nie, Xunyuan Liu, Yudong Bai, Yaqing Wang, Yang Liu, Quanming Yao, and Zhen Wang. Pspa-bench: A personalized benchmark for smartphone gui agent. *arXiv preprint arXiv:2603.29318*, 2026.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

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

Shuoxin Wang, Chang Liu, Gowen Loo, Lifan Zheng, Kaiwen Wei, Xinyi Zeng, Jingyuan Zhang, and Yu Tian. Me-Agent: A personalized mobile agent with two-level user habit learning for enhanced interaction, 2026a. URL https://arxiv.org/abs/2601.20162.

Yangyue Wang, Harshvardhan Sikka, Yash Mathur, Tony Zhou, Jinu Nyachhyon, and Pranav Guruprasad. Gui-perturbed: Domain randomization reveals systematic brittleness in gui grounding models. *arXiv preprint arXiv:2604.14262*, 2026b.

Dingbang Wu, Rui Hao, Haiyang Wang, Shuzhe Wu, Han Xiao, Zhenghong Li, Bojiang Zhou, Zheng Ju, Zichen Liu, Lue Fan, and Zhaoxiang Zhang. MobileGym: A verifiable and highly parallel simulation platform for mobile GUI agent research. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2026. URL https://arxiv.org/abs/2605.26114. To appear.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, Si Qin, Lars Liden, Qingwei Lin, Huan Zhang, Tong Zhang, Jianbing Zhang, Dongmei Zhang, and Jianfeng Gao. GUI-actor: Coordinate-free visual grounding for GUI agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=5fSkinHw7w.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026a.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Jiayu Huang, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Yuxiao Dong, et al. Mobilerl: Online agentic reinforcement learning for mobile gui agents. In *International Conference on Learning Representations*, volume 2026, pp. 35282–35315, 2026b.

Jingqi Yang, Zhilong Song, Jiawei Chen, Mingli Song, Sheng Zhou, Linjun Sun, Xiaogang Ouyang, Chun Chen, and Can Wang. Gui-robust: A comprehensive dataset for testing gui agent robustness in real-world anomalies. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2842–2853, 2026a.

Qinglong Yang, Haoming Li, Haotian Zhao, Xiaokai Yan, Jingtao Ding, Fengli Xu, and Yong Li. Fingertip 20k: A benchmark for proactive and personalized mobile LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=n3iFV0gLMc.

Wenkui Yang, Chao Jin, Haisu Zhu, Weilin Luo, Derek Yuen, Kun Shao, Junxian Duan, Huaibo Huang, Jie Cao, and Ran He. Are gui agents focused enough? automated distraction via semantic-level ui element injection. In *European Conference on Computer Vision*, pp. 115–132. Springer, 2026c.

Kuai Yu, Naicheng Yu, Han Wang, Rui Yang, and Huan Zhang. How do visual attributes influence web agents? a comprehensive evaluation of user interface design factors. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 34520–34536, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1723. URL https://aclanthology.org/2026.findings-acl.1723/.

Yuyang Zhao, Wentao Shi, Fuli Feng, and Xiangnan He. AppAgent-Pro: A Proactive GUI Agent System for Multidomain Information Integration and User Assistance, August 2025a. URL http://arxiv.org/abs/2508.18689. arXiv:2508.18689 [cs].

Zheng Zhao, Clara Vania, Subhradeep Kayal, Naila Khan, Shay B Cohen, and Emine Yilmaz. Personalens: A benchmark for personalization evaluation in conversational ai assistants. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18023–18055, 2025b.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024. URL https://proceedings.mlr.press/v235/zheng24e.html.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025.

A FINGERTIP-20K ANALYSIS

Data. All six models are evaluated on the same 103,199 steps from the FingerTip-20K (Yang et al., 2026b) training split: 12,016 episodes, 83 users and 102 app labels. We retain supported actions with available screenshots from apps used by at least five users. Each prediction uses the recorded screenshot, instruction and human action history. Later screenshots follow the human demonstration.

Screen labels and segments. A text label is user-specific if its normalized form appears for only one user within an app. A screen is user-conditioned (UC) if it contains such a label. Numeric and time labels are included; icons and unknown labels are excluded. Labels are not filtered by pixel visibility. Consecutive screens form a segment until the Jaccard similarity of labels outside scrollable regions falls below 0.475; two empty sets have similarity one. This gives 60,383 segments: 51,761 UC and 8,622 without UC elements.

Scoring. An action must have the correct type. For clicks and long clicks, the pixel distance from the reference position, divided by the image diagonal and rounded to four decimals, must be at most 0.14. Scroll direction must match; text is compared after removing whitespace and lowercasing. Other supported actions are scored by type, and parsing failures are incorrect. A segment succeeds only if all its actions are correct. All 60,383 segments are included.

Target exposure. The exposure analysis uses 29,640 segments with a labeled exit target. A clickable element matches the target label if character-bigram Jaccard similarity is at least 0.85, target-bigram coverage is at least 0.8, the shared prefix has at least six characters, or the matching substring has at least eight. This text-based measure does not establish visual attention. Differences between screen groups are descriptive; tasks, apps and segment lengths are not controlled.

B PAIR IMPLEMENTATION, SCREENS AND DATASET STATISTICS

B.1 USER-CONDITIONED APPLICATION STATE GENERATION

PAIR converts PersonaLens (Zhao et al., 2025b) interests into app-specific queries. Content is ranked by name matches (score 1) or cosine similarity using `paraphrase-multilingual-MiniLM-L12-v2` (Reimers & Gurevych, 2019) with mean pooling, L2 normalization and a 96-token limit. Duplicate IDs/names and reserved Common items are removed before selection. Common items are then added, and lists are shuffled using seed-0 hashes. Most lists contain 20 items. App-specific generators also create transactions, chats and meetings. The records keep their IDs and media links and are rendered by MobileGym (Wu et al., 2026).

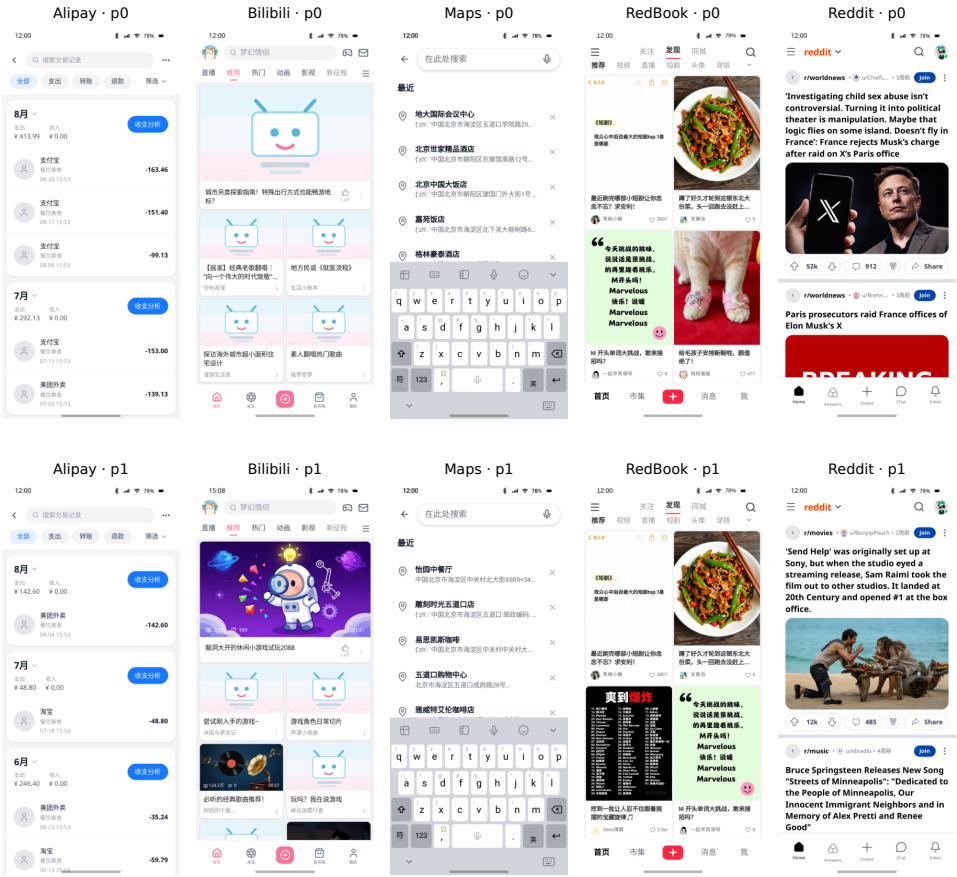


Figure 9: Generated screens for p0 (top) and p1 (bottom). Each column shows the same app and screen location for both personas.

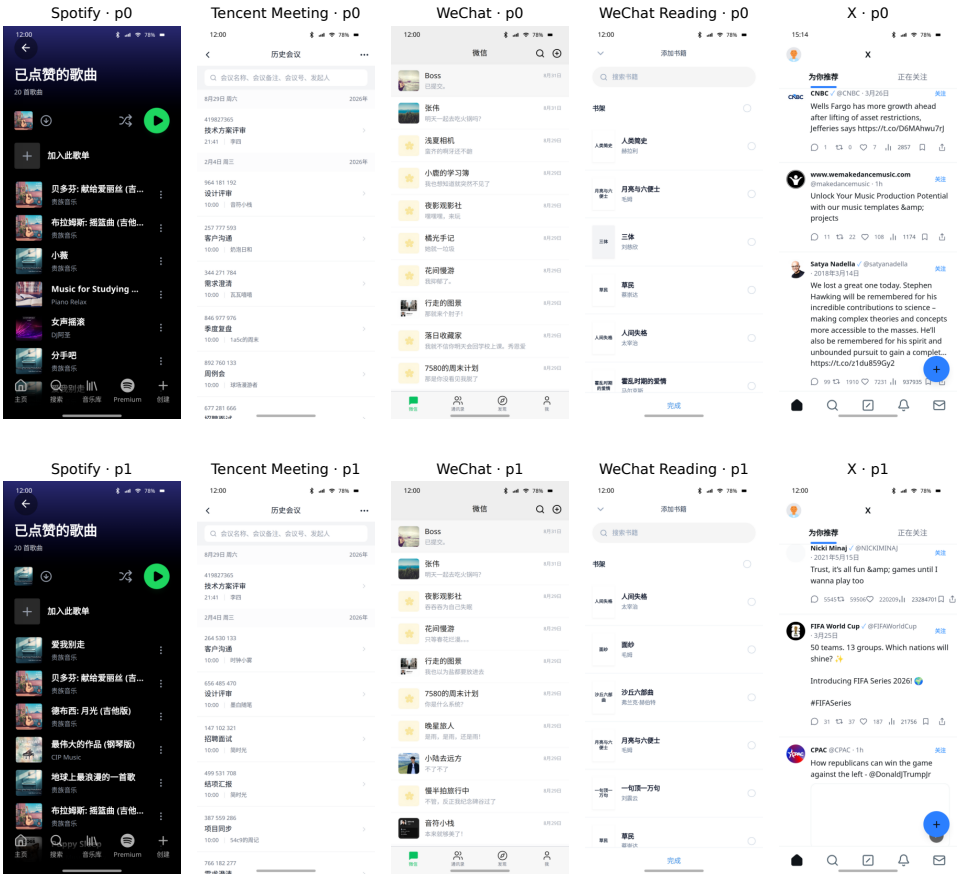


Figure 10: Persona comparison for the remaining five apps: p0 (top), p1 (bottom).

B.2 TASK GENERATION

Common targets are shared across personas; Personal targets come from each persona's data. Explicit and Implicit instructions refer to the same target. All four conditions use the same initial state for a template/persona pair, reset before each episode. The instruction writer (`openai/gpt-oss-120b` (OpenAI, 2025)) and reviewer use the candidate records. Agents receive only screenshots, instructions and action history.

All 1,200 test instances are evaluated against their designated targets, including cases with unresolved review flags. Automatic review does not guarantee a unique interpretation of every Implicit instruction.

Table 3: Apps and target screens. RedBook is also called Xiaohongshu.

Application	Target UI contexts
Alipay	Billing history; message list
Bilibili	Subscriptions; home recommendations; liked videos
Maps	Recent searches
RedBook	Saved notes; liked notes; discover feed
Reddit	Home feed
Spotify	Liked songs; recently played tracks; home playlists
Tencent Meeting	Meeting history; scheduled meetings
WeChat	Chat list; Moments
WeChat Reading	Home reading list; Add Favorite Book; recommendations
X	For You feed

Table 4: All 30 task templates. Each operation applies to the designated Common/Personal target under Explicit/Implicit instructions.

Application	**Task ID**	**Target**	**Task Template**
Alipay	`GG-ALI-01`	Transaction	Open the transaction in billing history.
Alipay	`GG-ALI-02`	Conversation	Open the conversation in the message list.
Alipay	`GG-ALI-03`	Conversation	Open the conversation; reply exactly "Understood".
Bilibili	`GG-BIL-01`	Video	Open the subscription video and like it.
Bilibili	`GG-BIL-02`	Video	Open the home recommendation video.
Bilibili	`GG-BIL-03`	Video	Open the video in Liked Videos.
Maps	`GG-MAP-01`	Place	Open place details from recent searches.
Maps	`GG-MAP-02`	Place	Get directions to the place in recent searches.
Maps	`GG-MAP-03`	Place	Open place details from recent searches.
RedBook	`GG-RBK-01`	Note	Open the saved note.
RedBook	`GG-RBK-02`	Note	Open the liked note.
RedBook	`GG-RBK-03`	Note	Open the discover-feed note and save it.
Reddit	`GG-RDT-01`	Post	Open the home-feed post.
Reddit	`GG-RDT-02`	Post	Open the post; comment exactly "Thank you".
Reddit	`GG-RDT-03`	Post	Open the home-feed post.
Spotify	`GG-SPT-01`	Song	Add the liked song to a new "Rainy Night" playlist.
Spotify	`GG-SPT-02`	Song	Play the recently played song.
Spotify	`GG-SPT-03`	Playlist	Open the home-screen playlist.
Tencent Meeting	`GG-MTG-01`	Meeting	Open the meeting in history.
Tencent Meeting	`GG-MTG-02`	Meeting	Open the scheduled meeting.
Tencent Meeting	`GG-MTG-03`	Meeting	Open the meeting in history.
WeChat	`GG-WCT-01`	Chat	Open the chat.
WeChat	`GG-WCT-02`	Chat	Open the chat; reply exactly "Understood".
WeChat	`GG-WCT-03`	Author	Open the author's profile from Moments.

Application	Task ID	Target	Task Template
WeChat Reading	GG-WRE-01	Book	Open the book in the home reading list.
WeChat Reading	GG-WRE-02	Book	Add the book using Add Favorite Book.
WeChat Reading	GG-WRE-03	Book	Open the recommended book.
X	GG-X-01	Post	Open the For You post.
X	GG-X-02	Post	Open the For You post.
X	GG-X-03	Post	Open the For You post.

B.3 Reference Subgoals by Task

Table 5 expands the task templates in Table 4 into their ordered reference subgoals. Each subgoal specifies an objective associated with a UI context and may involve multiple actions, including scrolling, typing, and clicking. The sequence is shared across Common/Personal targets and Explicit/Implicit instructions; the designated target is determined by the task instance. These descriptions specify the reference objectives; the RePAIR training reward is defined separately in Appendix C.

Table 5: The 67 reference subgoal positions across all 30 task templates included in our analysis. The optional library-to-history navigation step in GG-SPT-02 is excluded, and the remaining positions are numbered consecutively within each task.

Task ID	Ordered subgoals
GG-ALI-01	(1) Open My from the home screen; (2) Open billing history; (3) Open the designated transaction details.
GG-ALI-02	(1) Open the message list from the home screen; (2) Open the designated conversation.
GG-ALI-03	(1) Open the message list from the home screen; (2) Open the designated conversation; (3) Enter and send “Understood”.
GG-BIL-01	(1) Open Following from the recommendation home screen; (2) Switch to the video tab; (3) Open the designated video; (4) Like the designated video.
GG-BIL-02	(1) Open the designated video from the home recommendation feed.
GG-BIL-03	(1) Open Me from the recommendation home screen; (2) Open the personal space; (3) Open the liked-video list; (4) Open the designated video.
GG-MAP-01	(1) Open search from the map home screen; (2) Open the designated place details from recent searches.
GG-MAP-02	(1) Open search from the map home screen; (2) Select the designated place from recent searches; (3) Open directions to the designated place.
GG-MAP-03	(1) Open search from the map home screen; (2) Open the designated place details from recent searches.
GG-RBK-01	(1) Open Me from the discovery home screen; (2) Open the saved-notes tab; (3) Open the designated note.
GG-RBK-02	(1) Open Me from the discovery home screen; (2) Open the liked-notes tab; (3) Open the designated note.
GG-RBK-03	(1) Open the designated note from the discovery feed; (2) Save the designated note.
GG-RDT-01	(1) Open the designated post from the home feed.
GG-RDT-02	(1) Open the designated post from the home feed; (2) Enter and submit the comment “Thank you”.
GG-RDT-03	(1) Open the designated post from the home feed.
GG-SPT-01	(1) Open the library from the home screen; (2) Open Liked Songs; (3) Open the menu for the designated track; (4) Select Add to playlist; (5) Select the option to create a new playlist; (6) Create the playlist named “Rainy Night” and complete the track addition.
GG-SPT-02	(1) Open the library from the home screen; (2) Expand the recent-play group; (3) Play the designated track within the expanded group.
GG-SPT-03	(1) Open the designated playlist from the home screen.

Continued on next page

Task ID	Ordered subgoals (continued)
GG-MTG-01	(1) Open meeting history from the home screen; (2) Open the designated meeting details.
GG-MTG-02	(1) Open the designated scheduled meeting details from the home screen.
GG-MTG-03	(1) Open meeting history from the home screen; (2) Open the designated meeting details.
GG-WCT-01	(1) Open the designated conversation from the chat list.
GG-WCT-02	(1) Open the designated conversation from the chat list; (2) Enter and send "Understood".
GG-WCT-03	(1) Open Discover from the chat list; (2) Open Moments; (3) Open the designated author's profile.
GG-WRE-01	(1) Open the designated book from the home reading list.
GG-WRE-02	(1) Open Me from the reading home screen; (2) Open the book-list screen; (3) Open the list editor; (4) Open Add Favorite Book; (5) Select the designated book and confirm its addition.
GG-WRE-03	(1) Open the designated book from the home recommendations.
GG-X-01	(1) Open the designated post from the For You feed.
GG-X-02	(1) Open the designated post from the For You feed.
GG-X-03	(1) Open the designated post from the For You feed.

C Training Rewards and Algorithm

Reward calculation. We compute a binary reward for each subgoal from the recorded trajectory. A subgoal first requires evidence that its entry screen was observed before an action; appearing only in the final screen is insufficient. For a nonterminal subgoal, reward one additionally requires either a recorded route corresponding to a later subgoal in the reference path or a click on an expected item or control. For the terminal subgoal, defined as the last subgoal in the reference path, reward one instead requires the task judge to report success. Reward labels and matching criteria are used only for scoring and are not provided to the agent.

For rollout i and subgoal k, let E_{ik} indicate the required entry observation, L_{ik} a matching later-reference route in any observation or the final screen, and C_{ik} an expected click anywhere in the log. Let J_i be the binary task-judge outcome. The reward is

$$r_{ik} = \begin{cases} 0, & E_{ik} = 0, \\ J_i, & E_{ik} = 1 \text{ and } k \text{ is terminal}, \\ 1, & E_{ik} = 1,\ k \text{ nonterminal, and } (L_{ik} \vee C_{ik}), \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

Matching rules. Route matching compares URL paths after removing query parameters; app identity and screen state are not used. Expected items and controls are specified by the task template. A target-item click matches when its recorded IDs include the designated target ID or identity. Other controls match by trigger ID or by a substring match in either direction between whitespace-normalized labels. The scorer searches the entire log without enforcing visit order. Here, "later" refers to the order of subgoals in the reference path, not the order of observations in the rollout.

Policy update. Each iteration collects 124 rollouts with a fixed policy and performs one LoRA update per four-rollout group. Rollout-policy and SFT-reference token log probabilities are cached as specified in the pseudocode below.

Table 6: RePAIR training pseudocode ($K = 8$; $N \in \{5, 10\}$ training iterations). θ denotes trainable LoRA parameters; h_{it} is the recorded token prefix and y_{it} a response token. T_i counts response tokens over the entire trajectory, across all actions.

$\theta \leftarrow \mathrm{SFT}(\theta_{\mathrm{base}}, \mathcal{D}_{\mathrm{Explicit}})$ for one epoch; freeze $\theta_{\mathrm{ref}} \leftarrow \theta$.
Build fixed persona pools and a repeating five-iteration pair schedule.
for training iteration $j = 1, \ldots, N$ **do**
  Fix rollout policy $\theta_{\mathrm{old}} \leftarrow \theta$ for this iteration.
  for each of the 31 task–condition pairs q **do**
    Select the scheduled pair; collect two full rollouts per slot with vLLM.
    Cache the four trajectories and their rollout-policy token log probabilities.
  Cache SFT-reference token log probabilities; reset AdamW and the 31-update cosine schedule.
  for each cached group in fixed task order **do**
    Compute subgoal rewards r_{ik} using Section C.
    $a_{ik} \leftarrow (r_{ik} - \mu_k)/\sqrt{\sigma_k^2 + 10^{-8}}$, using the mean and population variance over $i = 1, \ldots, 4$.
    $A_i \leftarrow |\mathcal{U}_q|^{-1} \sum_{k \in \mathcal{U}_q} a_{ik}$; $\mathcal{U}_q$ is the UC-subgoal set (use zero if empty).
    Recompute current-policy token log probabilities; reuse cached SFT-reference scores.
    $\rho_{it} \leftarrow \pi_\theta(y_{it} \mid h_{it})/\pi_{\mathrm{old}}(y_{it} \mid h_{it})$; $d_{it} \leftarrow \log \pi_{\mathrm{ref}}(y_{it} \mid h_{it}) - \log \pi_\theta(y_{it} \mid h_{it})$.
    $\ell_{it} \leftarrow -\min\{\rho_{it} A_i, \mathrm{clip}(\rho_{it}, 0.8, 1.2) A_i\} + 0.01(e^{d_{it}} - d_{it} - 1)$.
    Take one AdamW step on $\frac{1}{4} \sum_{i=1}^{4} \frac{1}{T_i} \sum_{t=1}^{T_i} \ell_{it}$; clip gradient norm to 1.0 and advance the scheduler.
  Save the iteration checkpoint.
return the iteration-N adapter ($124N$ rollouts; $31N$ optimizer updates).

D EXPERIMENTAL AND TRAINING SETUP

Evaluation. We evaluate Qwen3-VL-8B, Qwen3-VL-30B-A3B, GUI-Owl-1.5-8B-Think, MAI-UI-8B, AutoGLM-Phone-9B-Multilingual and UI-TARS-1.5-7B on the same 1,200 test instances. Decoding uses temperature 0, top-p 1 and seed 0, with each model's action format. The action limit is 20 for 840 instances and 30 for 360. Output limits are 1,024 tokens for Qwen/SFT/RePAIR, 2,048 for MAI-UI and 3,000 for AutoGLM; GUI-Owl/UI-TARS limits are not specified in the saved request configs. Task SR uses all instances; all-success uses 120 ten-persona groups; SAR uses reached subgoals at the 67 reference positions.

SFT. Qwen3-VL-8B-Instruct is fine-tuned on successful Explicit-instruction oracle demonstrations across all 30 task templates. Of the 480 Explicit task instances from the eight training personas, three failed oracle executions are excluded, leaving 477 demonstrations containing 1,876 actions. Validation uses 119 successful demonstrations containing 469 actions, after excluding one failed execution from 120 instances. All four excluded executions are from X and failed to reach the designated target route.

Implicit instructions, designated targets, and initial states are available, but separate Implicit oracle demonstrations were not collected. SFT therefore uses only Explicit demonstrations. LoRA (Hu et al., 2021) updates the language model's q/k/v/o attention projections; the vision encoder is frozen. One epoch gives 235 updates using assistant-response loss. Training uses bf16, SDPA and gradient checkpointing on an RTX 6000 Ada (PyTorch 2.13.0+cu130, Transformers 4.57.6, PEFT 0.18.1).

Table 7: Core optimization settings. RePAIR inherits the SFT LoRA adapter.

Setting	SFT	RePAIR
LoRA rank / alpha	16 / 32	16 / 32
Dropout	0.05	Disabled during updates
Optimizer	AdamW (Loshchilov & Hutter, 2019)	AdamW
Learning rate	10^{-4}	10^{-5}
Weight decay / norm clipping	0 / 1.0	0 / 1.0
Scheduler / warmup	Cosine / 3%	Iteration-wise cosine / 3%
Update unit	8 action samples	4 full-task rollouts
Policy ratio clip	—	0.2
Reference KL coefficient	—	0.01 (fixed SFT)

RePAIR task selection. A task specification is a combination of task template, target type (Common or Personal), and instruction type (Explicit or Implicit). The 30 templates yield 120 specifications, each evaluated across ten personas, p0 to p9. For each model, a specification is classified as mixed when its final-goal outcomes include both successes and failures across these personas.

We select specifications classified as mixed by all five baselines in the frozen September 16, 2026 analysis: Qwen3-VL-8B, Qwen3-VL-30B-A3B, GUI-Owl, MAI-UI, and AutoGLM. This yields 31 specifications spanning 18 templates, comprising 12 Common and 19 Personal specifications, or 8 Explicit and 23 Implicit specifications.

Selection uses the historical AutoGLM results preceding its greedy rerun and excludes UI-TARS. The selected list is held fixed across subsequent training experiments rather than recomputed from updated baseline results.

RePAIR training. The 31 selected specifications are instantiated for the eight training personas p10 to p17, yielding 248 distinct task instances. Each training iteration collects four full-task rollouts per specification, totaling 124 rollouts and 31 optimizer updates. Both Explicit and Implicit tasks are included: the policy generates trajectories that are scored using recorded routes, actions, and the task judge, without requiring oracle demonstrations.

Seeded selection forms fixed pools of $K \in \{1, 2, 5, 8\}$ personas. K1 uses one persona, K2 a fixed pair, and K5/K8 cycle pairs; four Reddit task–condition pairs allow same-persona sampling to keep subgoal labels consistent. Group compatibility requires matching ordered subgoal IDs and UA/UC

labels across the full reference path. For each scheduled pair, we collect two rollouts per persona slot, resetting the corresponding task instance to its stored initial state before every rollout. Only UC-subgoal rewards are used to compute the trajectory advantage. Rollouts use temperature 0.8, top-p 1, top-k -1, min-p 0 and a 1,024-token output limit. Five/ten iterations give 620/1,240 rollouts and 155/310 updates. Iterations 6 to 10 repeat the persona schedule with new rollouts, continuing from iteration 5. The main result uses K8 at iteration 5; validation is not used to select it. SFT uses seed 0; RePAIR uses seed $j - 1$ at iteration j with derived update seeds.

Selection and review scope. Training rollouts use personas p10 to p17, disjoint from the evaluation personas p0 to p9. However, task specifications were selected using earlier baseline outcomes on p0 to p9. Thus, evaluation personas are held out from policy optimization, but their baseline results informed training-task selection. We interpret the RePAIR study as an exploratory intervention informed by the preceding analysis.

The p10 to p19 inputs have not completed semantic and browser review. Successful oracle execution and subsequent model rollouts do not establish completion of that review.

vLLM. We use vLLM (Kwon et al., 2023) for inference, with server seed 0 and GPU memory utilization 0.9. Qwen baselines use an 8,192-token batch budget and image bounds of 3,136 to 12,845,056 pixels; AutoGLM uses at most 5,000,000 pixels. K8 rollout logs for iterations 1 to 5 record vLLM 0.19.0, bf16, prefix caching and chunked prefill. These settings are not verified for all earlier FingerTip or recovery runs.

Table 8: vLLM settings from evaluation configs and training logs. TP: tensor parallelism; Context: token limit; Sequences/Images: per-server and per-request limits, respectively.

Model / use	TP	Context	Sequences	Images
Qwen3-VL-8B baseline	1	16,384	8	1
Qwen3-VL-30B-A3B	2	16,384	8	1
GUI-Owl	1	32,768	8	5
MAI-UI	1	32,768	8	4
UI-TARS	1	32,768	8	6
AutoGLM (PAIR / FingerTip)	1	25,480	16 / 32	10
SFT/RePAIR evaluation	1	32,768	8	2
RePAIR rollout collection	1	32,768	8	2

E Subgoal Failure Patterns Across Task Trajectories

To examine where task progress breaks down, we report subgoal-level failure rates along the reference trajectory for all 30 task templates. For each task, subgoals are ordered according to their position in the reference execution path. We compute the failure rate at each subgoal as $100 - \mathrm{SAR}$ after the agent reaches the corresponding subgoal context. Thus, failure rates at later positions are conditional on the agent having progressed far enough to reach those contexts.

Figure 11 shows that failures are not distributed uniformly across task trajectories. Instead, many tasks exhibit localized failure peaks at particular subgoals. Several pronounced failures occur in user-conditioned contexts, including target-related stages for Bilibili, Maps, Tencent Meeting, WeChat, and WeChat Reading. For example, the failure rate reaches 100% at the third subgoal of the Bilibili *subscribed anime* task, 90% at the first subgoal of *scheduled meeting*, and 94% at the first subgoal of *liked book*. These patterns indicate that particular stages involving user-specific application content can form substantial bottlenecks within otherwise multi-step tasks.

At the same time, high failure rates are not exclusive to user-conditioned contexts. Some user-agnostic stages also show large failure rates, such as the second subgoal of the Bilibili *liked video* and *subscribed anime* tasks. Thus, the trajectory-level analysis does not imply that all failures originate from personalization. Rather, it distinguishes general GUI interaction difficulties from localized failures that arise at user-conditioned stages.

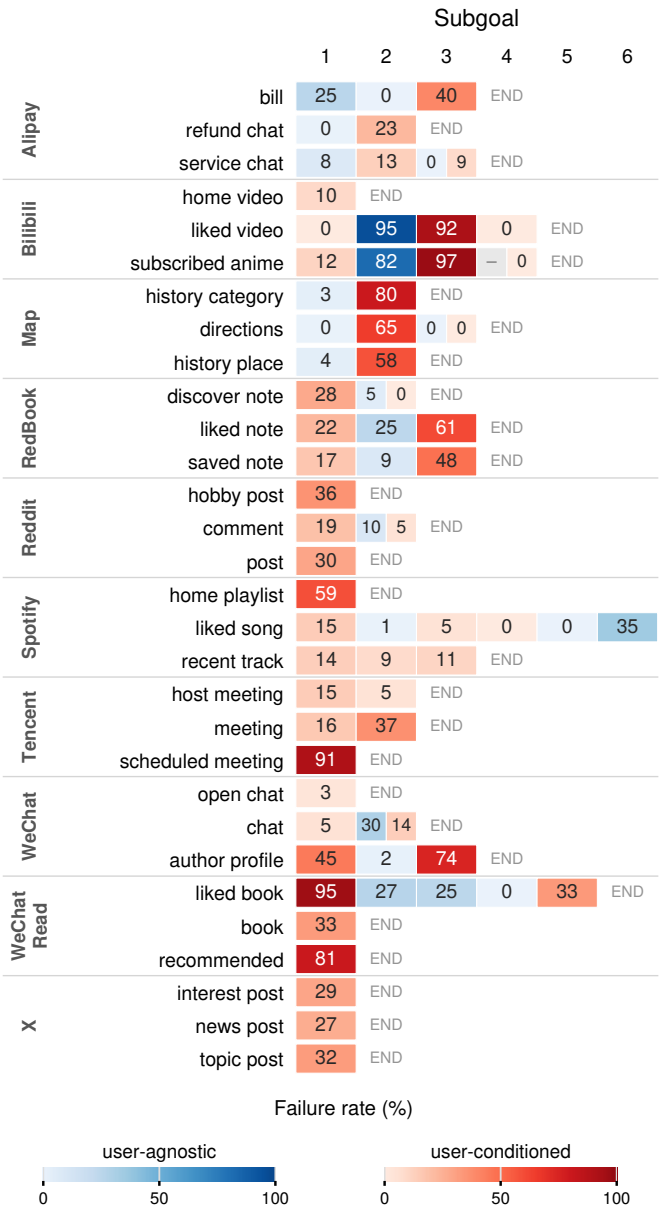


Figure 11: Subgoal failure rates across task trajectories. Each cell reports the failure rate ($100 - \mathrm{SAR}$, %) after entering the corresponding subgoal context, pooled across six complete agents. Blue and red denote user-agnostic and user-conditioned contexts, respectively; darker cells indicate higher failure rates. A dash denotes no entry, and END marks task termination. Later-position rates are conditional on reaching the corresponding context.

Overall, this analysis complements the aggregate SAR comparison by showing where the observed performance differences emerge within individual task trajectories. The results suggest that difficulty is often concentrated at specific interaction stages rather than being uniformly distributed throughout an entire task.

F COMPLETE PAIR FAILURE TRAJECTORIES

The two failures from the main paper use Personal targets and Explicit instructions. Each row shows all three recorded steps, including termination. Screenshots precede the labeled actions; no post-termination screenshot is available.

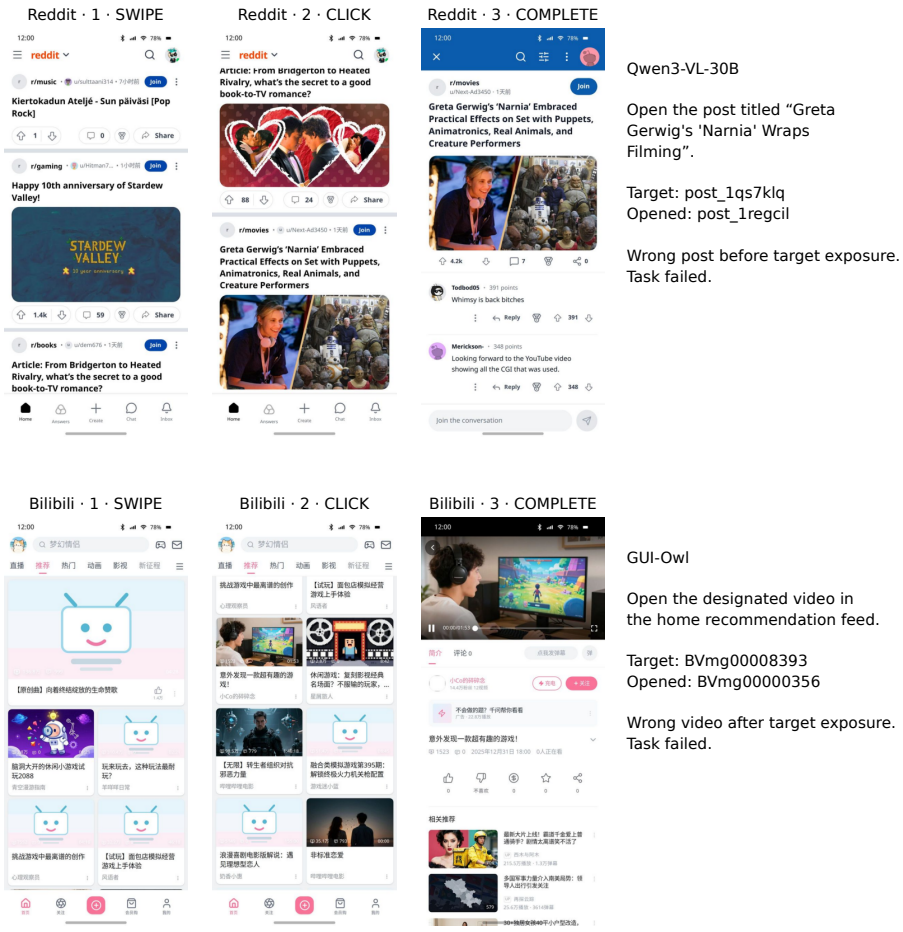


Figure 12: Reddit (top, Qwen3-VL-30B) selects the wrong post before target exposure. Bilibili (bottom, GUI-Owl) selects the wrong video after target exposure. Both fail the task’s success check; target and selected IDs are shown beside the trajectories.

The selected item is confirmed by the next screen’s route. These examples illustrate the two error types, not their overall frequency.