Reveal-and-Click: A Benchmark for VLM Agents under Hidden GUI Targets

Kyeongseon Kim [*1] **Jiyeon Son** [*1] **Tae-Hyun Oh** [1]
KAIST
{ellakim,daiisy7,taehyun.oh}@kaist.ac.kr

Abstract

GUI grounding benchmarks evaluate whether a VLM agent can localize a target element from a single static screenshot, under the assumption that the target is already visible. Real-world interfaces, however, often involve visibility constraints such as off-screen targets, hover-resolved visual ambiguity, occlusions, and delayed activation, so a correct click first requires an interaction that reveals the target. In these settings, grounding requires not only visual matching but also interaction to recover target visibility. We introduce ***Reveal-and-Click***, a benchmark for grounding under limited observability. In Reveal-and-Click, agents must first expose a hidden or non-actionable target, then click it. Reveal-and-Click defines a minimal action space and seven visibility-constraint subtypes spanning *viewport constraint*, *multi-state constraint*, and *temporal and manipulation-based* settings. Across 16 controllable mock GUI environments and 601 tasks, humans achieve 94.6% grounding accuracy, while strong open-source VLM agents achieve only around 30%, revealing a large gap in visibility recovery rather than ordinary target localization.

1. Introduction

Recent work on GUI grounding (Cheng et al., 2024; Wu et al., 2025; Mittal et al., 2026) has focused on mapping natural-language instructions to screen coordinates from a given screenshot. Existing benchmarks typically assume that the target element is already visible in the current frame (Cheng et al., 2024; Li et al., 2025), reducing the task to static visual matching. However, this assumption does not hold in many real interfaces. A target may be off-screen (Burigat et al., 2012), hidden behind a folded or inactive UI state, blocked by an occluding element, or become available only after a delay. In such cases, the agent cannot localize the target solely from the initial observation. Instead, it first needs to recover target visibility through interaction. This setting matters when structured UI metadata, such as the DOM, accessibility tree, or internal application state, is unavailable or restricted. In such cases, agents must rely only on screenshots (Lekies et al., 2013), as in browser-only, remote-desktop, or security-constrained deployments. Together, these conditions make screenshot-only interactive grounding a practically relevant evaluation setting.

We formulate this problem as ***Reveal-and-Click***. The objective remains grounding: the agent identifies and clicks the target element. However, unlike standard static grounding, the agent must first interact with the interface to expose the target before attempting the final localization click. To make this setting explicit, we define a minimal *action space* for visibility recovery, allowing the agent to reveal hidden or non-actionable targets before the final grounding click. The key question is not whether an agent can complete a workflow, but whether it can recover the visibility of a target element and localize it correctly once the target becomes observable. Building on this formulation, we introduce Reveal-and-Click, a benchmark that combines *Key Capabilities* with a minimal *Action Space* to derive *Visibility Constraint Types*, which cover *Viewport Constraint*, *Multi-State Constraint*, and *Temporal/Manipulation Constraint*. Reveal-and-Click tests a bounded reveal-and-click problem. The agent must choose the interaction that makes the target visible, then click the target, without solving a long-horizon workflow.

Our contributions are as follows:

- We formulate *Reveal-and-Click* as visibility-constrained grounding, where agents must reveal hidden or non-actionable targets before the final grounding click.
- We introduce a controlled benchmark with realistic GUI interaction patterns across diverse, reproducible mock environments
- We show that missing action priors are a key bottleneck in current VLM agents and can be partially learned through diagnostic training.

[*]Equal contribution [1]School of Computing, KAIST, Daejeon, Republic of Korea. Correspondence to: Kyeongseon Kim <ellakim@kaist.ac.kr>, Jiyeon Son <daiisy7@kaist.ac.kr>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

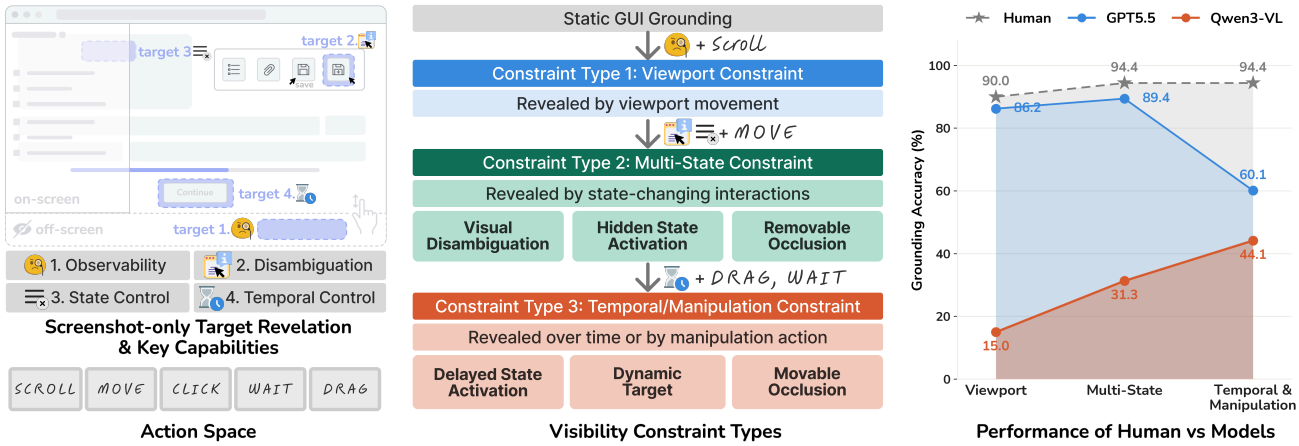


Figure 1. Overview of Reveal-and-Click. Reveal-and-Click evaluates grounding under visibility constraints, where the target is initially hidden or non-actionable and must be revealed before clicking. **Left**: the faded screenshot shows representative target-revelation cases in screenshot-only GUI grounding, including off-screen targets, hover-resolved ambiguity, state-dependent visibility, and temporal activation. These cases motivate four *Key Capabilities*: observability, disambiguation, state control, and temporal control, supported by a minimal action space. **Center**: Reveal-and-Click combines the *Key Capabilities* and *Action Space* to derive *Visibility Constraint Types*. **Right**: human and model performance across constraint types shows that current VLM agents remain far below humans, especially under temporal and manipulation constraints.

2. Related Work

GUI grounding ScreenSpot (Cheng et al., 2024), ScreenSpot-Pro (Li et al., 2025), and OSWorld-G (Xie et al., 2025) formulate GUI grounding as a coordinate-prediction task from a single screenshot in which the target is already visible. Universal visual grounding extends this to agent settings (Gou et al., 2025), but still maps an instruction to a target within a single observed state. See, Point, Refine (Mittal et al., 2026) adds multi-turn visual feedback for pixel-precise localization, but still refines an already-visible target. Reveal-and-Click instead drops the visibility assumption: the target is initially hidden or non-actionable, so the agent must first choose an interaction that changes what can be seen before localization is possible. This differentiates Reveal-and-Click from static grounding benchmarks, where the main failure mode is often coordinate localization among visible elements.

GUI navigation Mind2Web (Deng et al., 2023), WebArena (Zhou et al., 2024), OSWorld (Xie et al., 2024), and VisualWebArena (Koh et al., 2024) evaluate multi-step web or desktop task completion, where interaction serves a high-level user goal. These benchmarks judge end-to-end success, mixing planning, navigation, and grounding in one signal. Reveal-and-Click targets a smaller unit: choose the interaction that reveals the target, then click it. This makes Reveal-and-Click complementary to end-to-end navigation benchmarks rather than a replacement for them.

3. Method

3.1. Problem Formulation

We define Interactive GUI Grounding as a setting where the target element is not directly visible or actionable in the initial observation and must be exposed before it can be localized. At each time step t, the agent observes the current screenshot s_t, the natural-language instruction q, and the action history $h_{t-1} = (a_1, \ldots, a_{t-1})$. The instruction describes what to find, not how to reveal it: procedure hints such as "scroll down," "hover to reveal," or "wait for the button" are deliberately withheld.

The agent selects an action a_t from a minimal action space

$$\mathcal{A} = \{\textsc{Scroll}, \textsc{Move}, \textsc{Drag}, \textsc{Wait}, \textsc{Click}\},$$

and the environment executes the action and returns the next screenshot s_{t+1}. An episode terminates when the agent clicks the target element or exhausts the action budget. Non-target clicks do not count as success and are treated as failed grounding attempts, while the resulting interface state is returned to the agent (Table 1). This protocol allows us to distinguish failures in visibility recovery from failures in final target localization.

Table 1. Minimal action space $\mathcal{A}$ for visibility recovery.

Action	Role
$\texttt{SCROLL}(\text{dir}, \Delta_{\text{px}})$	Scroll to reveal off-screen regions.
$\texttt{MOVE}(x_{\text{px}}, y_{\text{px}})$	Move the pointer to trigger hover states or expose disambiguating cues.
$\texttt{DRAG}(x_s, y_s, x_e, y_e)$	Drag a movable UI element from start to end coordinates to move or adjust layout.
$\texttt{WAIT}(\Delta t_{\text{ms}})$	Pause for a specified duration to handle delayed or asynchronous UI updates.
$\texttt{CLICK}(x_{\text{px}}, y_{\text{px}})$	Click the specified location; grounding is evaluated at this step.

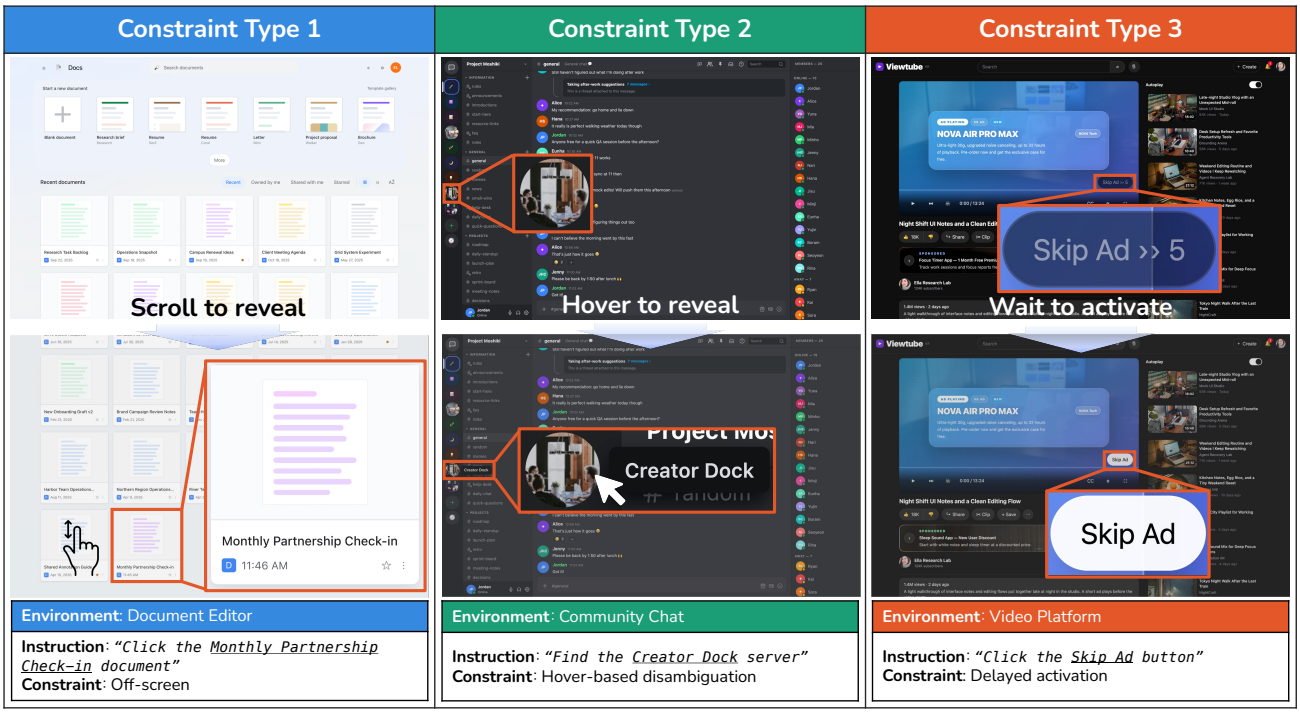


Figure 2. Examples of the three Reveal-and-Click constraint types. Constraint Type 1 is a Viewport Constraint that requires scroll-based exploration, Constraint Type 2 is a Multi-State Constraint that requires interaction-triggered state changes, and Constraint Type 3 is a Temporal & Manipulation Constraint. More detailed and diverse dataset examples are provided in Appendix B.

3.2. GUI Constraint Taxonomy

We categorize Reveal-and-Click tasks by the interaction needed to recover target visibility, rather than by difficulty alone (Figure 1). Figure 2 shows examples.

Constraint Type 1: Viewport Constraint. The target is hidden by the current viewport but remains reachable within the same UI state.

- **CT1-1. Scroll-based Exploration:** The target is off-screen and must be revealed by scrolling within the current state.

Constraint Type 2: Multi-State Constraint. The target, or the visual cue needed to identify it, is hidden by the current UI state and must be revealed through a state-changing interaction.

- **CT2-1. Visual Disambiguation:** The target is visually similar to other elements and can only be distinguished through hover-based cues such as tooltips or labels, requiring a MOVE before the final CLICK.
- **CT2-2. Hidden State Activation:** The target is contained in a collapsed or inactive UI region, such as a dropdown, tab, or side panel, that must be opened before clicking.
- **CT2-3. Removable Occlusion:** A blocking element, such as a popup or modal, hides the target and must be dismissed before the target can be clicked.

Constraint Type 3: Temporal & Manipulation Constraint. Reaching the target requires temporal control or spatial manipulation.

- **CT3-1. Delayed State Activation:** The target is initially inactive and becomes interactable only after a temporal delay or asynchronous event, requiring a WAIT before clicking.
- **CT3-2. Dynamic Target Interaction:** The target resides in an automatically transitioning element, such as a carousel or ticker, and must be reached by waiting for the correct state or by using navigation controls.
- **CT3-3. Movable Occlusion:** A UI element occludes the target due to its position or size, requiring a DRAG to reposition or resize the occluding element.

3.3. Dataset and Environments

Reveal-and-Click provides 16 controllable Next.js mockups spanning news, e-commerce, video, messaging, documents, design, maps, music, and developer tools. They are visually realistic while remaining deterministic for dataset construction and evaluation. The dataset contains 601 tasks across three constraint types and 16 mockups (Figure 3). It covers all seven sub-types, separating ordinary off-screen recovery from state-dependent, temporal, and manipulation-based visibility recovery.

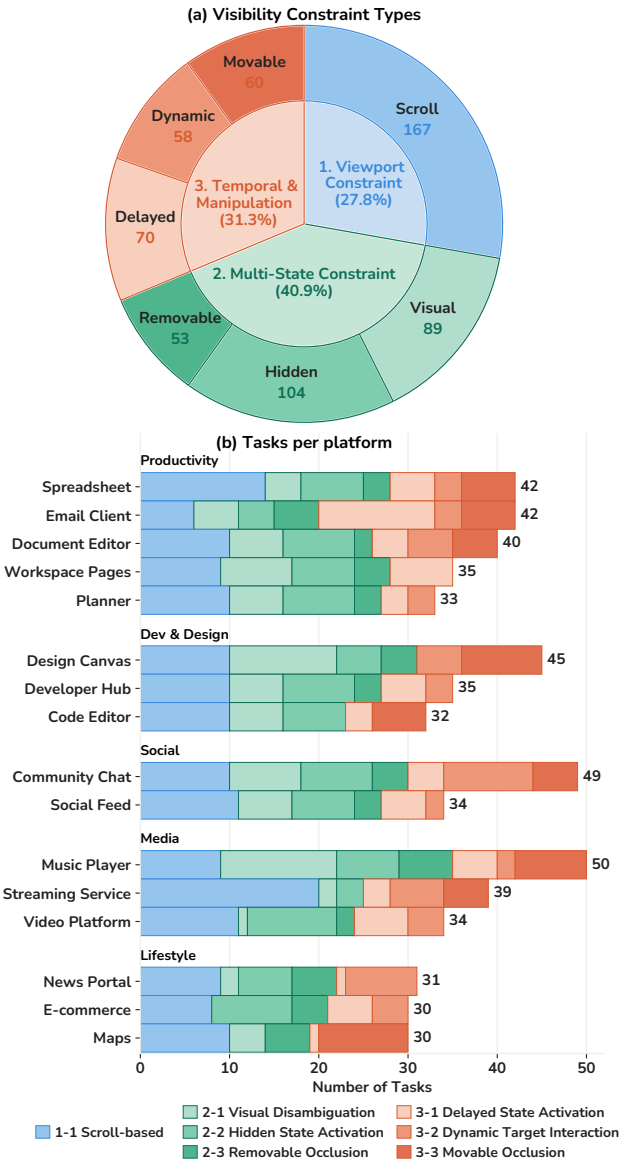


Figure 3. Distribution of Reveal-and-Click tasks. (a) Task distribution across the three constraint types and their sub-types. (b) Number of tasks per platform, broken down by sub-type.

3.4. Data Collection

We construct Reveal-and-Click in four stages: building deterministic mockups with stable target IDs, preparing task-specific initial states, human-correcting MLLM-drafted instructions and reference traces, and replaying all tasks to remove ambiguity, instability, and accidental initial target visibility.

4. Evaluation and Experiments

4.1. Evaluation Setup

Reveal-and-Click consists of task specifications, 16 controllable Next.js mockups, and a Playwright-driven Chromium interaction engine at a 1440×900 viewport. On each `CLICK`, the evaluator compares the clicked element ID with the target ID; a match ends the episode, and a mismatch returns the updated screenshot. Models receive only screenshots, the goal-only instruction, and action history. We do not expose the DOM, accessibility tree, target ID, or reference trace. Main runs use an eight-action budget. Open-source and trained Qwen3-VL variants use greedy decoding unless noted; open-source models are served locally with Transformers adapters on NVIDIA A6000/RTX PRO 6000-class GPUs, and commercial models use API backends.

Metrics. We report the following metrics:

- **Grounding Accuracy (GA) ↑ :** the fraction of tasks in which the agent `clicks` the target element within the action budget.
- **Action Cost (AC) ↓ :** mean number of actions executed until success or budget exhaustion.
- **Optimal Ratio (OR) ↑ :** mean of $|g_i|/|a_i|$ over successful tasks, where g_i is the annotated reference trace and a_i is the agent trace; values above 1 indicate shorter-than-reference successful trajectories.
- **Total Time (TT) ↓ :** mean elapsed wall-clock time per task.

GA measures success; AC, OR, and TT measure interaction efficiency.

4.2. Main Benchmark Results

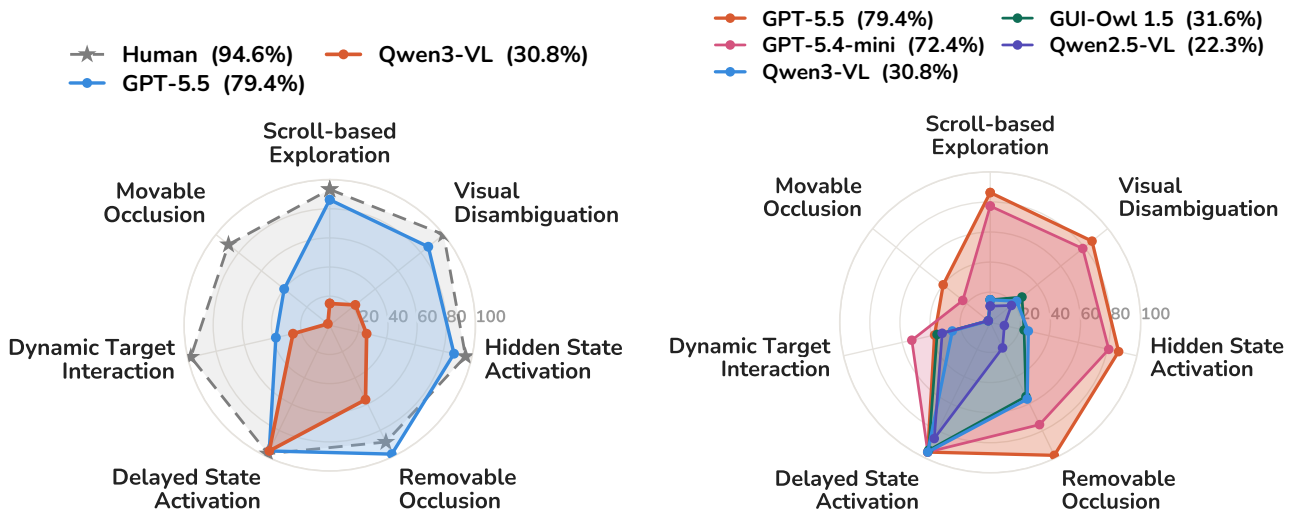
We evaluate three open-source VLM agents, Qwen2.5-VL-7B (Bai et al., 2025b), Qwen3-VL-8B (Bai et al., 2025a), and GUI-Owl-1.5-8B (Xu et al., 2026), plus the commercial reference model GPT-5.5. All models run in screenshot-only mode without DOM or accessibility tree access, with a budget of eight actions per task.

Table 2 reports grounding accuracy (GA) and efficiency by constraint type and sub-type, while Table 3 aggregates GA by application. Figure 4 compares humans, the strongest commercial model, and open-source baselines; Appendix A2 adds per-application AC, OR, and TT.

Two patterns stand out. First, even simple viewport recovery is inefficient: models often use small scrolls or wait actions instead of bringing the target into view. Second, Constraint Type 3 separates easy waiting from harder dynamics. Delayed State Activation reaches 95.7% GA for Qwen3-VL, but Dynamic Target Interaction and Movable Occlusion remain difficult; all open-source models reach only 1.7% GA on Movable Occlusion. Figure 5 shows the complementary cost trade-off: GPT-5.5 is much more accurate, but slower and paid. Our full 601-task GPT-5.5 run cost about $74.9; a validation-sized run scales to roughly $15.

4.3. Visibility-recovery ablation via oracle reveal.

To separate visibility recovery from final-action grounding, we run an oracle-reveal setting. We execute the reference trace up to the step immediately before the final click, then



(a) Human ceiling, strongest commercial, and open-source MLLM.

(b) All evaluated model baselines.

Figure 4. **Reveal-and-Click exposes a large interaction gap.** Humans solve the same visual-observation and action-space setting at 94.6% GA, while GPT-5.5 reaches 79.4% and the strongest open-source model reaches 31.6%. The full model comparison shows that this gap is consistent across most visibility constraint sub-types, rather than being driven by a single application or task family.

provide the resulting screenshot, original instruction, and action history to the model. The model predicts one final action (Pass@1, no retries).

Table 4 shows large gains once the target has already been revealed. This means many errors occur before the final click, when the model must choose and execute the reveal action. The remaining oracle-reveal errors show that revealed-state grounding is still nontrivial.

Failed Qwen3-VL traces show the same pattern. When the reference setup requires DRAG, SCROLL, or MOVE, failed trajectories omit that action in 59/59, 207/247, and 71/96 cases, respectively. We use SFT to add these reveal actions to the policy's support, then use GRPO to improve execution and reduce premature or ineffective actions.

4.4. Diagnostic Training Ablations

We use Qwen3-VL-8B to test whether these missing reveal-action priors can be learned. The gap is clearest for DRAG: on the 120-task validation split, Qwen2.5-VL, Qwen3-VL, and GUI-Owl 1.5 emit zero drag actions.

Diagnostic setup. We derive a training/validation split from tasks with human-verified reference traces: 120 held-out validation tasks. MLLMs draft the traces; humans correct and verify them by re-execution. The validation set is used only for model selection and ablation.

Injecting missing action priors (Hu et al., 2022). We first perform standard behavior cloning on the reference traces. Let x_t denote the instruction, screenshot, and interaction history at step t, and let a_t be the next reference action. We optimize a weighted next-action negative log-likelihood (Pomerleau, 1991):

$$\mathcal{L}_{\mathrm{BC}}(\theta) = -\mathrm{E}_{(x_t,a_t)\sim\mathcal{D}}\left[w(a_t)\log \pi_\theta(a_t \mid x_t)\right].$$

For standard behavior cloning, $w(a_t) = 1$. For frequency-balanced behavior cloning (Cui et al., 2019), we increase $w(a_t)$ for rare reveal actions such as MOVE, WAIT, and DRAG. The model still receives goal-only instructions. Only the training distribution changes.

Selecting among reveal actions (Shao et al., 2024). We then apply online GRPO under the same eight-action interaction budget as main evaluation. The reward is defined over browser-executed trajectories rather than text strings:

$$R(\tau) = R_{\mathrm{out}}(\tau) + R_{\mathrm{proc}}(\tau) - \lambda_{\mathrm{step}}|\tau| - R_{\mathrm{err}}(\tau).$$

R_{out} rewards click-time target success, R_{proc} gives bounded credit for visibility-changing progress, and R_{err} penalizes wrong clicks, empty clicks, invalid actions, and repeated ineffective actions.

The two stages serve different roles. Behavior cloning broadens action support. GRPO then selects among those actions using online feedback. If a seed policy never samples DRAG, GRPO sees no successful drag rollout to reinforce. Rare-action SFT gives GRPO something to improve.

Findings. Figure 6 shows that the two-stage recipe closes part of the gap: Qwen3-VL improves from 40.0% to 59.2% GA on the 120-task validation split. The gains are concentrated in task families that require non-click reveal actions. On Movable Occlusion, base Qwen3-VL never emits the required drag action; frequency-balanced behavior cloning

Table 2. Per-constraint-type and per-sub-type results on Reveal-and-Click.

Model	GA (%) ↑	AC ↓	OR ↑	TT (ms) ↓
Constraint Type 1: Viewport Constraint				
Qwen2.5-VL-7B	10.8	7.37	0.70	**14718**
Qwen3-VL-8B	15.0	7.26	**0.73**	15246
GUI-Owl-1.5-8B	15.0	7.27	0.62	21985
GPT-5.5	**86.2**	**4.29**	0.65	56355
Constraint Type 2: Multi-State Constraint				
Qwen2.5-VL-7B	14.6	7.20	1.16	**8773**
Qwen3-VL-8B	31.3	6.21	1.13	10862
GUI-Owl-1.5-8B	31.3	6.14	**1.16**	14682
GPT-5.5	**89.4**	**3.28**	1.08	49103
Constraint Type 2-1: Visual Disambiguation				
Qwen2.5-VL-7B	18.0	6.96	1.52	**8048**
Qwen3-VL-8B	22.5	6.51	**1.73**	11287
GUI-Owl-1.5-8B	27.0	6.22	1.71	13350
GPT-5.5	**86.5**	**2.91**	1.40	44982
Constraint Type 2-2: Hidden State Activation				
Qwen2.5-VL-7B	9.6	7.44	0.93	**9181**
Qwen3-VL-8B	26.0	6.54	**0.97**	11251
GUI-Owl-1.5-8B	23.1	6.67	0.87	17408
GPT-5.5	**87.5**	**3.73**	0.91	56739
Constraint Type 2-3: Removable Occlusion				
Qwen2.5-VL-7B	18.9	7.13	0.81	**9193**
Qwen3-VL-8B	56.6	5.08	0.89	9384
GUI-Owl-1.5-8B	54.7	4.96	**0.95**	11569
GPT-5.5	**98.1**	**3.02**	0.89	41038
Constraint Type 3: Temporal & Manipulation Constraint				
Qwen2.5-VL-7B	42.6	6.02	0.52	24467
Qwen3-VL-8B	44.1	6.08	0.63	**10907**
GUI-Owl-1.5-8B	46.8	5.78	0.73	11864
GPT-5.5	**60.1**	**4.82**	**0.97**	134642
Constraint Type 3-1: Delayed State Activation				
Qwen2.5-VL-7B	85.7	4.57	0.50	19820
Qwen3-VL-8B	**95.7**	3.74	0.63	**6626**
GUI-Owl-1.5-8B	94.3	3.53	0.67	7412
GPT-5.5	**95.7**	**2.50**	**0.97**	20603
Constraint Type 3-2: Dynamic Target Interaction				
Qwen2.5-VL-7B	32.8	**5.81**	0.58	45702
Qwen3-VL-8B	25.9	7.02	0.60	**12631**
GUI-Owl-1.5-8B	36.2	6.33	0.83	13558
GPT-5.5	**37.9**	6.47	**0.84**	134949
Constraint Type 3-3: Movable Occlusion				
Qwen2.5-VL-7B	1.7	7.90	1.00	**9362**
Qwen3-VL-8B	1.7	7.90	1.00	14235
GUI-Owl-1.5-8B	1.7	7.88	**2.00**	15421
GPT-5.5	**40.0**	**5.93**	1.10	267392
Overall				
Qwen2.5-VL-7B	22.3	6.88	0.72	15334
Qwen3-VL-8B	30.8	6.46	0.85	**12094**
GUI-Owl-1.5-8B	31.6	6.34	0.89	15830
GPT-5.5	**79.4**	**4.04**	**0.92**	77876

emits it in 7/8 required-drag tasks and solves 4/8, while the GRPO-refined model solves 5/8. Initialization also matters: on Dynamic Target Interaction, GRPO from the frequency-balanced seed reaches 60.0% GA, compared with 30.0% from the standard behavior-cloning seed. The best GRPO policy produces no invalid actions or empty clicks across 578 executed validation actions. These results make Reveal-and-Click useful both as an evaluation benchmark and as a training diagnostic for missing action priors.

Table 3. Per-application Grounding Accuracy (GA) (%) on Reveal-and-Click.

App	Qwen2.5-VL	Qwen3-VL	GUI-Owl-1.5	GPT-5.5
Social Feed	23.5	29.4	38.2	**61.8**
Planner	15.2	21.2	24.2	**81.8**
Developer Hub	11.4	25.7	25.7	**57.1**
Code Editor	25.0	21.9	28.1	**87.5**
Community Chat	22.4	26.5	34.7	**61.2**
Document Editor	32.5	37.5	42.5	**85.0**
Design Canvas	8.9	8.9	13.3	**82.2**
Spreadsheet	16.7	33.3	23.8	**83.3**
Email Client	47.6	50.0	54.8	**88.1**
News Portal	9.7	29.0	29.0	**71.0**
Workspace Pages	25.7	34.3	22.9	**85.7**
Streaming Service	7.7	20.5	17.9	**87.2**
E-commerce	16.7	43.3	33.3	**86.7**
Music Player	20.0	36.0	32.0	**86.0**
Maps	16.7	26.7	26.7	**70.0**
Video Platform	55.9	50.0	58.8	**94.1**
Overall (601)	22.3	30.8	31.6	**79.4**

Table 4. Oracle reveal evaluation on the Reveal-and-Click benchmark. The setting evaluates one final action predicted from the revealed state.

Model	Original Reveal-and-Click GA	Oracle Reveal GA	Gain
Qwen3-VL	30.8 (185/601)	81.9 (492/601)	+51.1
GPT-5.4 Mini (Low)	72.4 (435/601)	90.4 (543/601)	+18.0

Table 5. Human task-solvability results on Reveal-and-Click.

Sub-type	Trials	Success Rate (%) ↑	TT (ms) ↓
1-1	45	93.3	23317
2-1	45	100.0	15851
2-2	45	95.6	19962
2-3	45	88.9	20968
3-1	45	97.8	14239
3-2	45	97.8	22907
3-3	45	88.9	21124
Overall	315	94.6	19767

Table 6. Mock-fidelity study results. Scores are averaged over 15 participants on a five-point Likert scale, where higher scores indicate better perceived fidelity.

Application	VR ↑	IR ↑	TE ↑
Music Player	4.93	4.80	5.00
Video Platform	4.73	4.87	4.80
E-commerce	4.67	4.80	4.80
Developer Hub	4.67	4.53	4.80
Code Editor	4.67	4.80	4.87
News Portal	3.93	4.60	4.60
Maps	3.93	4.07	4.40
Average	4.50	4.64	4.75

4.5. Benchmark Validity Checks

We conduct two human studies to validate Reveal-and-Click along two complementary axes: task solvability and envi-

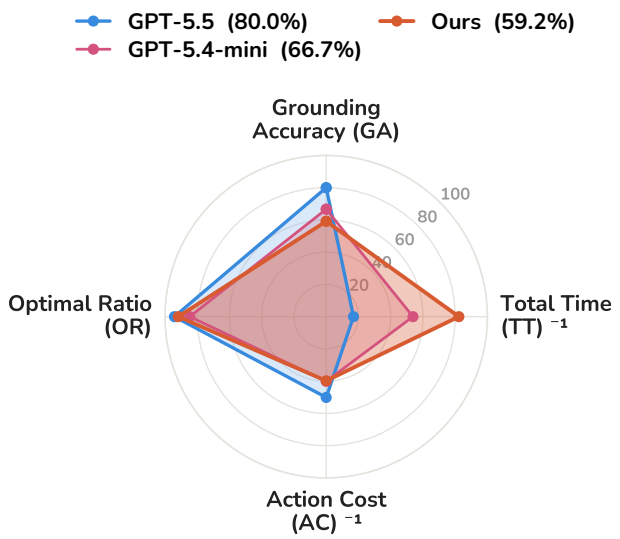


Figure 5. **Accuracy-efficiency trade-off on the 120-task validation split.** GPT-5.5 is substantially more accurate, but its average task time is much higher (66.5s per task) than our trained Qwen3-VL variant (14.2s per task), and it also requires paid API inference. We therefore treat GPT-5.5 as a strong reference model rather than the main deployable training target.

ronment fidelity. The first study examines whether humans can solve the benchmark tasks under the same interaction constraints as model agents. The second study evaluates whether the mock environments preserve realistic visual appearance, task executability, and interaction behavior compared to real applications.

Task solvability. We first test whether humans can solve Reveal-and-Click tasks using only the information provided to agents. We sample three tasks from each of the seven subtypes, resulting in 21 tasks in total, and ask 15 participants to solve them in the mock environments. Participants are given the natural-language instruction, the visual interface, and the available action space, but are not shown the constraint type, sub-type, target element ID, reference trajectory, target coordinates, or required action sequence. Participants interact with each task under the same eight-action budget used for model agents.

As shown in Table 5, humans achieve an overall success rate of 94.6% (298/315 trials), with an average completion time of 19.8 seconds. The model gap is therefore not explained by impossible tasks or broadly ambiguous instructions alone.

Fidelity test. The same 15 participants also complete a fidelity test to assess whether the mock environments provide realist ic and executable GUI settings. We evaluate seven representative applications rather than all 16 to keep the study length manageable while covering the benchmark's main UI families: media, commerce, development, productivity, news, music, and maps. Each participant evaluates these applications using a five-point Likert scale. For each evaluated application, participants compare the mock interface with the corresponding real-application panel and answer three questions.

The first criterion is *Visual Realism (VR)*: participants judge how realistic the mock-up looks compared to the real application, where 1 indicates *very unrealistic*, 3 indicates *somewhat realistic*, and 5 indicates *very realistic*. The second criterion is *Task Executability (TE)*: participants judge whether the given instruction appears executable in the mock site, where 1 indicates *not executable*, 3 indicates *somewhat executable*, and 5 indicates *clearly executable*. The third criterion is *Interaction Realism (IR)*: participants try the required interaction in the mock interface and rate how natural the interaction feels, where 1 indicates *very unnatural*, 3 indicates *somewhat natural*, and 5 indicates *very natural*.

As shown in Table 6, participants rate the mock environments highly across all three criteria, with an overall average of 4.63 out of 5. Task Executability receives the highest average score (4.75), followed by Interaction Realism (4.64) and Visual Realism (4.50).

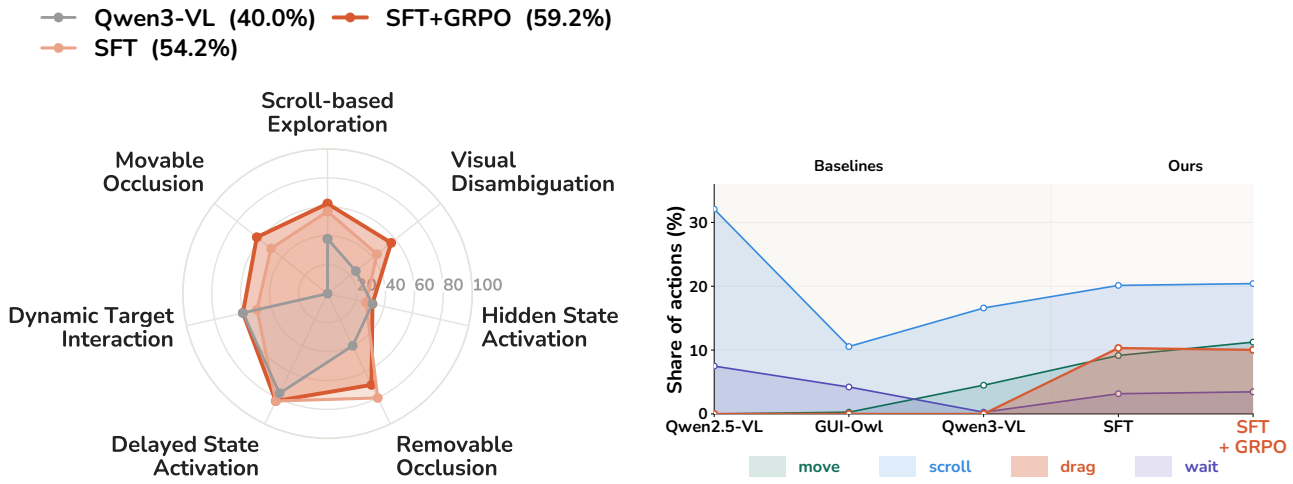
Together, the task-solvability and fidelity studies support Reveal-and-Click as a controlled yet realistic benchmark. The solvability study verifies that humans can complete challenging tasks under the same budget, visual observation setting, and action space as models, while the fidelity test shows that the mock environments preserve key properties of realistic GUI interaction.

5. Conclusion

We introduced Reveal-and-Click, a benchmark for grounding under visibility constraints. Unlike static GUI grounding, Reveal-and-Click requires agents to reveal initially hidden or non-actionable targets before clicking them. Across 16 controllable mock GUI environments and 601 tasks, Reveal-and-Click covers viewport movement, state-changing interactions, temporal dynamics, and manipulation-based reveal. Human studies show that the tasks are solvable and that the mock environments preserve realistic GUI interaction. Current VLM agents remain far below human performance. Oracle reveal and action-prior analyses show that many failures happen before localization: agents do not choose or execute the reveal action. Reveal-and-Click provides a testbed for studying visibility recovery in GUI agents.

6. Limitations

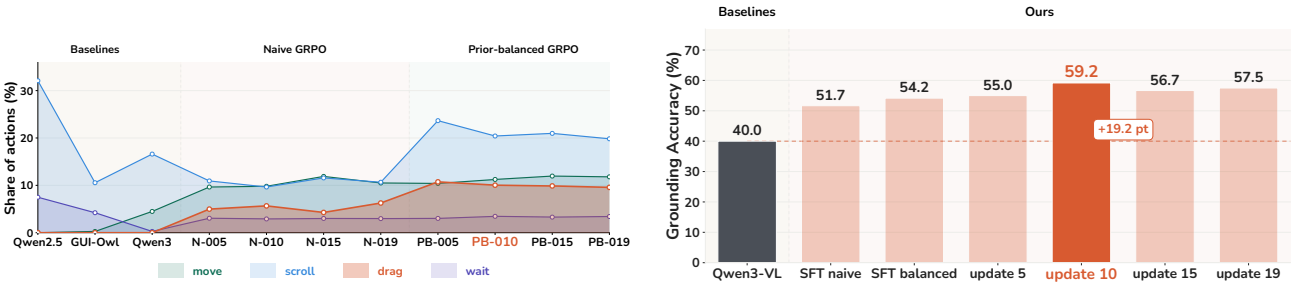
Reveal-and-Click is a controlled benchmark for visibility recovery, not a replacement for live-web or desktop-agent evaluation. Deterministic mockups give stable target identities and reproducible state transitions, but they do not cover production variability, personalization, authentication flows, or changing third-party layouts. We use them to isolate



(a) Per-sub-type gains from baseline to SFT and SFT+GRPO.

(b) Action-prior shift across baselines, SFT, and SFT+GRPO.

Figure 6. **Behavior cloning injects missing interaction priors, and GRPO selects among them.** The best trained model improves Qwen3-VL from 40.0% to 59.2% GA on the 120-task validation split. The action-share plot shows why this matters: baseline VLMs can sometimes succeed by shortcutting, but they emit zero drag actions on the validation split; SFT exposes the policy to those actions, and GRPO improves task success once they enter the policy's action support.



(a) Action share across GRPO variants.

(b) Validation GA across GRPO checkpoints.

Figure 7. **GRPO ablation.** GRPO is effective only when initialized from the frequency-balanced SFT seed: it reaches 59.2% GA, while GRPO from the standard behavior-cloning seed does not improve over its SFT seed. The best frequency-balanced GRPO policy preserves action validity across evaluation, with 0 invalid actions and 0 empty clicks over 578 executed actions, supporting a two-stage diagnostic recipe: first expose the policy to reveal actions, then use online reward to select among them.

grounding failures under controlled and reproducible conditions, and assess their visual and interaction fidelity through a human fidelity study.

The benchmark is screenshot-only and evaluates a fixed 1440×900 desktop viewport. This setting matches our goal of testing agents when DOM or accessibility metadata is unavailable, but it does not measure agents that rely on structured UI trees, keyboard shortcuts, text entry, file operations, or mobile layouts. Reveal-and-Click also uses a small action space focused on visibility recovery. Broader task-completion benchmarks remain necessary for studying long-horizon planning and application workflows. Some difficulty remains after the target is revealed; the oracle-reveal analysis in Table 4 measures this residual grounding cost.

Our training results are diagnostic, not a claim of an optimal optimizer. We fine-tune Qwen3-VL on a held-out 120-task validation split to test whether missing interaction priors can be learned. Larger training sets, different base models, or different rewards may change the best recipe. Commercial-model latency and cost also depend on API implementation and pricing; we report our observed values only to contextualize deployment trade-offs.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd. (Project Code: IO260114-15161-01), the InnoCORE program of the Ministry of Science and ICT (N10250156, KAIST InnoCore LLM) (50%), and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project) (50%).

References

Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., and Lin, J. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Burigat, S., Chittaro, L., and Vianello, A. Dynamic visualization of large numbers of off-screen objects on mobile devices: an experimental comparison of wedge and overview+ detail. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pp. 93–102, 2012.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

Cheng, K., Sun, Q., Chu, Y., Xu, F., YanTao, L., Zhang, J., and Wu, Z. 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, 2024.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Fu, S., Tamir, N., Sundaram, S., Chai, L., Zhang, R., Dekel, T., and Isola, P. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Advances in Neural Information Processing Systems*, volume 36, pp. 50742–50768, 2023.

Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., and Su, Y. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R., and Fried, D. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Lekies, S., Stock, B., and Johns, M. 25 million flows later: large-scale detection of dom-based xss. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 1193–1204, 2013.

Li, K., Meng, Z., Lin, H., Luo, Z., Tian, Y., Ma, J., Huang, Z., and Chua, T.-S. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 8778–8786, 2025.

Mittal, H., Mittal, G., Troncoso, N. D., and Hu, Y. See, point, refine: Multi-turn approach to gui grounding with visual feedback. *arXiv preprint arXiv:2604.13019*, 2026.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Wu, Q., Cheng, K., Yang, R., Zhang, C., Yang, J., Jiang, H., Mu, J., Peng, B., Qiao, B., Tan, R., et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Xie, T., Deng, J., Li, X., Yang, J., Wu, H., Chen, J., Hu, W., Wang, X., Xu, Y., Wang, Z., et al. Scaling computer-use grounding via user interface decomposition and synthesis. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Xu, H., Zhang, X., Liu, H., Wang, J., Zhu, Z., Zhou, S., Hu, X., Gao, F., Cao, J., Wang, Z., et al. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Zheng, B., Gou, B., Kil, J., Sun, H., and Su, Y. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=piecKJ2DlB.

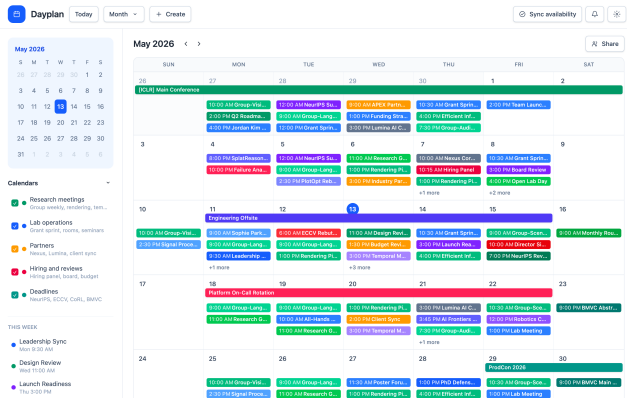
Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

A. Mockup Applications

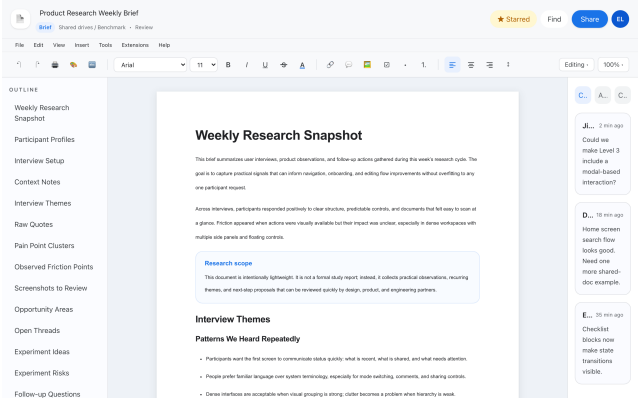
Reveal-and-Click comprises 16 controllable mockup applications spanning common GUI domains. Each mockup is designed to resemble a familiar real-world interface while remaining fully deterministic and instrumented for evaluation, allowing us to control initial states, dynamic behaviors, and target identities while preserving realistic visual and interaction patterns. Table A1 summarizes the mockup environments and their corresponding real-world interface references, and Figures A1–A5 show representative states grouped by application category.

Table A1. The 16 mockup environments and their reference interfaces.

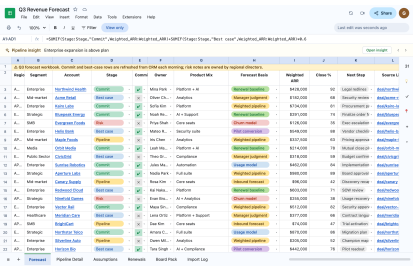
Mockup	Reference
Social Feed	Twitter/X
Planner	Google Calendar
Developer Hub	GitHub
Code Editor	VS Code Web
Community Chat	Discord
Document Editor	Google Docs
Design Canvas	Figma
Spreadsheet	Google Sheets
Email Client	Gmail
News Portal	Google
Workspace Pages	Notion
Streaming Service	Netflix
E-commerce	Walmart
Music Player	Spotify
Maps	Google Maps
Video Platform	YouTube



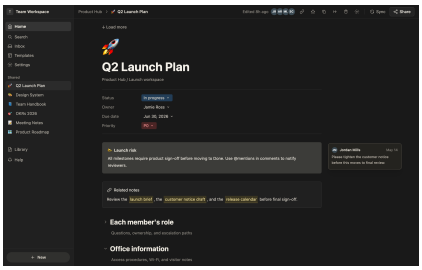
(a) Planner (Google Calendar)



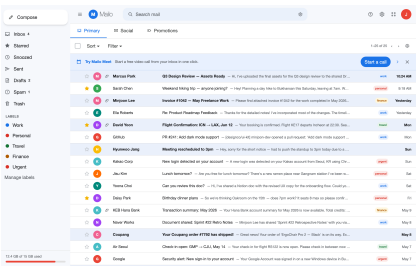
(b) Document Editor (Google Docs)



(c) Spreadsheet (Google Sheets)

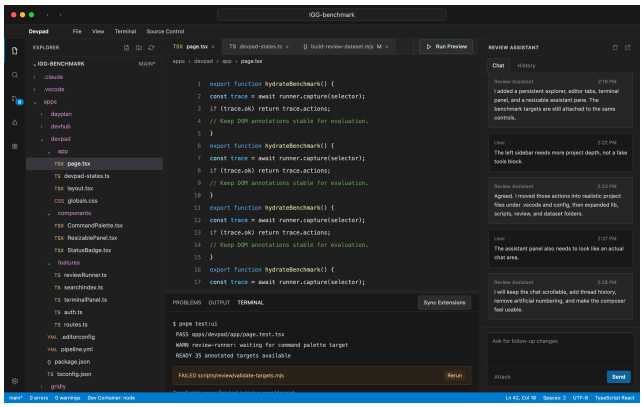


(d) Workspace Pages (Notion)

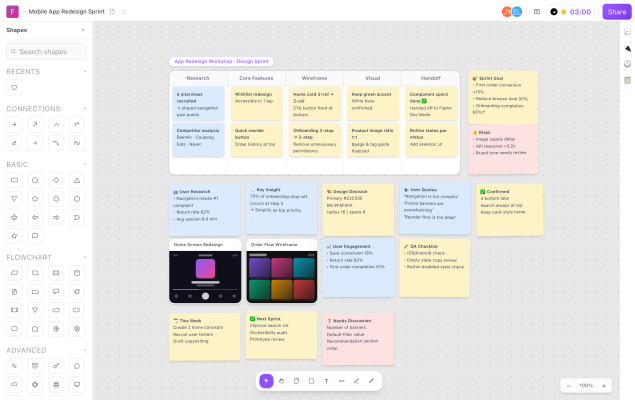


(e) Email Client (Gmail)

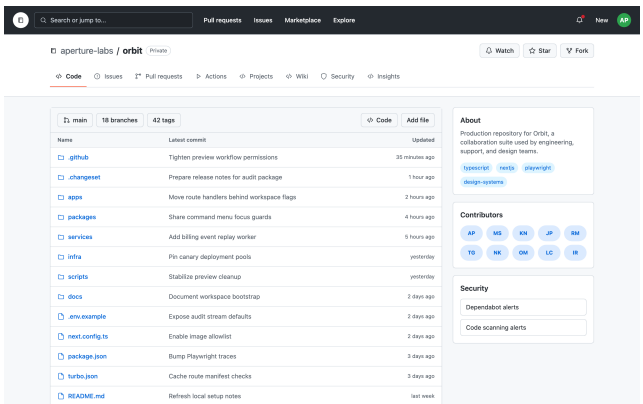
Figure A1. Representative states from productivity mockup applications.



(a) Code Editor (VS Code Web)

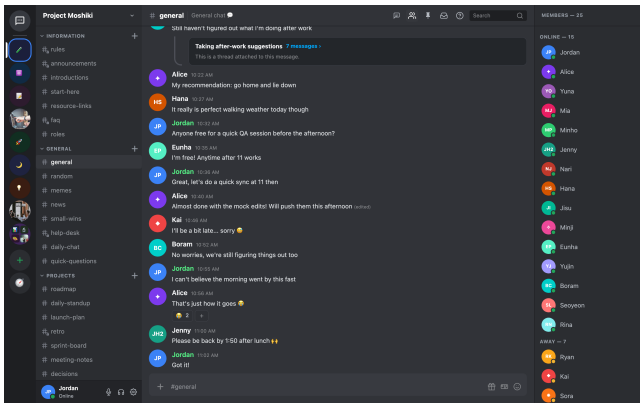


(b) Design Canvas (Figma)

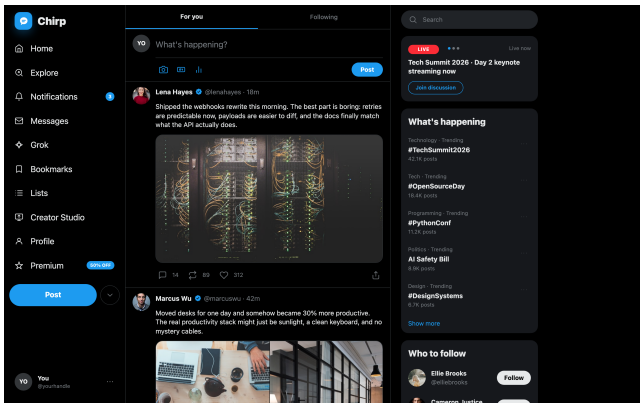


(c) Developer Hub (GitHub)

Figure A2. Representative states from development and design mockup applications.

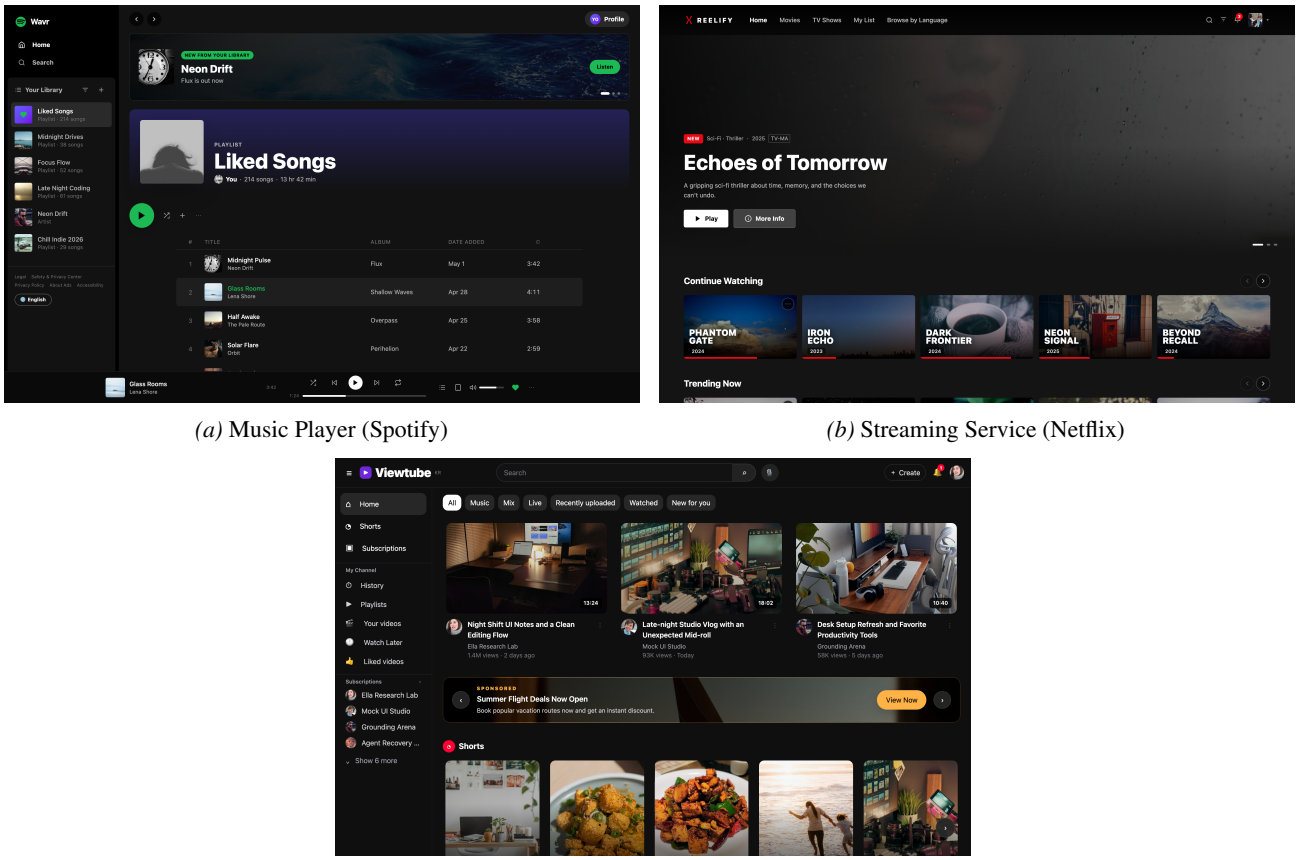


(a) Community Chat (Discord)



(b) Social Feed (Twitter/X)

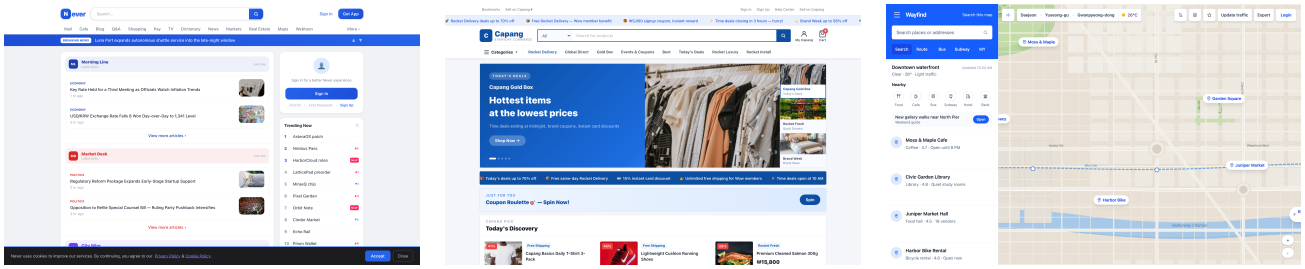
Figure A3. Representative states from social mockup applications.



(a) Music Player (Spotify) *(b)* Streaming Service (Netflix)

(c) Video Platform (YouTube)

Figure A4. Representative states from media mockup applications.



(a) News Portal (Google) *(b)* E-commerce (Walmart) *(c)* Maps (Google Maps)

Figure A5. Representative states from lifestyle mockup applications.

B. Dataset Examples

B.1. Task Specification Format

Each Reveal-and-Click task is represented as a structured JSON record that specifies the instruction, target element, initial state, and reference trace. The instruction is the target-seeking command given to the agent, while the target ID is used only by the evaluator for click-time success checking. The reference trace records one successful interaction sequence and is used to compute interaction-efficiency metrics such as Optimal Ratio (OR). Listing 1 shows a representative example.

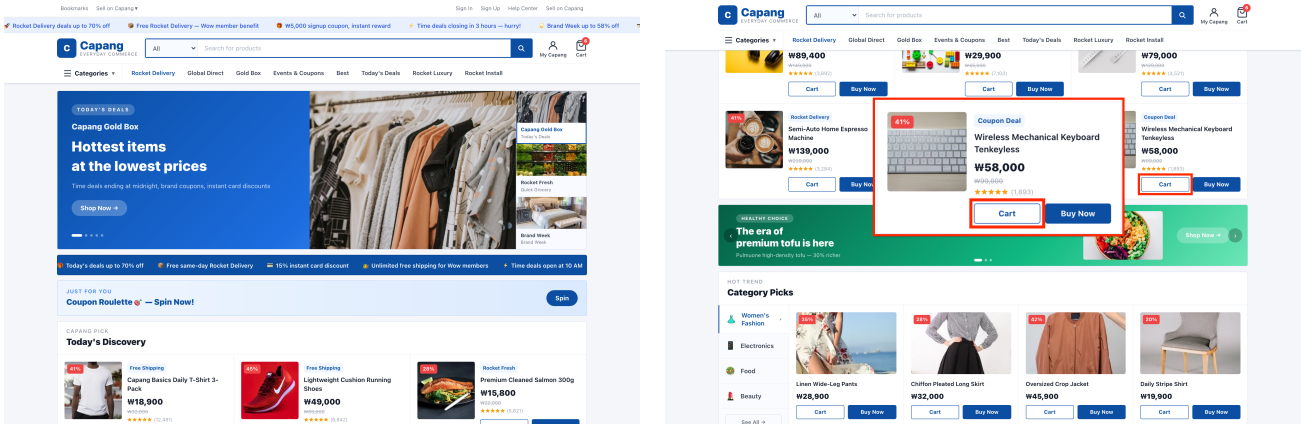
Listing 1. Task specification example. The task targets "Kai" in the direct message list (Messaging application, $\mathrm{CT}_{1\text{-}1}$: Scroll-based Exploration). The reference trace consists of a scroll action followed by a click on the revealed target.

```
{
  "id": "messaging-01-find-kai-direct-message-list",
  "sub_type": "1-1",
```

```
  "constraint_type": "off-screen",
  "instruction": "Find 'Kai' in the direct message list.",
  "target_id": "messaging.dm.kai",
  "initial_state_id": "1",
  "reference_trace": [
    {"action": "SCROLL", "dy": 1000},
    {"action": "CLICK", "selector": "messaging.dm.kai"}
  ]
}
```

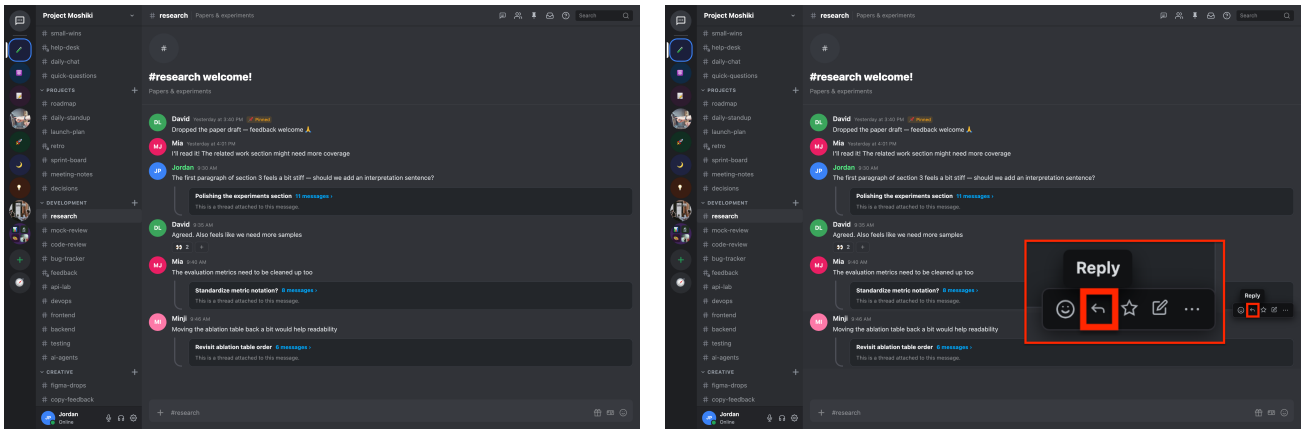
B.2. Sub-type Examples

Figures A6–A12 show one representative task for each Reveal-and-Click sub-type. Each example pairs the initial state, where the target is not directly actionable, with the final state at the successful click. Together, these examples illustrate how Reveal-and-Click separates off-screen search, visual disambiguation, hidden-state activation, removable occlusion, delayed state activation, dynamic target interaction, and movable occlusion.



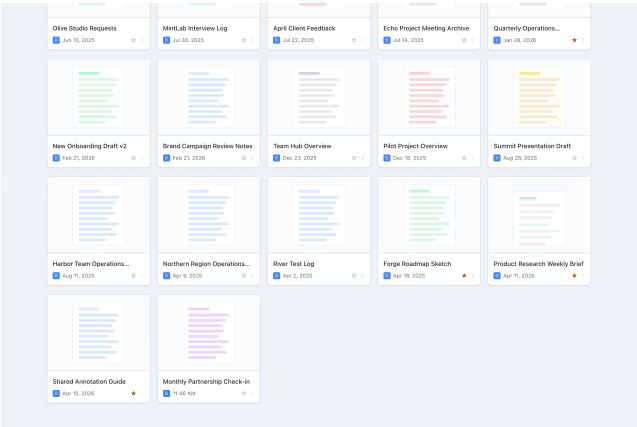
(a) Initial state *(b)* Final state at click time

Figure A6. **CT$_{1\text{-}1}$: Scroll-based Exploration** (E-commerce). *"Click 'Cart' for 'Wireless Mechanical Keyboard'."*

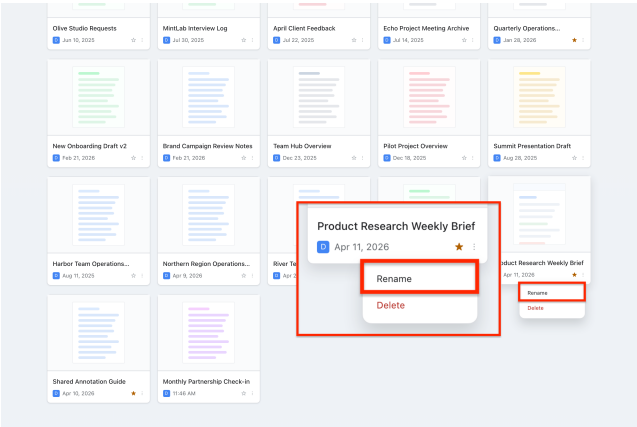


(a) Initial state *(b)* Final state at click time

Figure A7. **CT$_{2\text{-}1}$: Visual Disambiguation** (Community Chat). *"Click 'Reply' on Minji's message."*

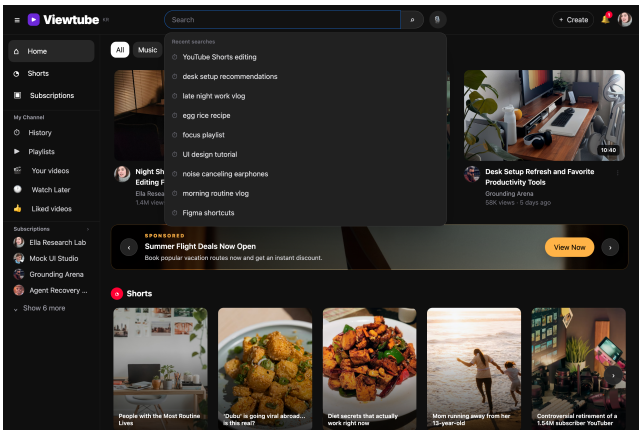


(a) Initial state

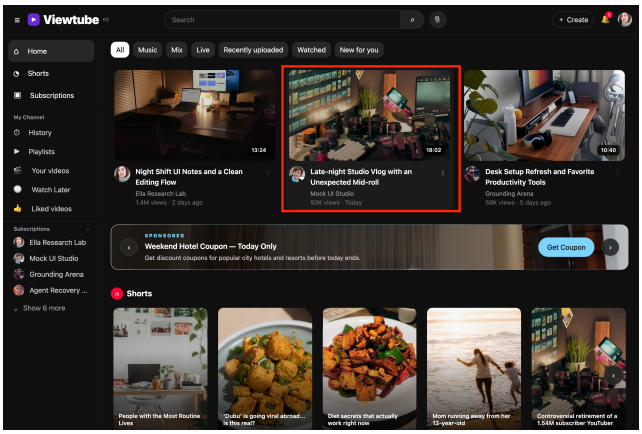


(b) Final state at click time

Figure A8. **CT$_{2\text{-}2}$: Hidden State Activation** (Document Editor). *"Click 'Rename' on 'Product Research Weekly Brief'."*

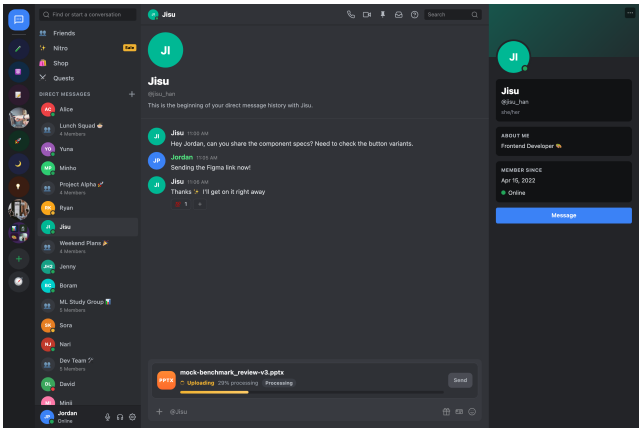


(a) Initial state

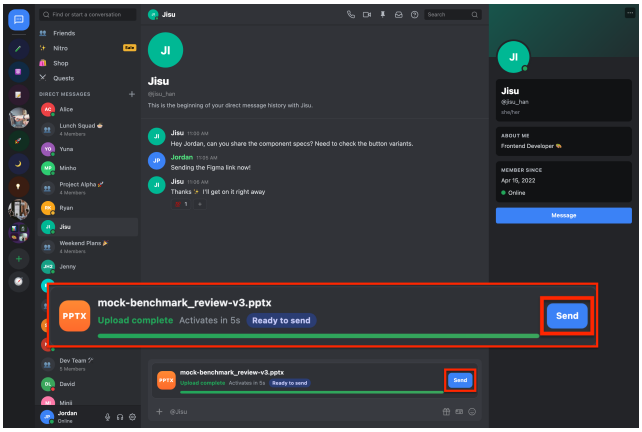


(b) Final state at click time

Figure A9. **CT$_{2\text{-}3}$: Removable Occlusion** (Video Platform). *"Click the 'Late-night Studio Vlog' video card."*

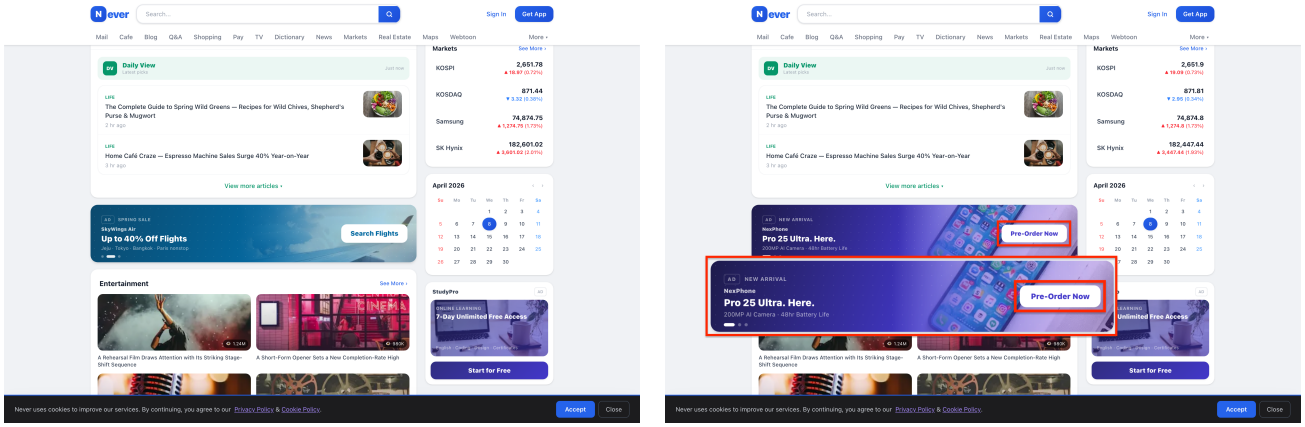


(a) Initial state



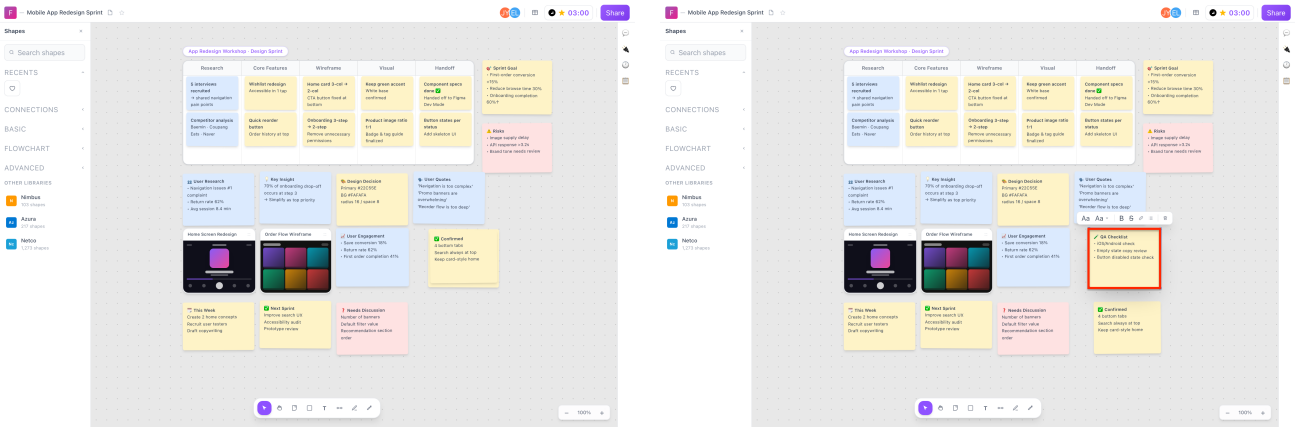
(b) Final state at click time

Figure A10. **CT$_{3\text{-}1}$: Delayed State Activation** (Community Chat). *"Click 'Send' for the pptx upload."*



(a) Initial state *(b)* Final state at click time

Figure A11. **$CT_{3\text{-}2}$: Dynamic Target Interaction** (News Portal). *"Click 'Pre-Order Now' in the 'NexPhone' ad."*



(a) Initial state *(b)* Final state at click time

Figure A12. **$CT_{3\text{-}3}$: Movable Occlusion** (Design Canvas). *"Click the 'QA Checklist' note without deleting any."*

Table A2. Extended per-application metrics for two representative full-benchmark models. The main text reports GA for all models; here we report GA, AC, OR, and TT for Qwen3-VL and GPT-5.5 on the same 601-task benchmark.

	Qwen3-VL				GPT-5.5			
App	**GA**	**AC**	**OR**	**TT**	**GA**	**AC**	**OR**	**TT**
Social Feed	29.4	6.47	0.93	17514	61.8	4.59	1.04	69757
Planner	21.2	6.88	0.79	13098	81.8	3.88	0.83	61011
Developer Hub	25.7	6.94	0.66	9763	57.1	5.57	0.69	138919
Code Editor	21.9	6.81	1.26	9575	87.5	3.88	0.97	59747
Community Chat	26.5	6.82	0.64	9357	61.2	4.82	1.03	108102
Document Editor	37.5	6.05	0.99	8451	85.0	3.48	1.04	63538
Design Canvas	8.9	7.44	1.25	13951	82.2	4.33	0.94	93199
Spreadsheet	33.3	6.24	0.84	8920	83.3	2.93	1.04	124023
Email Client	50.0	5.38	0.75	10493	88.1	3.38	0.97	63120
News Portal	29.0	6.55	0.80	19204	71.0	4.77	0.80	52984
Workspace Pages	34.3	6.40	0.86	8830	85.7	3.71	0.93	30489
Streaming Service	20.5	7.00	0.67	17224	87.2	4.05	0.71	52801
E-commerce	43.3	6.40	0.60	17047	86.7	4.40	0.72	49700
Music Player	36.0	5.80	1.15	12580	86.0	3.60	1.04	63597
Maps	26.7	6.47	1.25	11972	70.0	4.53	0.95	166930
Video Platform	50.0	5.91	0.63	8158	94.1	3.15	0.88	38354