
How Fragile Are GUI Agents? A Controlled Perturbation Benchmark for Interface Evolution

Opeyemi Akinniyi*
akinniyi@umich.edu

Carol Kang*
yhankang@umich.edu

Zuchen Li*
zuchenli@umich.edu

Xiangnong Wu*
xiangnongwu@umich.edu

Hao Yin*
haoyin@umich.edu

*Equal contribution. Authors listed in alphabetical order by last name.

Abstract

GUI agents powered by Vision-Language Models (VLMs) have shown promising capabilities in automating web-based tasks. However, their ability to remain robust as web interfaces evolve over time, a critical requirement for real-world deployment, remains underexplored. In this work, we first conduct a preliminary study on archived snapshots of real-world websites to systematically characterize how interfaces evolve, identifying five recurring categories of UI change each spanning multiple intensity levels. Grounded in these observations, we construct a controlled benchmark of synthetic websites that instantiates these perturbation categories across multiple intensity levels, enabling controlled analysis of how different types of interface change affect agent performance. We evaluate four VLM agents with one text-only baseline across three input modalities using the VisualWebArena harness and find that vision-based input, despite their generality and platform-agnostic deployment advantages, suffer substantially greater performance degradation under UI shifts than text-based agents, highlighting the practical importance of temporal robustness as a research challenge. We further show that GUI-specialized agents outperform larger general-purpose models on interaction-heavy interfaces but not on navigation-dominated ones, and that lightweight test-time interventions exhibit highly task-dependent recovery rates. Our results suggest that temporal UI robustness is a fundamental and underappreciated challenge for real-world GUI agent deployment.

1 Introduction

GUI agents built on Vision-Language Models (VLMs) have demonstrated the ability to automate computer tasks by interpreting visual interfaces and taking appropriate actions. However, maintaining reliable performance in real-world deployments remains a persistent challenge [Fu et al., 2025]. As digital environments continuously evolve through application updates and user interface (UI) modifications, agents trained on historical data often struggle to generalize, leading to degraded task performance. Current benchmarks on GUI agents typically operate in static environments, which does not capture this temporal variability. While recent work has sought to improve generalization through large-scale cross-platform data [Jang et al., 2025], the specific challenge of temporal UI evolution remains underexplored. This project investigates the temporal robustness of VLM-based GUI agents, focusing on how the evolution of the interface impacts their effectiveness.

In our initial study, we construct a temporal robustness benchmark using archived snapshots of popular websites (Craigslist, Reddit, and IMDb) from the Wayback Machine to systematically characterize how interfaces evolve. Through this analysis, we identify five recurring categories of UI change: *layout*, *semantic*, *visual*, *structural*, and *workflow* shifts, which together capture the dominant

ways interfaces evolve over time. Preliminary experiments with Qwen3-VL-30B on these archived snapshots revealed significant temporal degradation in agent performance. However, using historical snapshots introduces confounding variables, such as changes in the underlying content and missing assets, making it challenging to isolate the effect of individual interface changes. To address this, we construct synthetic websites where each variant differs from the baseline along controlled, labeled perturbation axes drawn from the five categories above. This allows for more controlled analysis of agent failures across different types of UI shifts. We then evaluate four VLM agents: Qwen3-VL-30B, UI-TARS-7B, Qwen2.5-VL-7B, InternVL2-8B, and a text-only baseline (GPT-oss-120B) on this benchmark using the VisualWebArena harness across three input modalities. Furthermore, we implement memory and chain-of-thought intervention strategies to assess the effectiveness of test-time assistance in mitigating performance loss caused by temporal UI shifts.

Our contributions are summarized as follows: (1) we introduce a benchmark for evaluating temporal robustness in GUI agents under UI changes, grounded in five recurring categories of real-world interface evolution; and (2) we provide empirical evidence characterizing failure modes under UI shifts, highlighting the limitations of vision-only inputs, task-dependent performance trade-offs between GUI-specialized and general-purpose agents, and the varying effectiveness of test-time interventions.

2 Related Work

Recent advancements in GUI agent evaluation have driven the development of several notable benchmarks. For desktop environments, OSWorld [Xie et al., 2024] provides 369 tasks with VM-based evaluation. In the web domain, WebArena [Zhou et al., 2024] offers 812 tasks within sandboxed replicas, while VisualWebArena [Koh et al., 2024] extends this to evaluate multimodal VLMs on visually complex web tasks. Alternative approaches include Mind2Web [Deng et al., 2023], which relies on cached HTML, and Online-Mind2Web [Xue et al., 2025], which evaluates agents on live websites but lacks historical snapshots. AndroidWorld [Rawles et al., 2024] has further expanded dynamic evaluation to mobile tasks.

To navigate these environments, several capable open-source VLMs have emerged. UI-TARS (7B) [Qin et al., 2025] currently achieves state-of-the-art performance on OSWorld, while efforts to improve cross-platform generalization have driven large-scale dataset construction; for instance, Jang et al. [2025] introduce MONDAY, a dataset of 313K annotated frames from instructional videos that demonstrates consistent performance gains on unseen mobile platforms.

Despite these advances, ensuring agent reliability remains a significant challenge. Existing robustness benchmarks primarily focus on spatial or visual disruptions: GUI-Robust [Yang et al., 2025] tests resilience against visual anomalies such as pop-ups and loading delays, while D-GARA [Chen et al., 2025] introduces dynamic anomalies, demonstrating up to a 33% performance drop under unexpected interface disruptions. Our work uniquely targets *temporal* robustness via synthetic websites with controlled perturbations, allowing us to isolate the causal effect of specific UI changes on agent performance: a challenge completely overlooked by current benchmarks.

3 Benchmark Construction

To enable controlled attribution of agent failures, we pivot from historical web archives to synthetic websites. Wayback Machine snapshots introduced confounding variables like content drift, missing assets, and simultaneous redesigns, making it difficult to isolate the effect of interface modifications.

Synthetic environments fix the underlying content and ground-truth outcomes, allowing each perturbed variant to differ from its base along controlled, labeled perturbation axes. Formally, we define our benchmark as $\mathcal{B} = \{(E_{d,s}^0, E_{d,s}^{c,\ell}, \tau)\}$, where $E_{d,s}^0$ is the base environment for domain d and visual style s ; $E_{d,s}^{c,\ell}$ is the perturbed variant under perturbation category c and intensity level $\ell \in \{L1, L2, L3\}$; and τ is the task instruction. For evaluation, each intensity level within category c is treated as a sequential snapshot $t \in \{1, 2, 3\}$ relative to the base $t_0 = E_{d,s}^0$, preserving the monotonic ordering required by the TRS formulation (Section 4.1). Because content and task semantics are held fixed across base and perturbed variants, any degradation in agent performance can be attributed

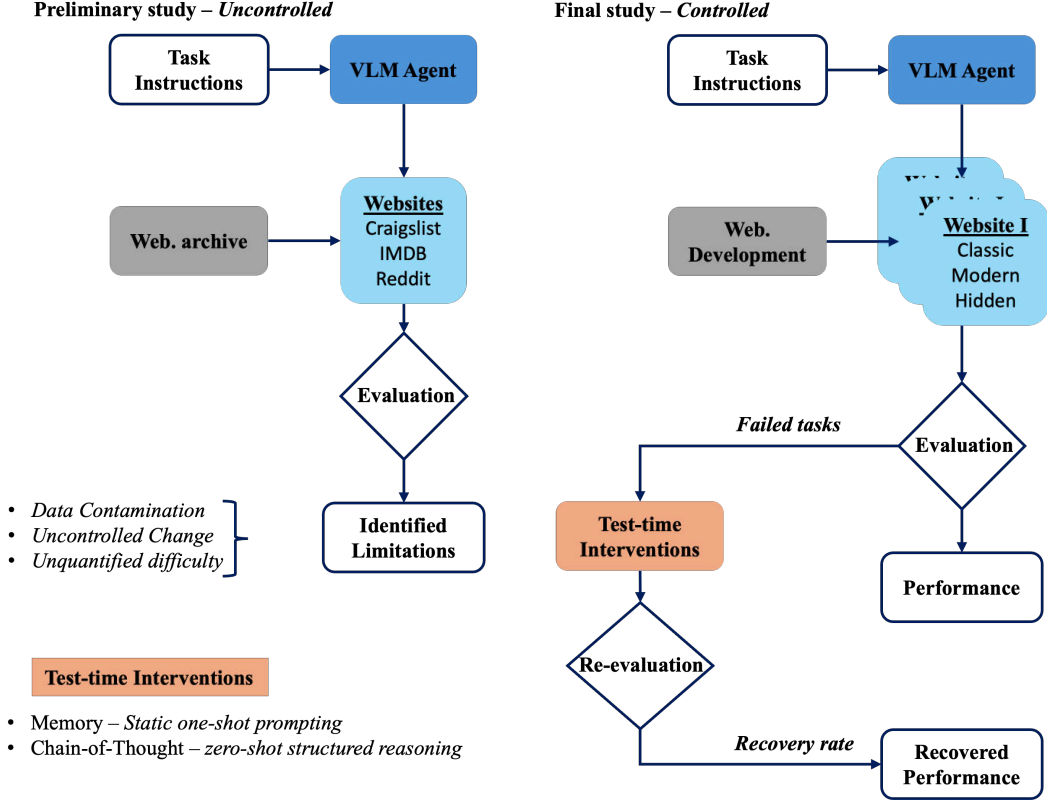


Figure 1: Simplified benchmark pipeline. This work focuses on the final stage—controlled synthetic perturbation.

to the controlled perturbations rather than unrelated content drift. The evaluated VLM agents and computational resources are listed in Table 5.

3.1 Synthetic Website Generation and Controlled Perturbation Design

Our choice of perturbation categories is grounded in a preliminary study of real-world UI evolution. We collected archived snapshots of Reddit, IMDb, and Craigslist from the Wayback Machine and manually analyzed how their interfaces evolved across multiple years. This analysis revealed five recurring and qualitatively distinct patterns of UI change. First, pages frequently underwent *layout* reorganization, where the same content was restructured from flat single-column displays into multi-column or card-based grids. Second, interface labels were often renamed without any change in underlying functionality, a pattern we term *semantic* shift. Third, the visual salience or appearance of key interactive elements changed, for instance, text-based links being replaced by icon buttons or floating action elements (*visual* shift). Fourth, information that was previously exposed inline became hidden behind accordions, tabs, or modal dialogs, requiring additional interaction to access (*structural* shift). Fifth, the sequence of actions required to complete a task changed substantially, such as single-step forms being replaced by multi-step wizards (*workflow* shift). These five categories, consistently observed across all three sites, form the axes of our controlled perturbation taxonomy and motivate the synthetic benchmark design described below.

Our synthetic websites span three domains: housing rentals, course registration, and personal portfolios. These domains encapsulate a broad spectrum of standard web interaction patterns, including information retrieval, filtering, navigation, and multi-step form submissions. For each domain, we utilize a coding-capable large language model to generate base websites from a fixed content specification. To prevent models from overfitting to a single aesthetic, this identical content is rendered across multiple distinct visual styles. To characterize the nature of the interface evolution,

each perturbed variant is described in terms of the five perturbation axes defined in Table 1, with website samples shown in Figures 2–4. Further details are provided in the appendix.

Table 1: Controlled perturbation categories and intensity progressions.

Perturbation Type	Description	Example Intensity Progression
Layout	Changes spatial organization	Single column → two column → card grid
Semantic	Renames interface elements	Contact → Get in Touch → Reach Out
Visual	Changes element appearance	Text link → icon button → floating CTA
Structural	Changes how information is exposed	Inline content → accordion → modal
Workflow	Changes the required action sequence	Single form → two-step wizard → four-step wizard

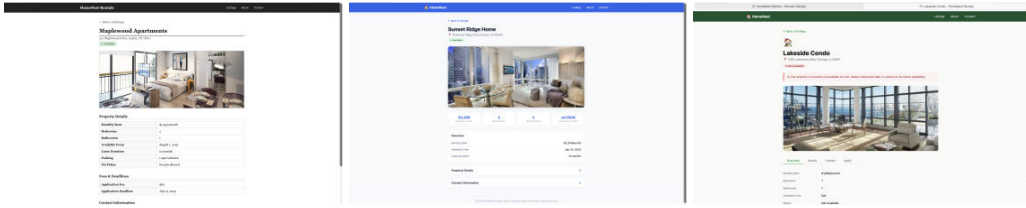


Figure 2: HomeNest Rentals. The classic template exposes listing information in flat tables, the modern introduces card-based layouts and expandable sections, and the hidden adds sidebar filters, tabs, and collapsible toggles. All templates use the same underlying property listings.



Figure 3: Course registration interfaces representing different design eras. The three versions use the same underlying course data while varying the layout, filtering controls, schedule display, and interaction flow required for enrollment tasks.

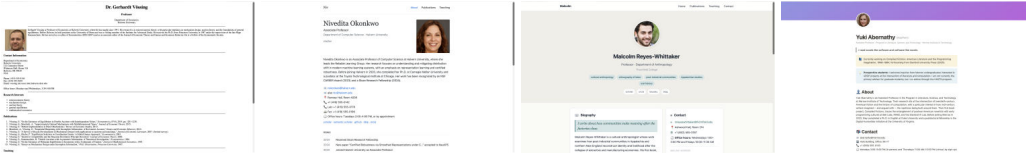


Figure 4: Faculty profile websites rendered in different visual styles. The examples use different synthetic personas to reflect realistic variation across academic websites. Controlled perturbations are applied within each generated website instance, holding that instance's content fixed.

From taxonomy to template variants. While the five categories above define orthogonal axes of UI change, real-world interface evolution rarely modifies a single axis in isolation: a redesign typically combines layout reorganization, visual restyling, and structural reorganization at once. Our template variants therefore instantiate *realistic combinations* of these perturbations rather than strict single-axis manipulations, with intensity reflecting the overall magnitude of departure from the baseline template. We treat the five-category taxonomy as a descriptive framework for characterizing how each variant differs from its base, and use it in Section 4.3 to interpret which axes drive the largest performance drops. A strictly factorial design that varies one axis at a time is left to future work.

4 Evaluation of Benchmark

We evaluate our benchmark using the VisualWebArena (VWA) harness [Koh et al., 2024], serving our synthetic websites as local environments accessible via Playwright. We test four VLM agents (Qwen3-VL-30B, UI-TARS-7B, Qwen2.5-VL-7B, InternVL2-8B) along with GPT-oss-120B as a text-only baseline. Agents are tested across three input modalities: text-only (accessibility tree extracted via Playwright), vision-only (screenshots with Set-of-Mark annotations), and multimodal (both combined); GPT-oss-120B is evaluated in text-only modality only. We also test two lightweight test-time interventions (memory and chain-of-thought (CoT)) to assess whether they can recover performance lost under temporal UI shifts without modifying model weights. Our CoT intervention extends VWA’s default prompt by requiring explicit three-step reasoning about UI element visibility before producing a final answer; our memory intervention is custom-built on top of VWA.

4.1 Success Rate & Temporal Robustness Score

In our synthetic benchmark, each perturbed variant is indexed by t , where $t = 0$ denotes the base environment $E_{d,s}^0$ and $t \in \{1, 2, 3\}$ denotes increasing perturbation intensity levels $\{L1, L2, L3\}$ within a given perturbation category c . For agent π_θ evaluated on environment snapshot $\mathcal{E}_t$ (with t_0 as baseline), the version-specific success rate is:

$$\text{SR}(t) = \frac{1}{|T_t|} \sum_{\tau \in T_t} \mathbb{1}[\pi_\theta \text{ completes } \tau \text{ in } \mathcal{E}_t] \quad (1)$$

where T_t is the set of tasks evaluated on snapshot t . The temporal degradation between snapshots t_i and t_j ($j > i$) is $\Delta(t_i, t_j) = \text{SR}(t_i) - \text{SR}(t_j)$, and the Temporal Robustness Score normalizes worst-case degradation from the baseline:

$$\text{TRS} = 1 - \frac{\max_j \Delta(t_0, t_j)}{\text{SR}(t_0)} \quad (2)$$

TRS = 1 indicates perfect resilience; TRS = 0 indicates complete failure on at least one later version. We report TRS both overall (maximizing over all perturbation categories c and intensity levels ℓ) and per category c , allowing us to isolate which axis of UI change causes the greatest robustness failure. When $\Delta(t_0, t_j) < 0$, the perturbed version is easier than the baseline and TRS > 1, indicating an inverse degradation pattern rather than a robustness failure (see Section 4.3). We note that TRS is normalized by each agent’s own baseline $\text{SR}(t_0)$ and is therefore primarily designed for within-agent comparison across perturbations rather than as an absolute cross-agent metric; we report it alongside raw SR to avoid masking cases where high TRS is achieved through uniformly low success rates.

4.2 Intervention Methods

To assess whether test-time assistance can mitigate performance loss caused by temporal UI shifts, we implement two intervention strategies applied without modifying model weights.

Memory. Before evaluating on a redesigned UI, we run the agent on the baseline UI and store correct task-answer pairs in a memory buffer. When the agent subsequently encounters the same task on a new UI version, we prepend the stored correct answer to the input prompt as a hint. This intervention helps when the underlying content is unchanged but its visual presentation has been rearranged. It can be misleading when content has moved to a different page entirely, causing the agent to hallucinate a previously seen answer rather than acknowledging the information is not visible.

Chain-of-Thought (CoT). We extend VWA’s default prompt with a mandatory three-step reasoning procedure before the agent produces a final answer: (1) *UI Scan* — enumerate all currently visible sections, panels, and interactive elements such as tabs, accordions, and dropdowns; (2) *Visibility Check* — determine whether the target information is directly visible or hidden behind an interactive component, and if hidden, identify which element likely contains it; (3) *Answer or Act* — extract the answer if visible, or specify the required interaction to reveal it. The agent is required to output its reasoning under a REASONING: tag followed by a final answer under an ANSWER: tag, which we

parse and evaluate against reference answers. In practice, we find that CoT often leads agents to overthink and hallucinate incorrect answers rather than recovering failed tasks (see Section 4.3).

Measuring intervention effectiveness. We measure intervention effectiveness using the *recovery rate*, defined as the fraction of tasks that the vanilla agent failed but the intervention agent answered correctly:

$$\text{RR}(\text{strategy}) = \frac{|\{\tau \in \mathcal{F}_{\text{vanilla}} : \pi_{\theta}^{\text{strategy}} \text{ completes } \tau\}|}{|\mathcal{F}_{\text{vanilla}}|} \quad (3)$$

where $\mathcal{F}_{\text{vanilla}}$ is the set of tasks on which the vanilla agent failed. A recovery rate of 1.0 means the intervention recovers all vanilla failures; 0.0 means no improvement. We additionally track the *hallucination rate*, defined as the fraction of still-failed tasks where the agent produced a specific wrong answer rather than admitting the information was not found.

4.3 Evaluation Results

4.3.1 Success Rate

Table 6 (deferred to the appendix for space) reports success rates across all three domains, templates, and input modalities. Text-only agents (GPT-oss-120B) consistently achieve the highest success rates, reaching 92.4% on average, confirming that DOM and accessibility tree extraction largely bypasses UI complexity. Multimodal agents perform substantially better than vision-only agents across all domains, with an average gap of 38.4 percentage points. Vision-only performance degrades sharply as UI complexity increases: on House Renting, scores drop from 57.9% on the Classic template to 16.5% on the Hidden template, a loss of 41.4 points. Course Registration shows a similar pattern, with the modern template proving hardest for vision agents (22.5% average) despite being the most visually polished, as it hides course details behind interactive panels.

Table 2 breaks down vision-only performance per agent. Despite a fourfold size difference, Qwen3-VL-30B (42.2%) and Qwen2.5-VL-7B (41.4%) achieve nearly identical overall scores, while UI-TARS-7B (41.0%) and InternVL2-8B (39.3%) trail by a small margin. Notably, all four vision agents converge to 46–48% on Personal Website regardless of model size or specialization, confirming that failures on that domain are dominated by multi-page navigation limitations rather than visual complexity.

Table 2: Per-agent success rate (%) in vision-only mode, averaged across all templates within each domain. GPT-oss-120B text-only is shown as an upper-bound reference. All vision agents receive screenshots only.

Domain	GPT-oss (text)	Qwen3-VL -30B	UI-TARS -7B	Qwen2.5-VL -7B	InternVL2 -8B
House Renting	99.6	32.3	36.6	32.9	31.1
Personal Website	90.0	47.5	46.3	47.5	46.3
Course Reg.	88.3	45.0	38.3	41.7	38.3
Overall	92.4	42.2	41.0	41.4	39.3

Vision-only mode only. House Renting averages across Classic, Modern, and Hidden templates. Course Reg. averages across 2000s, 2010s, and Modern. Personal Website averages across Raw HTML 1998, Hugo PaperMod, Notion, and Jekyll Al-Folio.

4.3.2 Temporal Robustness Score

Figure 5 shows TRS values per agent across all three domains and both vision-only and multimodal modes. In vision-only mode, UI-TARS-7B achieves the highest TRS on House Renting (0.414), outperforming the 30B Qwen3-VL model (0.236) despite being four times smaller, confirming that GUI specialization matters more than parameter count for robustness. In multimodal mode, UI-TARS again leads on House Renting (0.981), nearly matching the text-only reference (GPT-oss-120B, 0.988) while using a $17\times$ smaller model. Note that GPT-oss-120B’s high TRS reflects both high

baseline success ($SR(t_0) > 99\%$ on House Renting; Table 6) and minimal degradation, so this comparison is meaningful here. On Personal Website, all vision agents converge to TRS 0.417–0.500 regardless of size or specialization, as failures are dominated by multi-page navigation rather than visual complexity. Course Registration shows an inverse degradation pattern for multimodal agents ($TRS > 1.0$), where the 2000s template proves harder than later templates due to table-based HTML being more difficult to parse than modern semantic markup.

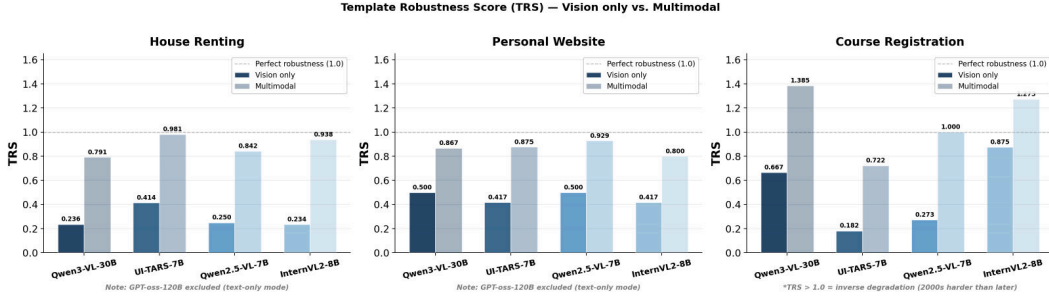


Figure 5: Temporal Robustness Score (TRS) per agent across all three domains in vision-only (left) and multimodal (right) modes. Higher is better. The dashed line at TRS = 1.0 indicates perfect robustness. GPT-oss-120B (text-only) shown as a text-only reference. TRS > 1.0 on Course Registration indicates an inverse degradation pattern where the 2000s baseline template is harder than later templates.

4.3.3 Effect of Interventions

Figure 6 shows recovery rates for memory and chain-of-thought (CoT) interventions across all four vision agents and three domains. Memory intervention is most effective on House Renting, recovering up to 52.1% of vanilla failures for Qwen2.5-VL-7B, while UI-TARS-7B shows the weakest memory recovery (23.2%) despite being the most robust vision agent overall, suggesting that GUI specialization does not transfer to memory-augmented settings. On Personal Website, all agents show near-zero recovery (0–9.5%) confirming that multi-page navigation failures are irreducible regardless of agent architecture or intervention strategy. CoT is consistently weak on House Renting and Personal Website (2–11%), but outperforms memory on Course Registration (13–23%), where failures require reasoning about interactive UI state rather than finding hidden content. Figure 7 shows hallucination rates alongside net improvement for each intervention. High hallucination rates accompany low recovery rates throughout, particularly for memory on Personal Website (hallucination rate up to 94%), where agents fabricate previously seen answers rather than admitting the information is not visible. CoT tends to produce lower hallucination rates than memory in most settings but also recovers fewer tasks, indicating a trade-off between confidence and correctness.

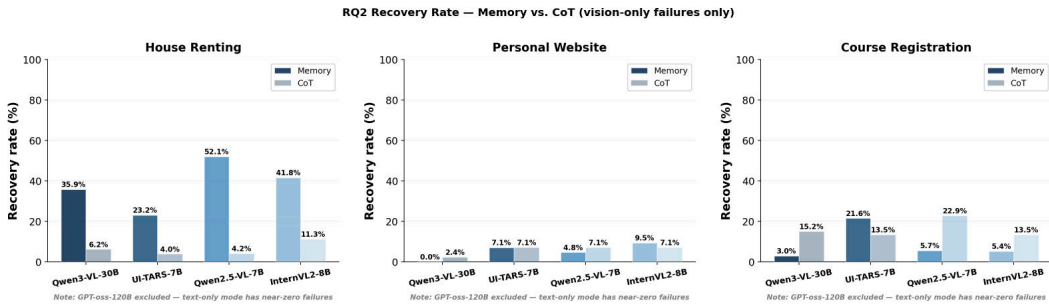


Figure 6: Recovery rate (%) for memory (solid) and chain-of-thought (faded) interventions per agent across all three domains. Recovery rate measures the fraction of failures that the intervention agent answers correctly. GPT-oss-120B excluded as it has near-zero baseline failures.

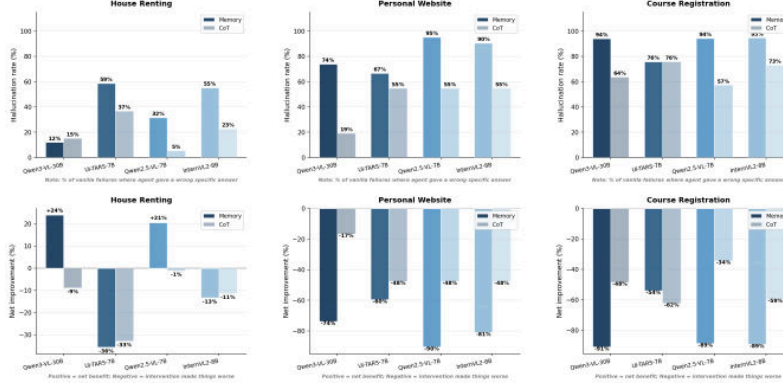


Figure 7: Hallucination rate and net improvement per agent and intervention strategy across all three domains.

5 Conclusion, Discussions, Limitations, and Future Directions

Evaluating four VLM agents and a text-only baseline across three input modalities yields three main findings. First, vision-based input suffers substantially greater degradation under UI shifts than text-based input: despite the generality and platform-agnostic advantages of screenshot-based deployment, vision-only agents face an irreducible ceiling because interactive elements such as tabs and accordions are invisible in static screenshots. Second, GUI-specialized agents outperform larger general-purpose models on interaction-heavy interfaces but not on navigation-dominated ones: UI-TARS-7B matches or exceeds Qwen3-VL-30B on house_renting at $4\times$ smaller scale, yet all vision agents converge on personal_website, where cross-page navigation rather than visual interpretation is the bottleneck. Third, lightweight test-time interventions exhibit highly task-dependent recovery rates: memory recovers tasks where content is preserved but rearranged yet hallucinates when content has moved off-page, while CoT shows the opposite profile, helping more when failures require reasoning about interactive UI state. We additionally observed an inverse degradation pattern on course_registration, where text-only and multimodal agents perform better on modern templates than on legacy ones, attributable to table-based HTML being harder to parse than modern semantic markup—a reminder that "newer" does not automatically mean "harder." Together, these results indicate that temporal UI robustness is a fundamental and underappreciated challenge for real-world GUI agent deployment.

We can further interpret these patterns through the lens of our perturbation taxonomy. The largest vision-only drops occur on templates dominated by *structural* shifts—House Renting Hidden and Course Registration Modern both relocate primary content behind tabs, accordions, and collapsible panels—consistent with the observation that screenshot-based observability, not visual reasoning per se, is the binding constraint. Templates dominated by *layout* and *visual* restyling without structural changes (e.g., Personal Website variants) cause more moderate and roughly uniform degradation across agents. *Workflow* shifts, when present, compound with structural ones rather than acting independently, which is why the most aggressive variants in House Renting and Course Registration are also the hardest. We report these associations as descriptive rather than causal, given the combined nature of the variants.

Our synthetic design enables clean causal attribution but varies perturbation factors independently, whereas real UI evolution couples them and drifts content simultaneously. Some vision-only degradation also reflects observability limits, information hidden behind interactions that a single screenshot cannot reveal, rather than intrinsic robustness gaps. Finally, our taxonomy covers interface- and workflow-level changes but leaves deeper functionality shifts, such as deprecated features and modified state logic, underexplored.

Each limitation suggests a direction. A leakage-aware hybrid benchmark combining controlled perturbations with curated real snapshots would capture coupled, content-drifting evolution while preserving causal interpretability. Extending evaluation to fully interactive multi-step environments would disentangle observability from robustness by letting agents explore hidden state. And instrumenting functionality-level perturbations would close the gap between our current taxonomy and the full space of real-world interface change.

References

- 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. *arXiv preprint arXiv:2511.16590*, 2025.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-Seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- Tang Fu, Alex Su, Chen Zhao, Haoyuan Wang, Minghao Wu, Zhiyuan Yu, et al. Mano technical report. *arXiv preprint arXiv:2509.17336*, 2025. URL <https://doi.org/10.48550/arXiv.2509.17336>.
- Yunseok Jang, Yeda Song, Sungryull Sohn, Lajanugen Logeswaran, Tiange Luo, Dong-Ki Kim, Kyunghoon Bae, and Honglak Lee. Scalable video-to-dataset generation for cross-platform mobile agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- Jing Yu Koh, Robert Lo, Lawrence Lawrence, Sewon Min, Hao Xing, Xiang Yue, et al. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.
- Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjuan Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.
- Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.
- Tianbao Xie, Danyang Zhang, Jiarui Chen, Xiaochuan Li, RuiBo Zhao, Ruisheng Cao, Yu Gu, Yanjun Gao, Hao Peng, Yining Hong, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Thirty-Eighth Conference on Neural Information Processing Systems (NeurIPS)*, 2024.
- Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*, 2025.
- 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. *arXiv preprint*, 2025.
- Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Appendix

Our codebase is available at:

<https://github.com/akinniye-okeyemi/EECS-Final-project>

Our benchmark dataset is available at:

<https://www.dropbox.com/scl/fo/34tjo8qqz8wd9ybffrifs/ACsX7lm4-ewwzMyiEVM-TsM?rlkey=wtuc9ozsj9ag3s6elvs4jne63&st=6njx4ksx&dl=0>

Table 3: Author Contributions. All authors contributed equally

Contribution	Akinniye	Kang	Li	Wu	Yin
Brainstorming & ideation	✓	✓	✓	✓	✓
Data collection & synthesis	✓	✓		✓	✓
Project organization			✓		
Running experiments	✓		✓		
Liaison with GSIs & advisors			✓		
Presentation	✓	✓	✓	✓	✓
Writing	✓	✓	✓	✓	✓

C.K., X.W., and H.Y. led data collection and synthesis, with O.A. also contributing to this effort. Z.L. organized and led the project throughout, with O.A. additionally taking on organizational responsibilities in the second half. O.A. and Z.L. ran the experiments. Z.L. served as the primary liaison with GSIs and faculty advisors. All authors contributed equally to brainstorming, the final presentation, and writing of this report.

Table 4: Legend of Extended Technical Notations and Metrics

Category	Symbol	Description
Entities & Models	$\mathcal{B}$	The Controlled Perturbation Benchmark , defined as the set $\{(E_{d,s}^0, E_{d,s}^{c,\ell}, \tau)\}$.
	$E_{d,s}^0$	The Base Environment for domain d and visual style s .
	$E_{d,s}^{c,\ell}$	The Perturbed Variant under perturbation category c and intensity level ℓ .
	π_θ	The VLM-powered GUI Agent parameterized by weights θ .
	τ	A Standardized Task provided to the agent as an instruction.
State & Modality	$\mathcal{T}_{\text{ally}}$	The Accessibility Tree extracted via Playwright for structured DOM representation.
	$\mathcal{M}_{\text{SoM}}$	Set-of-Mark Annotations applied to screenshots for visual element grounding.
	a_t	A Predicted Single-Turn Action (e.g., <code>CLICK [idx]</code> or <code>ANSWER: text</code>).
Perturbation Design	c	The Perturbation Category : layout, semantic, visual, structural, or workflow.
	ℓ	The Intensity Level of a perturbation, $\ell \in \{L1, L2, L3\}$.
	t	The Snapshot Index within a perturbation category, with t_0 denoting the base.
	$ T_t $	The Total Task Count evaluated within snapshot t .
Evaluation Metrics	$\text{SR}(t)$	Version-Specific Success Rate : average task completion rate within snapshot t .
	$\Delta(t_i, t_j)$	Temporal Degradation : performance drop from snapshot t_i to t_j .
	TRS	Temporal Robustness Score : a normalized metric measuring an agent’s worst-case degradation relative to its own baseline.
	RR	Recovery Rate : fraction of vanilla failures that an intervention strategy recovers.
	$\mathbb{1}[\cdot]$	The Indicator Function evaluating to 1 for success and 0 for failure.

Synthetic Website Templates: Detailed Descriptions

Housing Rental Website Generation

The housing rental domain is designed to evaluate whether GUI agents can retrieve and compare structured listing information under controlled changes in interface complexity. We generate synthetic

rental websites containing the same underlying property listings across multiple interface variants. The property data, including rent, location, availability, contact information, and application details, is held fixed, while the visual layout and interaction requirements are varied.

The generated templates progress from a simple baseline interface to more complex designs. In the simplest version, listing information is directly visible through a flat layout. More complex versions reorganize the same information into card-based layouts, expandable sections, sidebar filters, tabs, and collapsible content. This progression allows us to test whether agents can recover the same target information as it becomes less directly visible and requires additional interaction.

The tasks are modeled after common actions on rental platforms, such as finding a listing that satisfies a city or price constraint, retrieving contact details, checking availability, and locating application-related information. Because the same listing data is preserved across variants, differences in agent performance can be attributed primarily to changes in presentation and interaction structure rather than to changes in the underlying content.



Figure 8: Example HomeNest Rentals templates. The classic template exposes listing information in flat tables, the modern template introduces card-based layouts and expandable sections, and the hidden template adds sidebar filters, tabs, and collapsible toggles. All templates use the same underlying property listings.

Faculty Personal Website Generation

The personal website domain is designed to evaluate whether GUI agents can retrieve structured information from academic profile pages under realistic variation in presentation style. We generate synthetic faculty profiles containing fields such as name, title, institution, research interests, contact information, office hours, publications, and advising status. Rather than forcing the same faculty persona into every visual template, we generate multiple independent personas rendered in different academic website styles, such as minimalist HTML, *al-folio*, PaperMod, and Notion-style layouts. This better reflects the heterogeneity of real academic websites, where both the content and the visual presentation vary across faculty pages.

For controlled robustness evaluation, perturbations are applied within each generated website instance. That is, the base version and its perturbed variants share the same underlying content and task answers, while the interface is modified along a labeled perturbation axis. This separates two goals of the data generation process: diverse base-site generation for coverage, and matched perturbation generation for causal analysis of agent failures.



Figure 9: Examples of synthetic faculty profile websites rendered in different visual styles. The examples use different synthetic personas to reflect realistic variation across academic websites. Controlled perturbations are applied within each generated website instance, holding that instance’s content fixed.

Course Registration Website Generation

The course registration domain is designed to evaluate whether GUI agents can complete academic enrollment tasks under different generations of interface design. We generate synthetic course registration websites using the same underlying set of fifteen courses across all variants. Each course has fixed information such as course code, title, instructor, meeting time, location, remaining seats, and enrollment status. The core course data is preserved across versions, while the interface style, layout, and interaction complexity are varied.

The generated websites are designed to represent different eras of course registration systems, including a 2000s-style interface, a 2010s-style interface, and a modern interface. These versions expose the same course information through different UI conventions, such as older table-heavy layouts, intermediate search and filter interfaces, and more modern interactive designs with richer schedule and warning displays. This allows us to test whether agents can perform the same registration tasks as the interface shifts across visual style, information organization, and required interaction flow.

The tasks reflect common operations students perform during course registration. They include looking up course details, applying filters, searching for courses, adding and dropping courses, identifying scheduling conflicts, interpreting warnings, and viewing selected courses in a weekly schedule. More complex tasks combine several of these operations, such as filtering to a course category, adding a course, and verifying that it appears in the schedule. Because the same course data is used across all versions, differences in agent performance measure robustness to interface design and workflow changes rather than changes in the underlying registration information.



Figure 10: Example synthetic course registration interfaces representing different design eras. The three versions use the same underlying course data while varying the layout, filtering controls, schedule display, and interaction flow required for enrollment tasks.

Job Application Website Generation

While we developed templates for job-application domain, it was not included in the final evaluation due to time constraints; we describe it here for completeness as a starting point for future work. The job application domain is designed to evaluate whether GUI agents can retrieve and reason about structured recruitment information under controlled changes in interface layout and interaction workflows. We construct synthetic job application websites that present the same underlying job listings across multiple interface variants. The job data, including position title, application deadline, required materials, contact information, and application instructions, is kept fixed, while the visual presentation and interaction mechanisms are systematically varied.

The generated templates progress from a simple baseline interface to more complex designs with increasing interaction requirements. In the simplest version, all job-related information is directly visible in a flat, document-style layout. More complex versions reorganize the same content into card-based components, introduce buttons and expandable sections, and further hide critical information behind tabs or collapsible elements. This progression enables us to test whether agents can consistently recover target information as it becomes less directly accessible and requires multi-step interaction. These variants correspond to different types of interface changes, including layout shifts, workflow shifts, and visibility shifts.

The tasks are modeled after common actions on job application platforms, such as identifying the most recent job opening, retrieving application deadlines, finding submission instructions, checking required materials, and verifying whether a position is still open. These tasks are executed across all interfaces variants with identical underlying data, ensuring that any performance differences are

primarily caused by changes in interface structure and interaction complexity rather than differences in content.

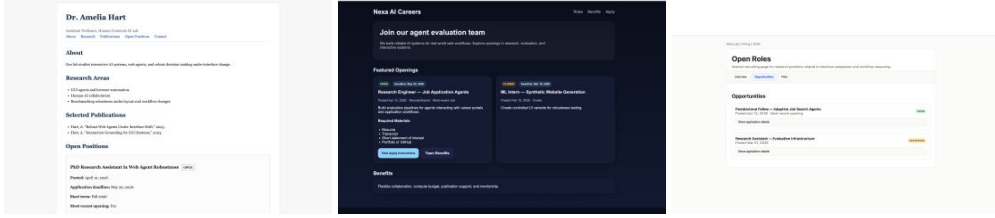


Figure 11: Example job application website templates. The classic template presents all job information in a flat academic-style layout, the modern template introduces card-based structures and interactive elements, and the notion-style template hides key information behind tabs and collapsible sections. All templates share the same underlying job listings.

Table 5: Evaluated VLM agents and computational resources.

Model	Parameters	Type	Endpoint
GPT-oss-120B	120B	Text-only	Class vLLM server
Qwen3-VL-30B	30B	General vision	Class vLLM server
UI-TARS-7B-DPO	7B	GUI-specialized	HPC clusters (Great Lakes & Bridges-2)
Qwen2.5-VL-7B	7B	General vision	HPC clusters (Great Lakes & Bridges-2)
InternVL2-8B	8B	General vision	HPC clusters (Great Lakes & Bridges-2)

Table 6: Success rate (%) by domain, template, and input modality. Text-Only (GPT-oss-120B) is shown as a DOM/accessibility-tree-based reference. Multimodal and Vision-Only columns report the mean across four vision agents (Qwen3-VL-30B, UI-TARS-7B, Qwen2.5-VL-7B, InternVL2-8B). Best vision-only score per domain marked [†]; worst marked [‡].

Domain	Template	Text-Only (GPT-oss)	Multimodal (avg)	Vision-Only (avg)	Vision-Only Drop
House Renting	Classic	100.0	94.6	57.9[†]	−42.1
	Modern	98.8	85.4	25.3	−73.5
	Hidden	100.0	94.0	16.5[‡]	−83.5
Personal Website	Raw HTML 1998	95.0	75.0	60.0[†]	−35.0
	Hugo PaperMod	95.0	77.5	40.0	−55.0
	Notion	90.0	77.5	27.5	−62.5
	Jekyll Al-Folio	80.0	65.0	60.0[†]	−20.0
Course Reg.	2000s	85.0	70.0	48.8	−36.2
	2010s	90.0	78.8	51.3[†]	−38.7
	Modern	90.0	76.3	22.5 [‡]	−67.5
Overall (macro avg.)		92.4	79.4	41.0	−51.4

Vision-Only avg. computed across Qwen3-VL-30B, UI-TARS-7B, Qwen2.5-VL-7B, InternVL2-8B. Drop = Vision-Only avg. — Text-Only. GPT-oss-120B is text-only and excluded from vision averages.