StressWeb: A Diagnostic Benchmark for Web Agent Robustness under Realistic Interaction Variability

Anonymous ACL submission

Abstract

Large language model-based web agents have demonstrated strong performance on realistic web interaction tasks. However, existing evaluations are predominantly conducted under relatively stable and well-behaved interaction conditions, which may overestimate agent robustness. High task success in such idealized settings does not necessarily reflect performance under realistic web interaction. To address this limitation, we introduce a diagnostic stress-testing benchmark for web agents. We first construct realistic and controllable web environments that provide clean and stable interaction workflows as reference baselines. We then introduce structured and controlled perturbations that emulate interaction variability, including shifting layouts, altered interaction semantics, and execution disruptions. By comparing agent behavior between clean and perturbed settings, our framework enables systematic diagnosis of robustness under what-if interaction scenarios. Through extensive evaluation of state-of-the-art multimodal web agents, we show that stress-based evaluation exposes failure modes and substantial robustness gaps that remain hidden under clean benchmark conditions.

1 Introduction

Large language model (LLM)–based web agents have recently achieved strong performance on complex web interaction tasks such as form filling, online shopping, and information management (Deng et al., 2023; Ning et al., 2025). These agents integrate multimodal perception and sequential decision-making to interact directly with complex web interfaces.

Alongside these advances, increasingly realistic benchmarks have been developed to evaluate agent performance in environments that resemble real-world websites. Platforms such as WebArena (Zhou et al., 2023) and WebVoyager (He et al., 2024) deploy full-stack websites to improve ecological validity, while works such as REAL (Garg et al., 2025) and MLA-Trust (Yang et al., 2025) emphasize simulation and reproducibility. However, despite increasing realism, evaluation is still largely conducted under relatively stable and well-behaved interaction conditions.

[0]Code and data will be released upon acceptance.

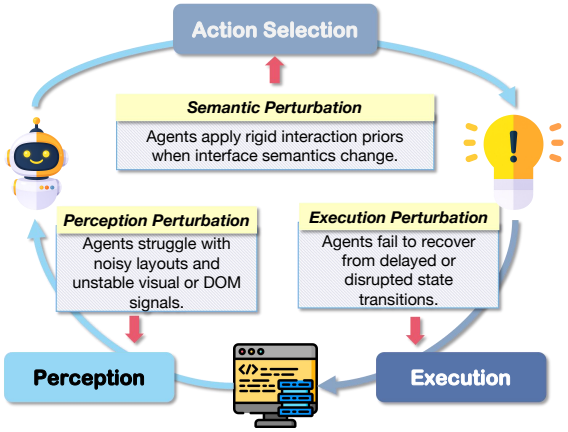


Figure 1: Perturbations Across the Interaction Pipeline

In real-world settings, web interfaces frequently exhibit asynchronous responses, delayed or failed updates, unexpected pop-up interruptions, shifting layouts, and evolving interaction semantics (Pei et al., 2025, 2026; Edgar, 2024; Mudryi et al., 2025; Guo et al., 2024; Brach et al., 2025). Such variability challenges agent perception, action selection, and execution in ways that are rarely stress-tested in standard benchmarks. Consequently, high task success in idealized benchmark environments does not necessarily reflect robustness under dynamic and irregular web interaction.

To address this gap, we propose a diagnostic stress-testing benchmark for web agents. Rather than passively measuring performance in fixed environments, we introduce controlled perturbations that simulate realistic interaction variability. This intervention-based design enables a structured analysis of agent robustness under irregular conditions.

We operationalize this framework by first constructing a collection of clean, realistic web environments that provide stable and comprehensive interaction workflows across diverse task types. These environments serve as reliable reference baselines. We then derive perturbed counterparts by injecting carefully designed stage-aligned interference into the interaction process (Figure 1): (1) perception perturbations that introduce visual and DOM irregularities, (2) semantic perturbations that alter action semantics under explicit or implicit cues, and (3) execution perturbations that induce action failures or disruptive interruptions. Each task is paired with a deterministic multi-checkpoint evaluation script that automatically verifies intermediate interaction milestones as well as final outcomes.

Through extensive evaluation of multimodal LLM web agents, we show that stress perturbations expose failure modes hidden under clean evaluation settings. Our results reveal substantial gaps between stable benchmark performance and behavior under dynamic interaction variability, highlighting the importance of intervention-based evaluation. In summary, this work makes three contributions:

- We introduce a diagnostic stress-testing benchmark that evaluates web agent robustness under irregular interaction conditions.
- We design stage-aligned perturbation mechanisms that enable robustness analysis across perception, action selection, and execution.
- Our stress-based evaluation reveals hidden failure modes and robustness gaps concealed by idealized benchmark settings.

2 Related Work

2.1 Web Agent Benchmarks: From Simplified to Realistic Environments

Early research on web agents focused on sandboxed environments that enable agents to interact with simplified web interfaces. World of Bits (Shi et al., 2017) introduced one of the first open-domain platforms for web-based agents, covering tasks such as form filling and navigation across diverse websites. Subsequently, WebShop (Yao et al., 2022) proposed a scalable simulated e-commerce environment to evaluate grounded language agents on goal-oriented shopping tasks. Recent benchmarks have significantly advanced the realism of web agent evaluation. Mind2Web (Deng et al., 2023) constructs a large-scale dataset of real-world web interaction trajectories, enabling supervised training and evaluation of generalist web agents. WebArena (Zhou et al., 2023) deploys full-stack realistic websites in controlled environments, enabling reproducible evaluation while preserving real-world complexity. Similarly, WebVoyager (He et al., 2024) demonstrated an end-to-end web agent that operates directly on screenshots and natural language instructions, highlighting the potential of large multimodal models for realistic web interaction. More recently, BrowserArena (Anupam et al., 2025) evaluates agents on realistic web navigation tasks across diverse websites, further emphasizing end-to-end interaction capabilities of modern LLM agents. In parallel, several recent benchmarks evaluate browsing and information-seeking capabilities of web agents. BrowseComp (Wei et al., 2025a) introduces a challenging benchmark focused on multi-step web browsing and information retrieval tasks. MM-BrowseComp (Li et al., 2025) further extends this setting to multimodal browsing scenarios where agents must reason over visual and textual web content. Similarly, DeepShop (Lyu et al., 2025) and WebMall (Peeters et al., 2025) study shopping-oriented web agents that retrieve and compare product information across multiple online stores. Despite these advances, most existing benchmarks remain largely static once instantiated. While increased realism improves ecological validity, it also introduces uncontrolled variability. Consequently, failures are difficult to attribute to perception, planning, or incidental environmental factors (e.g., asynchronous loading or interface changes). This limits their diagnostic value and motivates the need for environments that are both realistic and controllable.

2.2 Simulators and Environment Modeling

To address the tension between realism and controllability, several recent works have explored simulated web environments. REAL (Garg et al., 2025) evaluates agents on deterministic simulations of real websites, enabling execution errors to be isolated by eliminating stochastic environmental factors. WebCanvas (Pan et al., 2024) introduces an benchmark that emphasizes controllable environment replay while maintaining realistic interaction workflows. Beyond evaluation, simulators have also been adopted as tools for improving agent robustness through training. WebAgent-R1 (Wei et al., 2025b) leverages end-to-end multi-turn rein-

forcement learning in simulated web environments to enhance agent recovery behaviors. Similarly, GUI Exploration Lab (Yan et al., 2025) demonstrates that reinforcement learning in simulated GUI environments can significantly improve exploration efficiency and robustness in complex screen navigation tasks. These works suggest a growing consensus that simulated environments are valuable for controlled evaluation and training. However, prior approaches mainly use them for data generation or performance improvement. In contrast, our work uses controllable web environments to enable diagnostic evaluation, introducing structured perturbations that systematically probe agent reliability under realistic interaction variability.

2.3 Reliability and Robustness of LLM Agents

As large language model agents are increasingly deployed in interactive environments, understanding their reliability and robustness has become a critical research challenge. Recent studies have moved beyond aggregate task success rates to analyze agent failures in greater depth. WAREX (Kara et al., 2025) performs a large-scale reliability analysis of web agents across existing benchmarks. By repeatedly evaluating agents under identical task settings, it reveals substantial performance variance and execution instability. However, WAREX is not a benchmark or environment itself, but an analysis framework operating on fixed benchmarks, and thus relies on post-hoc outcome statistics without the ability to intervene in or manipulate environment conditions. In parallel, MLA-Trust (Yang et al., 2025) benchmarks the trustworthiness of multimodal LLM agents in GUI environments, focusing on perception, reasoning, and safety-related failures. While these efforts provide valuable insights into agent reliability, they primarily analyze failures under predefined task distributions and static environment configurations. In contrast, our work adopts an interventional perspective on reliability evaluation. By actively injecting controlled perturbations into the web environment, we enable validation of agent failures. This approach moves beyond correlational post-hoc analysis toward mechanistic understanding of how and why agents fail under specific environmental disruptions, complementing existing benchmarks and reliability analyses. A comparison with existing benchmarks is provided in Appendix A.

3 Methodology

3.1 Benchmark Overview

Our benchmark evaluates web agent robustness under controlled environmental stress. It consists of clean reference environments, structured perturbations, and deterministic multi-checkpoint evaluators (Figure 2).

To systematically organize perturbations, we view web agent interaction as a structured pipeline rather than a single atomic transition. Each interaction cycle can be decomposed into three stages: (1) perception, where the agent observes the current interface; (2) action selection, where the agent selects an action based on perceived content and learned action semantics; and (3) execution, where the environment realizes the issued action and updates its internal state. Perceptual perturbations modify state exposure, semantic perturbations alter action selection rules, and execution perturbations affect transition reliability. Each task is first evaluated in a clean setting, where the interaction pipeline is stable and deterministic across all three stages. The same tasks are then re-evaluated under controlled perturbations with instructions fixed, enabling diagnostic comparison across perception, semantic, and execution dimensions.

3.2 Reference Environments

To enable controlled stress testing, we construct a set of clean reference web environments that serve as the baseline reference for all subsequent evaluations. In these environments, all stages of the interaction pipeline, including perception, action selection, and execution, operate in a stable and fully deterministic manner. This provides a reliable baseline against which the effects of structured perturbations can be clearly isolated and measured.

Construction Pipeline. Each environment is generated through an agent-driven website construction pipeline that converts natural language task specifications into fully functional web applications. Unlike static template-based generation, the pipeline produces structurally coherent websites with realistic navigation flows, persistent internal state, and task-consistent data. The controllable simulation framework also makes the interaction dynamics remain stable and reproducible (Garg et al., 2025). The generation process is automated yet controllable, enabling scalable environment expansion and the construction of training data that is

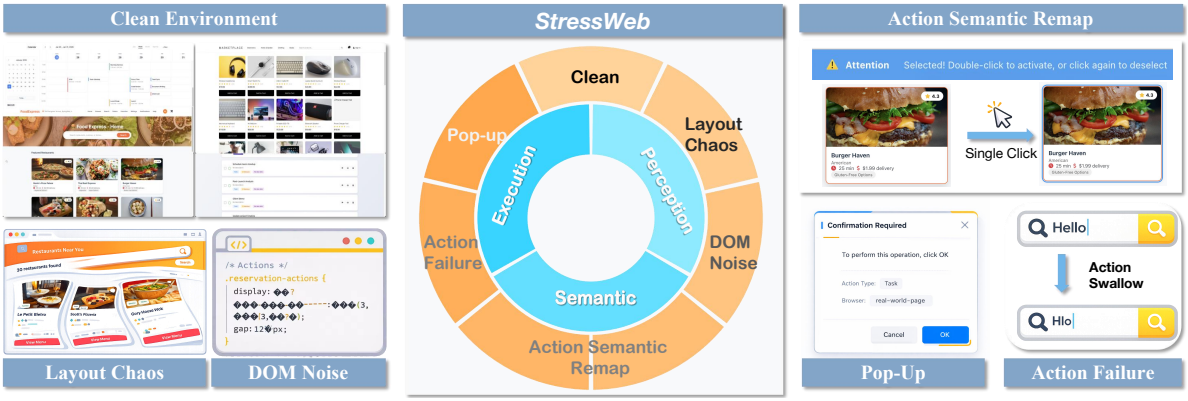


Figure 2: Overview of the StressWeb benchmark. Clean reference environments provide stable interaction workflows as reference. The benchmark introduces six perturbation types spanning perception (layout chaos, DOM noise), action selection (action semantic remap with or without explicit cues), and execution (pop-up interruptions, action failures) to systematically evaluate web agent robustness under controlled interaction variability.

distributionally aligned with evaluation settings. To ensure reliability, all generated websites undergo manual verification and light refinement to confirm functional correctness and interface stability. Implementation details are provided in Appendix B.

Benchmark Scope. We construct 10 websites covering different applications comprising 149 task queries (more details in Appendix C). Each query is evaluated under 7 environment settings, resulting in a total of 1,043 task evaluations. The benchmark covers representative end-to-end interaction patterns, including multi-step navigation, form completion, search and filtering, state modification, and batch operations. These tasks require consistent perception, coherent action selection, and reliable execution across multiple steps, reflecting common real-world interaction scenarios.

3.2.1 Perception Perturbations

Perception perturbations target the perception stage of the interaction pipeline. At this stage, the agent observes the rendered interface and identifies task-relevant elements from visual and structural signals. In real-world web systems, the observable interface often contains visual noise or structural irregularities due to legacy code, dynamic rendering, and anti-crawling mechanisms. Such conditions can disrupt agents that rely on clean or template-like interface representations. To evaluate robustness at this stage, we construct perception-perturbed environments that introduce controlled visual and structural variability. These environments require agents to remain robust to noise and to correctly identify task-relevant elements despite misleading or redundant interface signals.

Concretely, we construct two distinct perception-perturbed environments: a visual-perturbed environment and a DOM-perturbed environment. In the visual-perturbed setting (Chaos), we modify the rendered presentation of interface elements through changes such as altered text size, element rotation, and positional shifts. In the DOM-perturbed setting (Noise), we introduce structural redundancy and lightweight obfuscation patterns that resemble common real-world practices. These include text fragmentation, hidden or decoy elements, attribute perturbation, and mild content encoding (Sarabamoun, 2025). Perception perturbations modify how the underlying state is exposed to the agent, testing robustness under noisy or structurally misleading interface signals. The intensity of both visual and structural perturbations is parameterizable. Geometric distortion magnitude, noise density, and structural redundancy levels can be systematically adjusted to simulate different degrees of perceptual stress, enabling graded evaluation rather than a single fixed perturbation setting.

3.2.2 Semantic Perturbations

Semantic perturbations target the action selection stage of the interaction pipeline. At this stage, the agent selects actions based on learned assumptions about interface semantics, such as single-click for navigation or confirmation. In real-world web systems, however, interaction rules are often implicit or deviate from common conventions. When such deviations occur, agents that rigidly apply default action priors may systematically fail (Serim and Jacucci, 2019). To evaluate robustness at this stage, we construct semantic-perturbed environments that

modify the mapping between issued actions and their effective outcomes while keeping the observable interface unchanged. These perturbations require agents to infer altered interaction rules from contextual cues and revise default action assumptions accordingly.

Concretely, we instantiate this design using two semantic-perturbed environments. In both environments, selected single-click operations are remapped to require double-click execution. In the explicit environment (RemapE), the interface provides textual instructions indicating the modified interaction rule, requiring the agent to incorporate new semantic information into its decision process. In the implicit environment (Remap), no such instruction is provided; instead, a single click only produces a selection state, and the intended transition is triggered by a double click. The implicit environment therefore requires the agent to infer the altered rule through interaction feedback rather than relying on explicit guidance.

3.2.3 Execution Perturbations

Execution perturbations target the execution stage, where issued actions are realized and translated into state updates. In real-world web systems, action execution may be unreliable due to asynchronous processing, transient failures, or unexpected interface interruptions. As a result, agents cannot assume that every issued action immediately produces its intended effect. To evaluate robustness at this stage, we construct execution-perturbed environments in which action realization becomes unreliable. These perturbations require agents to verify execution outcomes and adapt their behavior when intended transitions do not occur.

Concretely, we design two execution-perturbed environments. In the first environment (Failure), actions may randomly fail to execute, including both click operations and text input submissions. In the second environment (Pop-Up), disruptive pop-up windows are introduced with varying prompts and interaction logic, requiring the agent to handle interruptions before continuing the intended task flow. Both mechanisms are parameterizable, allowing the failure rate and interruption frequency to be adjusted to simulate different levels of execution stress. This design enables systematic control over execution-stage variability while preserving the underlying task semantics.

3.3 Deterministic Evaluation

To ensure reliable and diagnostic assessment, we adopt a deterministic, script-based evaluation protocol that verifies agent behavior at multiple checkpoints throughout task execution. Rather than relying solely on final task completion, this design enables fine-grained attribution of failures to specific interaction stages.

For each task, we define a set of predefined checkpoints corresponding to critical milestones, such as successful navigation to a target page, correct completion of required form fields, or proper triggering of an intended action. After each agent action, the intermediate DOM state is recorded and automatically evaluated by task-specific checking scripts, which compare the state against expected conditions at that checkpoint. By grounding evaluation in observable states rather than agent-reported success signals, the protocol ensures consistent, reproducible, and stage-aware assessment, particularly under perturbation settings where agents may partially succeed before deviating.

4 Experiments

For completeness, we provide additional experimental results, detailed analyses, and failed pattern analysis in Appendix D, E, F, H, and I.

4.1 Experimental Setup

Models. We evaluate a diverse set of state-of-the-art multimodal and reasoning-capable LLM systems accessed via public APIs, including the Gemini family (Gemini-2.5-Pro), OpenAI GPT series (GPT-5.2, o4-mini), Anthropic Claude series (Claude-Sonnet-4.5, Claude-Opus-4.5, Claude-Haiku-4.5), GLM-V models (GLM-4.6V), and Qwen-VL models (Qwen3-VL-235B-A22B-Thinking). In addition, we include a human baseline to provide an upper-bound reference (conducted by computer science graduate students).

Run Protocol. All models interact with environments through a unified Playwright-based controller. At each step, the agent receives the current page screenshot and screenshots from the two most recent, the page DOM representation in HTML format, and the full history of previous actions and execution outcomes. Each run is limited to 100 interaction steps. A run terminates when the agent outputs DONE or FAIL, or when the step budget is exhausted. Task success is determined by task-specific evaluators that verify final page

Table 1: Overall Performance Across Different Environments. We report checkpoint pass rate (ckpt%) and average interaction steps (steps).

Model	Clean		Chaos		Noise		Failure		Pop-Up		RemapE		Remap		Avg	
	ckpt% ↑	steps ↓	ckpt% ↑	steps ↓	ckpt% ↑	steps ↓	ckpt% ↑	steps ↓	ckpt% ↑	steps ↓	ckpt% ↑	steps ↓	ckpt% ↑	steps ↓	ckpt% ↑	steps ↓
GLM-4.6V	46.7	61.0	38.3	59.2	42.1	61.6	40.1	59.3	38.0	66.6	20.8	89.1	20.5	88.4	35.2	69.3
Qwen3-VL-235B	52.7	60.0	44.9	61.2	49.2	58.8	43.4	62.3	40.9	68.5	26.5	84.8	24.8	92.0	40.3	69.7
Claude Haiku 4.5	43.0	79.8	38.5	77.3	40.9	83.1	39.7	83.1	38.5	85.8	26.0	95.0	24.1	97.7	34.6	87.0
Claude Opus 4.5	55.0	66.2	41.6	71.1	51.5	67.0	47.7	71.5	42.3	80.5	39.0	85.7	28.9	94.0	43.7	76.6
Claude Sonnet 4.5	57.5	64.4	45.7	74.6	52.4	67.9	47.5	72.7	45.2	74.6	31.6	89.8	25.4	90.1	43.6	76.3
Gemini-2.5-Pro	59.3	49.5	53.7	51.4	55.2	54.2	51.3	56.2	49.0	67.1	39.4	79.1	28.9	84.9	48.1	63.2
GPT-5.2	55.5	66.4	49.1	63.6	49.6	66.4	45.5	72.2	39.8	78.1	29.0	93.1	23.4	95.7	41.7	76.5
o4-mini	61.4	48.6	54.8	51.5	57.9	49.1	54.0	54.8	50.3	65.9	38.3	79.4	33.7	82.4	50.1	61.7
Human (Oracle)	100	7.0	100	7.0	100	7.0	100	10.3	100	9.7	100	7.8	100	7.7	100	8.1

states against predefined criteria. All models are evaluated under seven conditions (one clean setting and six perturbed variants) using the same task set and interaction protocol. In all reported experiments, perturbations are instantiated at a moderate stress level, with execution failure probability set to 0.35 and visual/structural distortions calibrated to intermediate severity. While this work focuses on a fixed stress configuration for controlled comparison, the benchmark supports parameterizable perturbation intensity.

4.2 Overall Performance under Environmental Stress

Table 1 summarizes the performance of representative multimodal models acting as web agents under clean and perturbed environments. Performance is measured by checkpoint success rate (ckpt%) and average interaction steps. Under the clean setting, agents achieve success rates between 46.7% and 61.4%. However, across perturbation settings, performance drops to 34.6%–50.1%, revealing a clear gap between capability and robustness. While agents perform well under stable conditions, they fail to maintain performance when interaction assumptions are perturbed, suggesting reliance on fragile environment-specific priors rather than adaptive strategies. Although o4-mini and Gemini-2.5-Pro achieve relatively higher average success rates (50.1% and 48.1%), all models exhibit substantial degradation, indicating that robustness limitations are systemic.

In contrast, the human oracle achieves near-perfect success across all environments with minimal increase in interaction steps. This suggests that the perturbations do not fundamentally increase task difficulty, but instead expose limitations in how agents perceive, reason, and adapt to changing environments. Therefore, the observed degradation reflects insufficient robustness and adaptability rather than intrinsic task complexity.

4.3 Perturbation Sensitivity Analysis

To understand the impact of environmental perturbations, Figure 3 reports checkpoint pass rates across models and perturbation modes. Environmental perturbations consistently degrade performance across all systems, indicating high sensitivity to deviations from expected interaction patterns. Compared with the clean environment, success rates decrease under every perturbation setting, and this trend is consistent across all models, suggesting that even moderate variability can undermine agent reliability. Different perturbation categories exhibit distinct levels of difficulty. Perception perturbations introduce relatively limited degradation, with average success decreasing from 54.3% to 49.8% under Noise and 45.8% under Chaos. Execution perturbations lead to moderately larger drops, to 46.1% under Failure and 43.0% under Pop-Up, indicating limited robustness to execution uncertainty and error recovery.

The most severe degradation occurs under semantic perturbations, where interaction rules deviate from common conventions. Performance drops sharply to 31.3% under RemapE and 26.2% under Remap. This suggests that agents rely on learned interaction priors rather than grounding actions in environment feedback. When these priors are invalidated, agents fail to adapt, even after repeated interactions. Notably, implicit remapping (Remap) causes more severe degradation than explicit remapping (RemapE), indicating difficulty in inferring updated interaction rules from context

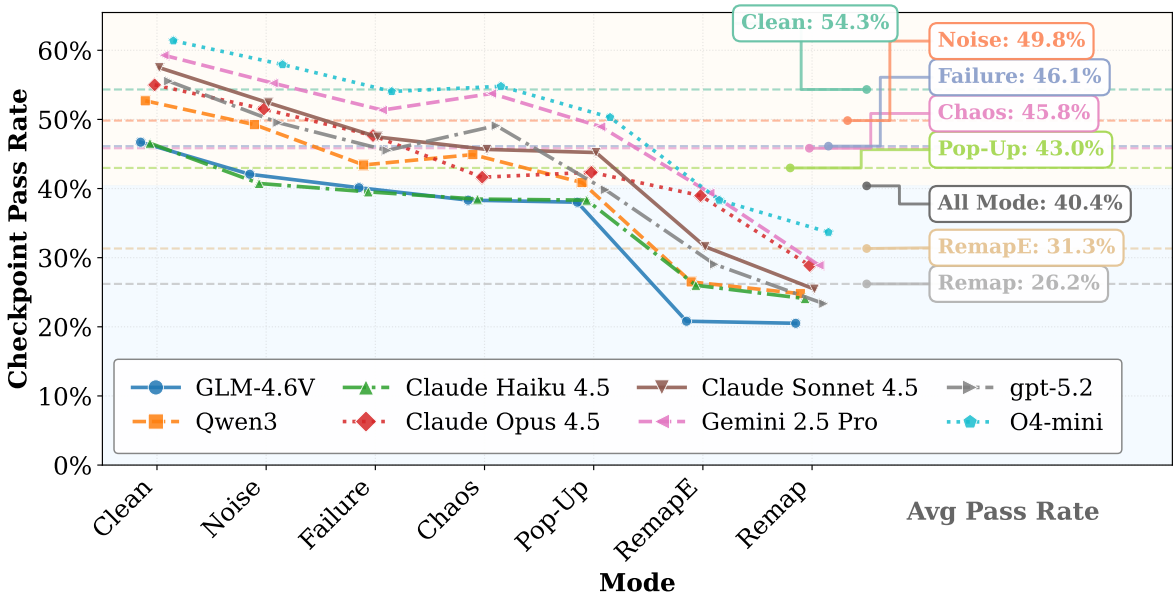


Figure 3: Checkpoint pass rates across different perturbation modes. Lines represent individual models and the dashed horizontal lines indicate the average performance for each mode across all models. Perception perturbations (Noise and Chaos) introduce relatively mild degradation, while execution perturbations (Failure and Pop-Up) lead to moderate drops. Semantic perturbations (RemapE and Remap), which alter interaction rules, cause the most severe performance decline.

and limited capability for online adaptation.

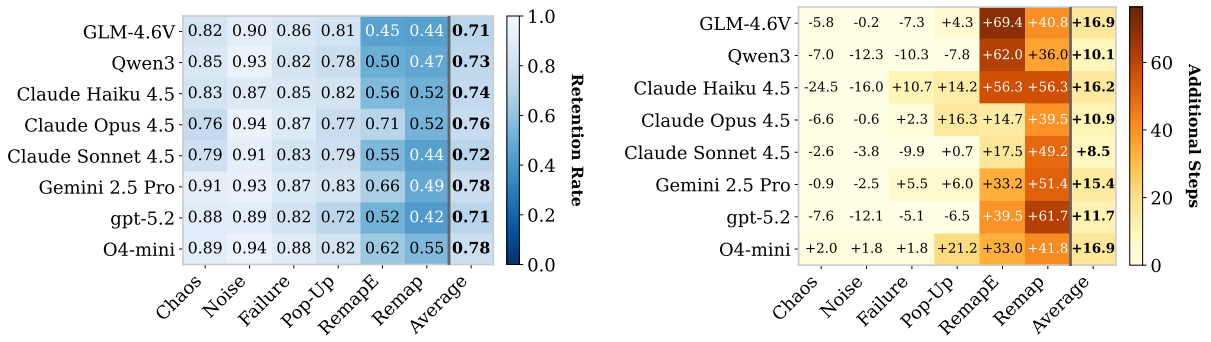
Overall, these results reveal a hierarchy of failure modes: agents are relatively robust to perceptual noise, moderately affected by execution uncertainty, and highly vulnerable to semantic shifts that violate interaction assumptions.

4.4 Performance and Interaction Cost Robustness

Figure 4a reports performance retention relative to the clean environment, while Figure 4b shows the additional interaction steps required under each perturbation. Across all models, semantic perturbations again produce the most severe degradation. When interaction rules are remapped (RemapE and Remap), performance retention drops dramatically. For example, GLM retains only 45% and 44% of its clean performance under RemapE and Remap, respectively, while GPT-5.2 drops to 52% and 42%. These results suggest that current web agents rely heavily on learned interaction conventions and struggle when action semantics change. In contrast, perception and execution perturbations have noticeably milder effects. Under Chaos, Noise, Failure, and Pop-Up conditions, most models retain a large portion of their clean performance, generally remaining close to or above 80%. Gemini-2.5-Pro and o4-mini show slightly stronger robustness in these settings, maintaining retention rates close to or above 0.85 across most non-semantic modes. Claude Opus 4.5 also demonstrates comparatively strong robustness under semantic remapping.

The step analysis provides a complementary perspective on interaction cost. While most perturbations increase the number of interaction steps, the largest increases occur under semantic perturbations. For instance, GPT-5.2 requires over 60 additional steps under Remap, and several other models exhibit increases exceeding 50 steps. Interestingly, performance degradation does not always correspond to longer interaction trajectories. We observe two distinct failure patterns. In some cases, agents terminate prematurely with fewer steps than in the clean environment, indicating early-stage failures caused by incorrect action assumptions. In other cases, agents exhibit substantially longer trajectories, repeatedly attempting invalid actions without adapting their strategies. For example, GLM and GPT-5.2 terminate earlier than in the clean environment, suggesting that agents commit to incorrect action hypotheses early and fail to recover.

Importantly, semantic perturbations not only reduce success rates but also substantially increase interaction cost, indicating that agents fail both to identify correct actions and to revise incorrect ones. This dual failure highlights a key limitation: cur-



(a) Performance Robustness

(b) Step Difference

Figure 4: Performance and cost robustness under environmental perturbations. (a) Performance retention relative to the clean environment. (b) Additional interaction steps compared to the clean setting. Darker colors indicate stronger degradation or higher cost.

rent agents lack mechanisms for online adaptation, leading to both incorrect decisions and inefficient exploration under changing interaction semantics.

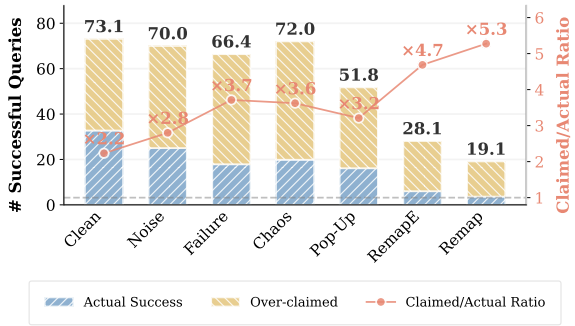


Figure 5: Claimed vs. actual successful queries under different modes. Environmental perturbations increase the discrepancy between perceived and actual task success, indicating reduced self-assessment reliability.

4.5 Self-Assessment Reliability under Perturbations

Beyond task success, it is critical to examine whether agents can reliably assess their own outcomes. In many web-agent frameworks, the model terminates an interaction by declaring task completion. If this self-assessment is unreliable, agents may prematurely stop even when the task has not been correctly executed. Figure 5 compares the number of actually successful queries with those claimed as successful by the agent. Even in the clean environment, agents exhibit noticeable miscalibration between claimed and actual task success. Perturbations substantially amplify this miscalibration, leading to systematic overestimation of task success. This suggests that agents lack reliable internal signals to detect failure under uncertain conditions. The effect becomes particularly pronounced under semantic perturbations. While the number of truly successful queries drops sharply under RemapE and Remap conditions, agents continue to report comparatively high success rates. As a result, the claimed-to-actual success ratio increases significantly, reaching up to 5.3×, indicating severe overconfidence. This behavior is consistent with the previously observed failure to adapt action strategies, indicating that agents not only make incorrect decisions but also fail to recognize these errors. These results indicate that environmental shifts impair not only task execution but also agents' ability to accurately evaluate their own performance. Such miscalibration poses a critical challenge for real-world deployment, as agents may prematurely terminate tasks while confidently reporting success, leading to silent failures.

5 Conclusion

We introduced StressWeb, a diagnostic benchmark for evaluating the robustness of web agents under realistic interaction variability. By constructing controllable web environments with structured perturbations across perception, action selection, and execution stages, StressWeb enables systematic analysis of agent behavior under environmental stress. Experiments on state-of-the-art multimodal agents reveal substantial robustness gaps that remain hidden under clean benchmark conditions. In particular, semantic perturbations that alter interaction rules cause the most severe performance degradation, while environmental stress also significantly impairs agents' ability to correctly assess their own task success.

Limitations

Despite enabling controlled robustness evaluation of web agents, StressWeb has several limitations. First, although the benchmark includes representative websites from multiple task domains, the overall coverage of website types remains limited compared with the diversity of real-world web ecosystems. Expanding the benchmark to additional websites and task categories would further improve its representativeness. Second, the current perturbation mechanisms are primarily applied individually to isolate their effects. In practice, multiple forms of environmental variability may occur simultaneously. Studying compositional perturbations and their combined effects remains an important direction for future work. Third, the perturbations used in StressWeb are controlled abstractions designed to approximate realistic web variability. While this design enables systematic analysis, real-world web environments may exhibit more complex and dynamic behaviors that are not fully captured in the current benchmark.

References

Sagnik Anupam, Davis Brown, Shuo Li, Eric Wong, Hamed Hassani, and Osbert Bastani. 2025. Browserarena: Evaluating llm agents on real-world web navigation tasks. *arXiv preprint arXiv:2510.02418*.

William Brach, Matej Petrik, Kristián Košt'ál, and Michal Ries. 2025. Ghosts in the markup: Techniques to fight large language model-powered web scrapers. In *2025 37th Conference of Open Innovations Association (FRUCT)*, pages 37–46. IEEE.

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

Matthew Edgar. 2024. Cumulative layout shift. In *Speed Metrics Guide: Choosing the Right Metrics to Use When Evaluating Websites*, pages 157–180. Springer.

Divyansh Garg, Shaun VanWeelden, Diego Caples, Andis Draguns, Nikil Ravi, Pranav Putta, Naman Garg, Tomas Abraham, Michael Lara, Federico Lopez, and 1 others. 2025. Real: Benchmarking autonomous agents on deterministic simulations of real websites. *arXiv preprint arXiv:2504.11543*.

Linqiang Guo, Wei Liu, Yi Wen Heng, Yang Wang, and 1 others. 2024. Popsweeper: Automatically detecting and resolving app-blocking pop-ups to assist automated mobile gui testing. *arXiv preprint arXiv:2412.02933*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Su Kara, Fazle Faisal, and Suman Nath. 2025. Warex: Web agent reliability evaluation on existing benchmarks. *arXiv preprint arXiv:2510.03285*.

Shilong Li, Xingyuan Bu, Wenjie Wang, Jiaheng Liu, Jun Dong, Haoyang He, Hao Lu, Haozhe Zhang, Chenchen Jing, Zhen Li, and 1 others. 2025. Mm-browsecomp: A comprehensive benchmark for multimodal browsing agents. *arXiv preprint arXiv:2508.13186*.

Yougang Lyu, Xiaoyu Zhang, Lingyong Yan, Maarten de Rijke, Zhaochun Ren, and Xiuying Chen. 2025. Deepshop: A benchmark for deep research shopping agents. *arXiv preprint arXiv:2506.02839*.

Mykyta Mudryi, Markiyan Chaklosh, and Grzegorz WĂłjcik. 2025. The hidden dangers of browsing ai agents. *arXiv preprint arXiv:2505.13076*.

Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao-yong Wei, Shanru Lin, Hui Liu, Philip S Yu, and 1 others. 2025. A survey of webagents: Towards next-generation ai agents

for web automation with large foundation models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6140–6150.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and 1 others. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Ralph Peeters, Aaron Steiner, Luca Schwarz, Julian Yuya Caspary, and Christian Bizer. 2025. Webmall–a multi-shop benchmark for evaluating web agents. *arXiv e-prints*, pages arXiv–2508.

Yu Pei, Jeongju Sohn, Sarra Habchi, and Mike Papadakis. 2025. Non-flaky and nearly optimal time-based treatment of asynchronous wait web tests. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–29.

Yu Pei, Jeongju Sohn, and Mike Papadakis. 2026. An empirical study of web visual flakiness: Characterisation and fix strategies. *Journal of Systems and Software*, page 112826.

Ephraiem Sarabamoun. 2025. Special-character adversarial attacks on open-source language model. *arXiv preprint arXiv:2508.14070*.

Barış Serim and Giulio Jacucci. 2019. Explicating" implicit interaction" an examination of the concept and challenges for research. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–16.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025a. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, and 1 others. 2025b. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*.

Haolong Yan, Yeqing Shen, Xin Huang, Jia Wang, Kaijun Tan, Zhixuan Liang, Hongxin Li, Zheng Ge, Osamu Yoshie, Si Li, and 1 others. 2025. Gui exploration lab: Enhancing screen navigation in agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2512.02423*.

Xiao Yang, Jiawei Chen, Jun Luo, Zhengwei Fang, Yinpeng Dong, Hang Su, and Jun Zhu. 2025. Mlatrust: Benchmarking trustworthiness of multimodal llm agents in gui environments. *arXiv preprint arXiv:2506.01616*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Comparison with Existing Web Agent Benchmarks

To provide a broader perspective on the design space of web agent benchmarks, Table 2 summarizes key properties of representative benchmarks and evaluation frameworks.

The comparison highlights several important dimensions that affect the evaluation of web agents. First, actionable benchmarks require agents to perform structured UI operations such as clicking, typing, and multi-step task workflows, rather than only browsing or reading web pages. Second, realistic environments preserve the fidelity of real-world web interfaces. Third, deterministic environments ensure reproducible execution across runs. Fourth, diagnostic benchmarks provide mechanisms for attributing agent failures to specific components. Finally, intervention environments support controlled perturbations of environment dynamics, enabling causal analysis of agent robustness.

As shown in Table 2, existing benchmarks typically emphasize only a subset of these properties. Interactive benchmarks such as WebArena and WebVoyager prioritize realism but lack deterministic execution and diagnostic capabilities. Browsing-oriented benchmarks such as BrowseComp and MM-BrowseComp focus primarily on information retrieval and do not require structured UI interaction. Simulation-based environments such as REAL and WebCanvas improve reproducibility but provide limited mechanisms for systematic intervention.

In contrast, our benchmark is designed to jointly support actionable interaction, realistic environments, deterministic execution, diagnostic analysis,

Table 2: Comparison of representative web agent benchmarks and evaluation frameworks. A capability is marked as supported only if it is explicitly designed and systematically supported. **Actionable** indicates whether the benchmark requires agents to perform structured UI operations such as clicking, typing, and multi-step task workflows, rather than only browsing or reading web pages. **Realistic** denotes whether the environment preserves the fidelity of real-world web interfaces. **Deterministic** indicates whether task execution is reproducible across runs. **Diagnostic** denotes built-in support for attributing agent failures to specific components. **Interventional** indicates whether the benchmark supports controlled and reproducible environment perturbations for robustness evaluation.

Benchmark	Actionable	Realistic	Deterministic	Diagnostic	Interventional
World of Bits	✓	✗	✓	✗	✗
WebShop	✓	✗	✓	✗	✗
Mind2Web	✓	✓	✗	✗	✗
WebArena	✓	✓	✓	✗	✗
WebVoyager	✓	✓	✗	✗	✗
BrowserArena	✓	✓	✗	✓	✗
BrowseComp	✗	✓	✗	✗	✗
MM-BrowseComp	✗	✓	✗	✗	✗
DeepShop	✗	✓	✗	✗	✗
WebMall	✓	✓	✓	✗	✗
REAL	✓	✓	✓	✗	✗
WebCanvas	✓	✓	✓	✗	✗
Ours	✓	✓	✓	✓	✓

and controlled environmental perturbations. This combination enables systematic evaluation of web agent robustness under controlled conditions.

B Task-Driven Website Construction

To support controlled and reproducible evaluation of web agents, we construct benchmark environments using a task-driven website generation methodology. Rather than starting from pre-existing websites and retrofitting evaluation tasks, our approach inverts this process: tasks are specified first, and websites are then constructed to faithfully support those tasks. This design choice ensures that navigation structure, interaction semantics, and internal state transitions are explicitly aligned with the evaluation objectives.

B.1 Task Decomposition and Specification

We begin by decomposing each benchmark scenario into a set of concrete, executable tasks. Tasks are derived from high-level product requirements and are designed to reflect realistic user goals, such as completing a purchase, managing a document, scheduling events, or updating persistent records. Importantly, all tasks are specified with explicit inputs and observable outcomes, avoiding underspecified or purely informational queries.

Each task is further decomposed into a sequence of interaction steps spanning multiple pages. This decomposition makes explicit the required navigation paths, intermediate state changes, and final success conditions. By grounding task definitions in step-level interaction structure, we ensure that task difficulty arises from environment behavior rather than ambiguity in task formulation.

B.2 Task-Driven Website Construction

Given a set of decomposed tasks, we construct web environments that are explicitly designed to support their execution. Each website is generated to include the pages, routes, data entities, and interaction affordances required by the task set. This task-driven construction ensures that page transitions, form submissions, and state updates follow coherent and deterministic logic, reducing spurious failures caused by missing links, inconsistent routing, or incomplete state propagation.

Because websites are generated *after* task specification, the resulting environments naturally encode valid navigation paths and interaction flows for all tasks. This contrasts with approaches that define tasks post hoc on existing websites, where agents may encounter undefined behavior due to incomplete coverage of the website's interaction space. In our setting, every task has a well-defined execution path by construction.

B.3 Structured Task Trajectories

To further bridge the gap between task specification and environment implementation, we represent each task as a structured trajectory. A trajectory specifies the expected sequence of pages, user ac-

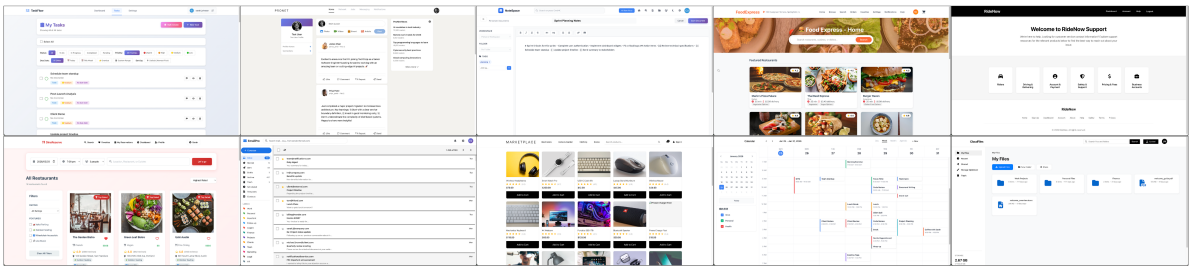


Figure 6: Example screenshots of the generated web environments used in StressWeb. The benchmark includes ten realistic website categories covering diverse domains such as e-commerce, scheduling, email management, transportation, and cloud file systems. Each environment supports multi-step interactions, persistent state updates, and common web operations including search, filtering, navigation, and form submission.

tions, and environment state transitions required for successful task completion. These trajectories act as an intermediate representation connecting high-level task intent with low-level website behavior.

Structured trajectories serve two purposes. First, they guide website construction by making explicit which pages, components, and backend operations must be implemented. Second, they provide a reference specification for evaluation, enabling deterministic verification of intermediate and final outcomes. This dual role ensures consistency between task design, website implementation, and evaluation criteria.

B.4 From Trajectories to Executable Websites

Based on the structured task trajectories, we generate fully functional websites that implement the required frontend interfaces, backend logic, and persistent storage. The generated websites support realistic interaction workflows, including multi-page navigation, form-based input, and stateful operations. While the generation process is automated, it is constrained by the task specifications and trajectories, which act as a blueprint for correct behavior.

This process results in websites that are not merely visually plausible, but behaviorally aligned with the benchmark tasks. As a consequence, agents evaluated in these environments are tested on robustness to environmental perturbations rather than on accidental inconsistencies or missing functionality.

B.5 Human Verification and Refinement

Although the construction process is largely automated, we apply a lightweight human verification and refinement step to ensure suitability for benchmark evaluation. This step focuses on verifying functional correctness of task workflows, clarity of interaction semantics, and consistency between task specifications and website behavior. When discrepancies are identified—such as ambiguous UI cues, incomplete state updates, or minor inconsistencies in navigation—we apply minimal corrective edits.

Crucially, this refinement process is constrained to alignment and correctness fixes and does not introduce new functionality beyond the task-driven design. As a result, the final environments remain faithful to the automated generation process while meeting the determinism and clarity requirements of reliable agent evaluation.

C Generated Website Environments

Figure 6 presents example screenshots of the ten generated websites used in the benchmark. All environments are produced through the agent-driven website construction pipeline described in the main paper, which converts natural language specifications into fully functional web applications. Unlike simplified benchmark interfaces, the generated websites maintain realistic layouts, multi-page navigation structures, and persistent internal states across interactions. This design ensures that agent behaviors are evaluated within interaction workflows that closely resemble real-world web applications.

To cover a diverse range of realistic scenarios, the benchmark includes ten representative categories of web services spanning common online activities such as shopping, communication, scheduling, productivity, and transportation. These environments include an e-commerce platform, food delivery service, restaurant reservation system, online notebook editor, calendar scheduling interface, email management system, list management application, professional networking platform, ride-

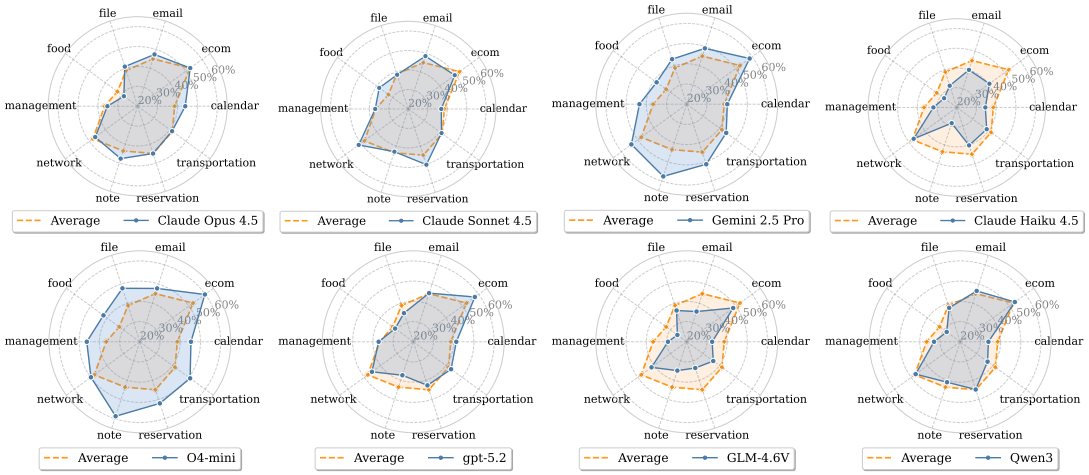


Figure 7: Radar plots showing model performance across different websites. Each radar plot contains two curves: the performance of the specific model and the average performance across all models.

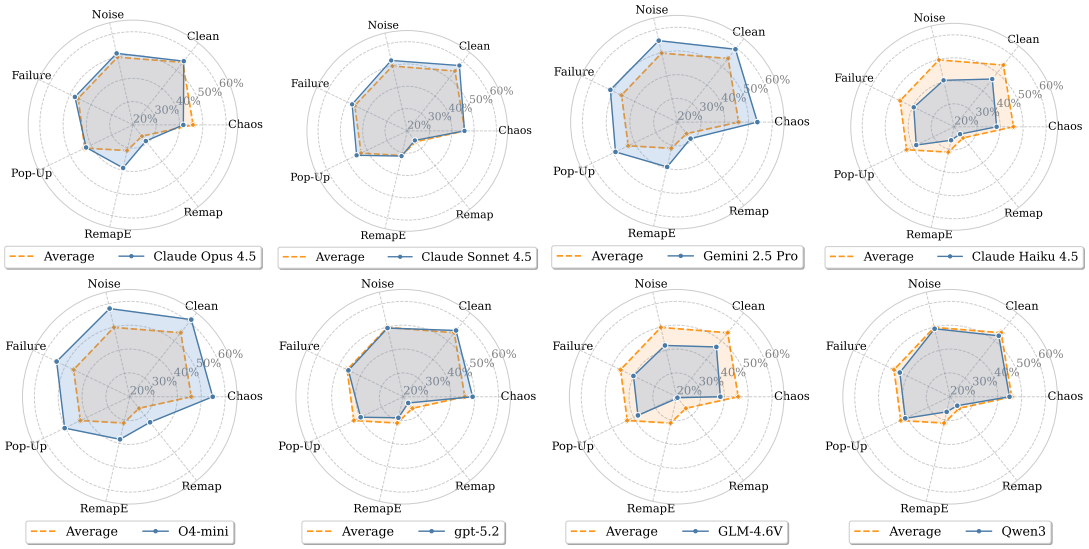


Figure 8: Radar plots showing model performance across different perturbation modes. Each plot compares the performance of an individual model against the average performance across all evaluated models.

hailing service, and cloud file management system. Together, these domains represent widely used classes of modern web applications and provide diverse interaction patterns for evaluating web agents.

Across these environments, the websites support a broad spectrum of common web interaction primitives. Users can navigate across multiple pages, perform keyword searches, apply filters and sorting options, interact with forms, submit multi-step requests, and update persistent data structures such as carts, reservations, schedules, messages, or file hierarchies. Many tasks also require multi-step workflows that involve intermediate state updates before a final confirmation step. As a result, agents must reason over both interface structure and evolving environment state in order to complete tasks successfully.

D Detailed Model Performance Analysis

This section provides detailed breakdowns of model performance across different websites and perturbation modes. To facilitate comparison, we report per-model radar plots showing checkpoint pass rates under all evaluation conditions. Each radar plot contains two curves: (1) the performance of the specific model and (2) the average performance across all evaluated models. The average curve serves as a reference baseline, allowing deviations in model-specific strengths and weaknesses to be visually identified. We provide two complementary views of the results. First, we visualize

performance across different websites, highlighting how models behave under different task domains. Second, we present performance across perturbation modes, revealing how each model responds to different categories of environmental disruptions.

D.1 Performance Across Websites

Figure 7 presents radar plots showing model performance across different websites. Each plot corresponds to one evaluated model and reports the checkpoint pass rate on all task websites. The model-specific curve is shown together with the average performance across all models, providing a direct reference for identifying relative strengths and weaknesses.

D.2 Performance Across Perturbation Modes

Figure 8 visualizes model performance across different perturbation modes.

Each radar plot shows checkpoint pass rates under all evaluation modes, including the clean environment and various perturbation settings. By comparing the model-specific curve with the average baseline, it becomes easier to identify models that exhibit stronger robustness or higher sensitivity under particular perturbation types.

E Model-Level Self-Assessment Reliability

To complement the aggregate analysis presented in Section 4.5, we further examine self-assessment reliability at the model level.

Figure 9 reports the relationship between claimed task success and actual task success for each evaluated model under all evaluation conditions. Each plot corresponds to one model and shows how the number of queries claimed as successful compares with the number of queries that are actually successful.

This model-level view reveals that the discrepancy between claimed and actual success is not limited to a small subset of models but appears consistently across most evaluated systems. While the magnitude of the gap varies across models, environmental perturbations generally lead to systematic overestimation of task success. In particular, semantic perturbations often produce the largest divergence, indicating that agents frequently fail to recognize execution errors when interaction semantics are altered.

F Model Ranking Under Clean and Perturbed Conditions

To further understand how environmental perturbations affect model performance, we analyze model rankings under clean and perturbed conditions.

Figure 10a first reports checkpoint pass rates across different perturbation modes for all evaluated models. Each group corresponds to one evaluation mode, while bars represent different models.

We then compare model rankings between the clean environment and perturbed environments. Figure 10b shows checkpoint pass rates under the clean condition. Figure 10c reports the average checkpoint pass rate across all perturbation modes.

To facilitate comparison, the same color is used for each model across both figures. This allows ranking differences between clean and perturbed settings to be visually identified.

These comparisons reveal that model rankings under the clean environment do not always translate to robustness under perturbations. Some models that perform strongly in the clean setting exhibit larger performance degradation when environmental conditions change, while others maintain more stable behavior across perturbation modes.

This observation highlights the importance of evaluating web agents beyond clean environments, as robustness to environmental variation represents a distinct capability dimension that is not fully captured by standard clean benchmarks.

G Agent Prompt Template

The following prompt template is used to instruct the agent to interact with web environments using Playwright. The prompt specifies the available actions, output format, and interaction constraints. Depending on the configuration, the agent may operate either with DOM information or purely from screenshots.

For brevity, the list of supported actions returned by `get_action_description()` is abbreviated here.

Web Agent Prompt Template

```
You are a web GUI automation agent that interacts with webpages
using Playwright.

Available actions include operations such as:
CLICK, TYPE, FILL, HOTKEY, WAIT, DONE, and FAIL.

Mode: <mode_name>

--------------------------------------------------
```

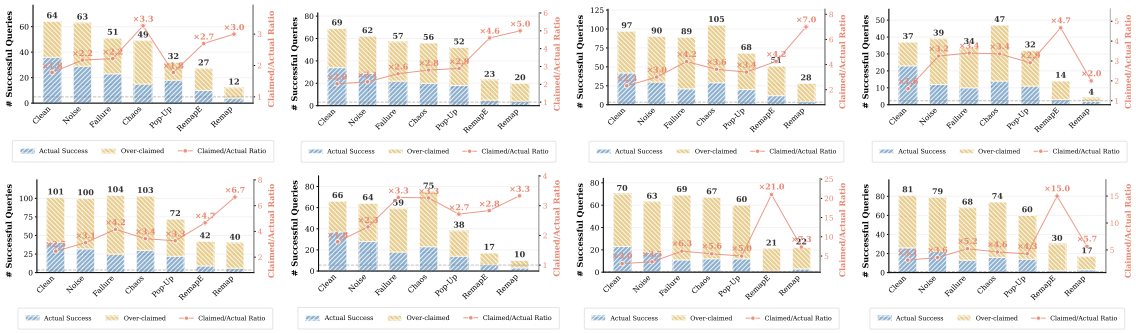
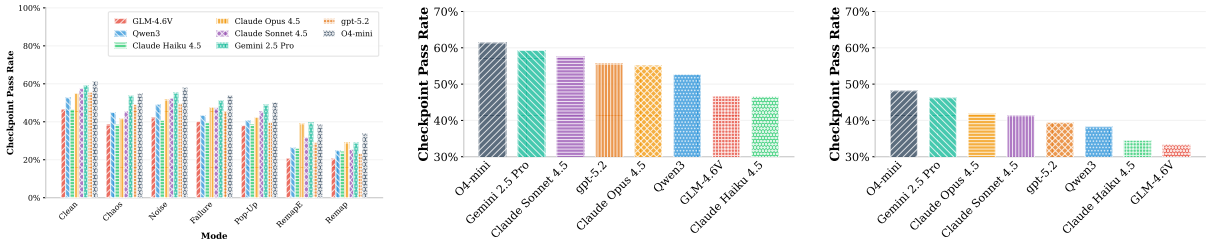


Figure 9: Model-level self-assessment reliability across different evaluation conditions. Each plot compares the number of queries claimed as successful by the agent with the number of queries that are actually successful. The gap between the two quantities reflects the degree of self-assessment error.



(a) Performance across perturbation modes.

(b) Model performance under the clean environment.

(c) Average performance under perturbed environments.

Figure 10: Model performance under clean and perturbed environments. (a) Checkpoint pass rates across different perturbation modes. (b) Model ranking under the clean environment. (c) Average model performance across all perturbed environments. Colors are consistent across (b) and (c), allowing ranking differences between clean and perturbed conditions to be visually identified.

```
CLICK actions

When DOM information is available, prefer CSS selectors.

Selector strategy (recommended order):

1. Text-based selectors
   button:has-text("Text")
   a:has-text("Link")
   text="Exact Text"

2. Element + class
   button.primary
   div.card
   input.search-input

3. ID selector
   #element-id

4. Attribute selector
   [placeholder="Search"]
   [type="submit"]

If selector-based interaction repeatedly fails,
visual coordinates may be used as a fallback.

Selectors should remain simple and avoid deep nesting.

--------------------------------------------------

Output format

All responses must be valid JSON objects.

Example:

{
  "action_type": "CLICK",
```

```
  "parameters": {
    "selector": "button:has-text('Submit')"
  },
  "reasoning": "Click the submit button"
}

Coordinate fallback example:

{
  "action_type": "CLICK",
  "parameters": {
    "x": 200,
    "y": 250
  },
  "reasoning": "Click based on visual location"
}

--------------------------------------------------

Other actions

FILL

{
  "action_type": "FILL",
  "parameters": {
    "selector": "input[placeholder='Title']",
    "text": "example text"
  }
}

TYPE

{
  "action_type": "TYPE",
  "parameters": {
    "text": "hello"
  }
```

```
}

HOTKEY

{
  "action_type": "HOTKEY",
  "parameters": {
    "keys": "Ctrl+A"
  }
}

--------------------------------------------------

Important rules

- Always return a valid JSON object.
- Never output plain text explanations outside JSON.
- Coordinates are relative to the viewport.
- Click the center of elements whenever possible.
- If an action fails, try an alternative selector or
adjust coordinates.
- Use DONE when the task is completed successfully.
- Use FAIL only when the task cannot be completed.
```

H Discussion

H.1 Robustness as a Missing Dimension of Agent Capability

Our results reveal a consistent gap between task capability and robustness in current web agents. While modern multimodal models can achieve strong performance under clean and stable environments, their performance degrades substantially under even moderate perturbations. This suggests that high benchmark performance does not necessarily imply reliable behavior in realistic, dynamic environments. Instead, robustness should be considered a first-class evaluation dimension alongside task success.

H.2 Failure to Adapt under Semantic Shifts

A key finding of our analysis is that semantic perturbations induce disproportionately large performance degradation. When interaction rules deviate from learned conventions, agents fail to adjust their behavior, even after repeated interaction attempts. This indicates that current agents rely heavily on fixed interaction priors rather than dynamically grounding their actions in environment feedback. The lack of online adaptation leads to both incorrect decisions and inefficient exploration, as agents repeatedly execute invalid actions without revising their strategies.

H.3 Failure Patterns: Early Collapse vs. Inefficient Exploration

By analyzing interaction steps, we identify two distinct failure patterns. In some cases, agents terminate prematurely with fewer steps, suggesting early-stage failures caused by incorrect action assumptions. In other cases, agents exhibit significantly longer interaction trajectories, indicating inefficient exploration without effective error correction. These patterns highlight that failures are not solely due to increased task difficulty, but arise from systematic limitations in decision-making and adaptation mechanisms.

H.4 Miscalibration and Overconfidence under Uncertainty

Beyond execution failures, we observe that agents exhibit substantial miscalibration between claimed and actual success, which becomes more severe under environmental perturbations. In particular, semantic perturbations lead to extreme overconfidence, where agents report successful task completion despite failing to achieve the intended outcome. This suggests that current agents lack reliable internal signals for failure detection, and are unable to accurately assess their own performance under uncertainty.

H.5 Implications for Real-World Deployment

These findings have important implications for deploying web agents in real-world settings. First, robustness to environmental variability is critical, as real-world interfaces are inherently dynamic and often deviate from training distributions. Second, the lack of adaptation and self-assessment reliability may lead to silent failures, where agents terminate tasks prematurely while confidently reporting success. Such behavior can undermine user trust and lead to downstream system errors. Therefore, improving robustness, online adaptation, and self-assessment calibration should be key directions for future research in web agents and multimodal systems.

I Repetitive Action Failure under Semantic Perturbations.

We further analyze repetitive action patterns to understand agent behavior under perturbations. Table 3 reports the proportion of trajectories containing consecutive identical actions, the total number of repeated operations, and the maximum repetition length.

Repetitive behavior is already present in the clean environment, with 58.4% of trajectories containing exact repeated actions. However, semantic perturbations dramatically amplify this effect. Under RemapE and Remap, over 96% of trajectories

Table 3: Repetitive action behavior under different perturbation settings. Exact Repeat (%) denotes the percentage of trajectories containing consecutive identical actions (same action type and parameters). Semantic perturbations lead to significantly higher repetition frequency and longer repeated sequences.

Setting	Exact Repeat (%)	Total Repeats	Max Repeat
Clean	58.4	1,599	54
Chaos	72.5	1,659	44
Failure	83.2	2,462	28
Perturbed	69.1	1,442	32
DOM	61.7	1,920	64
RemapE	96.6	4,588	65
Remap	96.0	4,874	92

exhibit repeated actions, indicating that such behavior becomes nearly universal. In addition, the total number of repeated operations increases by nearly three times compared to the clean setting, revealing a substantial escalation in repetition frequency.

More importantly, semantic perturbations induce extremely long repetition sequences. The maximum repetition length reaches up to 92 consecutive identical actions, suggesting that agents can become trapped in persistent action loops. Such behavior is rarely observed in the clean environment and is significantly less severe under other perturbation types.

These results indicate that semantic perturbations trigger a distinctive failure mode in which agents repeatedly execute the same action without adapting to environment feedback. This phenomenon reflects a form of policy collapse, where agents fail to revise incorrect action hypotheses and instead remain stuck in ineffective interaction patterns.