SAME WEBSITE, DIFFERENT IMPLEMENTATIONS: RETHINKING ROBUSTNESS OF WEB AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Web agents are increasingly capable of completing web tasks, but the diversity of real-world websites remains a major challenge to reliable deployment beyond controlled environments. One important but underexplored factor is website implementation: different implementations of the same functionality can provide similar user experiences while exposing agents to substantially different structural, textual, and visual signals. Existing evaluations do not account for such variation, making it unclear whether an agent can preserve its task-solving capability across functionally equivalent implementations. We introduce WEBVARIANTS, a controlled evaluation framework with nine websites, five implementations each, and 228 tasks shared across implementations, yielding 1,140 task–environment instances. Extensive evaluation reveals substantial sensitivity to implementation choice. Text- and screenshot-based agents exhibit different vulnerabilities, while richer multimodal observations do not consistently improve robustness across implementations. Implementation also affects whether agents recognize task completion after reaching a successful state. The resulting performance variation substantially alters agent rankings, with up to 50% of pairwise orderings reversing across implementations. We further investigate how to improve robustness across implementations. Training on a single implementation generalizes well to other implementations of websites seen during training, but its gains become less consistent on unseen websites. Under a matched training budget, training across multiple implementations improves robustness on unseen websites. Within this budget, covering more unique tasks is more effective than exposing fewer tasks to multiple implementations. Our results establish website implementation as an important dimension of both evaluation and training for robust web agents. [1]

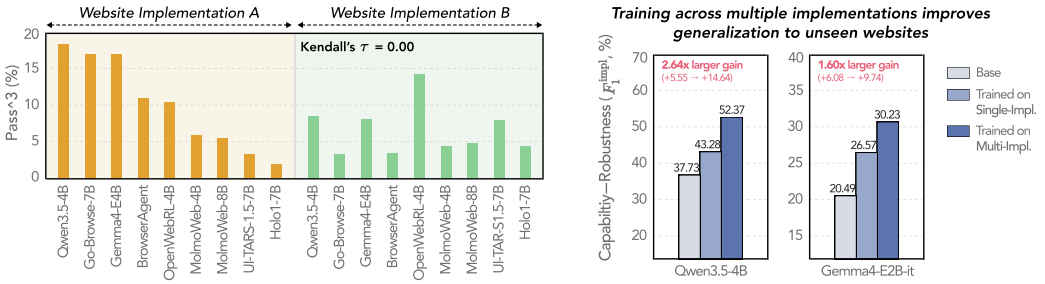


Figure 1: **Website implementation matters for both evaluation and training.** (*Left*) Performance of nine 4B–8B agents on the same tasks under two functionally equivalent website implementations, yielding *no* correlation between the resulting agent rankings (*i.e.*, Kendall's $\tau = 0.00$). (*Right*) Training on diverse implementations improves generalization to implementations of unseen websites.

1 INTRODUCTION

Autonomous web agents powered by large language models (LLMs) and vision-language models (VLMs) are increasingly capable of completing complex tasks, with steadily improving performance on web benchmarks (Gandhi & Neubig, 2025; Gupta et al., 2026; Yang et al., 2026). Yet strong performance in controlled environments does not necessarily translate into reliable real-world deployment. In practice, the same website functionality can be realized through different underlying

[1] Code and resources are available at https://anonymous.4open.science/r/WebVariants-C74E

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

implementations, differing in HTML structure, accessibility metadata, visual presentation, and interaction design. Such differences may have little effect on how a human interacts with a website or completes a task, but substantially alter the textual and visual observations available to a web agent. A web agent intended for real-world use should therefore preserve its task-solving capability across such implementation variation. Otherwise, apparent task competence may hinge on implementation-specific cues and disappear when the same functionality is presented differently.

This raises a fundamental question: to what extent do current web agents preserve their task-solving capability when functionality remains fixed but the website implementation changes? Existing evaluations provide limited evidence for answering this question because each task is typically evaluated under a single implementation, making it difficult to isolate the effect of implementation variation (Zhou et al., 2023; Deng et al., 2023; Xue et al., 2025; Drouin et al., 2024). Consequently, measured performance can conflate an agent's underlying task-solving capability with its dependence on implementation-specific cues. We therefore isolate website implementation as an experimental variable and examine how agent performance changes while holding the task, website functionality, and user experience fixed.

We first introduce WEBVARIANTS, a controlled evaluation framework that isolates variation in website implementation while holding the task, website functionality, and user-facing experience fixed. Starting from nine audited websites, we construct five implementations of each website through controlled changes to HTML semantics, accessibility metadata, textual labeling, and visual affordances. WEBVARIANTS contains 228 tasks that remain executable across all five implementations. Each task–implementation instance is evaluated using the same backend success criteria, yielding 1,140 task–environment instances in total.

Extensive experiments reveal substantial sensitivity to implementation choice. Figure 1 (left) illustrates its impact on evaluation: the same nine 4B–8B agents produce markedly different rankings under two functionally equivalent implementations, with Kendall's $\tau = 0.00$. OpenWebRL-4B (Yang et al., 2026), for example, moves from fifth to first. Across all implementations, agents that rely on accessibility tree observations are particularly affected when structural and accessibility cues are weakened, whereas screenshot-based agents are more sensitive to changes in textual labels and visual affordances. Providing richer observations does not eliminate this sensitivity. Adding screenshots or explicit Set-of-Marks (SoM) grounding (Yang et al., 2023) improves performance under some implementations but degrades it under others, with no observation interface consistently improving performance across implementations. Implementation also affects how accurately agents interpret the current web state. Specifically, the rate at which agents fail to terminate after reaching a successful state varies substantially across implementations, indicating that recognition of task completion is sensitive to implementation. These results identify website implementation as a distinct axis of web agent evaluation that can affect measured performance and relative agent rankings.

A natural next question is how to train web agents to remain robust to implementation variation. Training on a single implementation generalizes surprisingly well to unseen implementations of training websites, but these gains become less consistent on unseen websites and can even turn into performance degradation. Figure 1 (right) shows that training across multiple implementations improves generalization to implementations of unseen websites for both models, outperforming single-implementation training under the same budget. We further study how a fixed training budget should be allocated for generalization to unseen websites by comparing repeated exposure of the same tasks across multiple implementations with broader task coverage, using one of several implementations per task. The results show that broader task coverage is more effective than allocating multiple implementations to fewer tasks. These findings suggest that implementation diversity should be treated as an explicit dimension of training when developing robust web agents.

2 RELATED WORK

Web Agent Benchmarks and Evaluation. Web agent evaluation has progressed from simplified simulated environments such as MiniWoB++ (Shi et al., 2017) and WebShop (Yao et al., 2023) to more realistic settings. WebArena (Zhou et al., 2023), VisualWebArena (Koh et al., 2024), and WorkArena (Drouin et al., 2024) provide reproducible functional environments for long-horizon or enterprise tasks, while REAL (Garg et al., 2026) uses high-fidelity replicas of widely used websites. Other benchmarks, including WebVoyager (He et al., 2024) and Online-Mind2Web (Xue et al.,

2025), evaluate agents through direct interaction with real websites. A complementary line of work studies robustness under perturbed interaction conditions. StressWeb (Bai et al., 2026) introduces visual and DOM distortions, altered interaction semantics, action failures, and disruptive pop-ups. While such perturbations expose failures under adverse conditions, they can substantially alter the user-facing experience. In contrast, this work studies implementation differences that can arise from ordinary website construction while largely preserving functionality and user experience, enabling controlled evaluation of robustness to implementation variation found in real-world websites.

Observation Interfaces for Web Agents. Existing Web agents observe webpages through structured text derived from HTML, the DOM, or accessibility trees (Lai et al., 2024; Xiao et al., 2026; Chae et al., 2026), rendered screenshots (Cheng et al., 2024; Andreux et al., 2025; Gupta et al., 2026), or combinations of both (Chen et al., 2024; Zheng et al., 2024a; He et al., 2024). These interfaces expose different cues: structured text provides element semantics and accessibility metadata, whereas screenshots preserve visible labels, icons, layout, and styling. Prior work has focused on improving reasoning and grounding within a given interface (Zheng et al., 2024a; Cheng et al., 2024) or combining modalities to improve task performance (He et al., 2024; Koh et al., 2024). In contrast, we examine how changes in website implementation, while preserving functionality, affect agent performance across observation interfaces.

3 Controlled Evaluation of Website Implementation Robustness

3.1 Problem Setting

Why Website Implementation Matters. Web agents encounter substantial implementation variation across websites that provide the same or similar functionality. A deployable web agent should therefore preserve its task-solving capability across such variation rather than depend on incidental implementation details. In practice, functionally equivalent interfaces can differ substantially in HTML structure, accessibility metadata, textual cues, and visual presentation, changing the observations available to an agent while largely preserving the user experience. Accessibility tree observations depend on semantic HTML, roles, ARIA attributes, and alternative text that may be incomplete or absent, while visual observations depend on choices such as whether interactive elements use text or icons and how strongly styling conveys interactive affordances. Performance measured on a single implementation can therefore conflate an agent's capability with its dependence on implementation-specific cues. To isolate the effect of website implementation, we consider a controlled evaluation in which the task goal, functionality, and user experience remain fixed while the implementation varies.

Formulation. Let $\mathcal{W} = \{W_i\}_{i=1}^{n}$ denote a set of n seed websites. For each website W_i, we consider m functionally equivalent environments, $\mathcal{E}_i = \{E_i^{(1)}, \ldots, E_i^{(m)}\}$. All environments in $\mathcal{E}_i$ expose the same or similar user-facing functionality and support the same set of semantic tasks $\mathcal{Q}_i$, while differing only in how the website is implemented and presented. For a task $q \in \mathcal{Q}_i$, the task goal and success criterion remain fixed across all $E \in \mathcal{E}_i$.

At step t, an agent receives a textual observation T_t derived from the accessibility tree, a visual observation V_t derived from the rendered webpage, or both, and produces an action a_t. We investigate whether the agent's ability to complete the same task is preserved as the implementation varies within $\mathcal{E}_i$. We refer to this property as *implementation robustness*.

3.2 Constructing Functionally Equivalent Environments

To instantiate the controlled setting above, we construct WebVariants from nine seed websites, with five functionally equivalent implementations of each website, yielding 45 environments in total. WebVariants contains 228 tasks that can be completed under all five implementations, resulting in 1,140 task–environment instances. Each task is evaluated by the same deterministic Python verifier across implementations. The construction pipeline consists of three stages: (1) auditing the seed environment (Section 3.2.1), (2) generating implementation variants (Section 3.2.2), and (3) verifying task feasibility across implementations (Section 3.2.3). Additional details are in Appendix C.

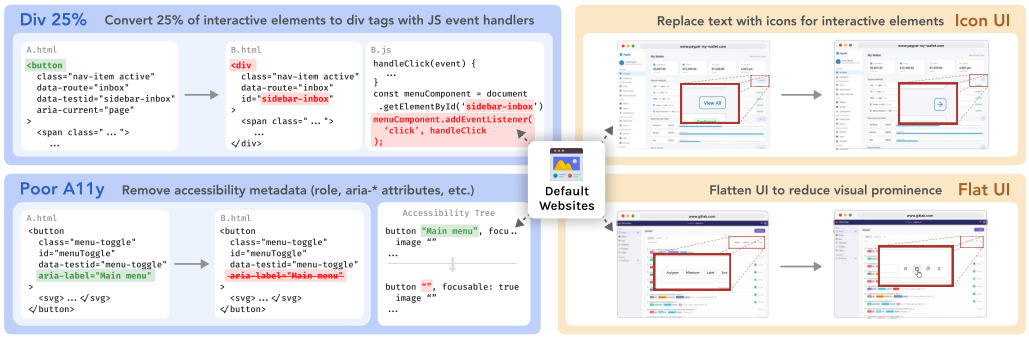


Figure 2: **WEBVARIANTS Overview**. Four functionally equivalent variants derived from default website by perturbing HTML semantics, visual affordances and semantics of interface elements.

3.2.1 SEED ENVIRONMENT AUDIT

We use WebArena-Infinity (Zhou, 2026) as the source of our seed environments. Out of its 13 environments, we retain nine and exclude four that require substantial domain-specific knowledge or overlap considerably in functionality with other websites. Since the WebArena-Infinity were generated entirely by coding agents, we first audit the retained environments for defects that could confound our evaluation. Common defects include modal dialogs that remain open after confirmation, layouts broken by text wrapping, and interactions that appear successful in the interface but fail to update the corresponding backend state. Two frontend engineers, each with more than five years of professional experience, manually inspect and repair all nine websites. The audit addresses functional and interface defects that could cause agent failures regardless of implementation variation.

3.2.2 VARIATION GENERATION

Starting from each audited seed environment, we construct four additional implementation variants that alter cues available to web agents while preserving user-facing functionality, as outlined in Figure 2. The variants manipulate HTML semantics, accessibility metadata, textual labeling, and visual affordances. They are not intended to form an exhaustive or mutually exclusive taxonomy of website implementation variation. These properties may co-occur in real interfaces, and many other forms of variation are possible. Instead, they provide controlled changes to the textual and visual observations used by current web agents.

- **Default** (E^{Default}): The audited seed environment without implementation perturbation.
- **Div-25%** (E^{Div}): We randomly replace 25% of interactive semantic elements, such as `buttons` and `a`, with generic `div` elements while preserving behavior through JavaScript event handlers. This reduces structural and semantic cues in the document representation. We use 25% as the primary setting and report results across replacement ratios in Appendix E.1.
- **Poor-A11y** (E^{Poor}): We reduce accessibility cues by removing explicit roles, aria-* attributes, and alternative text while preserving DOM structure, styling, and interaction behavior.
- **Icon-UI** (E^{Icon}): We replace visible text labels on selected interactive controls with corresponding icons while preserving their accessible names and interaction behavior.
- **Flat-UI** (E^{Flat}): Starting from E^{Icon}, we further reduce visual affordances by simplifying borders, backgrounds, and styling cues that distinguish interactive controls and interface regions.

3.2.3 CROSS-IMPLEMENTATION TASK VERIFICATION

We use the tasks and deterministic backend verifiers from WebArena-Infinity. Corresponding interactive elements across implementations are assigned shared canonical target IDs, which are used only for verification and never exposed to evaluated agents. For each task, we collect a successful human trajectory on E^{Default}, map its interaction targets to canonical IDs, and replay the same action sequence under the other four implementations by resolving each ID to the corresponding elements. We retain a task only if the replay succeeds and the same backend verifier confirms completion across all five implementations, yielding 228 tasks and 1,140 task–environment instances.

3.3 Evaluation Metrics

Rather than relying on a single execution of each task, we follow prior work on reliable evaluation of long-horizon agents (Yao et al., 2025; Khanal et al., 2026) and report Pass$\hat{}k$, the fraction of tasks that succeed in all k independent trials. We use $k = 3$, unless otherwise specified.

We further distinguish task-solving capability from implementation robustness. The resulting robustness curve is defined as

$$R_j = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbf{1}[m(q) \geq j], \text{ where } m(q) = \sum_{E \in \mathcal{E}_i} s_k(q, E), \ j \in \{1, \ldots, 5\}.$$

For each task $q \in \mathcal{Q}_i$, $m(q)$ denotes the number of implementations in $\mathcal{E}_i$ under which the agent succeeds; $s_k(q, E) = 1$ if the agent succeeds in all k independent trials for task–environment pair (q, E). Hence, R_j indicates the fraction of tasks reliably solved under at least j implementations.

The first point on this curve, $C = R_1$, corresponds to task coverage, i.e., the fraction of tasks that the agent can solve under at least one implementation. To measure how well this capability is preserved across implementation variations, we define normalized implementation robustness as $R = \frac{1}{5} \sum_{j=1}^{5} R_j / R_1$ with $R = 0$ when $R_1 = 0$. This score measures the average fraction of implementations successfully handled among tasks that the agent can solve under at least one implementation. Finally, $F_1^{\text{impl}} = 2 \cdot \frac{C \times R}{C + R}$ is the harmonic mean of C and R. A high F_1^{impl} therefore requires both broad task coverage and consistent performance across implementations.

4 Web Agents Are Sensitive to Implementation Variation

4.1 Experimental Setup

Web Agents. We evaluate web agents spanning text-based, screenshot-based, and general-purpose multimodal models. For text-based agents that operate on accessibility tree observations, we evaluate Go-Browse-WA-7B (Gandhi & Neubig, 2025) and BrowserAgent-7B (Yu et al., 2025), for which public checkpoints are available. For screenshot-based agents, we evaluate UI-TARS-1.5-7B (Qin et al., 2025), Holo1-7B (Andreux et al., 2025), MolmoWeb-4B/8B (Gupta et al., 2026), and OpenWebRL-4B (Yang et al., 2026). We additionally evaluate general-purpose VLMs, including GPT-5.6-luna (OpenAI, 2026), Gemini-3.5-Flash-Lite (Google DeepMind, 2026) and models from the Qwen3.5 (Qwen Team, 2026) and Gemma 4 (Gemma Team, 2026) families, which allow us to vary the observation and action interfaces while holding the underlying model fixed.

Observation and Action Interfaces. We consider two types of action interfaces. In the BID-based interface, each actionable node in the accessibility tree is assigned a browser element identifier (BID), and the agent predicts the identifier together with an action type. In the coordinate-based interface, the agent predicts an action type and a pixel coordinate on the rendered webpage. For BID-based agents, we evaluate textual observation $\mathcal{T}$ and multimodal observation $\mathcal{T} + \mathcal{V}$, where $\mathcal{T}$ denotes the accessibility tree and $\mathcal{V}$ denotes the webpage screenshot. Specialized screenshot-based agents use the coordinate interface with $\mathcal{V}$ observations. Because general-purpose VLMs exhibit substantially weaker coordinate grounding on web screenshots, our controlled observation analyses focus on the BID-based interface. Coordinate-based VLM results are reported in Appendix E.4.

4.2 Implementation Variation Changes Evaluation Outcomes

Performance Is Implementation-Dependent. Despite preserving the task and task-relevant functionality, performance shifts substantially across implementations. As shown in Table 1, these shifts appear across specialized text- and screenshot-based web agents as well as general-purpose VLMs. The direction and magnitude of these shifts vary across agents and implementations, rather than indicating a uniformly harder implementation. Agents relying on accessibility tree $\mathcal{T}$ are particularly affected when structural or accessibility cues are weakened, as in E^{Div} and E^{Poor}. For example, Go-Browse-WA-7B drops by 74.1% under E^{Poor} relative to E^{Default}. In contrast, screenshot-based agents are more sensitive to changes in visual labels or affordances, as in E^{Icon} and E^{Flat}. Multimodal agents also exhibit substantial variation despite access to both modalities. Thus, results on a single implementation do not reliably represent performance across alternative implementations.

Table 1: Main results on WEBVARIANTS across specialized web agents and general-purpose VLMs under different observation and grounding interfaces, reported using Pass^3 (%) and our proposed metric F_1^{impl}. $\mathcal{T}$ denotes the accessibility tree and $\mathcal{V}$ denotes the screenshot. Full results, including C and R, are in Appendix E.2.

Model	Obs.	Grounding	Environments					$F_1^{\text{impl}} \uparrow$
			Default	Div-25%	Poor-A11y	Icon-UI	Flat-UI	
Unimodal Web Agents								
Go-Browse-WA-7B	$\mathcal{T}$	BID	11.84	8.33	3.07	17.11	10.09	29.90
BrowserAgent-RFT (7B)	$\mathcal{T}$	BID	8.33	3.95	3.51	10.96	8.33	21.92
UI-TARS-1.5-7B	$\mathcal{V}$	Coordinate	5.70	5.26	7.89	3.51	3.07	17.29
Holo1-7B	$\mathcal{V}$	Coordinate	3.07	3.51	4.39	2.19	0.88	11.94
MolmoWeb-8B	$\mathcal{V}$	Coordinate	6.58	8.33	4.82	5.26	0.88	19.38
MolmoWeb-4B	$\mathcal{V}$	Coordinate	6.14	4.82	4.39	5.70	1.75	16.44
OpenWebRL-4B	$\mathcal{V}$	Coordinate	12.72	9.65	14.91	10.53	6.14	30.56
Vision Language Models (Closed-source)								
GPT-5.6-luna	$\mathcal{T}$	BID	43.42	21.05	24.12	46.49	41.67	59.32
	$\mathcal{T}+\mathcal{V}$	BID	50.88	28.51	34.21	53.95	51.75	66.00
Gemini-3.5-Flash-Lite	$\mathcal{T}$	BID	41.23	29.82	20.18	46.05	42.11	59.88
	$\mathcal{T}+\mathcal{V}$	BID	46.93	32.46	26.75	47.37	45.18	63.03
Vision Language Models (Open-source)								
Qwen3.5-122B-A10B	$\mathcal{T}$	BID	49.12	25.00	23.68	50.00	47.37	62.37
	$\mathcal{T}+\mathcal{V}$	BID	50.88	23.25	25.88	54.82	49.12	63.82
Qwen3.5-35B-A3B	$\mathcal{T}$	BID	36.40	16.23	17.11	39.47	36.84	53.80
	$\mathcal{T}+\mathcal{V}$	BID	28.51	17.54	17.11	30.70	28.95	48.82
Qwen3.5-9B	$\mathcal{T}$	BID	22.37	10.96	10.09	21.93	23.68	41.29
	$\mathcal{T}+\mathcal{V}$	BID	18.42	8.77	9.21	16.23	14.91	35.31
Qwen3.5-4B	$\mathcal{T}$	BID	17.11	7.02	6.14	12.28	10.09	31.95
	$\mathcal{T}+\mathcal{V}$	BID	17.54	6.58	8.33	17.98	16.67	35.76
Gemma4-31B-it	$\mathcal{T}$	BID	39.91	20.18	21.93	45.18	38.16	57.36
	$\mathcal{T}+\mathcal{V}$	BID	44.30	19.30	21.93	46.93	43.42	59.22
Gemma4-26B-A4B-it	$\mathcal{T}$	BID	19.30	12.72	7.89	21.49	19.30	37.52
	$\mathcal{T}+\mathcal{V}$	BID	19.74	10.53	5.70	19.74	20.18	36.14
Gemma4-E4B-it	$\mathcal{T}$	BID	19.30	9.21	5.70	16.23	14.47	33.31
	$\mathcal{T}+\mathcal{V}$	BID	17.98	8.77	8.33	17.11	17.98	35.62

Agent Rankings Change Across Implementations. Implementation variation affects not only absolute performance but also comparative conclusions between agents. To reduce model scale as a confounding factor, we only use nine agents with 4B–8B parameters: the seven unimodal web agents and Qwen3.5-4B and Gemma4-E4B-it using $\mathcal{T}+\mathcal{V}$. For each implementation, we compute Kendall's τ between its agent ranking and the ranking under E^{Default}.

As shown in Table 2, agreement with E^{Default} ranking varies widely, with Kendall's τ ranging from 0.197 to 0.817. Under E^{Poor}, 40% of agent pairs reverse their relative order compared with E^{Default}. Moreover, Kendall's τ captures disagreement in pairwise ordering but not the magnitude of individual reversals. For example, BrowserAgent-7B slightly outperforms OpenWebRL-4B under E^{Icon} (10.96 vs. 10.53), whereas OpenWebRL-4B significantly outperforms BrowserAgent-7B under E^{Poor} (14.91 vs. 3.07). Implementation choice can therefore change both the reported performance and which agent appears stronger, showing that the performance drawn from a single implementation partly reflect compatibility with that implementation.

	Div-25%	Poor-A11y	Icon-UI	Flat-UI
Kendall's τ	0.479	0.197	0.817	0.704
95% CI	[.171, .743]	[-.087, .556]	[.514, .889]	[.514, .889]

Table 2: Kendall's τ between the agent ranking under E^{Default} and those under each implementation variant. 95% CIs are obtained by bootstrap resampling over tasks. Full results are in Appendix E.3.

4.3 OBSERVATION DESIGN DOES NOT ELIMINATE IMPLEMENTATION SENSITIVITY

The implementation sensitivities observed in Section 4.2 raise a natural question: can implementation sensitivity be mitigated simply by choosing a richer observation interface? We therefore compare the same VLMs under $\mathcal{T}$ and $\mathcal{T}+\mathcal{V}$, and further examine whether explicit visual grounding through Set-of-Marks (SoM) improves robustness using $\mathcal{T}+\mathcal{V}_{\text{SoM}}$.

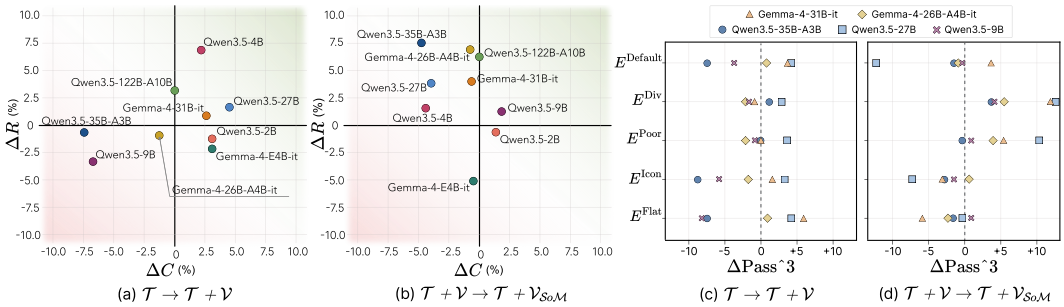


Figure 3: **Observation interface choice has implementation-dependent effects.** (a, b) Changes in C and R when adding screenshots or SoM grounding. (c, d) Corresponding changes in Pass^3 across implementations. Neither observation designs consistently improves performance across models and implementations.

Figure 3 (a) and (b) show that enriching the observation interface does not consistently improve both task coverage C and implementation robustness R. Adding screenshots moves different models in different directions rather than uniformly toward higher C and R, while SoM grounding similarly produces mixed changes across models. The per-implementation results in Figure 3 (c) and (d) reveal why: the same observation change can improve performance under one implementation while degrading it under another. Thus, implementation sensitivity cannot be explained simply by insufficient observation. Neither complementary visual input nor explicit grounding yields an observation interface that is uniformly better across implementations.

4.4 Recognizing Task Completion Is Implementation-Dependent

Different implementations can represent the same website state through different textual and visual observations. Implementation robustness therefore requires not only reaching a successful state, but also recognizing that the task goal has been satisfied despite these representational differences. To examine this, we retrospectively apply the verifier to intermediate states to identify rollouts that ever reach a successful state, and measure the fraction of these rollouts that fail to terminate successfully with `done()`. The verifier is used only for post-hoc analysis and provides no feedback to the agent.

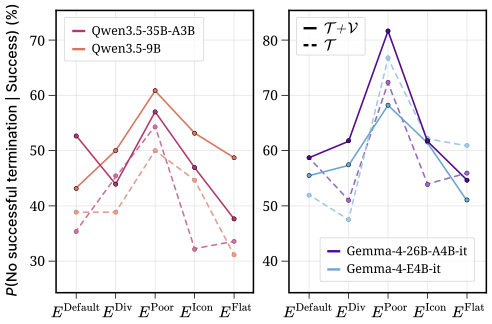


Figure 4: **Recognition of task completion varies across implementations.** Among rollouts that reach a successful state, we report the fraction that fail to terminate successfully with `done()`. Solid and dashed lines denote $\mathcal{T} + \mathcal{V}$ and $\mathcal{T}$, respectively.

Figure 4 demonstrates that the ability to identify task completion varies substantially across implementations, even with the model and observation interface fixed. For example, among rollouts in which Gemma-4-26B-A4B-it reaches a successful state using $\mathcal{T} + \mathcal{V}$, 81% fail to terminate successfully under E^{Poor}, compared with 55% under E^{Flat}. Similar variation appears across the other models and under both observation interfaces. Thus, implementation sensitivity persists even after the task goal has been achieved: changing only the implementation can affect whether an agent recognizes that the task is complete.

5 Learning to Generalize Across Website Implementations

Section 4 shows that current web agents are sensitive to implementation variation and that richer observations do not reliably eliminate this sensitivity. We next ask whether training can reduce this sensitivity and whether the resulting behavior generalizes beyond the training data. Since training cannot cover every website or every possible implementation, we consider two forms of generalization: to unseen implementations of websites represented during training, and to implementations of websites excluded entirely from training. We further examine when implementation diversity is beneficial and how it should be allocated under a fixed supervision budget.

5.1 Experimental Setup

We collect successful human trajectories for 176 tasks from seven source websites, using task instances disjoint from the 228 evaluation tasks in WebVariants. Following Section 3.2.3, we collect each trajectory on E^{Default} and replay its canonical action sequence under the other four implementations, verifying success in each environment. This yields 176 verified trajectories per implementation and 880 trajectories in total. The remaining two websites are excluded from all training data and used only for evaluation.

Let $\mathcal{D}_j$ denote the 176 trajectories collected under implementation E^j. **Single-Implementation** trains on $\mathcal{D}_j$ from one implementation. For an implementation E^h excluded from training, **Multi-Implementation** instead distributes the same 176 tasks across the other four implementations, with each task contributing one trajectory. Assignments are balanced across implementations and stratified by website. Across these regimes, we hold the task identities, number of trajectories, and source websites fixed while varying the implementation distribution represented during training.

We convert each trajectory into supervised examples at individual interaction steps and fine-tune Qwen3.5-4B and Gemma-4-E2B-it using $\mathcal{T} + \mathcal{V}$ observations. We evaluate each trained model on WebVariants under all five implementations, separately on the seven source websites and the two held-out websites. The first setting tests generalization across implementations when the website was represented during training, while the second additionally requires generalization to a new website. Training details and full results are provided in Appendix D.4 and E.6, respectively.

5.2 Generalization from One Implementation Weakens on Unseen Websites

We first ask whether supervision from one implementation generalizes to other implementations. For each E^j, we fine-tune on $\mathcal{D}_j$ and evaluate the resulting model on all five implementations. Figure 5 reports the change in Pass^3 relative to the base $\mathcal{T} + \mathcal{V}$ model, with rows indicating the training implementation and columns the evaluation implementation. Diagonal entries evaluate the implementation represented during training, while off-diagonal entries evaluate implementations absent from training.

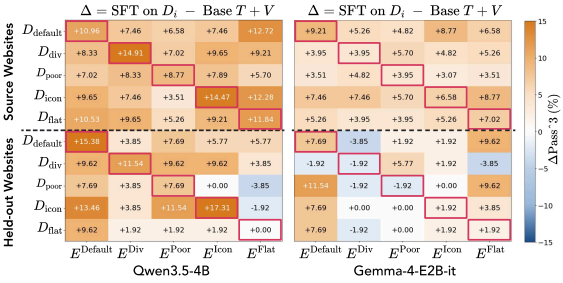


Figure 5: **Effects of training on a single implementation.** Each cell reports the change in Pass^3 relative to the base $\mathcal{T} + \mathcal{V}$ model after training on $\mathcal{D}_j$. Rows denote the training implementation and columns the evaluation implementation; outlined cells correspond to the implementation represented during training.

On the source websites, gains extend well beyond the training implementation: every implementation absent from training improves over the base model for both VLMs. The average gain on these implementations is 8.25% for Qwen3.5-4B and 5.39% for Gemma-4-E2B-it. On the held-out websites, however, the corresponding gains fall to 5.29% and 2.50% and become less consistent, with several implementation pairs showing no improvement or even degradation. Training on a single implementation therefore generalizes broadly to other implementations of websites represented during training, but this generalization weakens on unseen websites.

5.3 Implementation Diversity Matters Under Website Shift

The weaker and less consistent generalization to held-out websites raises a natural question: does training on multiple implementations improve generalization to websites absent from training? For each implementation E^h excluded from training, we compare Multi-Implementation with the average performance of the four Single-Implementation models trained on E^j, $j \neq h$. Thus, E^h is excluded from training in both conditions, and each model is trained with the same budget of 176 trajectories. Figure 6 reports C, R, and F_1^{impl} for each choice of E^h.

The effect of implementation diversity on the source websites varies across models. Multi-Implementation substantially improves C, R, and F_1^{impl} for Qwen3.5-4B, whereas the gains are smaller and less consistent for Gemma-4-E2B-it. On the held-out websites, however, both models

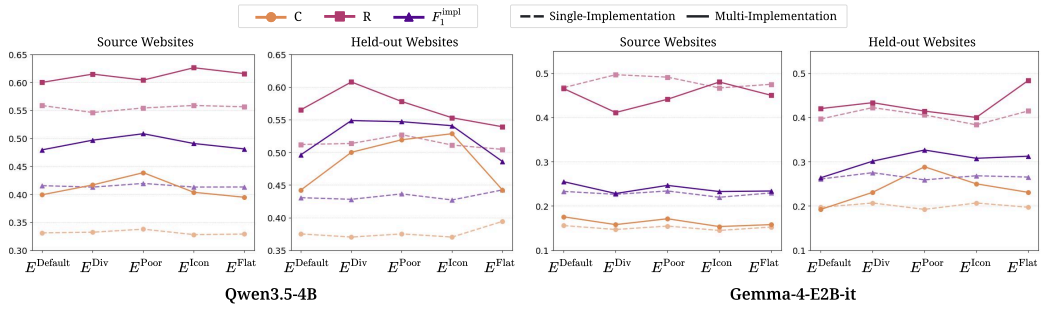


Figure 6: **Single- versus Multi-Implementation training under a fixed budget.** For each implementation E^h excluded from training, we compare Multi-Implementation with the average of the four Single-Implementation models trained on the remaining implementations.

benefit consistently from Multi-Implementation training, with improvements in both task coverage and implementation robustness and higher F_1^{impl} for every excluded implementation E^h. Thus, while the benefit of training on multiple implementations varies across models on the source websites, implementation diversity consistently improves generalization to unseen websites.

5.4 Allocating Multi-Implementation Supervision

Section 5.3 shows that implementation diversity is particularly beneficial on held-out websites. This raises a further question of how to allocate a fixed trajectory budget across tasks and implementations. Specifically, should the same tasks be represented across multiple implementations, or should the budget instead cover a larger number of unique tasks? To examine this trade-off, we fix the training budget at 176 trajectories and compare three allocations: 176 unique tasks each observed under one of the four implementation, 88 tasks each observed under two implementations, and 44 tasks each observed under all four implementations. Across all settings, the four implementations are used in balanced proportions within each source websites, and the task sets are nested.

On the held-out websites, increasing the number of implementations per task consistently reduces F_1^{impl} for both models, as shown in Figure 7. Under the same training budget, allocating supervision to more unique tasks therefore provides better generalization than repeatedly exposing fewer tasks to multiple implementations. The source websites show a less uniform pattern: Qwen3.5-4B exhibits the same preference for more unique tasks, whereas Gemma-4-E2B-it remains relatively insensitive to the allocation. These results suggest that, for generalization beyond the training websites, implementation diversity is most effective when combined with broader task diversity rather than concentrated on multiple implementations of the same tasks.

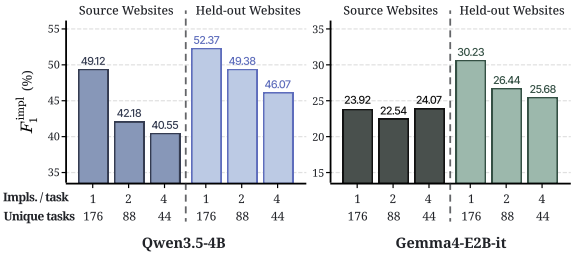


Figure 7: **Allocating supervision across tasks and implementations.** All settings use 176 trajectories while varying the number of implementations observed per task and, correspondingly, the number of unique tasks.

6 Conclusion

We introduce web implementation as a distinct axis for evaluating the robustness of web agents and present WebVariants, a controlled benchmark that varies implementation while preserving the task, website functionality and success criteria. Our experiments show that current agents remain sensitive to implementation changes across textual, visual, and multimodal observation interfaces, affecting agent rankings and even agents' recognition of task completion. We further find that training across multiple implementations improves generalization to unseen websites, and that, under a fixed training budget, covering more unique tasks is more effective than repeatedly observing fewer tasks across multiple implementations. These results establish web implementation as an important consideration in both the evaluation and training of robust web agents.

AI USE STATEMENT

We used generative AI tools to help identify relevant prior and related work, drafting and polishing portions of the manuscript, checking grammar, and rephrasing the text to improve clarity and meet space constraints. We also used generative AI to suggest color combinations for figures. In addition, coding agent assisted in writing parts of the experimental and analysis code. All AI-assisted code was reviewed by the authors and verified through repeated testing before use in our experiments.

The research questions, core ideas, experimental design, paper organization, and content were determined by the authors. Literature identified with the assistance of generative AI was manually verified against the original sources, and all AI-assisted writing was reviewed and revised by the authors to ensure that it accurately reflected our intended claims and results. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have uploaded WEBVARIANTS environments to an anonymous Github repository (https://anonymous.4open.science/r/WebVariants-C74E) and included the task set as part of the supplementary material. WEBVARIANTS will be made public upon acceptance. In Appendix D.3.1, we also provide the prompts to help reproduce and potentially extend our benchmark in the future. Our evaluation code will also be released to enable measuring implementation robustness on WEBVARIANTS.

REFERENCES

Mathieu Andreux, Breno Baldas Skuk, Hamza Benchekroun, Emilien Biré, Antoine Bonnet, Riaz Bordie, Matthias Brunel, Pierre-Louis Cedoz, Antoine Chassang, Mickaël Chen, Alexandra D. Constantinou, Antoine d'Andigné, Hubert de La Jonquière, Aurélien Delfosse, Ludovic Denoyer, Alexis Deprez, Augustin Derupti, Michael Eickenberg, Mathïs Federico, Charles Kantor, Xavier Koegler, Yann Labbé, Matthew C. H. Lee, Erwan Le Jumeau de Kergaradec, Amir Mahla, Avshalom Manevich, Adrien Maret, Charles Masson, Rafaël Maurin, Arturo Mena, Philippe Modard, Axel Moyal, Axel Nguyen Kerbel, Julien Revelle, Mats L. Richter, María Santos, Laurent Sifre, Maxime Theillard, Marc Thibault, Louis Thiry, Léo Tronchon, Nicolas Usunier, and Tony Wu. Surfer-h meets holo1: Cost-efficient web agent powered by open weights, 2025. URL https://arxiv.org/abs/2506.02865.

Haoyue Bai, Dong Wang, Long Chen, Bingguang Hao, Pengyang Shao, Yonghui Yang, Yicheng He, and Chenyi Zhuang. Stressweb: A diagnostic benchmark for web agent robustness under realistic interaction variability, 2026. URL https://arxiv.org/abs/2604.16385.

Hyungjoo Chae, Jungsoo Park, and Alan Ritter. Safe and scalable web agent learning via recreated websites. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=x1OLusJLNe.

Qi Chen, Dileepa Pitawela, Chongyang Zhao, Gengze Zhou, Hsiang-Ting Chen, and Qi Wu. Webvln: Vision-and-language navigation on websites. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.505. URL https://aclanthology.org/2024.acl-long.505/.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym

ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5950bf290a1570ea401bf98882128160-Paper-Datasets_and_Benchmarks.pdf.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Apurva Gandhi and Graham Neubig. Go-browse: Training web agents with structured exploration, 2025. URL https://arxiv.org/abs/2506.03533.

Divyansh Garg, Diego Caples, Andis Draguns, Nikil Ravi, Pranav Putta, Naman Garg, Prannay Hebbar, Youngchul Joo, Jindong Gu, Charles London, Christian Schroeder de Witt, and Sumeet Ramesh Motwani. REAL: Benchmarking autonomous agents on deterministic simulations of real websites. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=Un1sWxmZuI.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Google DeepMind. Gemini 3.5 flash-lite model card, July 2026. URL https://deepmind.google/models/model-cards/gemini-3-5-flash-lite/.

Tanmay Gupta, Piper Wolters, Zixian Ma, Peter Sushko, Rock Yuren Pang, Diego Llanes, Yue Yang, Taira Anderson, Boyuan Zheng, Zhongzheng Ren, Harsh Trivedi, Taylor Blanton, Caleb Ouellette, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmoweb: Open visual web agent and open data for the open web, 2026. URL https://arxiv.org/abs/2604.08516.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

Md Farhan Ishmam and Kenneth Marino. Timewarp: Evaluating web agents by revisiting the past, 2026. URL https://arxiv.org/abs/2603.04949.

Aaditya Khanal, Yangyang Tao, and Junxiu Zhou. Beyond pass@1: A reliability science framework for long-horizon llm agents, 2026. URL https://arxiv.org/abs/2603.29231.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *ACL*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5295—-5306, 2024.

OpenAI. Introducing codex, May 2025. URL https://openai.com/index/introducing-codex/.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition, July 2026. URL https://openai.com/index/gpt-5-6/.

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

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3135–3144. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/shi17a.html.

Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin, and Zuozhu Liu. Webworld: A large-scale world model for web agent training. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ES7Vy9rvRl.

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. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=6jZi4HSs6o.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. URL https://arxiv.org/abs/2310.11441.

Rui Yang, Qianhui Wu, Yuxi Chen, Hao Bai, Wenlin Yao, Hao Cheng, Baolin Peng, Huan Zhang, Tong Zhang, and Jianfeng Gao. Openwebrl: Demystifying online multi-turn reinforcement learning for visual web agents, 2026. URL https://arxiv.org/abs/2606.02031.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. {τ}-bench: A benchmark for \underline{T}ool-\underline{A}gent-\underline{U}ser interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

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

Tao Yu, Zhengbo Zhang, Zhiheng Lyu, Junhao Gong, Hongzhu Yi, Xinming Wang, Yuxuan Zhou, Jiabing Yang, Ping Nie, Yan Huang, and Wenhu Chen. Browseragent: Building web agents with human-inspired web browsing actions, 2025. URL https://arxiv.org/abs/2510.10666.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024a.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models, 2024b. URL https://arxiv.org/abs/2403.13372.

Shuyan Zhou. Webarena-infinity: Generating browser environments with verifiable tasks at scale. *shuyanzhou.com*, March 2026. URL https://webarena.dev/webarena-infinity/.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://webarena.dev.

A Limitations and Future Directions

Coverage of implementation diversity. WebVariants isolates several implementation dimensions that affect web agent observations, but these controlled variants do not exhaust the diversity of implementation choices found on real-world websites. Real-world websites may combine these variations or differ along other dimensions, such as layout structure, interaction patterns, client-side frameworks, and rendering behavior. Future work could extend controlled evaluation to a broader set of implementations, including combinations of multiple implementations, and examine whether the observed sensitivities and training effects extend beyond the variants studied here.

Coverage of real-world website dynamics. Our experiments use controlled website environments, which enable functionality and task semantics to remain fixed while varying only the implementation. This control is essential for isolating implementation effects, but it does not capture the full dynamics of live websites, such as evolving content, personalization, A/B testing, dynamic state, latency, and frequent interface updates. Evaluating implementation robustness under these naturally occurring dynamics is an important direction for future work.

Generalization beyond the benchmark distribution. Our held-out websites originate from the same underlying benchmark source. Although the held-out websites test generalization to unseen websites, whether the observed effects extend to substantially different website distributions remains an open question.

Training paradigm. Our training experiments use supervised fine-tuning on successful trajectories because our goal is to isolate the role of implementation diversity in training data rather than to compare optimization methods or maximize agent performance. Reinforcement learning introduces additional factors, including exploration, rollout generation, and reward design, which can make the effect of implementation diversity harder to isolate. Whether the observed effects of implementation diversity also hold under reinforcement learning in a similarly controlled setting remains an open question.

B Additional Related Work

User Interface Variation. Recent work has begun to examine web agent robustness to variations in user interfaces. VAF (Yu et al., 2026) studies controlled changes to visual attributes such as contrast, size, position, and card clarity while preserving semantic content. TimeWarp (Ishmam & Marino, 2026) evaluates agents across temporal versions of web environments that differ substantially in UI design and layout. Both works demonstrate that interface-level changes can substantially affect web agent behavior.

Our work examines a complementary axis of variation: implementation differences that can arise from ordinary website construction while largely preserving the user-facing experience. Website implementations can differ not only in visible presentation, but also in HTML semantics, accessibility metadata, and other underlying representations that may have little effect on how a user perceives or interacts with the website while substantially changing the observations available to an agent. Rather than focusing primarily on changes to the user-facing interface, we treat *website implementation* as a distinct axis of robustness that captures differences in how the same functionality is realized and represented to agents. By holding task, functionality, and success criteria fixed while largely preserving the user-facing experience, WebVariants evaluates whether agent capability persists across functionally equivalent website implementations. We further study the effects of this variation across observation interfaces, agent rankings, task-completion recognition, and training for implementation robustness.

C Additional Details of WebVariants

C.1 Implementation Variations

Selection of variations. The four variants were chosen in consultation with two frontend engineers to capture implementation choices that can vary substantially across developers while leaving the

intended functionality of an interface unchanged. We focused on properties that directly affect the observations consumed by current web agents: HTML semantics, accessibility metadata, textual labeling, and visual affordances. These variations are not intended to exhaust the space of possible web implementations. Rather, they represent common implementation decisions that may have little effect on whether a user can complete a task, while substantially changing the structural or visual evidence available to an agent.

Div-25% (E^{Div}). Native HTML elements such as `button` and `a` provide browser-defined semantics and behavior in addition to their visible appearance. In practice, however, the same interaction can be implemented using generic elements together with JavaScript event handlers. Frontend developers may choose this approach when building custom components whose appearance and behavior are controlled primarily through CSS and JavaScript rather than native element defaults. Modern design-to-code workflows can also reproduce a desired visual design without necessarily preserving the corresponding semantic HTML elements. Motivated by this implementation choice, E^{Div} replaces a subset of interactive semantic elements with generic `div` elements. The original interaction behavior is preserved through JavaScript, while structural cues exposed through the document and accessibility representations are weakened. Because replacing every semantic element would constitute an unusually strong intervention, we use a 25% replacement ratio in the main benchmark. Appendix E.1 reports additional results for replacement ratios of $\{0, 25, 50, 75, 100\}\%$.

Poor-A11y (E^{Poor}). Accessibility metadata such as `role`, `aria-*` attributes, and alternative text can provide explicit information about the semantics and purpose of interface elements. These annotations are particularly useful to agents that consume accessibility trees, but many of them are not required for the underlying interaction to function. For example, removing an `aria-label` from an icon button does not prevent its click handler from executing, although it can remove the textual description exposed through the accessibility tree. E^{Poor} therefore removes selected accessibility metadata while preserving DOM structure, styling, and interaction behavior. This variant isolates a setting in which the visible interface and task functionality remain largely unchanged, but the semantic information available through the accessibility tree is reduced. It reflects the fact that accessibility metadata can vary independently of the basic interaction bahavior of many web interfaces.

Icon-UI (E^{Icon}). The same interaction can be presented using explicit text, an icon, or a combination of the two. These alternatives may preserve the same functionality while providing different evidence to an agent. A text label such as "Delete" directly exposes the intended action in both the screenshot and textual observations. An icon-only control instead requires the agent to infer its meaning from visual appearance, surrounding context, or prior familiarity with the icon. E^{Icon} replaces visible text labels on interactive controls with corresponding icons while preserving their accessible names and interaction behavior. We focus on controls whose the function remains understandable from the icon or surrounding interface context, but is less explicitly conveyed than in the original text-labeled version. The variant therefore changes how the same semantics are presented visually without removing the corresponding accessibility information.

Flat-UI (E^{Flat}). Visual styling often indicates which regions of a webpage are interactive. Borders, background fills, shadows, spacing, and other visual distinctions can make buttons, menus, and input regions immediately recognizable even when their textual content is unchanged. Humans can additionally probe an interface through mouse movement and observe dynamic responses such as hover effects. Agents operating from static screenshots cannot observe such dynamic cues before selecting an action. Starting from E^{Icon}, E^{Flat} further reduces visual affordances by removing or weakening borders, backgrounds, and related styling that visually separates interactive controls from surrounding content. Interactive behavior is preserved, including dynamic effects such as hover responses where present. The resulting interface therefore remains functional while providing weaker static visual evidence about which elements are actionable.

C.2 Environment Construction

C.2.1 Seed Environment Selection

WebArena-Infinity provides 13 publicly available web environments generated by coding agents. We retain nine as seed environments and exclude four based on domain redundancy, specialized interaction requirements, and the amount of reconstruction required to obtain a stable environment suitable for controlled variation. Table 3 summarizes the selected environments and the number of evaluation tasks retained for each website.

We exclude elation-prescriptions because it belongs to the same application domain as the other Elation environments while requiring substantially more specialized medical and prescription-related knowledge. Retaining the other Elation environments preserves coverage of this domain without placing disproportionate emphasis on domain-specific knowledge.

We also exclude figma-slides and figma-text-and-typography. Unlike most of the retained environments, these environments model a graphical design tool and require interactions with a canvas-like editing interface, including operations that are uncommon in conventional web applications. The generated seed environments also contain substantially more functional detects and interface inconsistencies than the other websites. Bringing them to a comparable level of functionality would have required reconstructing substantial parts of the underlying design tool rather than auditing and repairing the existing environment.

Finally, we exclude superhuman-general because its primary functionality overlaps with the retained Gmail environment. We retain Gmail instead to avoid overrepresenting email workflows in WebVariants

Table 3: Seed environment selection and task statistics on WebVariants. For each website, we sample 20% of the tasks available in WebArena-Infinity while preserving the original proportions of easy, medium, and hard tasks.

Environment	Include	# Tasks
elation-clinical-records	✓	24
elation-patient-communication	✓	24
elation-prescriptions	✗	-
figma-slides	✗	-
figma-text-and-typography	✗	-
gitlab-plan-and-track	✓	28
gmail	✓	12
gmail-accounts-and-contacts	✓	24
handshake-career-exploration	✓	40
linear-account-settings	✓	24
paypal-my-wallet	✓	28
superhuman-general	✗	-
xero-invoicing	✓	24
Total		228

C.2.2 Seed Environment Audit

We first manually inspect each retained seed environment and repair visible functional and interface defects. Common issues include actions that do not update the corresponding backend state, table layouts that break when cells are empty, buttons or links that produce no response, malfunctioning dropdown menus, and form inputs that update the backend before an explicit save action is performed. These repairs are limited to restoring the intended functionality of the original environment rather than changing its task semantics or interface design.

We then conduct a task-driven audit using the tasks provided by WebArena-Infinity. From the original set of 1,140 tasks, we select 228 while preserving the original difficulty distribution. Each selected task is manually executed on the audited seed environment E^{Default}, and a successful human trajectory is recorded. This process exposes defects that are difficult to identify through interface inspection alone, including failures that arise only after particular sequences of interactions. We repair such defects and, when a task is infeasible in the provided environment, minimally revise task instance so that the intended goal can be completed and verified. The resulting 228 tasks form the evaluation set used throughout WEBVARIANTS.

C.2.3 VARIATION GENERATION

The implementation variants require only localized frontend modifications rather than substantial changes to application functionality or backend logic. For example, the transformations involve replacing selected HTML elements, simplifying accessibility metadata, substituting text labels with icons, or adjusting the styling of interactive elements. Because these edits are relatively lightweight and well scoped, we first generate the variants using Codex (OpenAI, 2025), instructing it to preserve existing functionality while modifying only the targeted implementation property. We then manually inspect each generated environment and repair any unintended functional or interface changes before proceeding to cross-implementation verification.

C.2.4 CROSS-IMPLEMENTATION TASK VERIFICATION

To replay the same human trajectory across implementation variants, we assign a canonical target ID to each interactive element in E^{Default}. Each ID refers to the same semantic interaction target regardless of its DOM position, rendered location, or implementation-specific representation. During variant construction, corresponding elements retain the same canonical ID. This allows an action sequence recorded on E^{Default} to be replayed even when the DOM hierarchy, element geometry, or visual presentation changes.

Each trajectory is represented as a sequence of interaction steps. A step specifies the canonical target ID, the action type, and, when applicable, the action value. Two example steps are shown below.

(1) A click on the search-options control on gmail.

```
id: gml-472862-searchoptionsbtn--instance-testid-search-options-btn-0
aflrjx
type: click
value: none
```

(2) Entering 12.50 into the savings-deposit amount field on paypal-my-wallet.

```
id: pmw-1015455-savings-deposit-amount
type: fill
value: 12.50
```

At replay time, the canonical ID is resolved to the corresponding element in the target implementation, and the recorded action is executed on that element. After replaying the complete action sequence, we apply the same backend verifier used for E^{Default}. A task is retained only if the replay succeeds and the verifier confirms completion across all five implementations. Canonical IDs are used only for benchmark construction and verification and are never exposed to evaluated agents.

D ADDITIONAL DETAILS OF EXPERIMENTS

All local experiments were conducted on 16 NVIDIA A40 GPUs, each with 48 GB of memory.

D.1 WEB AGENTS AND BASELINES

Table 4 summarizes the agents evaluated in our experiments. For specialized web agents, we follow the official evaluation protocol provided by the authors of each method, including the model-specific

Table 4: Models and checkpoints used in our experiments.

Model	Checkpoint / API
Go-Browse-WA-7B (Gandhi & Neubig, 2025)	`apurvaga/go-browse-wa-qwen-7B`
BrowserAgent-RFT (7B) (Yu et al., 2025)	`TIGER-Lab/BrowserAgent-RFT`
UI-TARS-1.5-7B (Qin et al., 2025)	`ByteDance-Seed/UI-TARS-1.5-7B`
Holo1-7B (Andreux et al., 2025)	`Hcompany/Holo1-7B`
MolmoWeb-4B (Gupta et al., 2026)	`allenai/MolmoWeb-4B`
MolmoWeb-8B (Gupta et al., 2026)	`allenai/MolmoWeb-8B`
OpenWebRL-4B (Yang et al., 2026)	`OpenWebRL/OpenWebRL-4B`
Qwen3.5-122B-A10B (Qwen Team, 2026)	`Qwen/Qwen3.5-122B-A10B`
Qwen3.5-35B-A3B (Qwen Team, 2026)	`Qwen/Qwen3.5-35B-A3B`
Qwen3.5-27B (Qwen Team, 2026)	`Qwen/Qwen3.5-27B`
Qwen3.5-9B (Qwen Team, 2026)	`Qwen/Qwen3.5-9B`
Qwen3.5-4B (Qwen Team, 2026)	`Qwen/Qwen3.5-4B`
Qwen3.5-2B (Qwen Team, 2026)	`Qwen/Qwen3.5-2B`
Gemma4-31B-it (Gemma Team, 2026)	`google/gemma-4-31b-it`
Gemma4-26B-A4B-it (Gemma Team, 2026)	`google/gemma-4-26B-A4B-it`
Gemma4-E4B-it (Gemma Team, 2026)	`google/gemma-4-E4B-it`
Gemma4-E2B-it (Gemma Team, 2026)	`google/gemma-4-E2B-it`
GPT-5.6-luna (medium)	OpenAI API
Gemini-3.5-Flash-Lite (minimal)	OpenRouter API

prompt format, action representation, and inference hyperparameters. For Holo1-7B, which uses separate policy, localizer, and validator roles, we use Holo1-7B for all three components. Prompt templates for general-purpose VLMs are provided in Appendix D.3.1.

D.2 Action Space

To enable controlled evaluation of general-purpose VLMs, we use the BID-based and coordinate-based action spaces summarized in Table 5. The BID interface targets elements through browser element identifiers (BIDs), whereas the coordinate interface operates directly on viewport coordinates. Specialized web agents retain their official model-specific action spaces.

D.3 Evaluation Details

All web interactions are executed through BrowserGym (Drouin et al., 2024; de Chezelles et al., 2025). Open-weight models are served locally using vLLM (Kwon et al., 2023), while Gemini-3.5-Flash-Lite is accessed through the OpenRouter API. For specialized web agents, we follow the inference settings released with each agent. For general-purpose VLMs, we use the default inference hyperparameters provided by each model or API provider rather than imposing a shared decoding configuration. Prompt templates are provided in Appendix D.3.1.

Each task–environment pair is evaluated over three independent trials. The environment is reset before every trial, and each rollout is limited to 60 interaction steps. A rollout terminates when the agent issues `done()` or `report_infeasible()`, reaches the interaction limit, or an unrecoverable execution failure occurs. Task success is determined by the corresponding backend verifier after termination. We use a viewport size of 1280×720.

D.3.1 Prompts for General-Purpose VLMs

We use a common prompt template for all general-purpose VLMs. The prompt differs across experimental settings only in the available action space and the observations provided to the model. BID-based settings use the BID action space in Table 5, whereas coordinate-based settings use the coordinate action space. Within each action interface, all other instructions, including the output

Table 5: **Action space.** BID-based and coordinate-based actions used in our controlled VLM experiments. Specialized web agents follow their official model-specific action spaces.

Action	Description
BID-based Actions	
`click(bid)`	Click the element identified by `bid`
`fill(bid, text)`	Fill the element identified by `bid` with text
`hover(bid)`	Hover over the element identified by `bid`
`go_back()`	Navigate to the previous page
`go_forward()`	Navigate to the next page
`goto(url)`	Navigate the browser to a URL
`select_option(bid, option)`	Select an option from the element identified by `bid`
`done()`	Terminate the task as successfully completed
`report_infeasible()`	Terminate the task as infeasible
`noop(wait_ms)`	Wait without performing an interaction
Coordinate-based Actions	
`scroll_at(x, y, dx, dy)`	Scroll by an offset at a viewport coordinate
`mouse_move(x, y)`	Move the pointer to a viewport coordinate
`mouse_click(x, y)`	Click at a viewport coordinate
`keyboard_type(text)`	Type a string of text using the keyboard
`go_back()`	Navigate to the previous page
`go_forward()`	Navigate to the next page
`goto(url)`	Navigate the browser to a URL
`done()`	Terminate the task as successfully completed
`report_infeasible()`	Terminate the task as infeasible
`noop(wait_ms)`	Wait without performing an interaction

format, memory mechanism, and termination criteria, are identical. Table 6 summarizes the inputs provided under each observation setting.

Table 6: **Prompt inputs for general-purpose VLMs.** All settings share the same prompt template; only the available action space and observation fields differ.

Setting	Action Interface	Inputs
$\mathcal{T}$	BID	Goal, accessibility tree
$\mathcal{T} + \mathcal{V}$	BID	Goal, accessibility tree, screenshot
$\mathcal{T} + \mathcal{V}_{\text{SoM}}$	BID	Goal, accessibility tree, SoM screenshot
$\mathcal{V}$	Coordinate	Goal, screenshot
$\mathcal{V} + \mathcal{T}_{\text{view}}$	Coordinate	Goal, screenshot, viewport accessibility tree

```
# SYSTEM PROMPT
You are a web UI agent. Your goal is to complete the user's browser task.

{input_description}

Executable browser actions:
{action_definitions}

Only a single action can be provided at once.
Grounding:
{grounding_instructions}

Response format:
- Return exactly one JSON object and no Markdown fence or surrounding
text.
- The object must contain exactly these fields: thought, action, memory.
```

```
- thought is useful step-local analysis.
- action is exactly one object with a name and an arguments object.
- memory is a concise full rewrite of persistent memory for future steps.
- Keep memory task-relevant: target entities, completed subgoals,
outcomes, important UI facts, and failed approaches.

Example response:
{"thought":"the Save button is the next required control.","action":{"
name":"click","arguments":{"bid":"123"}},"memory":"The form has been
filled; saving is the remaining step."}

Review the user instruction and current page state, then choose the best
next action.

# USER PROMPT
# Goal
{goal}

# Open Tabs
{website_title_and_url}

{observation_fields}

# Previous Memory
{memory}

{last_action_error}
```

For each setting, `action_definitions` is replaced by the corresponding action set from Table 5, while `observation_fields` is populated with the observations listed in the **Inputs** column of Table 6. Thus, $\mathcal{T}$ provides only the accessibility tree, whereas $\mathcal{T} + \mathcal{V}$ additionally provides the current screenshot. For $\mathcal{T} + \mathcal{V}_{\text{SoM}}$, the screenshot is replaced by its SoM-annotated version. The coordinate-based settings similarly provide either $\mathcal{V}$ alone or $\mathcal{V} + \mathcal{T}_{\text{view}}$.

D.4 Training Details

D.4.1 Training Data and Regimes

WebArena-Infinity provides 1,140 tasks across the nine websites retained in WebVariants. We first remove the 228 tasks used for WebVariants evaluation, leaving 912 tasks that are disjoint from the evaluation set. We designate elation-patient-communication and paypal-my-wallet as held-out websites and exclude them entirely from training. The remaining seven websites serve as source websites. From the tasks available on these source websites, we select 176 training tasks while approximately preserving the original task distribution across websites.

For each selected task, we obtain a verified trajectory under all five implementations, yielding a pool of 880 trajectories. The training regimes in Section 5 are constructed from this pool as illustrated in Figure 8. Single-Implementation draws all 176 trajectories from one implementation, whereas Multi-Implementation uses the same 176 task identities but assigns each task to one of the other four implementations. For the budget allocation experiments, we keep the total number of trajectories fixed at 176 while varying the number of implementations represented per task: 176 tasks with one implementation each (i.e., Multi-Implementation), 88 tasks with two implementations each, and 44 tasks with four implementations each.

D.4.2 Training and Hyperparameters

We perform all supervised fine-tuning with LLaMA-Factory (Zheng et al., 2024b). We employ full fine-tuning with DeepSpeed ZeRO-2 and enable Liger Kernel and Unsloth garbage collection for memory efficiency. We use a maximum cutoff length of 20,000 tokens and train for two epochs with a cosine learning-rate schedule. The detailed hyperparameters are summarized in Table 7.

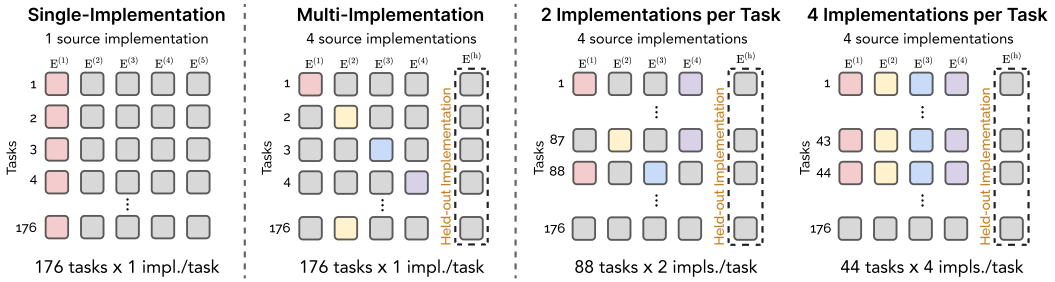


Figure 8: **Training regimes under a fixed trajectory budget.** All settings use 176 trajectories. Colored cells denote task–implementation pairs included in training, and E^h denotes the implementation excluded from training.

We use Qwen3.5-4B and Gemma-4-E2B-it as the backbone models for all experiments in Section 5, including the Single-Implementation, Multi-Implementation, and budget allocation experiments. The same training configuration is used across all regimes so that they differ only in the composition of the training data.

Table 7: Hyperparameters for supervised fine-tuning.

Hyperparameter	**Value**
Training Configuration	
Framework	LLaMA-Factory
Fine-tuning type	Full parameter
DeepSpeed	ZeRO-2
Precision	BF16
Liger Kernel	Enabled
Unsloth gradient checkpointing	Enabled
Number of epochs	2
Optimization	
Optimizer	AdamW
Learning rate	5×10^{-6}
Learning-rate schedule	Cosine with Warmup
Warmup ratio	0.05
Weight decay	0.01
Batch Configuration	
Per-device batch size	4
Gradient accumulation	1
Number of GPUs	4
Effective batch size	16
Input Processing	
Maximum sequence length	20,000 tokens
Image max pixels	921,600
Vision tower	Frozen
Multimodal projector	Frozen

E Additional Results and Analysis

E.1 Sensitivity to the Div Replacement Ratio

The E^{Div} environment replaces a fraction of interactive semantic elements with generic `div` elements while preserving their interaction behavior through JavaScript. To determine an appropriate perturbation strength for the main benchmark, we evaluate replacement ratios of

Table 8: **Sensitivity to the Div replacement ratio.** Pass@1 (%) as the fraction of interactive semantic elements replaced by generic `div` elements increases. A ratio of 0% corresponds to E^{Default}, and 25% is used as E^{Div} in the main benchmark.

Model	Obs.	Grounding	Replacement Ratio (%)				
			0 (E^{Default})	25 (E^{Div})	50	75	100
Qwen3.5-35B-A3B	$\mathcal{T}$		49.56	29.39	12.72	7.89	3.51
	$\mathcal{T}+\mathcal{V}$	BID	44.74	26.32	12.28	7.89	4.72
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$		42.11	32.89	27.63	25.88	10.09
Qwen3.5-9B	$\mathcal{T}$		42.98	18.42	6.58	4.82	1.32
	$\mathcal{T}+\mathcal{V}$	BID	28.51	20.18	7.46	4.82	1.32
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$		28.95	23.68	18.42	17.11	12.72
Qwen3.5-4B	$\mathcal{T}$		30.26	17.11	6.14	5.70	0.88
	$\mathcal{T}+\mathcal{V}$	BID	33.77	16.23	5.70	3.95	0.44
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$		31.58	27.63	17.98	16.67	10.96
Gemma-4-31B-it	$\mathcal{T}$		50.88	25.44	7.02	4.82	0.88
	$\mathcal{T}+\mathcal{V}$	BID	51.75	26.32	8.33	3.95	1.32
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$		46.49	39.91	28.07	22.37	17.11
Gemma-4-26B-A4B-it	$\mathcal{T}$		25.44	15.79	5.70	3.51	0.88
	$\mathcal{T}+\mathcal{V}$	BID	25.88	16.67	6.58	3.95	0.88
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$		25.44	21.93	18.42	13.60	10.96
Gemma-4-E4B-it	$\mathcal{T}$		32.02	13.60	3.07	1.75	0.00
	$\mathcal{T}+\mathcal{V}$	BID	22.81	12.28	2.63	1.32	0.44
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$		25.88	17.11	8.77	6.14	3.07

$\{0, 25, 50, 75, 100\}\%$, where 0% corresponds to E^{Default} and 25% corresponds to the E^{Div} used throughout the main experiments.

Table 8 reports Pass@1 (%) for this analysis. We use Pass@1 rather than Passˆ3 to better reveal the performance trend across replacement ratios, as Passˆ3 approaches zero for many models at the higher ratios. Performance generally decreases as a larger fraction of interactive semantic elements is replaced, with particularly sharp degradation beyond 25%.

We select 25% as the primary setting because it introduces a substantial change in the structural and semantic cues available to agents without turning the variant into an extreme stress condition. At higher replacement ratios, the removal of semantic structure becomes sufficiently severe that performance collapses for many models. Including such a variant in the main benchmark with five implementation would make aggregate implementation robustness disproportionately reflect the severity of E^{Div}, rather than sensitivity across multiple forms of implementation variation. The 25% setting therefore provides a moderate perturbation that remains informative alongside the other implementation variants.

Interestingly, the results reveal an additional pattern: $\mathcal{T}+\mathcal{V}_{\text{SoM}}$ degrades substantially more slowly as the replacement ratio increases. While performance under $\mathcal{T}$ and $\mathcal{T}+\mathcal{V}$ approaches zero for most models at high replacement ratios, performance with explicit SoM grounding remains considerably higher. This suggests that explicit visual grounding becomes particularly useful as semantic structure in the document representation is weakened.

E.2 All results on WebVariants

Table 11 reports the full results for WebVariants, which are omitted from the main text due to space constraints. It includes per-implementation Passˆ3, task coverage C, implementation robustness R, and F_1^{impl} for all evaluated agents and observation settings. These results provide the complete values underlying the summaries and visualizations in the main text.

E.3 Agent Ranking Correlation Across Implementations

Section 4.2 reports Kendall's τ between the agent ranking under E^{Default} and the rankings under the four alternative implementations. Figure 9 extends this analysis to all ten pairwise comparisons among the five implementations using the same set of agents. Ranking agreement varies substantially across implementation pairs, indicating that the instability observed in Section 4.2 is not an artifact of choosing E^{Default} as the reference.

The lowest agreement occurs between E^{Poor} and E^{Icon}, with $\tau = 0.00$ and a 95% CI of $[-0.286, 0.377]$, indicating almost no concordance in agent ranking between the two implementations. Other implementation pairs also exhibit relatively weak agreement, such as E^{Poor} and E^{Flat} ($\tau = 0.114$). These results further indiate that the relative ranking of web agents can depend considerably on how a website is implemented.

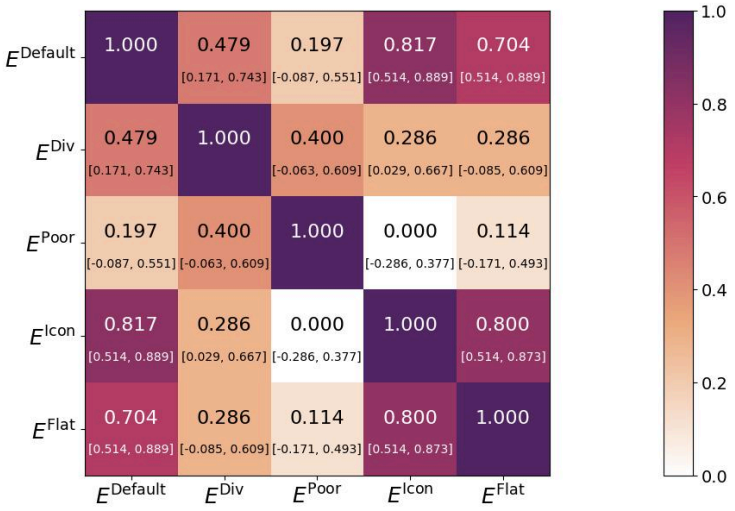


Figure 9: Each cell reports Kendall's τ between the agent rankings under a pair of implementations, using the same agents as in Section 4.2. Higher values indicate greater agreement in agent ordering.

E.4 Coordinate-based Interface

E.4.1 Coordinate Grounding Calibration

Coordinate-based interaction requires accurate visual grounding in addition to language understanding. We therefore first evaluate the visual-grounding capabilities of the general-purpose VLMs for coordinate-based interaction before using them as coordinate-based web agents. We construct a separate grounding set of 1,332 examples from the MolmoWeb-SyntheticGround dataset (Gupta et al., 2026). Each example consists of a webpage screenshot, an atomic task (e.g., subscribe to newsletters) and an interaction target (e.g., bbox $[32.0, 529.84, 332.0, 577.84]$). Given the screenshot and task instruction, the model predicts the next action with the corresponding interaction coordinate. The model use the screenshot and instruction and predict the the next action and corresponding interaction coordinate using th same output format as a single step in the web agent rollout.

Gemma4 predicts coordinates in a normalized $[0, 1000]$ coordinate space. We therefore linearly rescale its predicted coordinates to the pixel dimensions of the input screenshot before evaluating grounding accuracy or executing the corresponding browser action. Figure 10 shows that Gemma4-31B-it and Gemma4-26B-A4B-it achieve the strongest coordinate grounding performance among the general-purpose VLMs, although they still underperform specialized coordinate-based web agents. We therefore use these two Gemma4 models for the controlled coordinate-based experiments below.

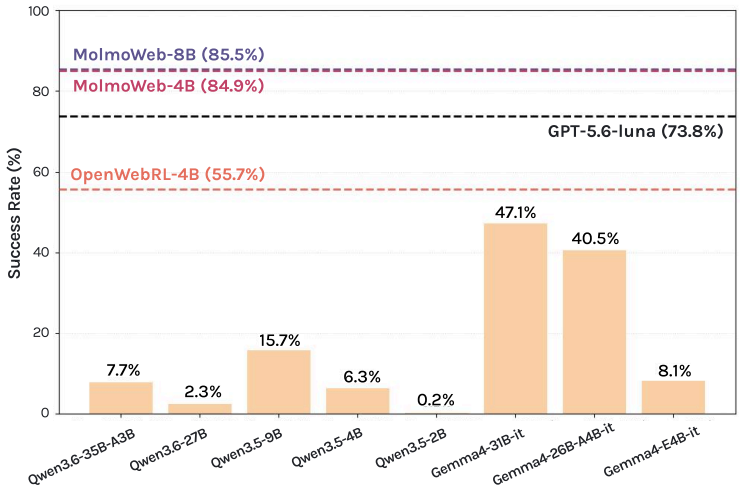


Figure 10: Coordinate grounding performance of general-purpose VLMs. Success rate on a separate set of 1,332 examples constructed from MolmoWeb-SyntheticGround (Gupta et al., 2026). Dashed lines show the grounding performance of specialized coordinate-based web agents.

E.4.2 Coordinate-Based Web Evaluation

We evaluate each model with either $\mathcal{V}$ or $\mathcal{V}+\mathcal{T}_{\text{view}}$, where $\mathcal{T}_{\text{view}}$ contains only accessibility tree nodes corresponding to elements visible in the current viewport. Restricting the textual observation to the viewport avoids providing information about off-screen elements that have no visual counterpart in $\mathcal{V}$.

Table 9 shows that coordinate-based VLMs remain substantially sensitive to website implementation. For both models, success rates vary considerably across the five implementations under both $\mathcal{V}$ and $\mathcal{V}+\mathcal{T}_{\text{view}}$, indicating that implementation sensitivity persists under coordinate-based interaction. Their overall performance is also substantially lower than that of the BID-based VLMs in the main experiments. Even with the additional viewport-restricted accessibility tree observation, the strongest coordinate-based configuration remains below the BID-based Qwen3.5-4B configuration.

Table 9: Coordinate-based results on WebVariants. Gemma4 models are evaluated with screenshot-only observation $\mathcal{V}$ and with viewport-restricted axtree $\mathcal{V}+\mathcal{T}_{\text{view}}$. Results report Pass^3 (%) for each implementation, together with C, R, and F_1^{impl}.

Model	Obs.	Grounding	Environments					$C\uparrow$	$R\uparrow$	$F_1^{\text{impl}}\uparrow$
			Default	Div-25%	Poor-A11y	Icon-UI	Flat-UI			
Gemma4-31B-it	$\mathcal{V}$	Coordinate	2.19	2.63	4.39	2.63	0.88	5.26	48.33	9.49
	$\mathcal{V}+\mathcal{T}_{\text{view}}$		8.33	9.21	8.77	12.28	5.26	17.54	50.00	25.97
Gemma4-26B-A4B-it	$\mathcal{V}$		1.75	1.75	0.88	1.32	0.88	3.07	42.86	5.73
	$\mathcal{V}+\mathcal{T}_{\text{view}}$		1.75	3.07	1.32	3.51	0.44	4.82	41.82	8.65

E.4.3 Grounding Failures During Web Interaction

We further examine how frequently coordinate-based agents fail to produce an observable state change when executing mouse clicks during full web agent rollouts. Table 10 reports statistics aggregated across all 1,140 task–implementation pairs and averaged over three runs, including the total number of mouse-click actions, the number and fraction of clicks that produce no observable state change, and the fraction of task–implementation pairs containing at least one such click.

General-purpose VLMs produce substantially more clicks that result in no observable state change than OpenWebRL-4B. Only 11.2% of OpenWebRL-4B clicks show this behavior, compared with 65.0–78.5% for the Gemma4 configurations. Although the absence of an observable state change does not necessarily imply incorrect grounding, the large gap provides further evidence that basic visual grounding remains a major bottleneck for general-purpose VLMs under coordinate-based

interaction. Consequently, their low coordinate-based performance is difficult to interpret solely in terms of implementation robustness, since failures to locate and interact with webpage elements constitute a substantial source of failure during rollouts.

Table 10: **Mouse-click actions with no observable state change.** Statistics are aggregated over all task–implementation rollouts. The final column reports the fraction of tasks containing at least one such click.

Model	Obs.	# Clicks	# No-change	No-change (%)	Tasks w/ $\geq$1 (%)
OpenWebRL-4B	$\mathcal{V}$	10,675	1,195	11.2	38.3
Gemma4-31B-it	$\mathcal{V}$	38,352	24,924	65.0	82.1
	$\mathcal{V} + \mathcal{T}_{\text{view}}$	34,368	22,458	65.3	82.8
Gemma4-26B-A4B-it	$\mathcal{V}$	35,153	24,885	70.8	87.5
	$\mathcal{V} + \mathcal{T}_{\text{view}}$	40,082	31,466	78.5	96.2

E.5 Benefits from Observation Augmentation Are Offset by Regressions

Section 4.3 shows that adding visual information or explicit grounding cues often yields only limited or implementation-dependent net improvements. To better understand these aggregate effects, we examine how the set of successfully completed tasks changes under each observation modification.

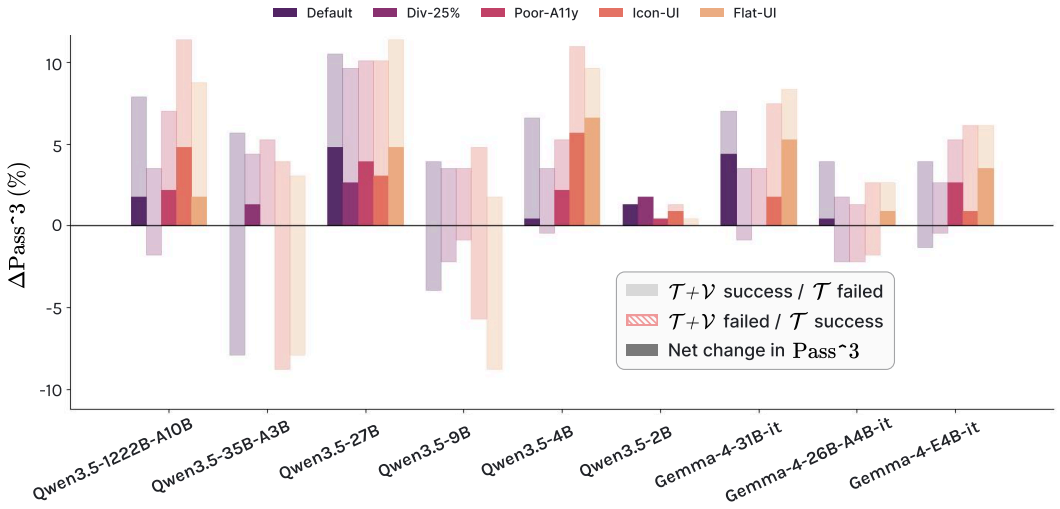
Figure 11 reveals substantial task-level gains even when the net change in Passˆ3 is small. Adding screenshots enables many tasks that fail under $\mathcal{T}$ to be solved, but the gap between these gains and the resulting net performance change indicates that many tasks previously solved under $\mathcal{T}$ fail under $\mathcal{T} + \mathcal{V}$. For $\mathcal{T} \rightarrow \mathcal{T} + \mathcal{V}$ in particular, these regressions often offset much of the benefit from newly solved tasks. A similar pattern appears when adding SoM grounding, although the balance between gains and regressions varies more strongly across implementations. These results show that the limited aggregate benefit of observation augmentation does not imply that additional visual information or grounding cues are unhelpful; gains on some tasks are often offset by regressions on others.

E.6 Trained Agents Results

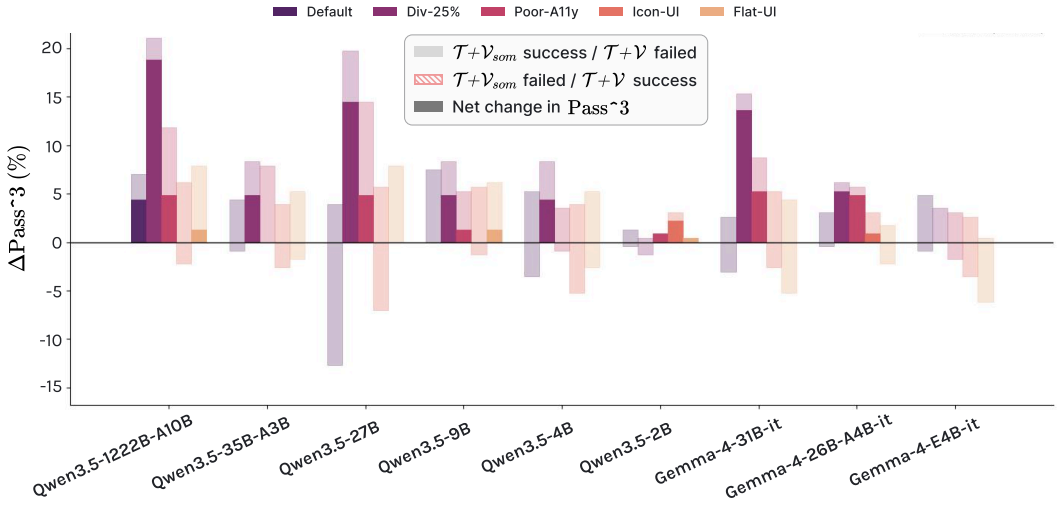
Tables 12 and 13 provide the full results for the trained agents evaluated in Section 5.2. They report per-implementation Passˆ3 and the aggregate metrics C, R, and F_1^{impl} on the source and held-out websites, respectively.

Table 11: Results are reported for specialized web agents and general-purpose VLMs under different observation and grounding interfaces. We report Pass^3 (%) for each implementation, with task coverage C, implementation robustness R, and F_1^{impl}. $\mathcal{T}$ denotes the accessibility tree, $\mathcal{V}$ the screenshot, and $\mathcal{V}_{\text{SoM}}$ the screenshot with Set-of-Marks.

Model	Obs.	Grounding	Environments					$C\uparrow$	$R\uparrow$	$F_1^{\text{impl}}\uparrow$
			Default	Div-25%	Poor-A11y	Icon-UI	Flat-UI			
Unimodal Web Agents										
Go-Browse-WA-7B	$\mathcal{T}$	BID	11.84	8.33	3.07	17.11	10.09	22.37	45.10	29.90
BrowserAgent-RFT (7B)	$\mathcal{T}$	BID	8.33	3.95	3.51	10.96	8.33	14.04	50.00	21.92
UI-TARS-1.5-7B	$\mathcal{V}$	Coordinate	5.70	5.26	7.89	3.51	3.07	10.53	48.33	17.29
Holo1-7B	$\mathcal{V}$	Coordinate	3.07	3.51	4.39	2.19	0.88	7.02	40.00	11.94
MolmoWeb-8B	$\mathcal{V}$	Coordinate	6.58	8.33	4.82	5.26	0.88	12.72	40.69	19.38
MolmoWeb-4B	$\mathcal{V}$	Coordinate	6.14	4.82	4.39	5.70	1.75	10.09	44.35	16.44
OpenWebRL-4B	$\mathcal{V}$	Coordinate	12.72	9.65	14.91	10.53	6.14	22.37	48.24	30.56
Vision Language Models (Closed-source)										
GPT-5.6-luna	$\mathcal{T}$	BID	43.42	21.05	24.12	46.49	41.67	63.60	55.59	59.32
	$\mathcal{T}+\mathcal{V}$	BID	50.88	28.51	34.21	53.95	51.75	71.93	60.98	66.00
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	41.23	34.21	32.02	40.35	37.72	54.39	68.23	60.52
Gemini-3.5-Flash-Lite	$\mathcal{T}$	BID	41.23	29.82	20.18	46.05	42.11	58.33	61.50	59.88
	$\mathcal{T}+\mathcal{V}$	BID	46.93	32.46	26.75	47.37	45.18	61.84	64.26	63.03
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	43.86	37.28	31.14	46.05	43.42	58.33	69.17	63.29
Vision Language Models (Open-source)										
Qwen3.5-122B-A10B	$\mathcal{T}$	BID	49.12	25.00	23.68	50.00	47.37	66.23	58.94	62.37
	$\mathcal{T}+\mathcal{V}$	BID	50.88	23.25	25.88	54.82	49.12	66.23	61.59	63.82
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	50.88	42.11	30.70	52.63	50.44	66.23	68.48	67.33
Qwen3.5-35B-A3B	$\mathcal{T}$	BID	36.40	16.23	17.11	39.47	36.84	49.12	59.46	53.80
	$\mathcal{T}+\mathcal{V}$	BID	28.51	17.54	17.11	30.70	28.95	41.67	58.95	48.82
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	27.63	22.37	17.11	28.07	27.19	36.84	66.43	47.40
Qwen3.5-27B	$\mathcal{T}$	BID	50.44	25.44	25.88	53.95	47.81	64.91	62.70	63.79
	$\mathcal{T}+\mathcal{V}$	BID	55.26	28.07	29.82	57.02	52.63	69.30	64.30	66.71
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	42.54	42.54	34.65	50.00	52.63	65.35	68.05	66.67
Qwen3.5-9B	$\mathcal{T}$	BID	22.37	10.96	10.09	21.93	23.68	34.21	52.05	41.29
	$\mathcal{T}+\mathcal{V}$	BID	18.42	8.77	9.21	16.23	14.91	27.63	48.89	35.31
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	18.42	13.60	10.53	14.91	16.23	29.39	50.15	37.06
Qwen3.5-4B	$\mathcal{T}$	BID	17.11	7.02	6.14	12.28	10.09	27.19	38.71	31.95
	$\mathcal{T}+\mathcal{V}$	BID	17.54	6.58	8.33	17.98	16.67	29.39	45.67	35.76
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	14.04	10.96	7.46	12.72	14.04	25.00	47.37	32.73
Qwen3.5-2B	$\mathcal{T}$	BID	0.44	0.00	0.00	0.44	0.44	0.88	30.00	1.70
	$\mathcal{T}+\mathcal{V}$	BID	1.75	1.75	0.44	1.32	0.44	3.95	28.89	6.95
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	1.32	0.44	1.32	3.51	0.88	5.26	28.33	8.88
Gemma4-31B-it	$\mathcal{T}$	BID	39.91	20.18	21.93	45.18	38.16	53.51	61.80	57.36
	$\mathcal{T}+\mathcal{V}$	BID	44.30	19.30	21.93	46.93	43.42	56.14	62.66	59.22
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	41.23	32.89	27.19	44.30	38.16	55.26	66.51	60.37
Gemma4-26B-A4B-it	$\mathcal{T}$	BID	19.30	12.72	7.89	21.49	19.30	27.63	58.41	37.52
	$\mathcal{T}+\mathcal{V}$	BID	19.74	10.53	5.70	19.74	20.18	26.32	57.67	36.14
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	19.30	15.79	10.53	20.61	17.98	25.88	65.08	37.03
Gemma4-E4B-it	$\mathcal{T}$	BID	19.30	9.21	5.70	16.23	14.47	24.12	53.82	33.31
	$\mathcal{T}+\mathcal{V}$	BID	17.98	8.77	8.33	17.11	17.98	27.19	51.61	35.62
	$\mathcal{T}+\mathcal{V}_{\text{SoM}}$	BID	17.11	8.77	6.58	13.60	16.23	26.75	46.56	33.98



(a) $\mathcal{T} \rightarrow \mathcal{T} + \mathcal{V}$



(b) $\mathcal{T} + \mathcal{V} \rightarrow \mathcal{T} + \mathcal{V}_{SoM}$

Figure 11: **Task-level gains and regressions from richer observation interfaces.** Light bars show the fraction of tasks that fail under the original observation but succeed after the observation change. Dark bars show the resulting net change in Pass^3. The difference between the two corresponds to tasks that were previously successful but become unsuccessful. (a) $\mathcal{T} \rightarrow \mathcal{T} + \mathcal{V}$. (b) $\mathcal{T} + \mathcal{V} \rightarrow \mathcal{T} + \mathcal{V}_{\text{SoM}}$.

Table 12: Full results on the seven source websites. **w/o** indicates that the corresponding implementation is excluded from Multi-Implementation training, while **only** indicates the implementation used for Single-Implementation training.

Model	Training Regime	Setting	Default	Div-25%	Poor-A11y	Icon-UI	Flat-UI	C	R	F_1^{impl}
Qwen3.5-4B	Base ($\mathcal{T}$)	–	13.16	5.26	5.70	9.65	8.33	20.61	40.85	27.40
	Base ($\mathcal{T}$+$\mathcal{V}$)	–	14.04	4.82	5.70	13.60	11.84	22.37	44.71	29.82
	Multi-Implementation	w/o Default	26.32	18.86	14.04	30.26	30.26	39.91	60.00	47.94
		w/o Div-25%	26.75	21.49	16.67	32.89	30.26	41.67	61.47	49.67
		w/o Poor-A11y	31.14	26.01	15.35	31.14	34.21	43.86	60.40	50.82
		w/o Icon-UI	28.07	21.93	16.23	28.51	31.58	40.35	62.61	49.07
		w/o Flat-UI	27.63	18.42	17.98	27.63	29.82	39.47	61.56	48.10
	Single-Implementation	Default only	25.00	12.28	12.28	21.05	24.56	33.33	53.95	41.21
		Div-25% only	22.37	19.74	12.72	23.25	21.05	32.89	58.93	42.22
		Poor-A11y only	21.05	13.16	14.47	21.49	17.54	30.70	55.71	39.59
		Icon-UI only	23.68	12.28	9.21	28.07	24.12	34.65	53.92	42.19
		Flat-UI only	24.56	14.47	10.96	22.81	23.68	34.21	54.87	42.15
Gemma4-E2B-it	Base ($\mathcal{T}$)	–	4.39	1.75	1.32	2.63	2.63	7.02	36.25	11.76
	Base ($\mathcal{T}$+$\mathcal{V}$)	–	3.51	0.88	0.44	3.51	2.19	5.26	40.00	9.30
	Multi-Implementation	w/o Default	10.53	7.89	5.26	10.09	7.02	17.54	46.50	25.48
		w/o Div-25%	8.33	4.39	3.95	7.46	8.33	15.79	41.11	22.82
		w/o Poor-A11y	8.77	5.70	5.70	9.21	8.33	17.11	44.10	24.65
		w/o Icon-UI	10.53	5.70	5.70	7.02	7.89	15.35	48.00	23.26
		w/o Flat-UI	7.89	5.70	4.39	8.33	9.21	15.79	45.00	23.38
	Single-Implementation	Default only	9.21	5.26	4.82	8.77	6.58	13.16	52.67	21.06
		Div-25% only	7.46	4.82	6.14	8.33	7.46	16.67	41.05	23.71
		Poor-A11y only	7.02	5.70	4.39	6.58	5.70	13.60	43.23	20.69
		Icon-UI only	10.96	8.33	6.14	10.09	10.96	17.54	53.00	26.36
		Flat-UI only	8.77	4.82	4.39	8.77	9.21	14.47	49.70	22.42

Table 13: Full results on the two held-out websites. **w/o** indicates that the corresponding implementation is excluded from Multi-Implementation training, while **only** indicates the implementation used for Single-Implementation training.

Model	Training Regime	Impl.	Default	Div-25%	Poor-A11y	Icon-UI	Flat-UI	C	R	F_1^{impl}
Qwen3.5-4B	Base ($\mathcal{T}$)	–	17.31	7.69	1.92	11.54	7.69	28.85	32.00	30.34
	Base ($\mathcal{T}$+$\mathcal{V}$)	–	15.38	7.69	11.54	19.23	21.15	30.77	48.75	37.73
	Multi-Implementation	w/o Default	38.46	15.38	23.08	25.00	23.08	44.23	56.52	49.63
		w/o Div-25%	42.31	19.23	25.00	36.54	28.85	50.00	60.77	54.86
		w/o Poor-A11y	30.77	23.08	28.85	42.31	25.00	51.92	57.78	54.69
		w/o Icon-UI	42.31	17.31	25.00	34.62	26.92	53.85	54.29	54.07
		w/o Flat-UI	19.23	17.31	25.00	34.62	23.08	44.23	53.91	48.59
	Single-Implementation	Default only	30.77	11.54	19.23	25.00	26.92	38.46	52.00	44.22
		Div-25% only	25.00	19.23	21.15	28.85	25.00	40.38	51.43	45.24
		Poor-A11y only	23.08	11.54	19.23	19.23	17.31	38.46	46.00	41.89
		Icon-UI only	28.85	11.54	23.08	36.54	19.23	40.38	52.38	45.61
		Flat-UI only	25.00	9.62	13.46	21.15	21.15	30.77	55.00	39.46
Gemma4-E2B-it	Base ($\mathcal{T}$)	–	11.54	7.69	1.92	9.62	7.69	17.31	44.44	24.91
	Base ($\mathcal{T}$+$\mathcal{V}$)	–	3.85	5.77	3.85	9.62	5.77	13.46	42.86	20.49
	Multi-Implementation	w/o Default	7.69	3.85	1.92	11.54	15.38	19.23	42.00	26.38
		w/o Div-25%	13.46	5.77	7.69	11.54	11.54	23.08	43.33	30.12
		w/o Poor-A11y	13.46	3.85	5.77	13.46	17.31	28.85	41.43	32.64
		w/o Icon-UI	11.54	1.92	9.62	13.46	13.46	25.00	40.00	30.77
		w/o Flat-UI	11.54	3.85	9.62	13.46	17.31	23.08	48.33	31.24
	Single-Implementation	Default only	11.54	1.92	5.77	11.54	15.38	21.15	43.64	28.49
		Div-25% only	1.92	3.85	9.62	11.54	1.92	17.31	33.33	22.78
		Poor-A11y only	15.38	3.85	1.92	9.62	15.38	23.08	40.00	29.27
		Icon-UI only	11.54	5.77	3.85	11.54	9.62	17.31	48.89	25.56
		Flat-UI only	11.54	3.85	3.85	11.54	7.69	21.15	36.36	26.75

Default E^{Default}	Div-25% E^{Div}
RootWebArea 'GitLab — Plan & Track', focused banner '' image '' StaticText 'Plan & Track' StaticText 'AcmeCorp' StaticText '/' StaticText 'platform-v4' button 'To-Do List' image '' StaticText '9' StaticText 'SC' navigation '' StaticText 'PLAN' button 'Issues 57' image '' StaticText 'Issues' StaticText '57' button 'Boards' image '' StaticText 'Boards' button 'Epics' image '' StaticText 'Epics' button 'Milestones' image '' StaticText 'Milestones' button 'Iterations' image '' StaticText 'Iterations' button 'Labels' image '' StaticText 'Labels' StaticText 'PERSONAL' button 'To-Do List 9' image '' StaticText 'To-Do List' StaticText '9' main '' heading 'Issues' button 'New issue' button 'Open 57' StaticText '57' button 'Closed 10' StaticText '10' button 'All' textbox 'Search issues...' button 'Assignee' button 'Milestone' button 'Label' button 'Sort' button 'Open issue Fix typo in 404 error page message' image '' button 'Fix typo in 404 error page message' StaticText '#53' StaticText 'type' StaticText 'bug' StaticText 'priority' StaticText 'low' StaticText 'good-first-issue' StaticText 'component' StaticText 'frontend' StaticText '1' StaticText '6mo ago' button 'Open issue Add tooltip to truncated sidebar labels' image '' button 'Add tooltip to truncated sidebar labels' StaticText '#54' ...	RootWebArea 'GitLab — Plan & Track', focused banner '' image '' StaticText 'Plan & Track' StaticText 'AcmeCorp' StaticText '/' StaticText 'platform-v4' button 'To-Do List' image '' StaticText '9' StaticText 'SC' navigation '' StaticText 'PLAN' image '' StaticText 'Issues' StaticText '57' image '' StaticText 'Boards' image '' staticText 'Epics' image '' StaticText 'Milestones' image '' StaticText 'Iterations' image '' StaticText 'Labels' StaticText 'PERSONAL' image '' StaticText 'To-Do List' StaticText '9' main '' heading 'Issues' button 'New issue' button 'Open 57' StaticText '57' button 'Closed 10' StaticText '10' button 'All' textbox 'Search issues...' button 'Assignee' button 'Milestone' button 'Label' StaticText 'Sort' button 'Open issue Fix typo in 404 error page message' image '' StaticText 'Fix typo in 404 error page message' StaticText '#53' StaticText 'type' StaticText 'bug' StaticText 'priority' StaticText 'low' StaticText 'good-first-issue' StaticText 'component' StaticText 'frontend' StaticText '1' StaticText '6mo ago' button 'Open issue Add tooltip to truncated sidebar labels' image '' StaticText 'Add tooltip to truncated sidebar labels' StaticText '#54' ...

Figure 12: An example comparison of accessibility tree observations for the same state under E^{Default} and E^{Div}. Despite identical user-facing functionality, the two implementations expose different accessibility tree structure and textual cues to available the agent. Highlighted regions mark corresponding elements whose representations differ across implementations.

Default E^{Default}	Poor-A11y E^{Poor}
RootWebArea 'Gmail', focused banner '' button 'Main menu' image '' image '' StaticText 'Gmail' button 'Search' image '' textbox 'Search mail' button 'Show search options' image '' button 'Settings' image '' StaticText 'AJ' navigation '' button 'Compose' image '' StaticText 'Compose' button 'Inbox 19' image '' StaticText 'Inbox' StaticText '19' ... main '' checkbox '', checked='false' button '▾' button 'Refresh' image '' StaticText 'Refresh' StaticText '1-48 of 48' button 'Prev', disabled=True image '' StaticText 'Prev' button 'Next', disabled=True image '' StaticText 'Next' button 'Primary 9' image '' StaticText 'Primary' StaticText '9' button 'Social 4' image '' StaticText 'Social' StaticText '4' button 'Promotions 2' image '' StaticText 'Promotions' StaticText '2' button 'Updates 3' image '' StaticText 'Updates' StaticText '3' button 'Forums 1' image '' StaticText 'Forums' StaticText '1' button 'Open email from Sarah Chen: Q1 Product Roadmap Review' checkbox '', checked='false' button '☆' button StaticText 'Sarah Chen' StaticText 'Q1 Product Roadmap Review' StaticText 'Work' StaticText 'Projects' StaticText '-' ...	RootWebArea '', focused button '' image '' image '' StaticText 'Gmail' button '' image '' textbox '' button '' image '' button '' image '' StaticText 'AJ' button 'Compose' image '' StaticText 'Compose' button 'Inbox 19' image '' StaticText 'Inbox' StaticText '19' ... checkbox '', checked='false' button '▾' button '' image '' StaticText '1-48 of 48' button '', disabled=True image '' button '', disabled=True image '' button 'Primary 9' image '' StaticText 'Primary' StaticText '9' button 'Social 4' image '' StaticText 'Social' StaticText '4' button 'Promotions 2' image '' StaticText 'Promotions' StaticText '2' button 'Updates 3' image '' StaticText 'Updates' StaticText '3' button 'Forums 1' image '' StaticText 'Forums' StaticText '1' checkbox '', checked='false' button '☆' button StaticText 'Sarah Chen' StaticText 'Q1 Product Roadmap Review' StaticText 'Work' StaticText 'Projects' StaticText '-' ...

Figure 13: An example comparison of accessibility tree observations for the same state under E^{Default} and E^{Poor}. Despite identical user-facing functionality, removing roles, ARIA attributes, and alternative text changes the accessibility tree representation and reduces the semantic cues available to the agent. Highlighted regions mark corresponding elements whose representations differ across implementations.