

WebAccessBench: Digital Accessibility Reliability in LLM-Generated Web Interfaces

Wednesday 25th February, 2026

Casey Kreer
casey@conesible.de

Abstract—Today I am introducing WebAccessBench, a novel benchmark for AI language models to assess accessibility quality and WCAG conformance in generated web interfaces under realistic prompting conditions. In this paper, I report the current benchmark snapshot, describe the evaluation framework, and document the gap between older/smaller model families and newer flagship systems. I provide aggregate quantitative findings that are useful for model selection, risk triage, and policy design. The main empirical result is a two-regime pattern: older/smaller models show lower raw error counts but much higher element-normalized error density, while newer flagship models show higher absolute error totals with substantially lower error density per DOM structure. Guidance improves outcomes across both cohorts, but the overall data shows that guiding a model with expert-grade prompts has very little effect over a small nudge. The benchmark results suggest that objective error count is too high to rely on LLM technology at all in digital accessibility work, even under explicit expert guidance.

I. INTRODUCTION

Large language models (LLMs) are increasingly used as front-end code generators in prototyping and production workflows. In principle, this shift could reduce accessibility defects by making correct patterns cheap to reproduce at scale, provided that models get trained and fine-tuned on modern guidance for semantic HTML and assistive technology compatibility. In practice, however, there is limited empirical evidence that current LLMs reliably produce WCAG-conformant web interfaces under realistic prompting. This matters because accessibility failures are not merely cosmetic. A single defect can block core tasks for disabled users and can create legal and operational risk in jurisdictions where accessibility requirements apply, like through Section 508 of the USA Rehabilitation Act and the European Accessibility Act.

Most established model evaluations in software engineering report a single aggregated success rate. For accessibility, such reporting can be misleading. Outputs vary substantially in structural scope, complexity and verbosity, so absolute defect counts, structure-normalized defect rates, and the frequency of fully clean outputs may disagree and can induce ranking reversals across models. A model that produces short, sparse interfaces can appear favorable on raw defect totals while remaining highly defect-dense when normalized by output structure. Conversely, a model that generates larger, more stylized interfaces can accumulate more total defects while

exhibiting lower defect density per unit of DOM structure. Because deployment decisions depend on the expected interface scope as well as on tolerance for residual risk, accessibility reliability requires a multi-metric evaluation framing.

WebAccessBench addresses this need by benchmarking accessibility outcomes of LLM-generated web interfaces under three guidance regimes that reflect common developer behavior: a baseline task instruction, a minimal accessibility nudge, and an expert-oriented prompt that enumerates typical problems and exact implementation details. For comparability with other industry standard benchmarks such as SWE-bench and "Humanity's Last Exam", WebAccessBench also defines an aggregate score, but treats it as a convenience rather than a sufficient summary.

II. EVALUATION FRAMING AND METHODOLOGY

WebAccessBench measures accessibility reliability in LLM-generated web interfaces under controlled and repeatable prompting conditions. It is designed to approximate common usage patterns in AI-assisted front-end work: a developer provides a standalone UI task, receives a complete HTML/CSS/JavaScript artifact, and evaluates whether the resulting interface exhibits detectable accessibility defects. The unit of analysis is one generated artifact for one task under one guidance condition.

A. Guidance conditions

Each task is evaluated under three prompt regimes intended to separate baseline behavior from responsiveness to accessibility intent:

- 1) **Unguided:** The prompt specifies only the UI to be built (for example, "Make a login form card") without accessibility framing.
- 2) **Little guidance:** The same prompt is augmented with a minimal directive to consider accessibility (specifically, "Make it accessible").
- 3) **Expert guidance:** The prompt is augmented with explicit, implementation-oriented accessibility requirements that reflect common pitfalls for the given component type (for example, explicit label associations, error-summary focus management, and correct control semantics). These details were curated by a human digital accessibility expert with a disability.

These regimes are not intended to exhaust all possible prompting principles and styles. Rather, they operationalize three practical levels of developer intent that can be applied consistently across tasks and models.

B. Task design and generation protocol

The benchmark uses 150 standalone UI implementation tasks covering common interactive patterns such as forms, dialogs, navigation components, and small stateful widgets. Tasks are framed to require complete front-end artifacts rather than bounded components embedded in an existing framework. This choice removes confounding guardrails provided by application scaffolding (for example, design systems or framework conventions) and forces models to jointly handle structure, semantics, interaction logic, and UI state in a single output. Because vanilla web technologies are widely documented and require less API-specific knowledge than framework-dependent code, this framing is expected to be a favorable setting for model performance. Consequently, measured outcomes should be interpreted as optimistic relative to deployment in large, heterogeneous production codebases.

For each model and each task, WebAccessBench generates two independent samples per guidance condition in fresh, uninitialized conversations. This yields $(150 \times 3 \times 2 = 900)$ total generations per model. For reporting, the benchmark retains the better of the two attempts per task and guidance condition, defined as the sample with the lower measured defect burden, resulting in $(150 \times 3 = 450)$ scored artifacts per model. The best-of-two protocol reduces run-to-run variance and produces a more stable estimate of capability, but it is also a cherry-picked procedure and therefore constitutes a lower bound on expected real-world defect burden. The protocol is used to characterize what a model can achieve under modest retry behavior, not what it will do on an arbitrary single attempt.

In rare cases, a model fails to produce assessable HTML within two attempts (for example, non-HTML output or malformed markup that prevents evaluation). Such cases are excluded from metric aggregation and are recorded separately as non-assessable generations.

C. Automated accessibility assessment

Each scored artifact is evaluated with axe-core, an industry-standard automated accessibility testing library. The benchmark records each detected issue as a distinct defect instance associated with a concrete DOM element. Defects are mapped to a level A/AA WCAG 2.2 success criterion using a deterministic rule-based mapping. This is used for interpretation and aggregation, not to claim exhaustive WCAG coverage. Automated testing captures only a subset of accessibility requirements, and some classes of barriers (for example, content appropriateness, interaction predictability across assistive technologies, or nuanced focus order problems) are generally not detected at all. WebAccessBench therefore measures automated defect burden as a lower bound on total accessibility risk.

To preserve benchmark integrity for future model tests and to mitigate overfitting risk, the paper does not contain the full

harness composition, the complete prompt set, or filtering and post-processing internals. A representative sample of tasks and prompt stems is provided in Appendix Section A to illustrate the evaluation style and guidance regimes. Future model results will be published online, through <https://conesible.de/wab>.

D. Reported metrics

WebAccessBench reports multiple complementary metrics because accessibility reliability is not well captured by a single scalar summary:

- **Average defect burden (E):** mean number of detected accessibility defects per artifact.
- **Clean output rate (Z):** share of artifacts with zero detectable defects.
- **Structure-normalized burden (R_{dom}):** defects per 100 DOM elements, using the artifact’s DOM element count (D) for normalization.
- **Aggregate benchmark score:** a 0–100 score derived from E , R_{dom} , and Z .

Reporting these metrics jointly makes explicit when rankings differ under absolute versus normalized burden and when a model’s average performance obscures low reliability due to few fully clean outputs.

E. Aggregate benchmark score

To summarize model performance in a single number, WebAccessBench maps accessibility results to a 0–100 score. The score uses three inputs: average error count (E), average DOM elements (D), and zero-error share (Z , in percent).

First, element-normalized burden is computed as:

$$R_{\text{dom}} = \begin{cases} \frac{E}{D/100}, & D > 0 \\ E, & \text{otherwise.} \end{cases}$$

Then error burden is converted to quality using an exponential decay with half-life 2.0:

$$Q_{\text{err}} = 100 \cdot 2^{-E/2}, \quad Q_{\text{dom}} = 100 \cdot 2^{-R_{\text{dom}}/2}.$$

All values are clamped to $[0, 100]$.

Per guidance condition, the reported score is:

$$S_{\text{guidance}} = 0.50 \cdot Q_{\text{dom}} + 0.50 \cdot Z.$$

The model-level overall score (across all guidance conditions) is:

$$S_{\text{overall}} = 0.50 \cdot Q_{\text{err}} + 0.30 \cdot Q_{\text{dom}} + 0.20 \cdot Z.$$

Scores are rounded to the nearest integer in $[0, 100]$.

a) *Rationale:* The scoring function is designed to penalize errors strongly at low counts. Because the quality terms are defined by an exponential decay with half-life 2.0,

$$Q_{\text{err}} = 100 \cdot 2^{-E/2}, \quad Q_{\text{dom}} = 100 \cdot 2^{-R_{\text{dom}}/2},$$

each additional two errors (or two normalized errors) reduce the corresponding quality score by half. This is intentional: Even a single accessibility barrier, regardless of category, can be a major blocker for disabled users.

b) *Accounting for output size*: The metric also considers that larger outputs create more opportunities for issues. Therefore it uses an element-normalized burden: Errors per 100 DOM elements. At the same time, fixing issues tends to become harder as the DOM and surrounding code grow, because locating causes and validating fixes requires more effort in larger and more complex code bases. This motivates including both raw error burden (E) and normalized burden (R_{dom}) in the final reporting.

c) *Rewarding reliability*: In addition to average burden, the score includes the zero-error share Z (in percent), which captures how often a model produces fully error-free outputs. The guidance-specific score emphasizes size-normalized quality within a prompting condition, while the overall score prioritizes reducing the absolute average error count due to aforementioned reasons.

d) *Achieving zero errors*: All issues that contribute to E are defined to be programmatically detectable in the evaluation pipeline. Under this assumption, such defects should be systematically preventable through automated checks and regression testing, as well as proper post-training techniques applied by LLM vendors. Consequently, the metric is calibrated so that even small nonzero error burdens lead to a substantial score reduction.

III. CURRENT SNAPSHOT RESULTS

The snapshot analyzed here is the current benchmark artifact generated on February 20, 2026 (UTC), covering 19 models. Per-model values for this snapshot are reported in Appendix Tables I–IV.

At a global level:

- overall score range: 11 to 62,
- global mean overall score: 26.84,
- top overall model in this snapshot: `openai/gpt-5-nano` (62).

Cross-model guidance effects are directionally strong:

- mean average errors: 3.22 (Unguided), 2.01 (Little guidance), 2.17 (Expert guidance),
- mean zero-error share: 7.97%, 11.47%, and 11.16%, respectively,
- mean errors per 100 DOM elements: 12.39, 6.91, and 6.86, respectively.

These aggregate guidance patterns are directly reflected in Appendix Tables II–IV.

All models improve from Unguided to Little guidance on average error count, and 18/19 improve from Unguided to Expert guidance.

A. *Small/Old vs New Flagship Models*

1) *Cohort Definition*: For this paper, I compare two explicit LLM cohorts:

- **Small/old cohort** (4): `openai/gpt-3.5-turbo`, `anthropic/claude-3-haiku`, `google/gemma-3-12b-it`, `openai/gpt-oss-20b`.

- **New flagship cohort** (4): `z-ai/glm-5`, `moonshotai/kimi-k2.5`, `anthropic/claude-opus-4.6`, `openai/gpt-5.2`.

Models with reasoning capability were allowed to reason on their default setting as provided by OpenRouter. Flagship models were selected by reported benchmark results on industry-recognized benchmarks such as SWE-bench Verified. `openai/gpt-3.5-turbo` and `anthropic/claude-3-haiku` are included in the old cohort to allow easier comparison between newer and older models. All models in the flagship cohort were released in the last three months.

2) *Cohort-Level Contrast*: It is important to note that the flagship cohort consistently produces more visually stylized outputs, while the small/old cohort generally produces more functional interfaces with limited visual styling. This also increases the amount of issues the automated detector is able to find in a given sample. However, multiple studies suggest that users with disabilities generally prefer simpler interfaces, akin to those generated by the old/small cohort. Therefore, in the final scoring, the total error count per task is weighed more aggressively.

The flagship cohort, trained specifically on agentic software tasks, performs substantially worse on raw average error count, but far better on element-normalized error density. Specifically, the small/old cohort has a higher mean overall score (31.0 vs 17.5) and lower raw average errors (Unguided 2.17 vs 5.13; Little guidance 1.65 vs 3.19; Expert guidance 1.82 vs 3.15). However, the flagship cohort produces larger outputs in DOM scope, and correspondingly lower normalized error burden (Unguided 9.69 vs 14.15 errors per 100 DOM elements; Little guidance 5.73 vs 9.38; Expert guidance 4.89 vs 9.61). The underlying per-model values for these comparisons are listed in Appendix Tables I, II, and IV.

This pattern indicates two different failure regimes:

- 1) **Compact-output regime (small/old)**: lower absolute error totals, but higher error concentration per DOM structure.
- 2) **Large-structure regime (flagship)**: higher absolute totals, but lower error concentration once normalized by DOM scope.

In practical terms, developers that optimize only for raw defect counts may prefer compact models; developers that optimize for defect density per DOM structure may prefer newer flagships.

3) *Flagship Results*: A common pattern in the requested flagship set is large Unguided \rightarrow Guided improvement, but with different preferred guidance styles. GPT-5.2 improves from 3.48 average errors (Unguided) to 1.98 (Little guidance), a 43.1% reduction. Claude Opus 4.6 improves from 5.17 to 2.02 under Little guidance, a 60.9% reduction. GLM-5 improves from 4.75 to 3.79 with Expert guidance as best, a 20.2% reduction. Kimi K2.5 improves from 7.11 to 3.66 with Expert guidance as best, a 48.5% reduction. All four models and guidance-condition values referenced here are reported in Appendix Tables II and IV.

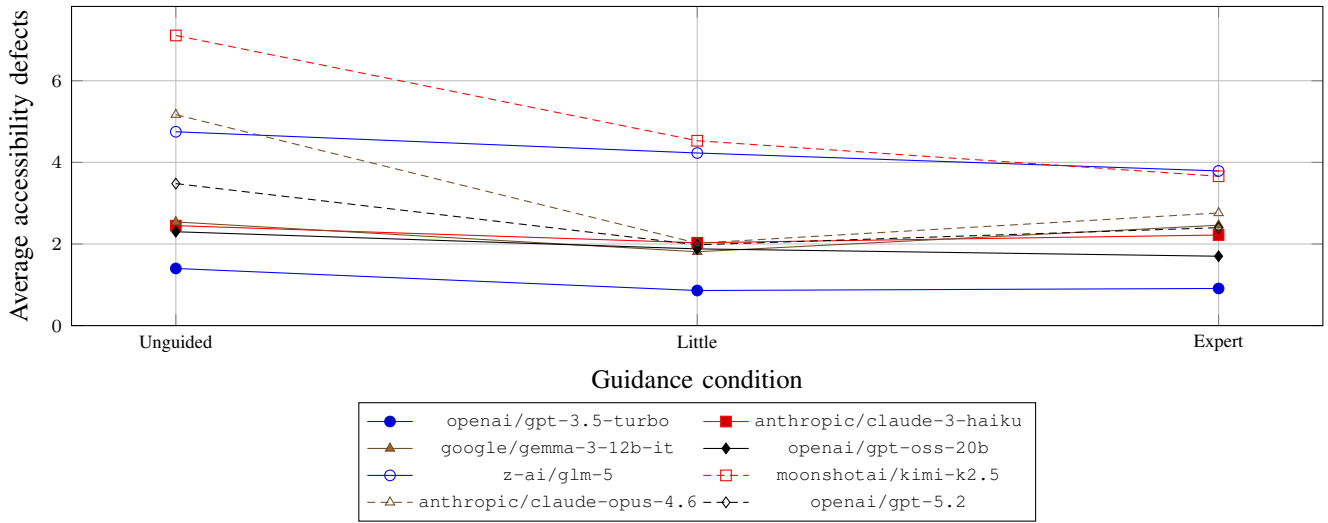


Fig. 1. Cohort-only comparison: average accessibility defects across guidance conditions

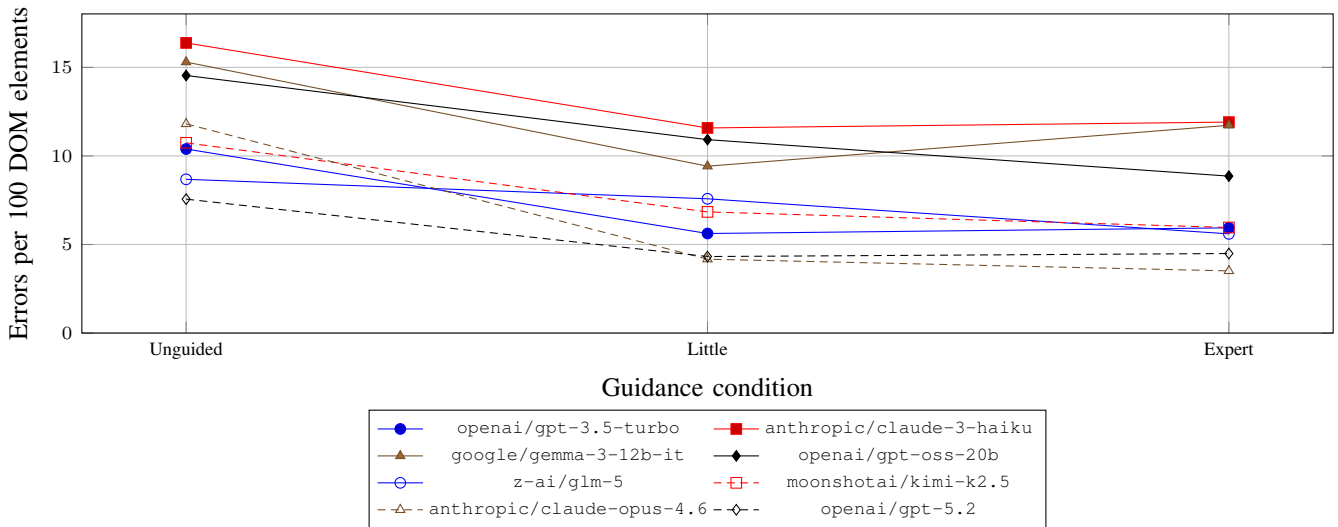


Fig. 2. Cohort-only comparison: element-normalized error burden across guidance conditions

a) *GPT-5.2.*: GPT-5.2 shows a strong response to lightweight guidance (from 3.48 to 1.98 average errors), with weaker performance under expert-style prompts (2.40). Its element-normalized error rates are moderate relative to this benchmark (4.32–7.56 errors per 100 DOM elements), indicating that guidance improves density as well as absolute burden.

b) *Claude Opus 4.6.*: Claude Opus 4.6 has the largest relative gain in this subset under Little guidance (60.9% reduction from Unguided). It also shows a strong element-normalized improvement (11.81 → 4.17 errors per 100 DOM elements under Little guidance; 3.51 under Expert guidance), but still accumulates non-trivial absolute defects.

c) *GLM-5.*: GLM-5 is atypical in this set: it benefits more from Expert guidance than Little guidance (best 3.79 vs 4.23 avg errors). This suggests that GLM-5 may require stronger

intent specification to realize accessibility improvements. Even with this gain, its absolute errors remain high compared with top-performing models in the full board.

d) *Kimi K2.5.*: Kimi K2.5 shows the highest Unguided burden in this subset (7.11 avg errors), then a substantial drop under guidance, with Expert guidance best (3.66). The improvement is large but still leaves a high residual burden versus the field median.

4) *Counterpoint from Older/Smaller Models.*: The small/old cohort produces smaller DOM footprints and, in this snapshot, frequently higher zero-error rates. For example, *openai/gpt-3.5-turbo* posts strong raw accessibility outcomes (Unguided 1.40, Little 0.86, Expert 0.91 avg errors; zero-error share up to 64.67%). However, because its DOM scope is small, normalized error density remains relatively high compared to the flagship cohort. See Appendix Tables II–IV for the complete

per-model breakdown.

This is exactly why I treat absolute and normalized metrics as complementary rather than substitutable.

IV. DISCUSSION

A. Comparison to human-written code

Every year, WebAIM of Utah State University conducts an automated analysis of one million home pages using WAVE, tooling that is methodologically similar to WebAccessBench and features approximate results. Across 2019–2025, WebAIM reports approximately 50–60 detected accessibility errors per page; in 2025 specifically, the reported averages are 51 errors and 1257 DOM elements per page, corresponding to ≈ 4.06 errors per 100 DOM elements (range using 50–60: ≈ 3.98 – 4.77).

Against this reference, WebAccessBench shows the expected absolute-count gap but a different normalized picture. Pooled across all benchmarked models, generated artifacts are much smaller and contain fewer absolute errors per sample, yet remain relatively defect-dense when normalized by structure. Cohort-level analysis sharpens this result. The small/old cohort has lower absolute error burden (overall mean 1.88 errors/sample) but very small DOM scope (17.02 elements/sample), yielding ≈ 11.05 errors per 100 elements. The flagship cohort has higher absolute burden (overall mean 3.82 errors/sample) but substantially larger outputs (57.34 elements/sample), yielding ≈ 6.77 errors per 100 elements. Thus, the cohort ordering reverses under element normalization: small/old models look better on raw counts, while flagships look better on structural error density.

Guidance effects are also cohort-specific under this normalization. In the small/old cohort, errors per 100 elements decrease from 14.15 (Unguided) to 9.38 (Little guidance) and 9.61 (Expert guidance). In the flagship cohort, they decrease from 9.69 (Unguided) to 5.73 (Little guidance) and 4.89 (Expert guidance). Even the strongest flagship condition remains above the WebAIM reference range, indicating that lower absolute error counts in generated artifacts are largely explained by reduced interface scope rather than superior accessibility quality per unit of DOM structure.

This remains a non-isomorphic comparison: WebAIM measures real deployed pages with heterogeneous legacy complexity, whereas WebAccessBench measures bounded generated artifacts under controlled prompts. Nevertheless, element-normalized results support the same operational conclusion: current AI-generated web code still requires expert human accessibility review before deployment.

B. Result Interpretation and Practical Guidance

I take six operational conclusions from this snapshot:

- 1) **Prompt guidance is non-optional.** Across all models, unguided prompting shows materially worse results.
- 2) **Guidance type should be model-specific.** Lightweight guidance is consistently beneficial, while expert-style guidance is beneficial only for specific models (for example, GLM-5 and Kimi K2.5).

- 3) **Absolute vs normalized metrics can reverse rankings.** If this is ignored, model selection can be systematically wrong for a given workload profile.
- 4) **Flagship scale does not guarantee lower raw accessibility defects.** In this assessment, several flagships trail older/smaller LLMs on raw error counts and zero-error frequency.
- 5) **Flagships can still be attractive for large interfaces.** Their lower element-normalized error density suggests higher local quality per DOM structure.
- 6) **Current model quality is still insufficient for autonomous accessibility-critical delivery.** Even under a cherry-picked best-of-two protocol, error rates remain high enough to require human accessibility review.

Overall, this means that current-generation LLMs are not yet suitable for unsupervised development of accessible web applications. As AI-assisted development is adopted at scale, accessibility defects can spread quickly and may feed back into future training corpora. This creates material societal risk for people with disabilities by increasing the probability of exclusion from key domains including education, healthcare, finance, leisure, and work.

Model vendors must improve web UI output quality substantially, not only visually but also semantically and behaviorally. Policymakers should enforce existing digital accessibility legislation and develop additional accountability mechanisms for both model vendors and software developers.

V. BENCHMARK LIMITATIONS

WebAccessBench remains an automated high-throughput system, so I keep the following caveats explicit:

- Automated assessment cannot fully substitute for expert manual accessibility review. The tooling used here captures only a subset of standards defined in WCAG 2.2 and EN 301 549. This undercounts barriers, especially with keyboard order, screen reader announcements, cognitive load, and interaction predictability.
- Prompt buckets are behaviorally useful but do not isolate all causal factors.
- Model and provider behavior are time-varying; results are timestamped measurements, not permanent rankings.

VI. CONCLUSION

WebAccessBench provides a structured measurement framework for accessibility reliability in LLM-generated web interfaces, using absolute error burden, zero-error frequency, element-normalized burden, and an aggregate score that combines these signals. In this snapshot (19 models; 150 tasks across three guidance conditions; best-of-two scoring), prompt guidance consistently improves outcomes relative to unguided generation (mean errors: 3.22 Unguided, 2.01 Little guidance, 2.16 Expert guidance), but improvement is heterogeneous by model and guidance style. This confirms that accessibility performance is not a monotonic function of model scale and that model-specific prompt policy remains necessary.

Cross-model comparisons also show that metric choice changes interpretation. Some models perform better on absolute defect counts, while others perform better on normalized burden due to broader DOM scope. The scoring framework therefore treats these measures as complementary rather than interchangeable. At the same time, external comparison to WebAIM indicates that absolute counts alone can be misleading: although benchmark artifacts have fewer total errors per output, they also contain far fewer DOM elements. Under element-normalized comparison, the benchmarked AI outputs remain more defect-dense than the large-scale human-written baseline.

The practical implication is urgent and political, not merely technical. If AI-generated interfaces are deployed at scale without rigorous accessibility safeguards, exclusion will be industrialized: people with disabilities will face compounded barriers to education, healthcare, employment, public services, finance, and democratic participation. Accessibility defects in generated code do not stay in a prototyping stage; they replicate across products, organizations, and future training data, turning preventable design failures into systemic discrimination. This is why accessibility review cannot be optional, and why responsibility must extend beyond individual developers to model vendors, platform operators, regulators, and public institutions.

APPENDIX A SAMPLE TASKS

Each task is evaluated under all three guidance regimes. The examples below use representative task stems from the benchmark task pool.

- 1) **Task T-001:** Make a login form card
Unguided: Make a login form card.
Little guidance: Make a login form card. Make it accessible.
Expert guidance: Make a login form card using explicit `<label for>` associations for both fields, and never rely on placeholder text as the only label. On submit, move focus to an error summary at the top, and link each error item back to its input with `aria-describedby` so screen readers announce the message in context. Mark invalid fields with `aria-invalid=true`, keep submit reachable by keyboard, and do not remove native focus outlines.
- 2) **Task T-002:** Make a toggle switch control
Unguided: Make a toggle switch control.
Little guidance: Make a toggle switch control. Make it accessible.
Expert guidance: Make a toggle switch control by implementing it with a native checkbox styled as a switch, instead of a clickable `<div>`, to avoid broken semantics. Ensure keyboard users can toggle it with Space without custom key hacks, and keep the visible focus ring in all states. Expose state with a persistent text label plus `checked/not checked` announcement; do not communicate state by color alone.
- 3) **Task T-003:** Make a modal dialog

Unguided: Make a modal dialog.

Little guidance: Make a modal dialog. Make it accessible.

Expert guidance: Make a modal dialog using the native `<dialog>` HTML tag. Focus should land on the first child, and an explicit close button should be last in focus order. Make sure that the focus is trapped within the dialog, and returns to the correct spot when the dialog is closed.

- 4) **Task T-004:** Make a chart legend block

Unguided: Make a chart legend block.

Little guidance: Make a chart legend block. Make it accessible.

Expert guidance: Make a chart legend block without encoding categories by color alone; include visible text labels and distinguishable markers/patterns. If legend items are interactive, render them as actual buttons with keyboard focus and pressed/selected state. Provide a short textual summary of the underlying values near the legend.

- 5) **Task T-005:** Make a multi-step form wizard

Unguided: Make a multi-step form wizard.

Little guidance: Make a multi-step form wizard. Make it accessible.

Expert guidance: Make a multi-step form wizard that exposes progress semantically (for example “Step 2 of 4”) and announces step changes in a polite live region. When moving steps, set focus to the new step heading, keep Back/Next as real buttons, and preserve entered values to prevent data loss. Validate after explicit user action, provide field-level errors plus a summary, and avoid trapping keyboard focus inside one subpanel.

APPENDIX B INFORMATION ON AI USE

Due to time constraints, this paper has been written with the assistance of an LLM. However, the research, including methodology, results and interpretation of those results were carried out fully without AI assistance.

APPENDIX C BENCHMARK DATA FOR ALL TESTED MODELS

TABLE I
ACCESSIBILITY SCORE BY MODEL (RANKED BY OVERALL SCORE)

Rank	Model	Score
1	openai/gpt-5-nano	62.00
2	openai/gpt-3.5-turbo	49.00
3	openai/gpt-5.1-codex-mini	46.00
4	openai/gpt-oss-20b	28.00
5	openai/gpt-4o-2024-11-20	25.00
6	openai/gpt-5.2	25.00
7	openai/gpt-oss-120b	25.00
8	xiaomi/mimo-v2-flash	25.00
9	deepseek/deepseek-v3.2	24.00
10	google/gemma-3-12b-it	24.00
11	anthropic/claude-3-haiku	23.00
12	google/gemini-3-flash-preview	23.00
13	qwen/qwen3-coder-next	23.00
14	x-ai/grok-4.1-fast	22.00
15	mistralai/devstral-2512	21.00
16	anthropic/claude-haiku-4.5	20.00
17	anthropic/claude-opus-4.6	20.00
18	z-ai/glm-5	14.00
19	moonshotai/kimi-k2.5	11.00

TABLE II
AVERAGE ACCESSIBILITY DEFECTS BY GUIDANCE CONDITION (RANKED BY AVERAGE ERROR BURDEN)

Rank	Model	Unguided	Little guidance	Expert guidance
1	openai/gpt-5-nano	1.26	0.43	0.84
2	openai/gpt-3.5-turbo	1.40	0.86	0.91
3	openai/gpt-5.1-codex-mini	2.05	0.93	0.93
4	openai/gpt-oss-20b	2.30	1.88	1.70
5	openai/gpt-4o-2024-11-20	2.48	1.73	1.84
6	openai/gpt-oss-120b	2.72	1.83	1.84
7	anthropic/claude-3-haiku	2.45	2.03	2.22
8	google/gemma-3-12b-it	2.54	1.81	2.46
9	google/gemini-3-flash-preview	3.43	1.73	1.79
10	xiaomi/mimo-v2-flash	3.27	1.82	1.99
11	deepseek/deepseek-v3.2	3.40	1.97	1.99
12	x-ai/grok-4.1-fast	3.18	2.19	2.14
13	qwen/qwen3-coder-next	3.43	2.17	2.18
14	mistralai/devstral-2512	3.57	1.99	2.30
15	openai/gpt-5.2	3.48	1.98	2.40
16	anthropic/claude-haiku-4.5	3.28	2.01	3.55
17	anthropic/claude-opus-4.6	5.17	2.02	2.76
18	z-ai/glm-5	4.75	4.23	3.79
19	moonshotai/kimi-k2.5	7.11	4.53	3.66

TABLE III
CLEAN OUTPUT RATE (% ZERO DETECTABLE ERRORS) BY GUIDANCE CONDITION

Rank	Model	Unguided (%)	Little guidance (%)	Expert guidance (%)
1	openai/gpt-5-nano	54.00	77.33	65.54
2	openai/gpt-3.5-turbo	50.00	64.67	59.46
3	openai/gpt-5.1-codex-mini	34.00	58.00	59.86
4	openai/gpt-oss-20b	6.67	11.33	18.92
5	openai/gpt-4o-2024-11-20	0.00	0.00	0.00
6	openai/gpt-oss-120b	0.00	2.00	0.68
7	anthropic/claude-3-haiku	0.00	0.00	0.00
8	google/gemma-3-12b-it	2.67	2.00	0.00
9	google/gemini-3-flash-preview	0.00	0.00	0.00
10	xiaomi/mimo-v2-flash	1.33	0.67	4.73
11	deepseek/deepseek-v3.2	0.00	0.00	0.00
12	x-ai/grok-4.1-fast	0.00	0.00	0.00
13	qwen/qwen3-coder-next	0.67	0.00	0.00
14	mistralai/devstral-2512	0.00	0.00	0.00
15	openai/gpt-5.2	0.00	0.00	0.00
16	anthropic/claude-haiku-4.5	0.00	0.00	0.00
17	anthropic/claude-opus-4.6	0.00	0.00	0.00
18	z-ai/glm-5	0.67	0.00	1.36
19	moonshotai/kimi-k2.5	1.34	2.00	1.43

TABLE IV
 ERRORS PER 100 DOM ELEMENTS BY GUIDANCE CONDITION (RANKED BY WEIGHTED DOM-NORMALIZED ERROR RATE)

Rank	Model	Unguided	Little guidance	Expert guidance
1	openai/gpt-5-nano	4.44	1.31	2.60
2	openai/gpt-5.1-codex-mini	9.79	3.77	3.57
3	openai/gpt-5.2	7.56	4.32	4.49
4	anthropic/claude-opus-4.6	11.81	4.17	3.51
5	deepseek/deepseek-v3.2	9.62	5.46	5.01
6	qwen/qwen3-coder-next	9.88	5.39	5.44
7	openai/gpt-3.5-turbo	10.39	5.62	5.94
8	z-ai/glm-5	8.68	7.58	5.60
9	xiaomi/mimo-v2-flash	12.43	5.29	5.61
10	moonshotai/kimi-k2.5	10.74	6.84	5.96
11	x-ai/grok-4.1-fast	12.75	7.58	6.76
12	anthropic/claude-haiku-4.5	11.89	6.92	7.68
13	mistralai/devstral-2512	17.23	7.41	8.76
14	openai/gpt-oss-120b	15.94	9.11	8.56
15	google/gemini-3-flash-preview	19.50	8.06	8.11
16	openai/gpt-oss-20b	14.54	10.92	8.86
17	openai/gpt-4o-2024-11-20	16.57	10.54	10.29
18	google/gemma-3-12b-it	15.30	9.42	11.73
19	anthropic/claude-3-haiku	16.38	11.58	11.91