Catching UX Flaws in Code: Leveraging LLMs to Identify Usability Flaws at the Development Stage

Nolan Platt
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
nolanplatt@vt.edu

Ethan Luchs
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
ethanluchs@vt.edu

Sehrish Nizamani
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
sehrishbasir@vt.edu

***Abstract*—Usability evaluations are essential for ensuring that modern interfaces meet user needs, yet traditional heuristic evaluations by human experts can be time-consuming and subjective, especially early in development. This paper investigates whether large language models (LLMs) can provide reliable and consistent heuristic assessments at the development stage. By applying Jakob Nielsen's ten usability heuristics to thirty open-source websites, we generated over 850 heuristic evaluations in three independent evaluations per site using a pipeline of OpenAI's GPT-4o. For issue detection, the model demonstrated moderate consistency, with average pairwise Cohen's Kappa of 0.50 and exact agreement of 84%. Severity judgments showed more variability: weighted Cohen's Kappa averaged 0.63, but exact agreement was just 56%, and Krippendorff's Alpha was near zero. These results suggest that while GPT-4o can produce internally consistent evaluations, especially for identifying the presence of usability issues, its ability to judge severity varies and require human oversight in practice. Our findings highlight the feasibility and limitations of using LLMs for early-stage, automated usability testing, and offer a foundation for improving consistency in automated User Experience (UX) evaluation. To the best of our knowledge, our work provides one of the first quantitative inter-rater reliability analyses of automated heuristic evaluation and highlights methods for improving model consistency.**

***Index Terms*—large language models, natural language processing, human-computer interaction, usability evaluations, heuristic evaluations, automated usability evaluations.**

I. Introduction

Usability is a critical quality attribute for web applications, directly influencing user satisfaction, adoption, and long-term success [1]. A feature-rich site still fails if users struggle to accomplish basic tasks [2], leading to costly redesigns and negative feedback. Detecting and resolving such usability flaws as early as possible is therefore highly desirable [3]. Traditionally, usability issues are discovered through *user testing*, where real users are observed while completing tasks [2]. Similarly, issues are also found via *expert heuristic evaluations*, where specialists review the interface against established principles. While effective, these approaches demand significant time, expertise, and coordination [4], [5]. Small teams or early-stage projects often lack the resources for comprehensive usability studies, allowing interface problems to remain hidden until after deployment [6]. This resource gap motivates our investigation into automated usability assessments [7] that can be run continuously during development with minimal human effort.

Recent advances in Large Language Models (LLMs), especially with regard to Natural Language Processing (NLP), offer a new novel approach to automate heuristic evaluations. LLMs are deep neural network models trained on vast text corpora and advanced methods in NLP. In some cases, these models are trained to interpret and understand code and images, enabling them to understand and generate complex language and reason about structured information. The past few years have seen a rapid development in LLMs ability to perform software engineering tasks, with specific regard to code generation, code review, and bug detection [8]. With some models able to simultaneously process files, interpret the source code, and understand natural language instructions, the applications of LLMs are nearly endless [9]. This raises the intriguing possibility that an LLM could serve as an *automated usability evaluator*, essentially acting as a virtual "expert" that inspects a web application's interface and code for usability guideline violations. If viable, such an approach could rapidly flag UX issues early on, providing feedback to developers without the need for costly usability studies.

This research investigates whether state-of-the-art LLMs can reliably identify usability flaws in web applications during early development by analyzing the respective source code. In particular, we examine the use of OpenAI GPT-4o to perform heuristic evaluations on web interfaces. We leverage Jakob Nielsen's ten usability heuristics [4] as a structured framework for analysis, prompting the model to check each heuristic with respect to potential issues in the given website's implementation. To evaluate consistency and reliability, we conduct an experiment on *30 open-source web applications*, instructing the nodel to review each application *three separate times* in fresh sessions. By treating each independent LLM session as a rater, we then compute inter-rater reliability metrics through various statistical measures: **Cohen's Kappa**, **Fleiss's Kappa**, and **Krippendorff's Alpha**. Through these measures, we effectively quantify how consistent the model identifies the same issue and severity rating across independent sessions. High agreement would indicate that the model's usability evaluations are stable and repeatable, whereas low agreement would signal variability or randomness in its re-

sponses. Through this study, we aim to assess the practical feasibility of LLM-driven usability inspection and understand its strengths and limitations as compared to traditional, human-based heuristic evaluation methods.

KEY CONTRIBUTIONS

In summary, the key contributions of this work are as follows:

1. LLM-Based Heuristic Evaluation Methodology. We propose a novel approach to automated usability testing using a large language model. Our method translates Nielsen's usability principles into structured prompts that guide GPT-4o to inspect web application source code for usability concerns, requiring no domain-specific fine-tuning of the model. We leverage a customized pipeline of GPT-4o that standardizes prompting and outputs results in JavaScript Object Notation (JSON) format.

2. Empirical Study on Web Applications. We conduct a comprehensive evaluation on 30 open-source websites, applying the LLM-driven heuristic evaluation to each. This design allows us to observe the types of usability issues a model can detect in real-world web projects and to gather qualitative examples of its feedback.

3. Inter-Rater Reliability Analysis. To our knowledge, we are among the first to examine the consistency of an LLM's UX evaluations. By repeating the analysis in triplicate for each site and calculating Cohen's Kappa, Fleiss's Kappa, and Krippendorff's Alpha, we provide quantitative insight into the reliability of GPT-4o's judgements.

4. Insights on LLM Efficacy for Usability Testing. We discuss the implications of our findings for the role of AI in usability engineering. Our results shed light on how well an off-the-shelf LLM aligns with human expert assessments, the kinds of usability problems it excels at or misses, and how it might be best integrated into development workflows.

The remainder of this paper is structured as follows: **II** provides background on usability evaluation methods and reviews related work in AI-assisted usability assessment. **III** then details our methodology for LLM-driven heuristic evaluation, including prompt design and experimental setup. **IV** presents the results of our study, focusing on the inter-rater reliability measures and examples of identified issues. **V** offers a discussion on the findings, their implications, and limitations. Finally, **VI** concludes the paper with a summary and outlook on future developments in automated usability testing.

II. BACKGROUND

A. Usability Evaluation and Heuristic Methods

Usability refers to how easy and effective it is for users to achieve their goals using a system. Key aspects include learnability, efficiency, error reduction, and user satisfaction. Common evaluation methods include user testing and heuristic evaluation. User testing involves observing participants complete tasks on an interface, offering rich empirical feedback but at the cost of time and resource investment. On the other hand, heuristic evaluation, introduced by Nielsen and Molich, requires experts to review an interface against recognized usability principles [4], [5]. Nielsen's ten usability heuristics remain a standard, covering principles such as visibility of system status, user control, consistency, and error prevention.

Heuristic evaluations are faster and less resource-intensive than user testing but suffer from subjectivity and limited coverage. Different experts may identify different issues or disagree on severity, leading to inconsistent results [5]. These inconsistencies are especially problematic in early-stage development, where iterative decisions depend on timely and actionable feedback. Moreover, many small teams lack access to usability experts, and even with multiple reviewers, variability persists. These challenges have motivated growing interest in automated heuristic evaluation as a way to reduce cost, improve coverage, and support continuous integration in agile workflows.

B. AI-Powered Usability Assessment with LLMs

To address the limitations of manual and expert-driven reviews, several tools have emerged to automate parts of the UX evaluation process. Widely used tools apply static rule-based analysis to detect violations of accessibility and usability standards in HTML and CSS. These systems are highly effective for identifying technical flaws—such as missing alternative text, improper contrast ratios, or non-semantic markup—but are fundamentally constrained by their deterministic nature. They operate by checking for predefined conditions, and as a result, they miss more nuanced, context-dependent problems such as ambiguous labels, ineffective navigation, or tone inconsistency.

Large language models (LLMs) offer a promising alternative to these hard-coded approaches. Trained on extensive corpora that include source code, natural language, documentation, and UX best practices [10], LLMs provide a semantic layer of reasoning that allows them to analyze both the structure and intent of an interface. These models can be prompted with heuristic criteria and interpret front-end code—including HTML, CSS, and JavaScript—to identify potential usability concerns [11]. Crucially, their capacity to understand language and context enables them to detect flaws that go beyond syntax, such as misleading User Interface (UI) metaphors, unclear instructions, or a lack of user control mechanisms. This flexibility positions LLMs as general-purpose evaluators capable of both qualitative and structured analysis.

Recent research has explored the use of vision-language and multimodal models to assist with usability assessment. For example, UX-LLM applies GPT-4o to evaluate iOS app screenshots and metadata, predicting violations based on visual layout and descriptive cues [12]. Other studies have incorporated behavioral logs, user interaction traces, and error messages to capture a fuller picture of the user experience [13]. While these multimodal approaches demonstrate promise, they typically depend on a rendered UI or assume access to runtime telemetry. They may flag superficial issues, such as visual clutter or alignment errors, but often overlook deeper concerns related to task clarity or content semantics. Furthermore, many

of these systems still require human review to validate or interpret the results, limiting their scalability in fully automated pipelines.

In contrast, our approach focuses on source code-only evaluation, leveraging LLMs to reason about usability directly from markup and scripting logic. This allows assessments to be conducted without the need for rendering or instrumentation, enabling seamless integration into early-stage development workflows. By encoding usability heuristics into structured prompts and querying the model in a standardized fashion, we produce machine-readable evaluations that include both quantitative severity ratings and detailed qualitative feedback.

To the best of our knowledge, prior work has not examined the reproducibility of such LLM-driven evaluations. While some studies report on detection accuracy or illustrative outputs, few if any quantify how consistent these models are when applied multiple times under the same conditions. In human-centered design, inter-rater reliability (IRR) is a critical measure of evaluation quality—used to assess whether different evaluators (or the same evaluator across time) produce similar judgments. In this work, we adapt IRR metrics, including Cohen's Kappa, Fleiss's Kappa, and Krippendorff's Alpha, to assess agreement across multiple independent GPT-4o sessions. This contribution extends the literature on automated usability assessment by offering an empirical foundation for evaluating not only what LLMs detect, but also how consistently they do so.

III. Methodology

A. Initial Approach

Initial research centered around using a consistent and exact prompt across various usability evaluations. This led to flawed and - in some cases - completely inaccurate results. GPT-4o initially struggled with consistently filling out templates, both in chat (i.e. directly to the user) and in downloadable document files. We quickly realized that we needed to design an acutely specific prompt, consisting of prerequisite knowledge, instructions for document processing, and clear guidelines on how output should be displayed. In natural language processing (NLP), there has been a significant challenge in fine-tuning models to comprehend the same instruction over time. Specifically, GPT-4o is known to behave inconsistently for semantically identical queries in different sessions [14]. This caused significant issues with regard to our initial data collection, in that the evaluations were frequently recorded wrong or the same heuristic was duplicated dozens of times.

B. Data Collection

To address the underlying issues discussed in III-A, we designed a pipelined version of GPT-4o. OpenAI allows users to create their own "versions" of different LLM models, in that the session will already contain prerequisite knowledge, instructions, and address misconceptions that arise in NLP. With regard to the issues we faced, we were able to design the application to:

- understand the user is uploading a compressed file containing a website's source code.
- have exact knowledge of Nielsen's heuristic principles and how they apply to modern web applications.
- only output the evaluation in a consistent JSON format.

In our customized GPT, the prompt is acutely specific, containing checklists for the model to follow, warnings of only following the exact instructions, and not making assumptions on what a user is instructing the model to do. The exact prompt utilized in our customized pipeline is shown in Appendix B. Through this, we were able to address the NLP challenges as best we could without domain-specific fine-tuning. The current methodology of conducting an evaluation is as follows:

1) identify an open-source web application via GitHub or other version control platforms.
2) verify the repository's licensing and our ability to use the code.
3) download the repository as a compressed ZIP file.
4) open three distinct sessions of the customized GPT.
5) upload the same compressed file to each of the three sessions and download each resulting JSON file.

After step five, each JSON file was then uploaded to our repository [16] at `\results\<repoName>\<eval#>`, where: `repoName` represents the name of the repository. `eval#` represents the evaluation in ascending order: `eval1`, `eval2`, `eval3`. By using three distinct evaluations for each website, we are able to quantify how GPT-4o performs with regard to consistency and accuracy over time. This consistent methodology proved highly effective, with GPT-4o consistently providing correctly structured JSON files, and not misinterpreting the natural language instructions. Figure 1 demonstrates our methodology visually.

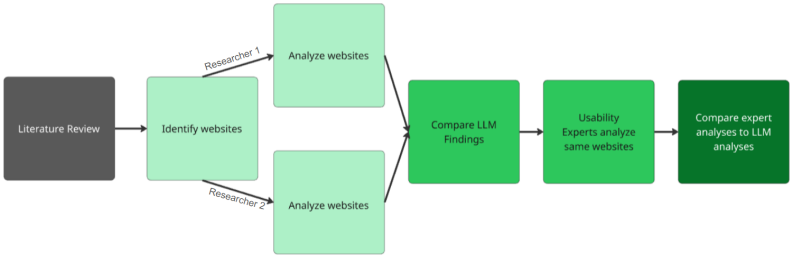

Fig. 1. Research methodology diagram

IV. Results

To obtain accurate results from our study, we created an open-source repository that includes all results (in JSON) from GPT-4o, along with the analysis code in a Jupyter notebook. This repository is publicly available at `github.com/nolanplatt/LLMUsabilityEvaluations` [16]. In this paper, all references to directories refer to this repository's root.

A. Issue Frequency Across Evaluations

Our analysis begins in `\analysis.ipynb`, where we iterate through every folder in `\results`, each representing an open-source website and the corresponding evaluations. For each site, we load the triplicate LLM evaluations into rows of a `DataFrame`, with each row holding three independent evaluations corresponding to the same site.

To visualize how often GPT-4o flags usability issues, Fig. 2 shows the percentage of heuristics marked as having an issue across all 30 websites. Each heuristic was reviewed three times per site, totaling 900 evaluations. The overwhelming majority of evaluations identify at least one usability flaw, reflecting the fact that our dataset largely consists of early-stage or demo-level web projects.

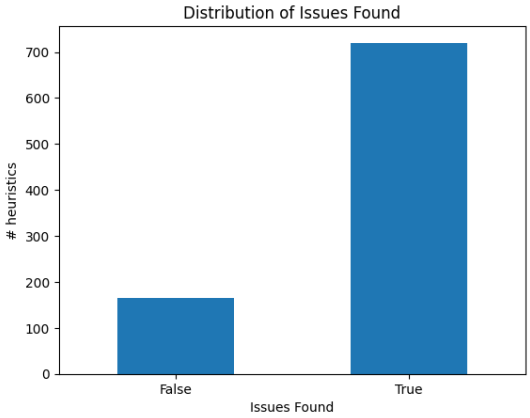

Fig. 2. Distribution of issues found across all evaluations. A large majority of heuristics were flagged as having usability issues by the model.

This high detection rate demonstrates that GPT-4o frequently identifies flaws when reviewing source code with respect to Nielsen's heuristics. Nonetheless, frequent detection does not necessarily imply consistent or accurate evaluation. Rather, this indicates a tendency toward sensitivity, often flagging something as an issue, which we further examine through severity analysis.

B. Distribution of Severity Ratings

To explore how serious GPT-4o identified each issue to be, we analyzed severity scores across all sessions and sites. Table I summarizes the full distribution. Ratings range from 0 (no issue) to 4 (usability catastrophe).

TABLE I
DISTRIBUTION OF SEVERITY RATINGS

Severity Level	Count
0 (No Issue)	164
1 (Cosmetic)	251
2 (Minor)	306
3 (Major)	155
4 (Catastrophic)	9

This distribution highlights the model's tendency to flag minor and moderate issues far more frequently than critical failures. The total number of severity ratings shown in Table I is represented by Equation (1), which computes the sum of all evaluations performed.

$$\sum_{i=0}^{4} \text{count}_i = 164 + 251 + 306 + 155 + 9 = 885 \quad (1)$$

where c_i is the count of severity level i. While 900 heuristic evaluations were expected (30 websites $\times$ 10 heuristics $\times$ 3 sessions), 15 entries were excluded due to missing or malformed severity values. These incomplete outputs were dropped during preprocessing to ensure consistency in statistical comparisons.

C. Agreement on Issue Presence

We first examined whether GPT-4o agreed on whether a usability issue *existed*. For this, we calculated **Cohen's Kappa** across each pair of evaluations for every site, as represented by Equation (2).

$$\kappa = \frac{p_o - p_e}{1 - p_e} \quad (2)$$

where p_o is the observed agreement and p_e is the expected agreement by chance. This was applied to evaluation pairs for every site, as shown in Table II.

Since the outcome is binary (i.e. either an issue is detected or not detected), a weighted statistical measure is unnecessary. Each disagreement should be treated equally. The κ values across all comparisons are moderately consistent, ranging from 0.47 to 0.53, with exact agreement rates between 82.3% and 85.0%. This suggests that GPT-4o is moderately reliable in consistently identifying whether or not a heuristic violation exists, even in multiple, distinct trials.

The high agreement percentages (>82%) show that the model is not randomly flagging issues but follows a repeatable logic in deciding when a heuristic has been violated or not. This underlines the idea that LLMs, even without domain-specific fine-tuning, can serve as dependable evaluations for identifying potential usability issues. While consistency in identifying the *presence* of an issue is promising, it is important to understand that this does not guarantee accuracy. This only shows that the model is internally consistent. While developers might trust the model to reliably identify potential issues, issues should still be validated for correctness.

TABLE II
UNWEIGHTED COHEN'S KAPPA ACROSS ALL ISSUE PAIRS

Comparison	Cohen's κ	Agreement	Exact Percent
eval1-eval2	0.471	247/300	82.3
eval1-eval3	0.530	255/300	85.0
eval2-eval3	0.502	254/300	84.7

We can further evaluate the significance of this data using multi-rater agreement metrics, including:

Fleiss's Kappa. Used to assess agreement among three raters. The general formula utilized is expressed in Equation (3) .

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{3}$$

where $\bar{P}$ is the mean of observed agreements, and $\bar{P}_e$ is the mean of expected agreement.

Krippendorff's Alpha. A general measure that accommodates missing data and all data types, as shown in Equation (4).

$$\alpha = 1 - \frac{D_o}{D_e} \tag{4}$$

where D_o is the observed disagreement and D_e is the expected disagreement.

Both (3) and (4) were applied to the same data for issue pairs, with the results shown in Table III.

While Cohen's and Fleiss's Kappa metrics suggest moderate agreement across different sessions (e.g., average $\kappa \approx 0.5$), Krippendorff's Alpha returned a value slightly negative. This discrepancy is not a calculation error: it reflects that Krippendorff's Alpha penalizes systematic disagreement much more severely and is highly sensitive to imbalanced distributions, even in small samples. In our case, GPT-4o outputs often overpredict issue presence, but the degree varies slightly between sessions, contributing to this divergence.

TABLE III
MULTI-RATER AGREEMENT METRICS

Metric	Value
Krippendorff's α	-0.000234
Fleiss's κ	0.500

D. Agreement on Severity Ratings

Fig. 3 visualizes the distribution of severity ratings across all heuristics. This graph more clearly shows the data in Table I, emphasizing how the model flags minor and moderate issues much more frequently than critical issues.

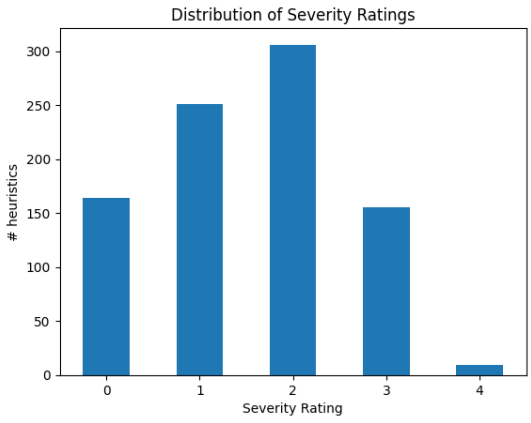

Fig. 3. The distribution of severity ratings found by the model.

As described in IV-B, severity ratings range from 0 (no issue) to 4 (usability catastrophe). We are thus looking at five possible ratings the LLM can rate *each heuristic*. So, statistically comparing these results cannot be achieved using the binary approach in IV-C. We must instead utilize a weighted Cohen's Kappa [17]. Fig. 4 represents the data-wide distribution of Cohen's Kappa for `eval1 vs eval2`. As visualized, most sites achieved $\kappa >= 0.6$, indicating substantial agreement, though a few had lower or even negative agreement, typically due to imbalanced detection rates between the two. Because severity is ordinal, we can understand the agreements in the data through a **Weighted Cohen's Kappa**. Equation (5) applies penalties to disagreements based on the significance of the displacement between two ratings.

$$\kappa_w = 1 - \frac{\sum w_{i,j} o_{i,j}}{\sum w_{i,j} e_{i,j}} \tag{5}$$

where $w_{i,j}$ is a weight matrix and $o_{i,j}, e_{i,j}$ are observed counts. The weighted κ_w was then computed for all severity pairs, as shown in Table IV.

GPT-4o is not only consistent in identifying whether a usability issue exists, but also exhibits reasonable agreement in how severe those issues are. Severity judgments do show more variability than binary issue presence, but this is expected; the severity data is ordinal, rather than binary. Most sessions still landed within a close scoring range. The weighted κ values near or above 0.6 (see Table IV) imply moderate agreement between evaluations, reinforcing the notion that GPT-4o is capable of reliably ranking issue significance.

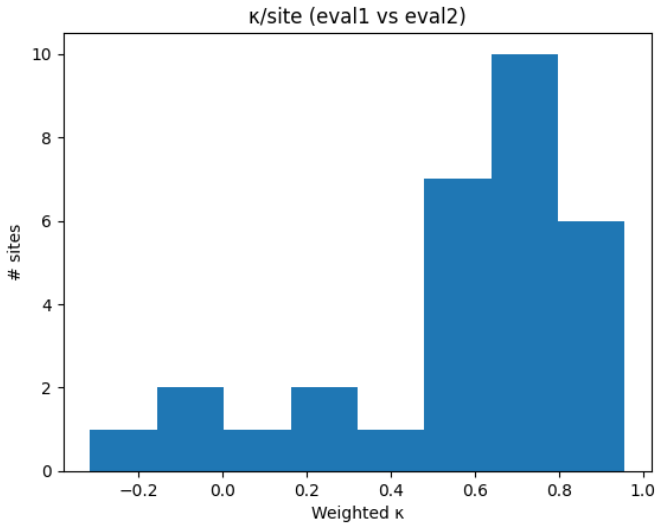

Fig. 4. Distribution of weighted Cohen's Kappa values per site (eval1 vs eval2) for severity ratings.

In particular, this consistency is crucial for developers or small teams in prioritizing fixes and, thus, workflow. Even if the exact severity scores vary slightly, the model does demonstrate a consistent understanding of what constitutes a minor usability flaw compared to a catastrophic issue. However, it is vital to understand that this is *internal consistency*, and does not imply actual correctness. The model's ratings should

be used to guide - not replace - expert judgment. We also computed reliability metrics across each individual metric - a table representing those values is shown in Appendix A.

TABLE IV
WEIGHTED COHEN'S KAPPA ACROSS ALL SEVERITY PAIRS

Comparison	κ_w	Agreement	Exact Percent
eval1-eval2	0.625361	169/300	56.3
eval1-eval3	0.668431	171/300	57
eval2-eval3	0.596722	167/300	55.6

V. DISCUSSION

Our findings show that automated heuristic evaluations can produce consistent results, especially when identifying whether a usability issue is exists. The relatively high agreement percentages (82–85%) and moderate Cohen's Kappa values ($\approx$0.5) show that GPT-4o applies its internal logic consistently across independent sessions. This is a promising result, particularly for teams lacking access to UX professionals.

When evaluating issue severity, we found slightly greater inconsistency. As shown, agreement rates were roughly 56–57% and the weighted Cohen's Kappa values near 0.6. While not as strong as the binary classification (for issue detection), this still indicates inter-rater reliability, even in the ordinal nature of the severity ratings. More remarkably, this level of reliability is achieved without domain-specific fine-tuning. In traditional human evaluations, severity judgments are also known to vary, even among expert raters [2]. The consistency observed here suggests that GPT-4o may offer comparable approaches when prioritizing usability problems (and ratings).

That said, the Krippendorff's Alpha value being slightly below zero does bring up an important discussion. Even small inconsistencies or biases are across raters are *heavily* penalized. In our case, the model has a tendency to overpredict issue presence and undervalue the severity of issues. While this may not significantly impact the Cohen's or Fleiss's Kappa, it notably drags the Krippendorf's Alpha value down. While this discrepancy is important to note - and is statistically significant - it does *not* indicate model failure. Rather, it shows the metric's sensitivity to specific cases.

When interpreting both methods of analysis (i.e. both issue detection and severity classification), our findings support the idea that LLMs can serve as a viable starting point for automated usability evaluation. They are not a replacement for expert review, but do offer a consistent baseline for flagging and prioritizing issues, especially when expert resources are limited or consistent, large-scale, iteration is needed. In practice, these evaluations could add to human evaluations: serving as a type of preliminary triage tool. Automated evaluations could provide initial evaluations for how teams should distribute work, or could provide second opinions.

Traditional usability evaluation tools, such as aXe DevTools, primarily rely on rule-based engines that scan web content against predefined accessibility standards like the Web Content Accessibility Guidelines (WCAG). These automated tools are effective at identifying a significant portion of common accessibility issues and are widely used due to their efficiency and ease of integration into development workflows. However, they often fall short in detecting more nuanced usability problems that require contextual understanding, such as issues related to user intent, content clarity, or the overall user experience.

Our approach leverages large language models (LLMs) to provide a semantic, language-based layer of analysis that complements traditional tools. By understanding context and user intent, LLMs can generate qualitative feedback and identify usability issues that are not easily captured by rule-based systems. This method builds upon the foundations laid by early tools like aXe and traditional usability testing scripts, offering a more generalized and insightful evaluation of user interfaces. The integration of LLMs enables the detection of complex usability concerns, such as ambiguous navigation cues or inconsistent content tone, thereby enhancing the overall effectiveness of usability assessments.

Nevertheless, several limitations must be acknowledged. The consistency shown in our data does not necessarily imply correctness or alignment with user experience principles in all contexts. Moreover, LLMs lack awareness of real user behavior and context-specific aspects of UX that codebases only partially reveal. To evaluate the correctness and scalability of automated heuristic evaluations, there would need to be, at a minimum, comparisons to expert evaluations. Future work may address these limitations by

1) comparing model output to expert evaluations
2) using active learning to improve evaluation accuracy.
3) domain-specific fine-tuning

More broadly, as LLMs continue to evolve in their interpretive and reasoning abilities [10], the difference between AI-assisted and expert heuristic evaluations may begin to decrease. This would open us up to new and exciting usability testing frameworks.

VI. CONCLUSION

In this paper, we demonstrate the potential of large language models, specifically GPT-4o, to perform consistent and structured heuristic evaluations on web application source code. By applying Nielsen's usability principles across thirty open-source websites and evaluating results through statistical agreement metrics, we find that GPT-4o exhibits moderate consistency, particularly for issue detection. Severity ratings show reasonably internal consistency with slight variation, highlighting the model's ability to distinguish between different levels of usability concern.

While LLMs are not a replacement for expert evaluations, our results suggest they can serve as reliable early-stage tools for identifying usability flaws. This capability is especially valuable for small teams where traditional evaluations may be impractical and far too time consuming. Further, the methodology we introduce: prompt standardization, triplicate evaluation, and data analysis lays the foundation for reproducible AI-assisted usability evaluations. In particular, our research

methodology can reasonably be expanded to evaluate other models, larger datasets, and more.

Future work may explore comparing model output with expert human evaluations, improving prompt robustness, and domain-specific fine-tuning. As LLMs continue to improve in accuracy and natural language processing, their role in usability evaluations will likely expand from supporting human evaluators to driving entirely new systems of automated usability assessments.

ACKNOWLEDGMENT

We would like to thank the Virginia Tech Center for Human-Computer Interaction (CHCI) for providing invaluable feedback during the preliminary stages of our research. Further, we would like to especially thank the Virginia Tech Department of Computer Science; this paper would not have been possible without the feedback and experiences from ACM's CAPWIC, which we received a funded trip to.

REFERENCES

[1] International Organization for Standardization, "ISO 9241-11:2018 Ergonomics of human-system interaction—Part 11: Usability: Definitions and concepts," ISO, 2018. [Online]. Available: https://www.iso.org/standard/63500.html

[2] J. Rubin and D. Chisnell, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 2nd ed. Wiley, 2008.

[3] A. Smith and B. Johnson, "Early Detection of Usability Flaws in Software Development," in *Proceedings of the 2025 IEEE International Conference on Software Engineering*, 2025, pp. 123-130. doi:10.1109/ICSE.2025.10700097.

[4] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1990, pp. 249–256.

[5] R. Molich and J. Nielsen, "Improving a human-computer dialogue," *Communications of the ACM*, vol. 33, no. 3, pp. 338–348, 1990.

[6] Choi, H., Kim, Y., & Jang, W. (2025). Enhancing the Usability of Patient Monitoring Devices in Intensive Care Units: Usability Engineering Processes for Early Warning System (EWS) Evaluation and Design. *Journal of Clinical Medicine*, *14*(9), 3218. https://doi.org/10.3390/jcm14093218

[7] N. Bevan, "ISO 9241-11 Revised: What have we learnt about usability since 1998?" in *Proceedings of HCI International*, 2009. [Online]. Available: https://www.researchgate.net/publication/300644710

[8] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A Survey on Large Language Models for Code Generation," *arXiv preprint arXiv:2406.00515*, 2024. Available: https://arxiv.org/abs/2406.00515

[9] F. Chiarello, V. Giordano, I. Spada, S. Barandoni, and G. Fantoni, "Future applications of generative large language models: A data-driven case study on ChatGPT," *Technovation*, vol.133, p.103002, 2024.

[10] OpenAI et al., "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[11] S. Bubeck et al., "Sparks of Artificial General Intelligence: Early experiments with GPT-4," *arXiv preprint arXiv:2303.12712*, 2023. [Online]. Available: https://arxiv.org/abs/2303.12712

[12] Y. Zhang, Q. Zhou, J. Liu, X. Page, Y. Shi, and F. Tian, "Does GenAI Make Usability Testing Obsolete? Evaluating UX-LLM for Mobile App Interfaces," *arXiv preprint arXiv:2411.00634*, 2024.

[13] D. Yang, S. Chen, and J. Wang, "Enhancing UX Evaluation Through Collaboration with Conversational AI Assistants," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023.

[14] Y. Liu, H. Jiang, Y. Jiang, et al., "Applying LLMs to User Experience Testing: Challenges and Opportunities," *Electronics*, vol. 13, no. 23, p. 4633, 2023.

[15] M. Jang and T. Lukasiewicz, "Consistency analysis of ChatGPT," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023, pp.15970–15985.

[16] Platt, Nolan; Luchs, Ethan; Nizamani, Sehrish Basir (2025). Data Analysis for Catching UX Flaws in Code: Leveraging LLMs to Identify Usability Flaws at the Development Stage. *figshare*.

[17] Li, M., Gao, Q. & Yu, T. Kappa statistic considerations in evaluating inter-rater reliability between two raters: which, when and context matters. *BMC Cancer* 23, 799, 2023.

APPENDIX

A. Per-Heuristic Pairwise Cohen's κ Values

TABLE V
PER-HEURISTIC PAIRWISE COHEN'S κ VALUES

Heuristic	κ **(1–2)**	κ **(1–3)**	κ **(2–3)**
Aesthetic & Minimalist Design	0.328	0.143	0.154
Consistency & Standards	-0.024	0.302	-0.029
Error Prevention	0.416	0.551	0.438
Flexibility & Efficiency	0.103	0.280	0.032
Help & Documentation	-0.068	0.123	0.188
Error Recovery Support[1]	0.406	0.292	0.231
Match to Real World	0.569	0.364	0.267
Recognition vs Recall	0.238	0.211	0.132
User Control & Freedom	0.327	0.534	-0.010
Visibility of Status	0.427	0.298	0.252

B. LLM Prompt Instructions

As discussed, we utilized a customized and consistent version of GPT-4o. This ensured the model received identical instructions and context across evaluations. The prompt used is shown below.

```
You are a usability evaluation
expert specializing in website
usability based on code analysis.
Users will provide a .zip file
containing the full code of a
website (HyperText Markup Language
(HTML), Cascading Style Sheets
(CSS), JavaScript (JS)). Your task
is to perform a detailed heuristic
evaluation based on Jakob Nielsen's
10 usability heuristics. For each
heuristic, you will:
- Assign a SeverityRating (0
through 4)
- Indicate IssueFound (true/false)
- Write an IssueDescription
- Provide CodeReference (file
name(s) and line number(s))
- Provide a CodeSnippet
- Complete EvaluationAnswers (with
explanations)
- Offer a clear Recommendation

If no issue is found,
SeverityRating must be 0,
IssueFound must be false, and
placeholders used as directed.
Always return a single, clean,
valid JSON object for all 10
heuristics, without skipping any
heuristic or reusing responses. Use
professional, detailed language.
Do not proceed unless the .ZIP is
provided. Verify internally that
all heuristics are evaluated and
JSON is valid before replying.

Accuracy and clarity are mandatory.
Nothing can be skipped or
overlooked.

Always send the output as a
downloadable .JSON file. The format
must always be:

{
  "Heuristic": "Heuristic",
  "SeverityRating": 0-4,
  "IssueFound": true/false,
  "IssueDescription": "Issue Description",
  "CodeReference": "Code reference",
  "CodeSnippet": "Code snippet",
  "EvaluationAnswers": {
    "Is the system providing appropriate
feedback?": "...",
    "Is the language understandable and
uses real-world conventions?": "...",
    "Can users undo and redo actions?":
"...",
    "Are standards consistent across the
system?": "...",
    "Is the design preventing problems
before they occur?": "...",
    "Is recognition prioritized over
recall?": "...",
    "Is the interface flexible for novices
and experts?": "...",
    "Is the design aesthetically
minimal?": "...",
    "Are error messages helpful and
clear?": "...",
    "Is help and documentation
accessible?": "..."
  },
  "Recommendation": "Clear and actionable
recommendation."
}
```

[1]Shortened from "Help Users Recognize, Diagnose, and Recover from Errors."