An Exploratory Study on Prompting Strategies for LLM-Assisted Heuristic Evaluation of Web User Interfaces

Ana Beatriz Farias
Federal Institute of Paraíba (IFPB)
Campina Grande, PB, Brazil
ana.farias.1@academico.ifpb.edu.br

Danyllo Albuquerque
Federal Institute of Paraíba (IFPB)
Campina Grande, PB, Brazil
danyllo.albuquerque@ifpb.edu.br

Emanuel Dantas Filho
Federal Institute of Pernambuco (IFPE)
Jaboatão dos Guararapes, PE, Brazil
emanuel.filho@jaboatao.ifpe.edu.br

Rohit Gheyi
Federal University of Campina Grande (UFCG)
Campina Grande, PB, Brazil
rohit@dsc.ufcg.edu.br

Francisco Petrônio Medeiros
Federal Institute of Paraíba (IFPB)
João Pessoa, PB, Brazil
petronio@ifpb.edu.br

Abstract

Usability is a key non-functional requirement in interactive software systems, and heuristic evaluation can support the verification and validation of user interfaces by identifying violations of usability principles. However, this process still depends on human expertise, is subject to evaluator variability, and may be difficult to scale. Although recent advances in Large Language Models (LLMs) create opportunities for AI-assisted heuristic evaluation, there is limited empirical evidence on the reliability of LLM-generated diagnoses. This paper presents an empirical study of GPT-5.4-assisted heuristic evaluation using Nielsen's heuristics (H), with diagnoses evaluated against an expert-derived ground truth. We evaluate zero-shot, few-shot, and chain-of-thought prompting on 12 web user interfaces, comparing model diagnoses with the reference evaluation using precision, recall, F1-score, and Cohen's kappa. Few-shot prompting achieved the best precision-recall balance (F1=0.51, κ=0.193), while chain-of-thought achieved the highest recall but produced more false positives. Heuristic-level results revealed distinct error patterns, with high recovery for H8 and H10 and frequent unsupported predictions for H4, H5, and H6, although these patterns are affected by uneven ground-truth prevalence. These findings suggest that prompting strategy is an important design consideration in LLM-assisted heuristic evaluation, while human validation remains necessary.

Keywords

AI-Assisted Software Testing, Large Language Models, Prompt Engineering, Usability Inspection, Heuristic Evaluation, Non-Functional Requirements, User Interfaces

1 Introduction

Usability is a fundamental quality attribute of interactive software systems [13, 20]. Even when a system is functionally correct, poor usability may compromise effectiveness, increase user effort, and reduce user satisfaction, ultimately affecting the perceived quality and adoption of the software [12, 19]. For this reason, usability is commonly treated as a non-functional requirement that should be evaluated throughout the software development lifecycle (SDLC) [4].

Among the various usability evaluation methods, heuristic evaluation remains one of the most widely adopted inspection techniques [7]. In this approach, specialists analyze user interfaces according to established usability principles, such as Nielsen's heuristics, and identify potential usability problems together with recommendations for improvement [18, 19]. Because it does not require recruiting users or conducting controlled experiments, heuristic evaluation can be performed early and repeatedly during development. However, the process remains highly dependent on human expertise, is subject to evaluator variability, and may become costly when large numbers of interfaces or frequent design iterations need to be assessed [7]. From a software testing perspective, the identified usability problems can be interpreted as failures of a non-functional requirement, making heuristic evaluation a form of inspection-based testing and verification [1].

Recent advances in Large Language Models (LLMs) have created new opportunities for automating or supporting software engineering activities that require reasoning over artifacts and generating structured feedback [10]. In software engineering, LLMs have been explored for tasks such as requirements analysis, code review, test generation, and software quality assessment [24]. More recently, vision-capable LLMs have enabled the analysis of visual software artifacts, including screenshots and user-interface mockups [8]. In this context, LLMs can analyze interface evidence, identify interface elements, relate observations to usability principles, and produce structured diagnoses describing potential usability violations [5, 6]. This capability creates the possibility of employing LLMs as AI-assisted evaluators, supporting the early detection of potential usability defects before expert validation.

Despite this potential, the reliability of LLM-based heuristic evaluation remains unclear. Recent studies have shown that LLMs can identify some usability and accessibility issues in digital interfaces. For example, Guerino et al. [9] reported that GPT-4o recovered only part of the usability issues identified by specialists while also generating unsupported findings. Similarly, Zhong et al. [27] and Pourasad and Maalej [21] highlighted limitations related to evidence availability, contextual reasoning, and diagnostic reliability. Consequently, an important question is not simply whether LLMs can perform heuristic evaluation, but under which conditions their

outputs become sufficiently reliable to support human reviewers during usability inspection activities.

One factor that may influence this reliability is the prompting strategy. Different prompting approaches, such as zero-shot, few-shot, and chain-of-thought, can affect how models interpret interface evidence, formulate diagnoses, and balance sensitivity against specificity [3, 25]. From a testing perspective, these strategies can be viewed as alternative configurations for the same inspection task, potentially influencing the number of detected violations, false positives, false negatives, and agreement with expert assessments. However, there is still limited empirical evidence on how prompting strategies affect the reliability of LLM-generated usability diagnoses when evaluated against expert ground truth.

This paper addresses this gap through an empirical study of LLM-assisted heuristic evaluation, in which a multimodal LLM performs heuristic inspections, and its diagnoses are compared against expert-derived ground truth. We investigate whether different prompting strategies influence the ability of an LLM to detect usability violations from static user-interface screenshots. More specifically, we compare zero-shot, few-shot, and chain-of-thought prompting with respect to precision, recall, F1-score, and Cohen's kappa. We also analyze whether different usability heuristics exhibit distinct agreement and error patterns when the available evidence is restricted to static screenshots.

Our results show that the prompting strategy affects diagnostic performance. Few-shot prompting achieved the best overall balance between precision and recall, producing the highest F1-score and the strongest agreement with expert ground truth. Chain-of-thought prompting achieved the highest recall, recovering more expert-identified violations, but also generated the largest number of false positives. Furthermore, the heuristic-level analysis revealed distinct patterns of agreement, over-identification, and under-identification across heuristics, although these patterns are influenced by the uneven prevalence of violations in the evaluated interfaces. These findings suggest that prompt design should be treated as a first-class configuration decision in LLM-assisted heuristic evaluation workflows and that human validation remains necessary when LLMs are used to support usability inspection.

This paper makes the following contributions: an empirical evaluation of LLM-assisted heuristic evaluation under zero-shot, few-shot, and chain-of-thought prompting; a quantitative comparison of these strategies using precision, recall, F1-score, and Cohen's kappa against a specialist ground truth; an analysis of heuristic-level agreement and over-identification from static screenshot evidence; and a replication package comprising interfaces, prompts, specialist evaluations, model outputs, and analysis data.

2 Background

This section introduces the concepts underlying this study, including usability as a software quality attribute, heuristic evaluation based on Nielsen's heuristics, and the prompting strategies adopted for LLM-assisted evaluation.

2.1 Usability as a Software Quality Attribute

Usability is a central factor in the development of interactive systems, directly influencing users' experience, effectiveness, efficiency, and satisfaction when interacting with digital interfaces [13, 20]. The ISO 9241-11 standard defines usability as the extent to which specified users can use a product to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use [12]. In software engineering, usability is commonly treated as a key non-functional requirement that shapes product quality alongside functional correctness, reliability, security, and performance.

From a testing and quality assurance perspective, usability is relevant because a functionally correct interface may still fail to support users in completing their tasks effectively. Interfaces with low usability may compromise task achievement, increase user effort, and reduce perceived product quality [19]. For this reason, usability evaluation should be integrated into the SDLC, enabling teams to identify interaction problems early, reduce rework, and provide evidence about the quality of user-facing software artifacts.

Usability evaluation methods can be broadly divided into user-based and inspection-based approaches. In user testing, representative participants perform tasks under observation. In expert inspection, specialists analyze the interface against predefined criteria without involving real users [22]. This work focuses on the latter category, and specifically on heuristic evaluation, because it provides a structured setting in which interface observations can be compared against an expert reference. In this sense, the specialist's judgment acts as a reference ground truth for assessing whether an LLM-assisted heuristic evaluation correctly identifies usability violations.

2.2 Heuristic Evaluation of User Interfaces

Heuristic evaluation is a usability inspection technique in which specialists analyze digital interfaces based on a set of predefined usability principles, known as heuristics, to identify potential interaction problems [22]. Unlike user testing, it does not require recruiting representative users or setting up controlled experimental sessions, making it relatively lightweight and applicable across different development stages.

The process generally follows four steps: (1) selection of heuristics and interfaces; (2) systematic inspection of the interface; (3) identification and categorization of usability problems; and (4) consolidation of findings into a structured report with recommendations [17]. Each identified problem is typically associated with one or more heuristics, each accompanied by a justification and suggestions for improvement. In automated or AI-assisted settings, this structure is particularly useful because it defines an expected diagnostic format that can be compared with a human reference.

Among the various sets of heuristics available in the literature, Nielsen's 10 Usability Heuristics are the most widely recognized and adopted in the field of Human-Computer Interaction (HCI) [19]. These heuristics provide a set of general principles that guide the evaluation of interfaces and help identify recurring usability problems. Table 1 lists the heuristic (H) codes and names adopted throughout this study. These heuristics constitute the evaluation framework used both by the specialist and by the LLM throughout the empirical study.

Although Nielsen's heuristics are general in nature, their broad scope covers different dimensions of HCI, including system-user

Table 1: Nielsen's usability heuristics used in this study.

Code	Heuristic
H1	Visibility of System Status
H2	Match Between System and the Real World
H3	User Control and Freedom
H4	Consistency and Standards
H5	Error Prevention
H6	Recognition Rather Than Recall
H7	Flexibility and Efficiency of Use
H8	Aesthetic and Minimalist Design
H9	Help Users Recognize, Diagnose, and Recover from Errors
H10	Help and Documentation

communication, error prevention and recovery, consistency, flexibility, and efficiency of use. However, their generality also introduces subjectivity, since the identification of violations depends on the evaluator's interpretation and on the evidence available in the interface being analyzed. This is especially important in static screenshot-based inspection, where some heuristics can be directly assessed from visible evidence, while others may require interaction, cross-screen comparison, or domain context. This characteristic motivates the investigation of automated and AI-assisted approaches that can support inspection while still requiring human validation.

2.3 Prompting Strategies for LLMs

LLMs are based on Transformer architectures trained to model the distribution of linguistic sequences and predict the next unit of text given a context [3, 23]. From this training, these models develop the ability to perform different natural language processing (NLP) tasks through conditioning with textual instructions, without requiring task-specific training. Recent LLMs (e.g., GPT, Claude, Gemini, LLaMA, and DeepSeek) are increasingly used in software engineering tasks that require interpreting artifacts, following instructions, and generating structured outputs [10], including software quality assurance, testing, and inspection activities [24].

In AI-assisted testing and inspection, the quality of an LLM output depends on the model and how the task is formulated [24]. This process, known as prompting, defines the instructions, examples, constraints, and output format used to guide the model [15]. In this work, prompting is treated as an experimental factor because different strategies may influence the number of reported violations, the specificity of diagnoses, and the balance between false positives and false negatives.

Three prompting strategies are investigated in this study:

- **Zero-Shot Prompting**: The model receives only a description of the task, without prior examples of input and output. The model is expected to generalize based on the provided instructions and the knowledge acquired during pre-training. In this study, this strategy evaluates the baseline capability of the model to perform usability inspection without demonstrations.
- **Few-Shot Prompting**: The model receives one or more examples of input and output before the target task. These examples serve as demonstrations of the expected response pattern, allowing the model to reproduce the desired structure in the new analysis [3]. In this study, the examples include annotated interface descriptions with associated heuristic violations, aiming to calibrate the level of evidence required to report a diagnosis.
- **Chain-of-Thought (CoT) Prompting**: The model is guided to decompose the task into intermediate reasoning steps before producing the final response. This strategy aims to make the analysis process explicit, leading the model to identify interface elements, relate them to usability heuristics, and then formulate the diagnosis [25]. Previous studies have shown that CoT prompting can improve performance in tasks that require structured reasoning and multi-step analysis.

These three strategies represent different levels of guidance to the model and constitute the independent variable investigated in this study. They are expected to produce different behaviors in terms of diagnostic coverage, precision, recall, agreement with expert assessments, and specificity of the generated usability findings.

3 Related Work

This section reviews studies on LLM-based interface evaluation, automated usability inspection, accessibility analysis, and AI-assisted testing. The goal is to position this work within the growing body of research on using LLMs to evaluate user-facing software artifacts and support software quality activities.

Recent studies have explored the use of LLMs and multimodal models for interface evaluation. Duan et al. [6] investigated GPT-4 for heuristic evaluation of Figma mockups through a plugin that converts interface structures into JSON and generates recommendations. In a subsequent study, Duan et al. [5] introduced UICrit, a dataset of design critiques for mobile interfaces, showing that multimodal models can support automated design feedback from screenshots. Other works have explored richer sources of evidence. Hsueh et al. [11] combined screenshots and interaction logs from user sessions for automated UX evaluation, while Pourasad and Maalej [21] used screenshots and source code to predict usability failures in mobile applications. Zhong et al. [27] proposed ScreenAudit, which combines accessibility metadata, navigation traces, and LLM reasoning to detect accessibility issues. Collectively, these studies demonstrate the potential of LLMs for evaluating user-facing software artifacts while highlighting the importance of evidence quality and contextual information.

The work most closely related to ours is that of Guerino et al. [9], who evaluated GPT-4o for heuristic inspection of web interfaces using screenshots and structured prompts. Their results showed that LLMs can recover part of the usability issues identified by specialists, but also generate unsupported findings and false positives. This reinforces the need to treat LLM outputs as candidate diagnoses that require human validation.

From a software testing perspective, recent research has investigated LLMs as assistants for test generation, test interpretation, and GUI testing. Barbosa et al. [2] explored LLM-generated tests for semantic conflict detection, while Fagundes and Teixeira [8] evaluated LLMs for multimodal GUI test generation in Android applications. These studies illustrate a broader trend in which LLMs are incorporated into testing workflows and assessed in terms of reliability, coverage, and diagnostic quality. Our work follows this direction by investigating LLM-assisted heuristic evaluation as a

software quality assessment activity focused on usability, a key non-functional requirement.

Table 2 summarizes the reviewed studies according to evaluation target, input evidence, prompting strategy, and ground truth. The comparison shows that prior work explores diverse input modalities and evaluation settings. Moreover, few studies explicitly analyze how prompting strategies influence the agreement between LLM-generated evaluations and specialist judgments.

Table 2: Summary of related work.

Study	Target	Input	Prompting	Ground Truth
Duan et al. [6]	UI critique	JSON	Structured	Specialist
Duan et al. [5]	Design critique	Screenshot	Few-shot / visual	Specialist
Hsueh et al. [11]	UX eval.	Screenshot + logs	Structured	Specialist
Pourasad et al. [21]	Usability failures	Screenshot/code	One-shot	Specialist
Zhong et al. [27]	Accessibility	Metadata + logs	CoT variants	Specialist
Guerino et al. [9]	Heuristic eval.	Screenshot	Structured	Specialist
Barbosa et al. [2]	Conflict testing	Code + tests	Test gen.	Test output
Fagundes et al. [8]	GUI testing	GUI evidence	Multimodal	Test exec.
This study	**Heuristic Evaluation**	**Screenshot**	**Zero-Shot/Few-Shot/CoT**	**Specialist**

Table 2 shows that prior studies have explored LLM-based interface evaluation, accessibility analysis, and software testing using a variety of input modalities, including screenshots, source code, interaction logs, and accessibility metadata. While these studies demonstrate the potential of LLMs for supporting software quality activities, the prompting strategy is typically fixed rather than treated as a controlled experimental factor. Furthermore, although several works compare LLM outputs against expert assessments, less attention has been given to the reliability of LLM-generated heuristic evaluations, particularly regarding false positives, false negatives, and agreement with expert ground truth. This study addresses these gaps by systematically comparing zero-shot, few-shot, and chain-of-thought prompting for LLM-assisted heuristic evaluation using Nielsen's heuristics and expert-derived ground truth.

4 Study Design

This study evaluates how different prompting strategies influence the reliability of LLM-assisted heuristic evaluation of user interfaces. Specifically, we investigate whether zero-shot, few-shot, and chain-of-thought prompting affect a multimodal LLM's ability to identify usability violations from static screenshots and whether their performance varies across Nielsen's heuristics.

To address these objectives, we designed an empirical study in which a multimodal LLM performs heuristic evaluations of user-interface screenshots using different prompting strategies. The diagnoses generated by the model are compared against an expert-derived ground truth to assess their reliability. The replication package described in the Data Availability section includes the interface set, prompts, expert evaluations, model outputs, and analysis data.

The study is guided by two Research Questions (RQ):

> *RQ1:* How do prompting strategies affect the performance of an LLM in heuristic evaluation of user interfaces?
> *RQ2:* What agreement and error patterns emerge across usability heuristics when LLM-assisted evaluation is performed from static screenshots?

RQ1 is operationalized by comparing three prompting strategies, *zero-shot*, *few-shot*, and *chain-of-thought*, in terms of their ability to identify heuristic violations relative to a specialist reference. In this study, the prompting strategy is the independent variable, while the dependent variables are precision, recall, F1-score, false positive and false negative counts, and Cohen's kappa. RQ2 is operationalized by analyzing the distribution of true positives, false positives, and false negatives across Nielsen's heuristics. This analysis allows us to examine heuristic-specific patterns of agreement, over-identification, and under-identification in the evaluated interfaces. To answer these RQs, the study follows six stages as described in Figure 1.

Stage 1: Interface Set. The experimental object consists of 12 static user interface screenshots representing typical e-commerce scenarios. The interfaces were generated using Google Stitch [1], an AI-based prototyping tool that produces interface mockups from natural language descriptions. The generated screens include common e-commerce components (e.g., navigation menus, search bars, forms, banners, shopping carts, and product detail areas).

The e-commerce domain was selected because it combines visual density, navigation structures, decision-support elements, and interaction-oriented components. These characteristics make it suitable for investigating usability violations related to heuristics such as consistency, error prevention, recognition rather than recall, user control, and aesthetic and minimalist design. The use of generated mockups also allowed us to control the experimental setting by avoiding dynamic content, personalization, and external system behavior.

All interfaces were evaluated as static screenshots. No DOM, HTML, source code, accessibility tree, or interaction logs were provided to either the model or the specialist. This decision ensures that both the automated inspection and the reference evaluation rely on the same type of evidence: visible information available in a single screenshot.

Stage 2: Expert Ground Truth. A manual heuristic evaluation of the 12 interfaces was initially produced by the first author, who has experience in applying Nielsen's heuristics, and subsequently reviewed by a usability specialist with experience in these heuristics. Disagreements or ambiguities identified during the review were discussed and resolved by consensus. The evaluation was performed without access to any model output and produced the "expert ground truth" used to assess the diagnoses generated by the LLM.

For each interface, the specialist inspected the visible elements and recorded which of Nielsen's 10 heuristics were violated. The reference was represented at the interface-heuristic level: for each interface and each heuristic, the ground truth indicates whether a violation was present or absent. This representation allows the model outputs to be compared with the specialist reference using classification-oriented metrics. The specialist evaluation was restricted to static visual evidence. Therefore, heuristics that require interaction, cross-screen comparison, or runtime behavior could only be marked when there was visible evidence in the screenshot. This restriction is important for interpreting false positives, since model-generated diagnoses referring to non-visible behavior were considered unsupported by the available evidence.

[1] https://stitch.withgoogle.com/

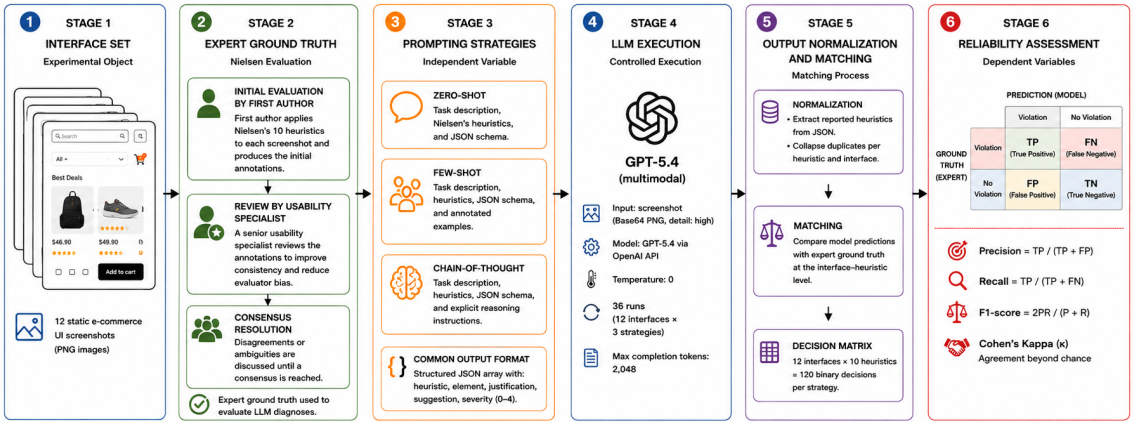

Figure 1: Overview of the empirical study design.

Stage 3: Prompting Strategies. Three prompting strategies were evaluated. All strategies shared the same system prompt and the same structured JSON output format. The system prompt instructed the model to act as a usability specialist, analyze only what was directly visible in the static screenshot, avoid inferring behavior that depends on interaction, and return only a valid JSON array. The strategies differed only in the user-turn prompt:

- *Zero-shot*: the prompt included the task description, the list of Nielsen's heuristics, and the expected output format, but no examples.
- *Few-shot*: the prompt included the same elements as zero-shot, plus annotated examples illustrating the expected diagnostic format and level of evidence. These examples were drawn from interfaces outside the evaluation set and were not derived from the specialist ground truth.
- *Chain-of-thought*: the prompt instructed the model to reason through the inspection process by identifying visible elements, relating them to applicable heuristics, and producing the final diagnosis as a JSON array.

All strategies used the same output schema. Each diagnosis included the violated heuristic, the affected interface element, a justification, and an improvement suggestion. Figure 2 summarizes the structure of the three prompting strategies.

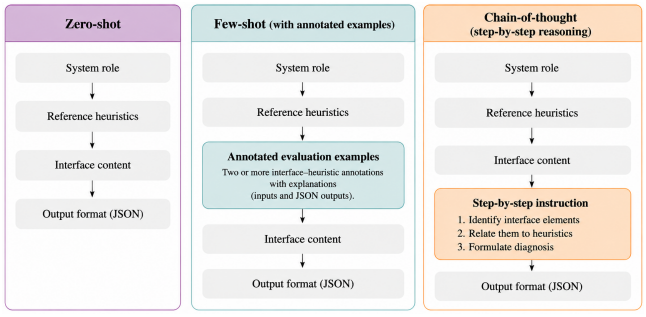

Figure 2: Structure of the prompting strategies.

Stage 4: LLM Execution. The model used in this study was GPT-5.4 (OpenAI), accessed via the OpenAI API with multimodal support. Each interface screenshot was provided as a Base64-encoded PNG image and included as an image content block in the user message. The image detail parameter was set to high to support fine-grained visual analysis.

When an image exceeded the API size constraints, it was resized to a maximum width of 1,280 pixels using the Lanczos resampling algorithm while preserving the aspect ratio. All executions were performed with the temperature set to 0 to reduce sampling variability. The API completion-token limit was set to 2,048 to accommodate structured JSON outputs.

For each combination of interface and prompting strategy, one API call was issued, resulting in 36 executions (12 interfaces × 3 prompting strategies). The raw model outputs were post-processed to remove residual markdown formatting, when present, before JSON parsing. Unparseable outputs would be treated as missing. All 36 responses were successfully parsed as valid JSON.

Stage 5: Output Normalization and Matching.

The model could generate multiple diagnoses for the same interface. Since the reference ground truth was defined at the interface-heuristic level, model outputs were normalized before metric computation. For each interface, the model's reported heuristics were extracted from the JSON array. Duplicate diagnoses associated with the same heuristic in the same interface were collapsed into a single predicted violation.

The comparison was then performed for each *interface–heuristic* pair. For each of the 12 interfaces and 10 heuristics, the model prediction was compared with the specialist ground truth, resulting in 120 binary decisions per prompting strategy. This design supports the computation of true positives, false positives, false negatives, true negatives, and agreement metrics.

Stage 6: Reliability Assessment. The diagnoses generated by the model were compared against the specialist ground truth using the following categories:

- *True Positive (TP)*: the model identified a heuristic violation that was also present in the specialist ground truth.

- *False Positive (FP)*: the model identified a heuristic violation that was not present in the specialist ground truth.
- *False Negative (FN)*: the specialist ground truth identified a heuristic violation that was not detected by the model.
- *True Negative (TN)*: both the model and the specialist ground truth agreed on the absence of a violation for a given heuristic.

Based on these categories, we computed precision, recall, F1-score, and Cohen's kappa for each prompting strategy. Precision measures the proportion of model-reported violations that were confirmed by the specialist ground truth. Recall measures the proportion of specialist-identified violations that the model recovers. F1-score summarizes the precision-recall trade-off. Cohen's kappa measures agreement between the model and the specialist beyond chance, which is particularly relevant in this study because a high number of false positives may inflate apparent coverage while reducing agreement.

Together, these metrics characterize the reliability of each prompting strategy as an LLM-assisted heuristic evaluation approach: precision reflects the cost of human validation, recall reflects the ability to recover specialist-identified violations, and kappa reflects the overall agreement with the expert ground truth. Finally, the objective of this study is not to establish definitive performance estimates for LLM-assisted heuristic evaluation, but rather to investigate the influence of prompting strategies under controlled conditions and provide an initial empirical baseline for future replications.

5 Results

This section presents the results of the empirical study on LLM-assisted heuristic evaluation of user interfaces. We first analyze the effect of the prompting strategy on model performance, considering precision, recall, F1-score, false positives, false negatives, and agreement with the specialist ground truth (RQ1). We then analyze the error profile across Nielsen's heuristics, examining heuristic-specific patterns of agreement, over-identification, and under-identification in the evaluated interfaces (RQ2). Detailed experimental data, including interface-level evaluations, model outputs, confusion matrices, and supplementary analysis artifacts, are available in the replication package described in the Data Availability section.

5.1 Effect of Prompting Strategy (RQ1)

Table 3 presents the aggregated results for the three prompting strategies. The reference ground truth was produced through expert heuristic evaluation and subsequently reviewed by a usability specialist, as described in Section 4. The evaluation was conducted at the interface-heuristic level, resulting in 120 binary decisions per strategy. A true positive indicates that the model identified a heuristic violation also present in the specialist ground truth. In contrast, a false positive indicates that the model reported a violation not supported by the ground truth.

The results show that the prompting strategy affects the reliability of LLM-generated usability evaluations. Few-shot prompting achieved the best overall balance, obtaining the highest precision (0.40), the highest F1-score (0.51), and the lowest number of false positives (39). This suggests that providing an annotated example helped calibrate the model's diagnostic threshold, reducing over-reporting while still recovering most of the violations identified in the expert ground truth.

Table 3: Performance metrics by prompting strategy.

Strategy	TP	FP	FN	TN	Prec.	Rec.	F1
Zero-shot	23	43	14	40	0.35	0.62	0.45
Few-shot	26	39	11	44	0.40	0.70	0.51
CoT	28	52	9	31	0.35	0.76	0.48

Compared with zero-shot prompting, few-shot improved precision by approximately 14% (0.35 to 0.40) and recall by 13% (0.62 to 0.70), while simultaneously reducing false positives by 9.3% (43 to 39). These results indicate that examples served not only as formatting demonstrations but also as implicit guidance regarding the level of evidence required to report a usability violation. Consequently, the model became more selective in reporting violations without sacrificing coverage.

Chain-of-thought prompting achieved the highest recall (0.76), identifying 28 of the 37 violations present in the expert ground truth. Relative to zero-shot prompting, recall increased by approximately 22%. However, this improvement came at the cost of a substantial increase in false positives, which rose from 43 to 52. As a result, precision remained unchanged at 0.35, and the F1-score (0.48) remained below that of few-shot prompting. These findings suggest that explicit reasoning encourages the model to explore a broader space of potential usability problems, increasing sensitivity but also amplifying unsupported diagnoses.

Zero-shot prompting served as the baseline configuration. Although it identified a substantial portion of the violations present in the expert ground truth (recall = 0.62), it produced a considerable number of unsupported findings (43 false positives). This result indicates that task instructions alone are insufficient to consistently constrain the model to visually grounded evidence, leading to diagnoses that extend beyond what can be reliably inferred from a static screenshot.

From an inspection perspective, the three strategies exhibit distinct evaluation profiles. Few-shot prompting produced the most balanced behavior, combining relatively high violation recovery with lower validation effort. Chain-of-thought behaved as a high-sensitivity detector, maximizing coverage at the expense of a larger review burden. Zero-shot served as the lower bound on both F1 and agreement, while occupying an intermediate position in terms of over-reporting. Therefore, the choice of prompting strategy depends on whether the objective is to maximize defect discovery or to minimize the number of unsupported findings that specialists must later review.

Another relevant observation is that all strategies produced more false positives than false negatives. Even the best-performing configuration (few-shot) generated 39 unsupported diagnoses against only 11 missed violations. This pattern suggests that the model adopts a high-sensitivity inspection behavior, preferring to report potential usability concerns rather than risk omitting them. While this tendency may be useful during exploratory inspections, it reinforces the need for human validation before generated diagnoses can be used as evidence of usability problems.

To complement the analysis of classification performance, Table 4 presents Cohen's kappa coefficient for each prompting strategy, together with observed agreement (P_o) and expected agreement by chance (P_e). While precision, recall, and F1-score measure the ability of the model to identify heuristic violations, kappa provides an estimate of agreement with the expert ground truth beyond chance.

Table 4: Agreement by prompting strategy.

Strategy	P_o	P_e	Kappa	Interpretation
Zero-shot	0.525	0.481	0.085	Slight
Few-shot	0.583	0.484	0.193	Slight
Chain-of-thought	0.492	0.436	0.099	Slight

All three strategies produced kappa values within the "slight" agreement range according to the interpretation proposed by Landis and Koch [14]. Few-shot achieved the strongest agreement with the expert ground truth ($\kappa = 0.193$), while zero-shot and chain-of-thought remained below 0.10. Although these values are relatively low, they are consistent with the high number of false positives observed across all strategies.

The difference between recall and kappa highlights an important characteristic of LLM-assisted heuristic evaluation. While recall measures the ability to recover violations present in the expert ground truth, kappa reflects the overall reliability of the generated diagnoses. Chain-of-thought, for example, recovered the largest number of expert-reported violations but simultaneously produced the largest number of unsupported findings. Consequently, improvements in recall did not translate into stronger agreement with the expert assessment.

From an AI-assisted evaluation perspective, these results indicate that high coverage alone is insufficient to characterize a prompting strategy as reliable. A strategy that generates excessive false positives may appear effective because it recovers many violations, but it also increases the cost of human validation and reduces trust in the generated diagnoses. Therefore, both coverage-oriented metrics (recall) and reliability-oriented metrics (precision and kappa) are necessary to assess the practical usefulness of LLM-assisted heuristic evaluation approaches.

Answer to RQ1. Prompting strategies affected the performance of the GPT-5.4 model in the heuristic evaluation of user interfaces. Few-shot achieved the best overall performance, yielding the highest precision (0.40), F1-score (0.51), and agreement with the specialist ground truth ($\kappa = 0.193$). In contrast, CoT maximized recall (0.76), recovering a larger number of specialist-identified violations but also generating the highest number of false positives (52). Zero-shot achieved intermediate results. Overall, the results show that prompting strategies influence the balance between coverage and reliability: few-shot prompts produce more balanced diagnoses, whereas CoT prompts favor broader violation detection at the cost of increased over-identification.

5.2 Error Profile by Heuristic (RQ2)

To answer RQ2, we analyzed the heuristic-level diagnoses produced by each prompting strategy and compared them with the expert ground truth. Figure 3 summarizes the identified heuristics for each interface. Green entries represent heuristics that matched the expert ground truth, whereas red entries correspond to heuristics reported by the model but absent from the expert evaluation.

UI	Ground Truth	GPT Zero-shot	GPT Few-shot	GPT Chain-of-Thought
01	H2, H3, H7, H10	H1, H4, H5, H6, H7, H8, H10	H1, H4, H6, H8, H10	H2, H4, H5, H6, H8, H10
02	H3, H10	H4, H5, H6, H8, H10	H4, H5, H6, H8, H10	H2, H4, H5, H6, H8, H10
03	H3, H4, H8, H10	H1, H3, H4, H5, H6, H8, H10	H1, H3, H4, H5, H6, H8, H10	H1, H3, H4, H5, H6, H7, H8, H10
04	H2, H3, H7, H8, H10	H1, H4, H6, H8, H10	H4, H5, H6, H8, H10	H1, H3, H4, H5, H6, H7, H8, H10
05	H1, H3, H8, H10	H4, H6, H8, H10	H4, H6, H8, H10	H1, H4, H5, H6, H8, H10
06	H10	H1, H4, H6, H7, H8, H10	H4, H6, H8, H10	H1, H4, H6, H7, H8, H10
07	H2, H3, H8, H10	H1, H4, H6, H8, H10	H3, H4, H5, H6, H8, H10	H1, H4, H5, H6, H8, H10
08	H8, H10	H4, H5, H6, H7, H8, H10	H4, H5, H6, H8, H10	H1, H4, H5, H6, H8, H10
09	H3, H8, H10	H1, H4, H5, H10	H3, H4, H5, H6, H8, H10	H1, H3, H4, H5, H6, H7, H8, H10
10	H3, H10	H4, H5, H6, H8, H10	H4, H5, H6, H8, H10	H2, H4, H5, H6, H8, H10
11	H3, H10	H3, H4, H5, H6, H8, H10	H3, H4, H5, H6, H8, H10	H1, H3, H4, H5, H6, H7, H8, H10
12	H3, H4, H8, H10	H4, H5, H6, H7, H8, H10	H1, H3, H4, H5, H6, H8, H10	H1, H4, H5, H6, H8, H10

Figure 3: Heuristics identified by each strategy.

The results reveal variation across heuristics. H10 (Help and Documentation) and H8 (Aesthetic and Minimalist Design) showed the highest recovery of expert annotations. H10 was identified in all interfaces by both the expert and the model, regardless of prompting strategy. However, because H10 was present in the ground truth for all 12 interfaces, its perfect recovery does not demonstrate the model's ability to distinguish between its presence and absence. H8 also showed high recall across prompting strategies, although it was frequently reported in interfaces where it was absent from the reference evaluation.

In contrast, H4 (Consistency and Standards), H5 (Error Prevention), and H6 (Recognition Rather Than Recall) exhibited frequent unsupported predictions. Although H4 appeared only twice in the ground truth, it was reported in almost every interface by all prompting strategies. H5 and H6 were absent from the expert evaluation but were repeatedly reported by the model. Thus, predictions for H5 and H6 constitute false positives in this dataset; however, the absence of positive cases prevents assessing the model's ability to correctly identify these heuristics when they are actually present.

A different pattern was observed for H3 (User Control and Freedom). This heuristic was one of the most frequent violations identified by the expert (10 occurrences), yet the model often missed it. Zero-shot recovered only two occurrences, while few-shot and chain-of-thought improved coverage to five and four detections, respectively. This result indicates substantial under-identification of H3 in the evaluated interfaces, even when examples or explicit reasoning instructions were provided.

The prompting strategies also produced distinct heuristic-level behaviors. Few-shot prompting achieved the most balanced profile, improving the detection of reference violations while limiting unsupported diagnoses. Chain-of-thought expanded coverage for heuristics such as H1, H2, and H7, recovering violations missed

by the other strategies, but also produced more unsupported predictions for H4, H5, and H6. Zero-shot generally produced lower coverage and missed a larger number of expert-identified violations.

Overall, the findings show that heuristic-level error profiles vary across prompting strategies and heuristics. However, these patterns should be interpreted cautiously because the prevalence of violations is highly uneven across heuristics. The results therefore characterize the behavior observed in the evaluated interfaces rather than establishing that particular heuristics are inherently more or less suitable for screenshot-based evaluation. Future studies with more balanced heuristic prevalence and richer evidence, such as interaction traces, task flows, or multi-screen contexts, are needed to investigate these differences further.

Answer to RQ2. Within the evaluated dataset, H8 and H10 showed high recovery of reference annotations, whereas H4, H5, and H6 exhibited frequent unsupported predictions. However, these heuristic-level patterns should be interpreted in light of the highly uneven ground-truth prevalence, including the absence of positive H5 and H6 cases and the presence of H10 in all evaluated interfaces. Thus, the results characterize heuristic-specific error patterns in this dataset rather than establishing that particular heuristics are inherently more reliable for screenshot-based evaluation.

6 Discussion

The findings of this study contribute to the ongoing discussion on the role of Generative AI in usability evaluation [21]. Although the prompting strategy affected model behavior, it did not fundamentally alter the level of agreement with expert assessments. Few-shot prompting achieved the highest precision (0.40) and agreement with the ground truth ($\kappa = 0.193$), whereas Chain-of-Thought (CoT) achieved the highest recall (0.76). However, all strategies remained within the "slight agreement" range, reinforcing observations from previous studies that current LLMs are not yet capable of reliably replicating expert heuristic evaluation judgments [6, 11]. Instead, the models appear more suitable as assistants that generate candidate findings requiring human validation.

The heuristic-level analysis further helps characterize these limitations and complements recent findings reported by Guerino et al. [9]. H8 (Aesthetic and Minimalist Design) and H10 (Help and Documentation) showed the highest recovery of reference annotations, whereas H4 (Consistency and Standards), H5 (Error Prevention), and H6 (Recognition Rather Than Recall) exhibited frequent unsupported predictions. However, these patterns should be interpreted cautiously because ground-truth prevalence was highly uneven: H10 was present in all interfaces, whereas H5 and H6 had no positive cases. Therefore, the results characterize heuristic-specific error patterns in this dataset rather than demonstrating that particular heuristics are inherently more suitable for screenshot-based evaluation.

One possible explanation for the superior performance of few-shot prompting is that the provided example acts as an implicit calibration mechanism. Rather than merely demonstrating the expected output structure, the example appears to communicate the level of evidence required to justify a usability diagnosis. In contrast, CoT prompting encourages broader exploration of potential violations, which increases sensitivity but also amplifies speculative findings that are not supported by the available screenshot evidence.

The results also highlight the importance of input modality. While approaches such as ScreenAudit [27] leverage richer sources of information, including accessibility metadata and interaction traces, our study deliberately restricted the model to static screenshots. Under these conditions, CoT prompting increased recall and false positives, suggesting that additional reasoning steps may encourage speculative diagnoses when contextual evidence is limited. This behavior indicates that prompt engineering alone may not be sufficient to overcome the limitations imposed by restricted interface evidence and that improvements in evaluation reliability may require richer contextual inputs.

Overall, our findings suggest that future advances in LLM-assisted usability evaluation are likely to depend on improved prompting strategies, richer contextual inputs, and more capable evaluation agents. Integrating screenshots with complementary artifacts, such as DOM structures, accessibility trees, interaction logs, and task flows, may help reduce unsupported diagnoses and improve agreement with expert evaluations. In this sense, recent research on LLM-driven GUI testing and autonomous interface exploration [8] points toward a promising direction for extending usability inspection beyond static interface analysis.

7 Implications

The findings of this study have implications for both research and practice in LLM-assisted heuristic evaluation. **From a research perspective**, the results show that the prompting strategy is a relevant experimental factor rather than a fixed configuration choice. Although all strategies used the same model and input evidence, they produced distinct precision-recall profiles. Few-shot prompting achieved the highest precision and agreement with the expert ground truth, whereas chain-of-thought achieved the highest recall at the cost of more false positives. These findings suggest that prompt design should be systematically investigated alongside other factors, such as model families, evaluation protocols, and input modalities.

The results also reveal distinct error profiles across usability heuristics. However, the uneven prevalence of heuristic violations in the ground truth limits conclusions about whether specific heuristics are inherently more suitable for screenshot-based evaluation. These findings motivate future studies using richer evaluation artifacts, such as DOM structures, accessibility trees, interaction logs, and task flows, as well as stronger benchmarks and multi-evaluator designs. The replication package accompanying this study may further support future benchmarking and replication efforts in LLM-assisted usability evaluation.

From a practical perspective, the findings indicate that GPT-5.4 can support usability inspections rather than replace human evaluators. Even the best-performing configuration achieved only slight agreement with the expert ground truth, indicating that generated diagnoses should be treated as candidate findings requiring validation. Consequently, organizations adopting LLM-assisted usability evaluation should position LLMs as assistants for preliminary

inspection, triage, and issue discovery rather than as automated decision-makers.

The observed trade-off between precision and recall also guides practitioners. Few-shot prompting may be preferable when reviewer effort is limited and reducing false positives is important. In contrast, chain-of-thought may be useful in exploratory inspections where maximizing issue discovery is the primary objective. Therefore, practitioners should carefully align prompting strategies with their evaluation goals and, whenever possible, complement screenshot-based analyses with richer interface artifacts and human review.

8 Threats to Validity

This section discusses the main threats to the validity of the study, organized according to the categories proposed by Wohlin et al. [26].

Construct Validity. The comparison between model outputs and the expert ground truth was conducted at the interface-heuristic level using exact heuristic matching. This representation supports the computation of classification-oriented metrics but may not capture partial agreement between diagnoses. For example, a usability problem categorized by the expert as H3 (User Control and Freedom) but reported by the model as H5 (Error Prevention) would be counted as both a false negative and a false positive, even if both diagnoses refer to the same interface issue. As a result, the reported metrics may underestimate cases in which the model identified a relevant problem but assigned a different heuristic.

Another construct threat concerns the granularity of the ground truth. The reference evaluation indicates only whether each heuristic was violated in each interface, without representing the number of distinct problems, their locations, or their severity. Consequently, the analysis focuses exclusively on the detection of heuristic violations. It does not assess whether the model identified the same interface element, produced equivalent explanations, assigned appropriate severity ratings, or generated actionable recommendations. Therefore, the reported results should be interpreted as measures of violation-detection capability rather than overall diagnostic quality. These aspects are important for practical adoption and should be investigated through qualitative assessments involving usability specialists in future work.

Internal Validity. The ground truth was produced by the first author and subsequently reviewed by a usability specialist with experience in Nielsen's heuristics, with disagreements resolved by consensus. However, because the evaluation was not conducted independently by multiple evaluators, inter-rater agreement could not be assessed, and evaluator-specific bias may remain. To mitigate this threat, the human evaluators and the model were restricted to the same evidence: information visible in a single screenshot.

The results may also be sensitive to prompt design. Small changes in the system prompt, few-shot example, chain-of-thought instructions, or output schema could affect model behavior. To reduce this threat, all strategies shared the same system prompt, output format, model configuration, and temperature setting, differing only in the prompting strategy. Finally, the post-processing procedure may influence the reported metrics. Model outputs were normalized by collapsing multiple diagnoses of the same heuristic into a single prediction. While necessary to align with the interface-heuristic representation adopted in the ground truth, this procedure may hide differences in the number and specificity of diagnoses generated by each strategy.

External Validity. The study used 12 AI-generated e-commerce mockups produced with Google Stitch. Although this controlled setting helped standardize the experiment and reduce confounding factors such as personalization, runtime behavior, and dynamic content, it may not fully represent real-world interfaces, which often contain more complex layouts, accessibility features, and interaction states.

The findings are also limited to a single application domain, a single input modality (static screenshots), and a single LLM family. Results may differ in domains such as healthcare, education, government services, or enterprise systems, where usability issues often require domain knowledge or richer contextual information. Likewise, evaluations based on screenshots may not generalize to settings that incorporate DOM structures, accessibility trees, source code, interaction logs, or task flows. Therefore, the observed superiority of few-shot prompting should be interpreted as evidence for this specific experimental setting rather than as a general claim across interface types, evidence sources, or LLM families.

Conclusion Validity. The sample size of 12 interfaces limits the statistical stability of the reported metrics. Although the interface-heuristic representation yields 120 binary decisions per strategy, these observations are not fully independent because decisions are grouped by interface and by heuristic. Moreover, each interface–strategy combination was executed once, so run-to-run variability in model outputs was not quantified. Cohen's kappa is also sensitive to class distribution and prevalence effects, which should be considered when interpreting the low agreement values.

The study was designed as an exploratory empirical comparison of prompting strategies rather than a statistical test of superiority; therefore, no hypothesis testing was performed. The results should thus be interpreted as preliminary evidence about zero-shot, few-shot, and CoT prompting in LLM-based usability evaluation. Future replications with larger interface sets, repeated executions, multiple specialists, additional model families, and richer input modalities are needed to strengthen the conclusions.

9 Final Remarks

This paper presented an empirical study evaluating how different prompting strategies influence the reliability of GPT-5.4-assisted heuristic evaluation of user interfaces. To this end, we compared zero-shot, few-shot, and chain-of-thought prompting against expert-derived ground truth using 12 e-commerce interface screenshots and Nielsen's heuristics, considering heuristic evaluation as an AI-assisted software testing activity.

The results show that the prompting strategy influences model behavior. Few-shot prompting achieved the best overall balance, with the highest F1-score (0.51), the highest precision (0.40), and the strongest agreement with the ground truth ($\kappa = 0.193$). Chain-of-thought prompting achieved the highest recall (0.76), but also produced the largest number of false positives. Zero-shot prompting provided an intermediate baseline. Across all strategies, agreement with the ground truth remained within the slight range, indicating room for improvement in the reliability of LLM-assisted heuristic

evaluation. At the heuristic level, H8 and H10 showed high recovery of reference annotations, whereas H4, H5, and H6 exhibited frequent unsupported predictions. However, these patterns should be interpreted in light of the uneven ground-truth prevalence, including the presence of H10 in all interfaces and the absence of positive H5 and H6 cases. Thus, the results reveal heuristic-specific error patterns but do not establish that particular heuristics are inherently more reliable for screenshot-based evaluation.

These findings suggest that LLM-assisted heuristic evaluation has the potential to support early-stage usability inspection by providing rapid, structured feedback on interface screenshots. However, the low agreement and frequent unsupported diagnoses observed in this study reinforce the need for specialist validation. In this sense, LLMs are better positioned as assistants for preliminary inspection and issue triage than as replacements for expert heuristic evaluation.

Future work will extend this study in six directions: (i) investigating richer input modalities, such as DOM structures, accessibility trees, interaction logs, and task flows, to determine whether additional evidence improves the reliability of LLM-assisted heuristic evaluation; (ii) evaluating additional LLM families and versions; (iii) expanding the interface set to include real-world systems and domains beyond e-commerce; (iv) involving multiple usability specialists to assess inter-rater reliability and construct a consensus ground truth; (v) evaluating the correctness, relevance, clarity, severity assessment, usefulness, and actionability of generated diagnoses; and (vi) exploring agentic LLM strategies for multi-step interface inspection and diagnosis refinement, building on recent agentic approaches in software quality tasks [16].

Generative AI Use

Generative AI tools were used to support language revision and the generation of illustrative figures. All research design decisions, experimental procedures, data analysis, interpretation of results, and final manuscript content remain the responsibility of the authors.

Artifact Availability

The replication package for this study - including the interface set, specialist evaluations, prompts, and model outputs - is publicly available at: http://doi.org/10.5281/zenodo.21891321.

References

[1] Issa Atoum, M. Baklizi, I. Alsmadi, A. Otoom, Taha Alhersh, Jafar Ababneh, Jameel Almalki, and S. M. Alshahrani. 2021. Challenges of Software Requirements Quality Assurance and Validation: A Systematic Literature Review. *IEEE Access* 9 (2021), 137613–137634. doi:10.1109/access.2021.3117989

[2] Nathalia Barbosa, Paulo Borba, and Léuson M. P. da Silva. 2025. Detecting Semantic Conflicts with LLM-Generated Tests. In *Proceedings of the 10th Brazilian Symposium on Systematic and Automated Software Testing (SAST 2025)*. Sociedade Brasileira de Computação, Recife, Brazil, 18–27.

[3] Tom B Brown et al. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

[4] Larissa Chazette and Kurt Schneider. 2020. Explainability as a non-functional requirement: challenges and recommendations. *Requirements Engineering* 25, 4 (2020), 493–514.

[5] Peitong Duan, Chin-yi Chen, Gang Li, Bjoern Hartmann, and Yang Li. 2024. UICrit: Enhancing Automated Design Evaluation with a UI Critique Dataset. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. ACM, 1–17. doi:10.1145/3654777.3676381

[6] Peitong Duan, Jeremy Warner, Yang Li, and Bjoern Hartmann. 2024. Generating Automatic Feedback on UI Mockups with Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. ACM, 1–20. doi:10.1145/3613904.3642782

[7] Andrea Esposito, Giuseppe Desolda, and R. Lanzilotti. 2024. The fine line between automation and augmentation in website usability evaluation. *Scientific Reports* 14, 1 (2024), 10129. doi:10.1038/s41598-024-59616-0

[8] Nayse Fagundes and Leopoldo Teixeira. 2025. Evaluating LLMs for Multimodal GUI Test Generation in Android Applications. In *Proceedings of the 10th Brazilian Symposium on Systematic and Automated Software Testing (SAST 2025)*. Sociedade Brasileira de Computação, Recife, Brazil, 28–36.

[9] Guilherme Guerino, Luiz Rodrigues, Bruna Capeleti, Rafael Ferreira Mello, André Freire, and Luciana Zaina. 2026. Can GPT-4o Evaluate Usability Like Human Experts? A Comparative Study on Issue Identification in Heuristic Evaluation. In *Human-Computer Interaction – INTERACT 2025*. Springer Nature Switzerland, Cham, 381–402.

[10] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024), 1–79. doi:10.1145/3695988

[11] Nien-Lin Hsueh, Hsuen-Jen Lin, and Lien-Chi Lai. 2024. Applying Large Language Model to User Experience Testing. *Electronics* 13, 23 (2024). doi:10.3390/electronics13234633

[12] International Organization for Standardization. 2018. ISO 9241-11:2018 — Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts. Available at: https://www.iso.org/standard/63500.html. Accessed: August 11, 2026.

[13] Malik Asif Jilani, Iftikhar Azim Niaz, Saad Naeem Zafar, Faiz Ali Shah, Shujaat Ali, Dilawar Shah, and Muhammad Tahir. 2026. Usability Quality Model: An Enhancement of Dromey's Model. *Engineering Reports* 8, 4 (2026), e70747.

[14] J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33, 1 (1977), 159–174.

[15] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. arXiv:2107.13586 [cs.CL] https://arxiv.org/abs/2107.13586

[16] Rian Melo, Pedro Simoes, Rohit Gheyi, Marcelo d'Amorim, Marcio Ribeiro, Gustavo Soares, Eduardo Almeida, and Elvys Soares. 2026. Agentic LMs: Hunting Down Test Smells. *IEEE Software* 43, 1 (2026), 32–40.

[17] Kate Moran and Kelly Gordon. 2023. How to Conduct a Heuristic Evaluation. https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/. Accessed: August 11, 2026.

[18] Jakob Nielsen. 2024. 10 Usability Heuristics for User Interface Design. https://www.nngroup.com/articles/ten-usability-heuristics/. Originally published in 1994; updated January 30, 2024. Accessed: August 11, 2026.

[19] Jakob Nielsen and Robert L. Mack. 1994. *Usability Inspection Methods*. John Wiley & Sons, New York.

[20] Donald A. Norman. 2006. *O design do dia a dia*. Rocco, Rio de Janeiro.

[21] Ali Ebrahimi Pourasad and Walid Maalej. 2025. Does GenAI Make Usability Testing Obsolete?. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. 437–449. doi:10.1109/ICSE55347.2025.00138

[22] Raquel Oliveira Prates et al. 2003. Avaliação de interfaces de usuário–conceitos e métodos. In *Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, Capítulo*, Vol. 6. sn, 28.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in neural information processing systems* 30.

[24] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering* 50, 4 (2024), 911–936.

[25] Jason Wei et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[26] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Vol. 236. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-29044-2

[27] Mingyuan Zhong and others. 2025. ScreenAudit: Detecting Screen Reader Accessibility Errors in Mobile Apps Using Large Language Models. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, Article 1163, 19 pages. doi:10.1145/3706598.3713797