Solaria-GPT: A Tailored ChatGPT Tool for Usability Inspection

Lennon Chaves
Federal University of Amazonas
Manaus, Amazonas, Brazil
lennon@icomp.ufam.edu.br

Márcia Lima
Federal University of Amazonas and
State University of Amazonas
Manaus, Amazonas, Brazil
msllima@uea.edu.br

Tayana Conte
Federal University of Amazonas
Manaus, Amazonas, Brazil
tayana@icomp.ufam.edu.br

ABSTRACT

Usability defects in software systems result in challenges for users during their interactions with the software. To address these challenges, usability inspection is key for detecting defects during software development, allowing them to be fixed before the defects reach end users. It also plays a critical role after software implementation, during the software maintenance phase. This paper presents Solaria-GPT, a tool based on a tailored version of ChatGPT, designed to assist software engineers in identifying and classifying usability defects following Nielsen's heuristics. Solaria-GPT facilitates two interactions: the detection of usability defects in user interfaces and the classification of each defect based on the violated heuristic. To evaluate the tool, we conducted a study to assess the performance of Solaria-GPT across textual and media-based inputs (screenshots and videos), focusing on its accuracy rate (correct heuristic violation), utility rate (valid defect identified by Solaria-GPT), and new defect rate (new defect identified by Solaria-GPT). The results indicated a 96.23% accuracy rate in heuristic classification from textual inputs. For utility rate and new defect rate, the Solaria-GPT achieved 86.67% and 86.67%, respectively, for screenshots, and 16.13% and 87.10%, respectively, for videos. Comparisons with other large language models (Claude, Qwen, Gemini, and Deepseek) demonstrated that Solaria-GPT outperformed all alternatives across metrics. These findings suggest that Solaria-GPT is a promising tool for enhancing software usability during the software development lifecycle.
Solaria-GPT Video: https://doi.org/10.5281/zenodo.15275798

KEYWORDS

ChatGPT, LLM, Heuristic Evaluation, Usability Inspection

1 Introduction

Usability is a critical quality attribute in software design, reflecting the extent to which users can effectively interact with and utilize a software [10]. As a fundamental aspect of software development, usability plays a role in shaping user experience, influencing factors such as satisfaction, productivity, and product success [8, 11, 14]. To assist users in operating software systems effectively, it is crucial to reduce their vulnerability to usability defects [17]. Intending to address these issues, two primary approaches are available: usability testing and usability inspection [7]. Usability testing involves end users who do not require technical expertise. In contrast, usability inspections are performed by usability experts who are familiar with human-computer interaction and usability evaluation [9, 15, 24, 28].

Usability inspection is typically conducted during the software development lifecycle, particularly during software implementation, to identify defects before the product is released to end users. Additionally, it can also be performed during the software maintenance phase, when bug fixes and new features are implemented. This approach enables developers to resolve issues early, preventing usability problems and improving software quality [16].

One widely used usability inspection technique is Nielsen's heuristic evaluation [17], which relies on 10 heuristics to assess the usability of a system interface. If one or more heuristics are violated during inspection, the system is considered to have usability defects that require correction before deployment.

To support usability inspection during software implementation and maintenance, we introduce Solaria-GPT, a tool based on a tailored version of ChatGPT [23]. Solaria-GPT assists software engineers in identifying usability defects and classifying them according to the violated Nielsen heuristics [17]. Users can interact with Solaria-GPT in two main ways: (1) detecting usability defects in user interfaces and (2) classifying each defect based on the heuristic it violates. To assess the effectiveness of the tool, we conducted a study and compared the performance of Solaria-GPT with those of other generative AI platforms, including Claude [2], Qwen [3], Gemini [26], and Deepseek [5].

Solaria-GPT is the result of a research process in which we followed the Research Playbook for Disruptive Innovations [25]. As first step, we employed McLuhan's tetrad [13, 27], where we conducted a reflection process to analyze the phenomena that the use of LLM can generate in the usability inspection process by identifying which the disruptive technology *enhances*, make *obsolete*, which concepts *retrieve* and what *reverses* when used exhaustively. In this analysis, (1) We identified what the use of LLM *improves* or *enhances* in usability inspection, which is the automatic classification of violated heuristics and detection of usability defects; (2) We also verified what the use of LLM can make *obsolete*, such as the need for manual inspection made only by humans; (3) What the use of LLM *retrieves*, which is using it as an interactive tutorial through questions (asking instructions about defect analysis) and answers (defects and heuristics violated); and (4) What usability inspection *reverses*, when using LLM exhaustively, such as autonomy and dependence on users to perform usability inspection. Thus, a first step of this research is to analyze the introduction of LLM for usability inspection as a tool, to understand how LLM can complement the usability inspection process. To achieve it, we discuss in this paper how the LLM *enhances* the usability inspection by proposing a tool to detect defects and classifying them according to the violated heuristic.

We organized this paper as follows: Section 2 provides the theoretical background about usability inspection and heuristic evaluation; Section 3 introduces Solaria-GPT; Section 4 presents the study; Section 5 discusses limitations; and Section 6 offers conclusions and directions for future work.

2 Heuristic Evaluation

Usability is a quality attribute that measures how easily users can interact with a system's interface [9, 12, 21]. When a software system contains usability defects, it becomes difficult to use, reflecting its low usability [16]. In particular, the usability inspection is a widely used method for identifying existing defects [17], and Nielsen's heuristic evaluation is a commonly adopted inspection technique that involves a set of 10 general usability principles[1] that software might follow to ensure a good user experience [17]. During this evaluation, a software engineer reviews the interface to identify violations of these principles. The presence of usability defects indicates that one or more heuristics have been violated and that the system must be improved to address these issues [17].

3 Solaria-GPT: A Tool for Usability Inspection

To support the usability inspection process throughout the software development lifecycle, particularly during implementation and maintenance, we developed Solaria-GPT[2], a tailored version of ChatGPT designed to detect usability defects and classify them according to the violated Nielsen heuristics. The Solaria-GPT is distributed under the MIT license. This section presents Solaria-GPT, outlining its features, users, architecture, and usage for usability inspection throughout the software development lifecycle.

3.1 Tool Description

GPT technology enables users to create tailored versions of ChatGPT adapted to perform specific tasks [19]. In particular, Solaria-GPT was designed to work as a usability specialist to perform heuristic evaluation, supporting the inspection process in two primary ways: (1) detecting usability defects in user interfaces and (2) classifying the violated Nielsen heuristics. Tailoring a GPT involves providing it with detailed instructions, which guide the model to perform well-defined tasks. Once tailored, the GPT can be published in the GPT Store [20], making it accessible to any user.

We developed Solaria-GPT following the steps outlined below:

(1) **Prompt Creation:** We created a prompt containing the necessary instructions to tailor ChatGPT to detect defects and identify the violated Nielsen heuristic [17].
(2) **GPT Construction:** We then embedded the prompt created into a GPT instance using the GPT Builder [19] interface.

3.1.1 Prompt Creation. To develop Solaria-GPT, we adopted the CRISPE template (Capacity and Role, Insight, Statement, Personality, and Experiment) [30] as described below:

- **Capacity and Role:** Solaria-GPT was defined to act as an expert in usability inspection.
- **Insight:** This component is related to the context we provided in our prompt, and it is represented by a file[3] describing Nielsen's heuristics and 10 examples for each heuristic violated.
- **Statement:** This component includes the explicit instructions the model must follow. For Solaria-GPT, the prompt specified the actions to be performed during inspection.
- **Personality:** We configured Solaria-GPT to adopt a formal tone in its responses.
- **Experiment:** We defined a structured format for the model's outputs, ensuring clarity when reporting usability defects and classifying the violated heuristics. For defect classification, the output structure is the violated Nielsen Heuristic [17]. For defect detection, the output structure is the defect description, the location where the defect happens, and the violated Nielsen heuristic [17].

3.1.2 GPT Construction. To create a tailored GPT, the creator must have a subscription to ChatGPT Plus [18], where the creator gains access to the GPT Builder. All details about the process to create Solaria-GPT are available in our artifacts (See Section 6).

3.2 Tool Features

Solaria-GPT includes the following features:

- **Text Input Processing:** Solaria-GPT can interpret defect descriptions in textual format and return the corresponding violated heuristic;
- **Media Input Processing:** Solaria-GPT is capable of processing images and videos to detect usability defects. In both cases, users upload the desired media type, and Solaria-GPT identifies the existing defects;
- **ChatGPT Interface:** As a tailored version of ChatGPT, Solaria-GPT shares the same user interface, making it familiar and easy to use;
- **Structured Response:** For text input, the output is always the heuristic violated by the reported defect. For image or video input, the response is structured as follows: (1) Description of the defect, (2) Location of the defect, and (3) Violated heuristic;
- **Questions and Answer Chat:** Users can interact with Solaria-GPT through a chat-based interface, engaging in a dialogue to ask questions (e.g., requesting defect analysis and violated heuristics) and receive as a response the defect description and the respective heuristic violated;
- **Multiplatform Support:** Solaria-GPT is available on both web and mobile platforms, allowing users to interact through multiple channels.

3.3 Tool Users

The Solaria-GPT is a tool designed to assist development teams that do not have dedicated usability specialists, *i.e.*, the users of Solaria-GPT are software engineers concerned with software quality.

3.4 Tool Architecture

To define the architecture of Solaria-GPT, we draw an analogy to the Back-end and Front-end architectural pattern [6]. The **Back-end** is responsible for business logic and rules, which in Solaria-GPT correspond to the prompt instructions along with the description of Nielsen's heuristics (Described in Section 3.1.2). In the back-end, ChatGPT processes user requests, applies the logic and rules (i.e.,

[1]The 10 heuristics are described at https://www.nngroup.com/articles/ten-usability-heuristics/
[2]Available at https://chatgpt.com/g/g-gO0eGq3rN-solaria-gpt
[3]The file with examples and details about prompt are available in our supplementary material: https://github.com/lennonchaves/Solaria-GPT

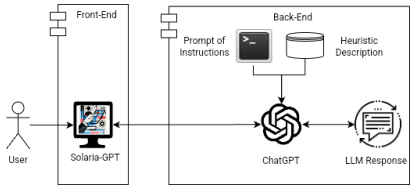


Figure 1: Solaria-GPT Architecture

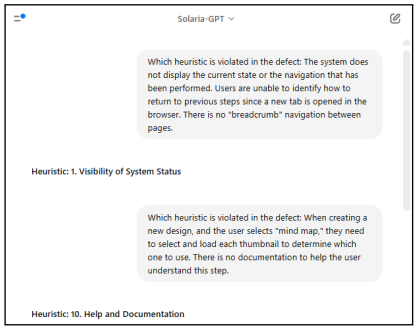
prompt and heuristic descriptions), and returns the output generated by the LLM. The **Front-End** is the user interface of Solaria-GPT, where users interact by submitting requests and receiving feedback or responses after data processing. Figure 1 graphically illustrates the implemented architecture of Solaria-GPT.

3.5 Tool Usage

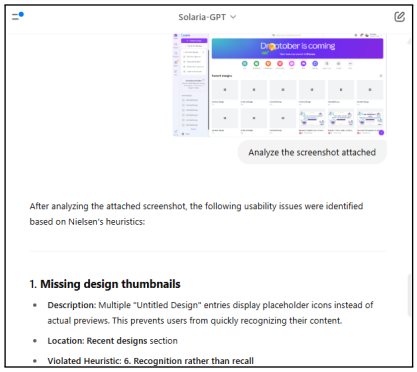
The Solaria-GPT was implemented to accept input data in two formats: text and media (screenshots and video). To use Solaria-GPT, the user might perform the following steps:

(1) **Login to the ChatGPT account**: https://chatgpt.com/auth/login
(2) Once logged in, the user must **access the Solaria-GPT** tool: https://chatgpt.com/g/g-gO0eGq3rN-solaria-gpt
(3) Then, the usage of Solaria-GPT might be according to the input data:
   - **For textual input**, the user must provide a detailed description of the usability defect, and then click to submit the text description, and Solaria-GPT classifies the defect and returns the violated heuristic.
   - **For screenshots and video**, the user might attach the media file (screenshot or video), whose usability is being evaluated, and then click to submit the file for evaluation. As a result, the Solaria-GPT will answer with a list of defects, their specific locations, and violated usability heuristics.

When a software engineer identifies usability defects within the software, seeks to verify the presence of such defects, or receives a defect report from a user, the software engineer can use Solaria-GPT. By inputting a textual description of the defect or providing a screenshot or video in the Solaria-GPT, a software engineer can identify usability defects and receive a response indicating which heuristic is violated in the reported defect. The Figure 2 illustrates the example of usage for textual description input (Figure 2a) and screenshot input (Figure 2b).



(a) Solaria-GPT Usage with Textual Input



(b) Solaria-GPT Usage with Screenshot Input

Figure 2: Example of Usage in the Solaria-GPT

4 Comparative Study

In this section, we describe the comparative study conducted to evaluate Solaria-GPT. In particular, we outline how the study was carried out, present the results, and provide a comparison with similar approaches.

4.1 Study Planning

4.1.1 Scope. We adopted the GQM (Goal-Question-Metric) framework [4] to define the main goal of the comparative study, as follows: Analyze **the Solaria-GPT** for the purpose of **evaluating** with respect to their **accuracy, utility, and new defect rate** from the point of view of the **researchers** in the context of **the usability inspection performed in the Canva website**[4].

4.1.2 Dataset Description. To conduct this study, we evaluated Solaria-GPT using two types of input data: text and media (screenshots and videos). For this purpose, we selected the Canva system, a widely accessible application designed for general use. For text-based input, the goal was to evaluate whether Solaria-GPT could correctly classify the violated heuristic given a defect description. For media-based input, the objective was to assess whether Solaria-GPT could process the media, identify usability defects, and indicate which heuristics were being violated.

For the text-based evaluation, we consolidated a dataset containing 53 usability defects. We analyzed each defect in collaboration with a usability inspection expert, and we identified the corresponding violated heuristic. Thus, the defect dataset includes both the defect description and the associated violated heuristic. The main objective of building this dataset was to compare Solaria-GPT's output with the expert-validated classification and verify whether the tool can correctly identify the appropriate heuristic.

For the media-based evaluation (screenshot or video), we collected 10 screenshots from different areas of the Canva system, all stored in PNG format. Additionally, we recorded 4 video clips in MP4 format simulating user interactions with Canva. The screenshot and video were also analyzed by an usability expert.

[4]https://www.canva.com/

> Which heuristic is violated in the following defect:
> [Defect Description]

Figure 3: Prompt Used for Text Input

> Perform a usability inspection of the attached screen?
> [Attached screenshot of the screen]

Figure 4: Prompt Used for Media-Based Input (Screenshot)

> Analyze the attached video and perform a usability inspection.
> [[Attached video with screen interaction]

Figure 5: Prompt Used for Media-Based Input (Video)

4.1.3 Prompt Used. To validate the Solaria-GPT, we also adopted a specific prompt for each type of input data. We adopted the zero-shot technique [29], as the sole objective was to illustrate how to use the Solaria-GPT. In particular, we created these prompts to enter the inputs in Solaria-GPT, however, users can freely use Solaria-GPT without adhering to specific prompts or methods, allowing them to easily use Solaria-GPT. The prompt used to evaluate text-based input is illustrated in Figure 3, where the goal is to analyze the inserted text and classify the usability heuristic being violated. For media-based input, we developed two different prompts. Figure 4 presents the prompt used to evaluate screenshot input, while Figure 5 shows the prompt used for video input analysis.

4.1.4 Metrics. Regarding the evaluation metrics for the Solaria-GPT tool, we defined 3 metrics:

- **Accuracy Rate:** This metric applies to text-based inputs. Its goal is to assess whether Solaria-GPT can correctly classify the heuristic associated with a defect based on a textual description. Accuracy rate is calculated as follows:

$$\text{Accuracy Rate} = \frac{\text{Number of Correct Classified Heuristics}^{*}}{\text{Total Number of Defects Found}} \times 100\% \quad (1)$$

  * Number of Defects Classified in the Correct Heuristics

- **Utility Rate:** This metric applies to media-based inputs (screenshots or videos). Its purpose is to determine whether the defect identified by Solaria-GPT is valid, that is, whether the defect actually exists in the context of the provided media. If the defect does not correspond to the content of the media, it is considered invalid. Utility rate is calculated as follows:

$$\text{Utility Rate} = \frac{\text{Number of Valid Defects}}{\text{Total Number of Defects Found}} \times 100\% \quad (2)$$

- **New Defect Rate:** This metric assesses whether a defect identified in the media input is uncatalogued in the defect dataset, which means it is an unknown defect. To support the computation, we removed the duplicated defects and consolidated them into a list of unique defects. If the defect is unique and not cataloged in the dataset yet, it indicates that the LLM is capable of identifying new defects (not previously known). New Defect Rate is computed as follows:

$$\text{New Defect Rate} = \frac{\text{Number of New Defects}}{\text{Total Number of Defects Found}} \times 100\% \quad (3)$$

4.1.5 Execution. We conducted the study as follows:

(1) **Validation of Text-Based Input:** We submitted the prompt described in Figure 3 to Solaria-GPT along with each defect from the defect dataset. Solaria-GPT processed each textual description and returned the corresponding violated heuristic. We then compared the heuristic classified by Solaria-GPT with the heuristic identified in the dataset. If the returned heuristic matched the one from the dataset, the classification was considered correct. Otherwise, it was marked as incorrect. In total, all 53 defects from the dataset were evaluated.
(2) **Validation of Media-Based Input (Screenshot):** Each of the 10 screenshots was submitted individually to Solaria-GPT along with the prompt described in Figure 4. For each input, Solaria-GPT returned the identified defects, their respective locations, and the violated heuristics.
(3) **Validation of Media-Based Input (Video):** Similarly, each of the 4 videos was submitted to Solaria-GPT along with the prompt described in Figure 5. Solaria-GPT returned the detected defects, the location where each defect occurred, and the corresponding violated heuristics.
(4) **Computation of Evaluation Metrics:** After completing the experiments for each type of input, we computed the evaluation metrics. For text-based inputs, we calculated the accuracy rate. For media-based inputs (screenshots and videos), we computed the utility rate and the new defect rate.
(5) **Comparison with Other Approaches:** After evaluating Solaria-GPT, we compared its performance with other similar LLMs, including Claude[5] [2], Qwen[6] [3], Gemini[7] [26], and Deepseek[8] [5]. We conducted comparative analyses based on the accuracy rate, utility rate, and new defect rate metrics. Additionally, we compared the features between the LLMs. These results are presented in Section 4.3.

4.2 Solaria-GPT Results

In this section, firstly, we will discuss the results of the accuracy rate, utility rate, and new defect rate for Solaria-GPT. In the next section, we will compare the Solaria-GPT results with other LLM approaches.

4.2.1 Validation of Text-Based Input. The analysis for text-based input aimed to evaluate the classification of the heuristic violated by the defect description. Our defect dataset consisted of 53 defects, and each one was submitted individually to the tool. Solaria-GPT returned the corresponding violated heuristic for each input. By comparing Solaria-GPT's output with the expected heuristic in our dataset, we determined whether the classification was correct. After evaluating all 53 defects, we observed that Solaria-GPT correctly classified 51 cases and misclassified only 2. Therefore, Solaria-GPT achieved an accuracy rate of 96.23% (51/53). One of the incorrect classifications highlighted a flaw characterized by icons that were too similar and could lead to user confusion. The expected heuristic classification was "*H4. Consistency and Standards*"; however, Solaria-GPT incorrectly classified it as "*H3. User Control and Freedom*". Another misclassified defect was related to the lack of freedom for users to copy content to their preferred location. Despite this, the system did not permit the task from being completed. Consequently, Solaria-GPT categorized it as "*H2. Match Between System and the Real World*", whereas the expected heuristics were actually "*H7.*

[5] https://claude.ai/
[6] https://chat.qwen.ai/
[7] https://gemini.google.com/
[8] https://chat.deepseek.com/

Flexibility and Efficiency of Use" and "*H3. User Control and Freedom*". These heuristics were mentioned in Section 2.

4.2.2 Validation of Media-Based Input (Screenshot). In the analysis of media-based input, specifically the screenshots of the evaluated system, we observed that from 10 screenshots, Solaria-GPT detected a total of 30 usability defects. After using the tool, we manually validated whether each reported defect was actually present in the context of the submitted screenshot. In this analysis, we found that 26 defects were valid, yielding a utility rate of 86.67% (26/30). In addition, we computed the new defect rate metric, *i.e.*, the capability of Solaria-GPT to detect new defects, and we found that 26 defects were unique and different from those already present in the defect dataset, resulting in a new defect rate of 86.67% (26/30).

4.2.3 Validation of Media-Based Input (Video). For this analysis, we submitted 4 videos to Solaria-GPT, which returned a total of 31 defects. We also evaluated the utility rate, and 5 defects were found to be valid, resulting in a utility rate of 16.13% (5/31). This indicates that although Solaria-GPT identified several defects in the videos, only a few defects were actually valid. Additionally, we observed that 27 defects detected by Solaria-GPT were unique and new, resulting in a computed new defect rate of 87.1% (27/31).

4.3 Comparison with Similar Approaches

Pourasad et al. [22] presented a study on the application of GenAI in usability testing. Although the main focus is on testing, the work also offers contributions to usability evaluations. In contrast, our proposal focuses specifically on the context of usability inspection. To complement the study of Solaria-GPT, we conducted a comparison with similar approaches, including: (1) Qwen by Alibaba Cloud, version 2.5 Max; (2) Deepseek, version V3; (3) Claude by Anthropic, version Sonnet 3.7; and (4) Gemini by Google, version 2.0. These models were selected because they are competitive LLMs on the market [1–3, 5, 26]. The same instructions (See Section 3.1) used for tailoring the Solaria-GPT were applied to these models to ensure a fair comparison under the same conditions. As these other approaches do not allow the creation a tailored version, we entered the instructions via prompt, and then we performed the evaluations.

In our results, we identified that almost all evaluated metrics (accuracy rate, utility rate, and new defect rate), Solaria-GPT consistently outperformed the other LLMs, demonstrating its potential as a valuable tool to *enhance* the usability inspection during or after software development. Additionally, we compared the feature sets of Solaria-GPT with those of the other models, as shown in Table 1. Unlike the other models, Solaria-GPT supports input via text, screenshots, and videos. Moreover, the ChatGPT platform allows for tailored GPT versions that can be published and shared through the GPT Store [20], making them accessible to any user. The other models still have limitations that are already addressed by Solaria-GPT, positioning Solaria-GPT as the most suitable approach for supporting usability inspection in software development.

4.3.1 Comparison for Text-Based Input. Table 2 presents a comparison of heuristic classification accuracy across different approaches. Solaria-GPT (GPT-4o) outperformed all others with an accuracy rate of 96.23%, followed by Deepseek (92.45%), Qwen (88.68%), Claude (84.91%), and Gemini (66.04%). This evaluation was based on all 53 textual defect descriptions in the dataset. **The results indicate the Solaria-GPT presents a better classification when heuristics are violated in comparison with other approaches.** Although Deepseek achieved a performance similar to that of Solaria-GPT in terms of text evaluation, its inability to process images or videos represents a disadvantage compared with our tool.

Table 2: Accuracy Rate Comparison

Approach	No. of CCH*	Accuracy Rate
Qwen	47	88.68%
Claude	45	84.91%
Gemini	35	66.04%
Solaria-GPT	51	96.23%
Deepseek	49	92.45%

* Number of Correct Classified Heuristics

4.3.2 Comparison for Media-Based Input (Screenshot). Table 3 compares the *utility rate* and *new defect rate* metrics for screenshot-based input. This analysis includes only models that support image processing. Deepseek was excluded since it does not currently support media analysis. The results show that Solaria-GPT achieved the best performance on both metrics (86.67% for utility rate and 86.67% for new defect rate). All 10 screenshots were used for each LLM in this comparison. These results showed that **Solaria-GPT is capable of detecting valid usability defects (utility)** — that is, defects actually present in the context of the screenshots submitted for inspection. Moreover, it **contributes to the usability inspection process by *enhancing* the identification of previously unknown defects (new defect rate)**.

Table 3: Utility Rate and New Defect Rate Comparison for Screenshot Input

Approach	Qwen	Claude	Gemini	Solaria-GPT	DeepSeek
Defects	96	47	30	30	-
No. of VD*	43	36	22	26	-
No. of ND**	40	37	19	26	-
Utility Rate	44.79%	76.60%	73.33%	86.67%	-
New Deffect Rate	41.67%	78.72%	63.33%	86.67%	-

* Number of Valid Defects

** Number of New Defects

4.3.3 Comparison for Media-Based Input (Video). Table 4 summarizes the comparison for video-based input. Only 4 videos were used in this analysis. Among the models evaluated, only Solaria-GPT and Qwen support video processing. Solaria-GPT did not outperform Qwen in terms of utility, however, it achieved a better result in new defect rate. Regarding utility rate, Qwen achieved a score of 65%, while Solaria-GPT reached only 16.13%. The low utility rate of Solaria-GPT during video-input analysis is attributed to defects that are not aligned with the context of the attached video. In contrast, Solaria-GPT demonstrated a higher new defect rate, with 87.1%, whereas Qwen achieved only 42.5%. Deepseek, Claude, and Gemini do not support video input and were therefore excluded from this comparative study. Based on the utility rate and new defect rate results, we observed that **Solaria-GPT contributes by *enhancing* the identification of new defects; however, it faced problems finding valid defects when the validated input data consisted**

Table 1: Feature Comparison Across Different Approaches

Approach	Qwen	Deepseek	Claude	Gemini	Solaria-GPT
Version	Qwen 2.5 Max	DeepSeek-V3	Claude 3.7 Sonnet	Gemini 2.0 Advanced	GPT-4o
Text Processing	Yes	Yes	Yes	Yes	Yes
Image Processing	Yes	No	Yes	Yes	Yes
Video Processing	Yes	No	No	No	Yes
Multiplataform*	No**	Yes	Yes	Yes	Yes
Tailored Version	No	No	No	Yes, but it cannot be shared with other users	Yes, and it can be shared with any user

* Web and Mobile Version

** It works only in Web Version. There is no mobile application

of video recordings. Nonetheless, Solaria-GPT illustrates its potential utility in defect screening, as it identifies both valid and novel defects that may be fixed before reaching end users.

Table 4: Utility Rate and New Defect Rate Comparison for Video Input

Approach	Qwen	Claude	Gemini	Solaria-GPT	DeepSeek
Defects	40	-	-	31	-
No. of VD*	26	-	-	5	-
No. of ND**	17	-	-	27	-
Utility Rate	65%	-	-	16.13%	-
New Deffect Rate	42.5%	-	-	87.1%	-

* Number of Valid Defects

** Number of New Defects

5 Limitations

During the development of Solaria-GPT, some limitations were identified in the use of this tailored version:

- **Collaboration:** It is not possible to add other researchers to collaboratively edit the Solaria-GPT prompt, as OpenAI currently does not support shared editing of tailored GPTs.
- **Daily Usage:** Since Solaria-GPT is based on the GPT-4o model, users without a ChatGPT Plus [18] subscription face daily usage restrictions, which limit the number of interactions within a five-hour window[9]. In contrast, ChatGPT Plus users are limited to 80 messages every three hours[10].
- **Classification:** The current version of Solaria-GPT assigns only one heuristic per defect. However, in some cases, a single defect may simultaneously violate multiple usability heuristics.
- **Dynamic Interaction Defects:** Usability defects could appear in dynamic interactions rather than static states. For example, usability defects related to keyboard shortcuts could not be identified by Solaria-GPT owing to their hidden nature. Specifically, when a shortcut is used, the defect must be explicitly described in the text for Solaria-GPT to understand it. This type of defect cannot be identified through screenshots, and with video input, and due to this reason, Solaria-GPT is still unable to interpret dynamic-based defects.
- **Defects Discrimination**: Software engineers can use Solaria-GPT to identify defects for media-based input. However, It is still necessary to triage defects, analyze false positives, and even prioritize critical defects that might be addressed.

[9]https://help.openai.com/en/articles/9275245-using-chatgpt-s-free-tier-faq
[10]https://help.openai.com/en/articles/6950777-what-is-chatgpt-plus

6 Conclusions and Future Works

In the software development lifecycle, usability inspection is a task to prevent end users from experiencing issues while using the system. In this manner, usability inspection plays an essential role both during implementation (while the software is still under development) and after implementation, in the maintenance phase (when the software has been released and requires fixes). In this paper, we investigated how the introduction of an LLM to support the usability inspection *enhances* the detection of usability defects and the classification of them according to the violated heuristic. To achieve it, we propose Solaria-GPT, an inspection assistant based on ChatGPT, adapted to aid usability inspection during software development. To evaluate Solaria-GPT, we conducted a study that analyzed its accuracy rate, utility rate, and new defect rate across different input types, including text and media (screenshots and videos). The results showed that Solaria-GPT achieved an accuracy rate of 96.23% in heuristic classification based on textual input. For the utility rate, it reached 86.67% for screenshots and 16.13% for video inputs. Regarding the new defect rate, the Solaria-GPT achieved 86.67% considering screenshots and 87.10% for videos. To complement this study, we compared Solaria-GPT with other LLMs, and Solaria-GPT outperformed them across all metrics. As future work, we are planning to explore different phenomena using McLuhan's tetrad [13], such as: how Solaria-GPT can *enhance* productivity and collaboration, what the introduction of LLM makes *obsolete*, *retrieves*, and *reverses* in the software engineers' experience. In addition, we also intend perform a comparison between Solaria-GPT and the general GPT model to show how effective the tool makes the usability inspection.

ARTIFACT AVAILABILITY

We organized the datasets, the Solaria-GPT prompt and the results of this study in the supplementary material, which is available on GitHub: https://github.com/lennonchaves/Solaria-GPT.

ACKNOWLEDGMENTS

We thank USES Research Group members for their support. We would like to thank the financial support granted by Project No 017/2024 Divulga CT&I FAPEAM; Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Finance Code 001; FAPEAM - through the POSGRAD project 2025/2026; CNPq 314797/2023-8; CNPq 443934/2023-1; CNPq 445029/2024-2; Amazonas State University (UEA) through Academic Productivity Program 01.02.011304.026472/2023-87.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).

[4] Victor R Basili. 1994. Goal, question, metric paradigm. *Encyclopedia of software engineering* 1 (1994), 528–532.

[5] DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[6] Yifei Gong, Feng Gu, Kengbin Chen, and Fei Wang. 2020. The Architecture of Micro-services and the Separation of Frond-end and Back-end Applied in a Campus Information System. In *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. IEEE, 321–324.

[7] Sabda Norman Hayat and Fatwa Ramdani. 2021. A comparative analysis of usability evaluation methods of academic mobile application: are four methods better?. In *Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology* (Malang, Indonesia) *(SIET '20)*. Association for Computing Machinery, New York, NY, USA, 136–141. https://doi.org/10.1145/3427423.3427435

[8] James Lewis and Jeff Sauro. 2021. *Usability and User Experience: Design and Evaluation*. 972–1015. https://doi.org/10.1002/9781119636113.ch38

[9] I Scott MacKenzie. 2024. Human-computer interaction: An empirical research perspective.

[10] Ankita Madan and Sanjay Kumar. 2012. Usability evaluation methods: a literature review. *International Journal of Engineering Science and Technology* 4 (02 2012).

[11] Leonardo Marques, Patrícia Matsubara, Walter Nakamura, Igor Wiese, Luciana Zaina, and Tayana Conte. 2019. UX-Tips: A UX evaluation technique to support the identification of software application problems. In *Anais do XXXIII Simpósio Brasileiro de Engenharia de Software* (Salvador). SBC, Porto Alegre, RS, Brasil. https://sol.sbc.org.br/index.php/sbes/article/view/9256

[12] Suéllen Martinelli, Nicolas Nascimento, Jonathan Souza, Afonso Sales, and Luciana Zaina. 2022. UX Requirements Matters: Guidelines to Support Software Teams on the Writing of Acceptance Criteria. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering* (Virtual Event, Brazil) *(SBES '22)*. Association for Computing Machinery, New York, NY, USA, 398–408. https://doi.org/10.1145/3555228.3555230

[13] Marshall McLuhan. 1977. Laws of the Media. *ETC: A Review of General Semantics* (1977), 173–179.

[14] Walter T. Nakamura, Edson Cesar de Oliveira, Elaine H.T. de Oliveira, David Redmiles, and Tayana Conte. 2022. What factors affect the UX in mobile apps? A systematic mapping study on the analysis of app store reviews. *J. Syst. Softw.* 193, C (Nov. 2022), 28 pages. https://doi.org/10.1016/j.jss.2022.111462

[15] Muhammad Nasir, Naveed Ikram, and Zakia Jalil. 2022. Usability inspection: Novice crowd inspectors versus expert. *Journal of Systems and Software* 183 (2022), 111122.

[16] Roberto Natella, Stefan Winter, Domenico Cotroneo, and Neeraj Suri. 2020. Analyzing the Effects of Bugs on Software Interfaces. *IEEE Transactions on Software Engineering* 46, 3 (2020), 280–301. https://doi.org/10.1109/TSE.2018.2850755

[17] Jakob Nielsen. 1995. Usability inspection methods. In *Conference Companion on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. Association for Computing Machinery, New York, NY, USA, 377–378. https://doi.org/10.1145/223355.223730

[18] Open AI. 2025. *ChatGPT Plus*. https://openai.com/index/chatgpt-plus/

[19] Open AI. 2025. *Creating a GPT*. https://help.openai.com/en/articles/8554397-creating-a-gpt

[20] Open AI. 2025. *Introducing GPT Store*. https://openai.com/index/introducing-the-gpt-store/

[21] Eduardo Gouveia Pinheiro, Larissa Albano Lopes, Tayana Uchôa Conte, and Luciana Aparecida Martinez Zaina. 2018. The contribution of non-technical stakeholders on the specification of UX requirements: an experimental study using the proto-persona technique. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering* (Sao Carlos, Brazil) *(SBES '18)*. Association for Computing Machinery, New York, NY, USA, 92–101. https://doi.org/10.1145/3266237.3266268

[22] 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. https://doi.org/10.1109/ICSE55347.2025.00138

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[24] Luis Rivero, Guto Kawakami, and Tayana Uchoa Conte. 2014. Using a Controlled Experiment to Evaluate Usability Inspection Technologies for Improving the Quality of Mobile Web Applications Earlier in their Design. In *2014 Brazilian Symposium on Software Engineering*. 161–170. https://doi.org/10.1109/SBES.2014.24

[25] Margaret-Anne Storey, Daniel Russo, Nicole Novielli, Takashi Kobayashi, and Dong Wang. 2024. A Disruptive Research Playbook for Studying Disruptive Innovations. *ACM Trans. Softw. Eng. Methodol.* 33, 8, Article 195 (Nov. 2024), 29 pages. https://doi.org/10.1145/3678172

[26] Gemini Team. 2024. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL] https://arxiv.org/abs/2312.11805

[27] Christoph Treude and Margaret-Anne Storey. 2025. Generative AI and Empirical Software Engineering: A Paradigm Shift. arXiv:2502.08108 [cs.SE] https://arxiv.org/abs/2502.08108

[28] Natasha Malveira Costa Valentim and Tayana Conte. 2014. Improving a Usability Inspection Technique Based on Quantitative and Qualitative Analysis. In *2014 Brazilian Symposium on Software Engineering*. 171–180. https://doi.org/10.1109/SBES.2014.23

[29] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652* (2021).

[30] Rui Zhong, Yang Cao, Jun Yu, and Masaharu Munetomo. 2024. Large Language Model Assisted Adversarial Robustness Neural Architecture Search. In *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*. 433–437. https://doi.org/10.1109/DOCS63458.2024.10704419