Does GenAI Make Usability Testing Obsolete?

Ali Ebrahimi Pourasad and Walid Maalej
Department of Informatics
Universität Hamburg
Hamburg, Germany
{ali.ebrahimi.pourasad,walid.maalej}@uni-hamburg.de

***Abstract*—Ensuring usability is crucial for the success of mobile apps. Usability issues can compromise user experience and negatively impact the perceived app quality. This paper presents UX-LLM, a novel tool powered by a Large Vision-Language Model that predicts usability issues in iOS apps. To evaluate the performance of UX-LLM, we predicted usability issues in two open-source apps of a medium complexity and asked two usability experts to assess the predictions. We also performed traditional usability testing and expert review for both apps and compared the results to those of UX-LLM. UX-LLM demonstrated precision ranging from 0.61 and 0.66 and recall between 0.35 and 0.38, indicating its ability to identify valid usability issues, yet failing to capture the majority of issues. Finally, we conducted a focus group with an app development team of a capstone project developing a transit app for visually impaired persons. The focus group expressed positive perceptions of UX-LLM as it identified unknown usability issues in their app. However, they also raised concerns about its integration into the development workflow, suggesting potential improvements. Our results show that UX-LLM cannot fully replace traditional usability evaluation methods but serves as a valuable supplement particularly for small teams with limited resources, to identify issues in less common user paths, due to its ability to inspect the source code.**

***Index Terms*—App Development, Large Language Model, Foundation Models, Usability Engineering, AI4SE, Recommender Systems, Quality Requirements, AI-Inspired Design.**

I. Introduction

With the rapid growth of the app market over the last decade, developing "good" apps has become crucial to vendor success [1], [2]. Studies have shown that users tend to favour apps that perform as expected and are easily understandable [3], [4]. Software usability is a key factor that crucially influences how users perceive the quality of an app [2], [5]. Usability can be broadly defined as "a concept that essentially refers to how easy it is for users to learn a system, how efficient they can be once they have learned it, and how enjoyable it is to use it" [2], [6]. Usability issues are problems that compromise the usability of an app, hindering a positive user experience [7]. It is thus crucial for developers to thoroughly detect and address usability issues for improving their apps [2], [5].

There are different ways to systematically identify usability issues. Conventional usability evaluation methods include usability testing in labs with users as well as theoretical analyses with experts [8], [9]. Once usability issues are identified, developers can address them and offer a refined app to their users [10]. However, it can be challenging, especially for small app development teams, to channel the resources and expertise needed for implementing an effective usability evaluation [11].

Generative Artificial Intelligence (GenAI) is a maturing technology that is increasingly getting attention for the purpose of automating various tasks in different domains, as it is capable of generating meaningful texts, images, and videos [12], [13]. Particularly, Foundational Models such as Large Language Models (LLMs) process a user textual prompt and construct responses by predicting next tokens in a partially formed context [14], [15]. Recently Foundation Models gained considerable interest in the software engineering domain, as they can be employed for a variety of purposes, including generating code and documentation, or fixing bugs [16]–[18].

This work investigates the extent to which GenAI can support or even automate usability evaluation for mobile apps. We introduce UX-LLM, a novel open-source tool that uses Foundation Models to detect usability issues in the separate views of native iOS apps. As input, UX-LLM requires a brief description of the app context, the source code, and an image of the analysed view. Section II introduces the implementation details of UX-LLM and the underlying prompt engineering. This constitutes our first contribution.

To evaluate UX-LLM performance and how it compares with conventional usability evaluation methods, we conducted a multi-method study consisting of expert assessments, expert reviews, and usability testing for two open-source iOS apps of medium complexity: a Quiz and a To-Do app. To understand how development teams perceive the support of such tools and explore possible concerns, we conduced a focus group within an app development project. Section III presents the design of our evaluation study.

The encouraging results confirm that LLM-based approaches are able to identify valid app usability issues. The results also indicate that GenAI approaches do not fully replace traditional methods but rather complement them—highlighting the potential as a supportive tool that can enhance usability evaluations during the development process. Section IV reports on the results of our evaluation study, which constitutes together with the data [19] our second contribution. The remainder of the paper discusses the work limitations and the threats to validity in Section VI, related work in Section V, and summarises the findings with their implication in Section VII.

II. UX-LLM Approach

UX-LLM is an application that predicts usability issues for a view of an iOS mobile app. For instance, in the case of a registration view, UX-LLM might predict the absence of

1558-1225/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSE55347.2025.00138

Fig. 1: User Interface of UX-LLM.

placeholders for input fields, leaving users uncertain about what information to input. As input, UX-LLM requires the app context, source code, and an image of the analysed view. The app context consists of two texts: a brief overview of the app (e.g. as found on app pages in app stores) and the user task, which describes the main goal of interacting with the view. For example, when looking at a meditation app, the overview could be: "A meditation app focused on improving stress relief and wellness". When analysing the progress tracking view, the user's task could be: "Review meditation history and achieved milestones". In addition, the source code provided needs to be SwiftUI[1] code from the view components and logic. The input image can be a screenshot from the running app or from the design file. The input data is packaged into a prompt using prompt engineering techniques and sent to a multimodal LLM, namely OpenAI's GPT-4 Turbo with Vision[2]. Finally, the output is a list of predicted usability issues with brief explanations.

Figure 1 displays the GUI of UX-LLM, showing an example where usability issues were predicted for a view of a Quiz app. A screenshot of the view is provided on the left side. On the right side, three text fields contain the app context and the source code, as mentioned earlier. Below, a start button initiates the testing to generate usability issues. Under this button, the identified issues are displayed, offering insights into potential areas of improvement in the user interface.

A. Prompt Engineering

Prompt engineering optimises the input to an LLM to enhance the output performance. White et al. describe it as: "the means by which LLMs are programmed via prompts" [20].

Listing 2: User Prompt

```
I have an iOS app about: [Inserted App Overview]

The user's task in this app view is about: [Inserted
     User Task].

An image of the app view is provided.

Below is the incomplete SwiftUI code for the app
    view.
This code includes the view's user interface and a
    view model for logic handling.
It may also include additional components like
    subviews, models, or preview code.
Source Code:
[Insert Source Code]
```

OpenAI's API[3] accepts a list of messages as input for their LLMs. UX-LLM utilises two types of messages: one for the system and another for user input, which can be seen in Listings 1 and 2, respectively. The system prompt provides high-level instructions on the behaviour expected from the model. Conversely, the user message is assembled using the information provided by the user about their app view.

Listing 1: System Prompt

```
You are a UX expert for mobile apps.
Your task is to identify usability issues with the
    information you get for an app's view.
An example of a usability issue could be: 'Lack of
    visual feedback on user interactions'.
Respond using app domain language; you must not use
    technical terminology or mention code details.

Enumerate the problems identified; add an empty
    paragraph after each enumeration; no preceding
    or following text.
```

Using several app projects which we had access to and were familiar with (but different from those used in the evaluation), we experimented with various Prompt Engineering strategies and tactics from the OpenAI documentation[4] to enhance the LLM performance. One tactic is to "Ask the model to adopt a persona", which we used in the system prompt where the LLM is instructed to act as a UX expert. Furthermore, the model was instructed to respond using domain-specific language, avoiding any mention of code. This helps focus the responses, facilitates more user-like feedback, and ensures accessibility to non-technical stakeholders.

Using the strategy: "Write clear instructions", we tested different user and system prompts. We particularly tested more detailed system prompts, where the LLM was guided to pinpoint issues using a pre-defined set of usability attributes such as effectiveness. However, in the end, we opted for a more open-ended question, allowing the LLM to freely identify any usability issues it could find, thereby enabling a more exploratory discovery of potential issues.

[1]https://developer.apple.com/xcode/swiftui/
[2]https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo
[3]https://platform.openai.com/docs/api-reference
[4]https://platform.openai.com/docs/guides/prompt-engineering

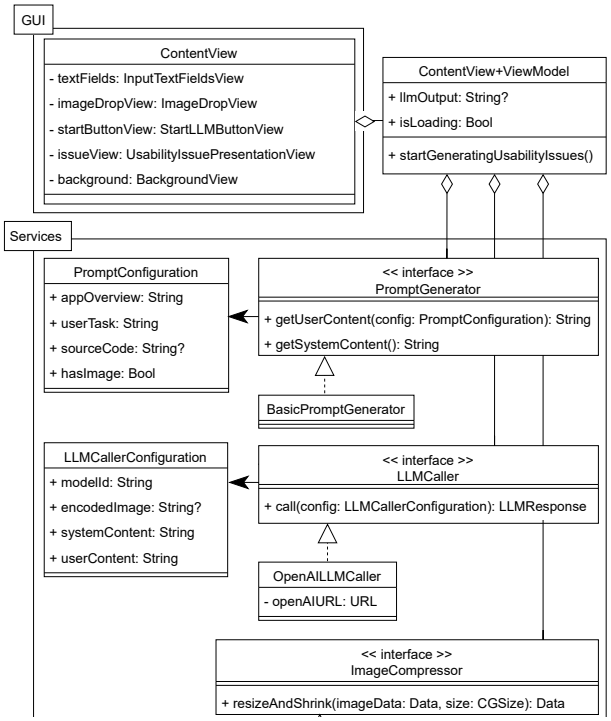

Fig. 2: Simplified class diagram of UX-LLM architecture.

Another tactic employed is: "Provide examples", commonly referred to as few-shot prompting [21]. It enhances the LLM results by directing the output towards a desired direction [22]. Similarly, we implemented one-shot prompting by presenting an example usability issue in the system prompt: "Lack of visual feedback on user interactions".

Another tactic we used is: "Use delimiters to clearly indicate distinct parts of the input". This is particularly useful in organising the user prompt, thus dividing the different sections: app overview, user task, image, and source code. Based on the documentation, this structure should help the LLM understand and process each part of the prompt effectively.

Furthermore, "Specify the desired length of the output" is a tactic to ensure that the responses are concise and focused. In the system prompt, we initially asked the LLM to list exactly 10 identified usability issues. However, in the testing, we ran into cases, where less than 10 usability issues were found and the LLM fabricated non-existent issues. Also, we found that when more than 10 issues could be found, the model would discard valuable information. In the end, we left out the exact number of issues to identify, allowing an unrestricted discovery.

B. Implementation

UX-LLM is a macOS application developed using `SwiftUI`. When entering an image, it uses TinyPNG API[5] to compress it. While waiting for the LLM response, a shader animation is shown, which is sourced from the Inferno project by twostraws[6].

[5]https://tinypng.com

[6]https://github.com/twostraws/Inferno

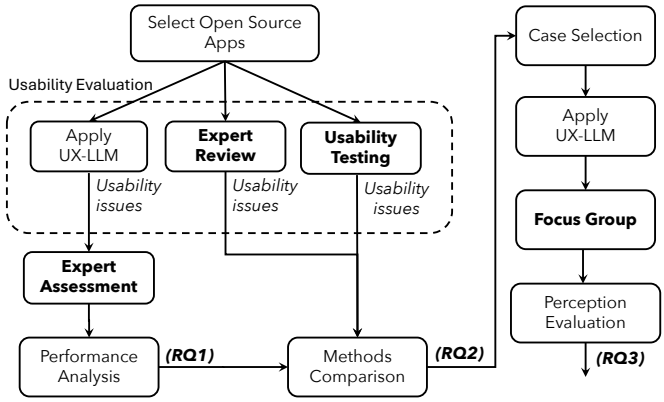

Fig. 3: Overview of our evaluation study.

Figure 2 shows a simplified Class diagram of UX-LLM architecture. The diagram particularly abstracts the complexity of the SwiftUI MVVM architecture by condensing the GUI layer and modelling combinations of views and their `ViewModels` as one. The primary focus is on the service layer, which operates using the "bridge pattern" [23] to separate the abstraction (the high-level logic) from the implementation (the low-level details), allowing them to change independently. This separation enhances modularity and makes the system more flexible and maintainable. The three services `PromptGenerator`, `LLMCaller`, and `ImageCompressor` are abstracted from their actual implementation, allowing UX-LLM to easily adapt to different models or APIs without significant changes. Additionally, the bridge pattern has simplified GUI testing by enabling the substitution of mock objects for actual services.

III. Evaluation Design

We conducted a thorough multi-method evaluation to assess the potential and limitation of GenAI-powered usability evaluation, focusing on the following research questions:

RQ1 How accurately can UX-LLM predict usability issues?

RQ2 How do UX-LLM predicted issues compare with those identified by traditional usability evaluation methods, and to what extent can UX-LLM replace these methods?

RQ3 How would an app development team perceive UX-LLM support during app development?

To answer RQ1 and RQ2, we applied UX-LLM to two apps and asked 2 usability experts to assess the predicted issues. We also performed usability testing and expert review on the same apps and compared the results to UX-LLM. To answer RQ3 we applied UX-LLM to an ongoing app development project and conducted a focus group with its development team where we discussed the results and limitations. Figure 3 overviews the entire evaluation study, which we discuss in the following.

A. Performance Analysis and Methods Comparison (RQ1+2)

For the evaluation of the UX-LLM prediction performance, we first had to select evaluation apps, which we *did not* use during the development and prompt-engineering. The evaluation apps must have accessible source code that is allowed to be

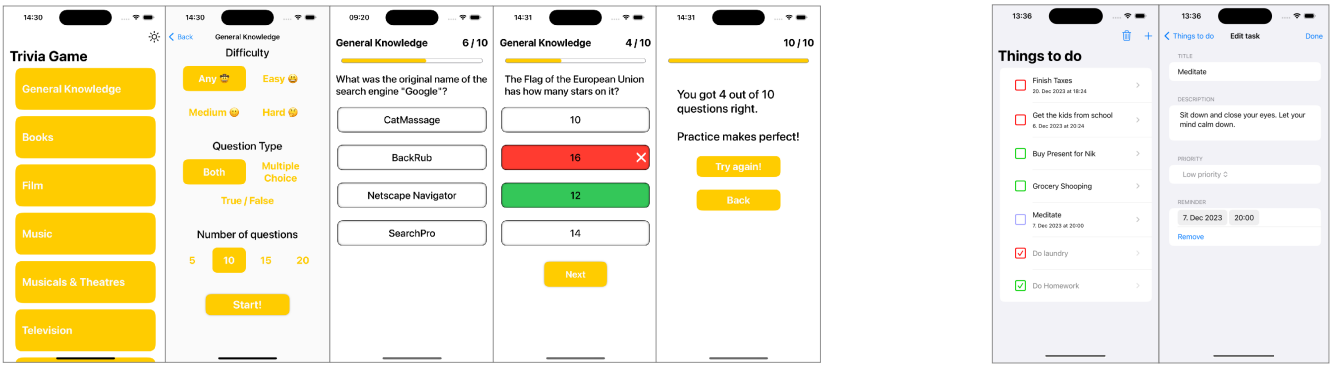

Fig. 4: Reference apps for the evaluation. Left: Quiz App, Right: To-Do App.

processed by OpenAI's GPT-4. We weren't able to use apps from partner companies, as none wanted to share their code with OpenAI due to security and privacy concerns. Consequently, we focused on open-source apps, particularly structurally straightforward apps of medium complexity to ensure a comprehensive, yet manageable evaluation. Furthermore, it was essential to have UX experts assess the predicted usability issues generated by UX-LLM to calculate precision and recall scores, which are established benchmarks for a tool like UX-LLM [24]. In this context, precision calculates of all the issues identified by UX-LLM, how many were actual usability issues. Recall calculates of all the actual usability issues available, how many did UX-LLM successfully identify.

For a comprehensive evaluation of UX-LLM, it is also crucial to compare its results with them of other methods used to identify usability issues (RQ2). To the best of our knowledge no dedicated dataset of open-source apps with pre-identified usability issues (to compare UX-LLM to) was available when this research was conducted. We also explored GitHub Issues and App Store reviews, as these could have been used to extract usability issues. Many open-source apps lacked a substantial user base, resulting in limited feedback documented on GitHub or the App Store. While large open-source apps like Firefox offered extensive user feedback, they were unsuitable due to not being written in SwiftUI and their rather high complexity.

We followed a systematic approach for the app selection. We first searched GitHub for "SwiftUI" and Google for "Open Source + SwiftUI". We then manually checked 50 apps looking for structurally straightforward apps of medium complexity to ensure a comprehensive, yet manageable evaluation. We excluded niche, very specialized, or inactive apps. Finally we chose two app: a Quiz app[7] and a To-Do app[8] shown in Figure 4. The Quiz app consists of four screens. Initially, the user selects a category in which to take a quiz, e.g. Geography. Next, they set up the quiz by adjusting settings, such as the length. Then, they proceed to take the quiz. In the end, there is a score screen that displays the results with options to either retry the quiz or return. The To-Do app consists of two screens.

[7]https://github.com/nealarch01/TriviaQuizApp
[8]https://github.com/fredrik9000/TodoList_SwiftUI

Participant	Age	Gender	Occupation
P1	24	M	Quality assurance specialist
P2	26	F	Civil engineer
P3	26	M	Marketing consultant
P4	27	M	Civil engineering student
P5	27	M	Law student
P6	28	F	Teacher
P7	30	M	Software developer
P8	31	M	Informatics student
P9	34	M	Software developer
P10	60	F	Dentist

TABLE I: Participants in usability testings of reference apps.

The first screen provides a list of all the tasks that have been added. The second screen is a task detail view that is used to create or edit a task.

After app selection, we conducted three usability evaluations in parallel on each app: applying UX-LLM, expert reviews, and usability testings. First, we gathered the source code of the two mentioned apps from GitHub and generated usability issues for each view with UX-LLM. Two UX experts assessed these generated issues, to check their correctness and enable calculating precision and recall [24].

1) Usability Testing: According to Nielsen Principles of Usability Testing, participants should represent actual users [8]. The selected apps are widely recognised: Quiz apps are popular and To-Do apps often come pre-installed on devices [25], [26]. Our screening process also ensured that all participants had prior experience with using similar apps.

According to Nielsen tests with five participants uncovers about 80% of usability issues [27]. However, this approach is debated, with some studies suggesting significant variability in the number of usability issues identified with five participants [28]. As we aim at identifying as many usability issues as feasible, we conducted tests for each app with 10 participants, whose details are depicted in Table I. We recruited participants within our personal network and met at convenient locations to conduct the testing, such as at home or in the office. The participants were seated at a table and used an iPhone

13 to interact with the apps. The iPhone was configured to record the screen, ensuring a comprehensive capture of the interactions. One author sat next to the participants with a laptop, documenting their behaviour.

The test sessions followed four steps. First, we welcomed and thanked participants and explained the obligatory details, such as the option to opt out whenever they want. Second, we explained the purpose of the study and demonstrated the think-aloud protocol they should use. We stressed that the focus of the study was on the apps and their usability, ensuring that participants did not feel they were being tested. Third, we gave participants the following list of usual tasks to perform [8]:
For the Quiz App:

- T1: Put the app in light mode. (*Goal:Find&use light mode*)
- T2: Take a short quiz on a topic that interests you. (*Goal: Take a quiz*)
- T3: You want to test your knowledge again on the same topic. (*User goal: Replay quiz*)
- T4: Next week, you will have an exam in Geography. Take a quiz to boost your knowledge. (*Goal: Search and setup a specific quiz*)

For the To-Do App:

- T5: Prepare for your upcoming days. Some To-Do's on your mind are: buying groceries, watering plants, and taking out the trash. (*Goal: Write down simple tasks*)
- T6: You must plan your vacation next Friday. (*Goal: Write down a high-priority task with a reminder*)
- T7: Groceries have been bought, except for one item "Bread". (*Goal: Edit task*)
- T8: You have emptied the trash and watered your plants. (*Goal: Mark tasks complete*)
- T9: Remove the "watering plants" task from your list, since it is no longer needed. (*Goal: Remove one task*)
- 10: Clear the entire list to make room for new tasks. (*Goal: Remove all tasks*)

Fourth, participants performed the tasks using think-aloud while we observed and took notes. This setup aimed to foster a comfortable environment that resembled typical usage scenarios for the participants. The full list of resulting usability issues can be found in the replication package [19].

2) Expert Review and Expert Assessment: As the second usability evaluation method, we conducted expert reviews with two UI/UX professionals (1 male, 1 female) each with six years of work experience. Their daily work is to conduct usability evaluations and UX design. The first session was conducted in person, and the second via zoom. Each session was divided into two distinct phases, with an introduction at the beginning to explain the obligatory details and the purpose of the study. In the **first phase**, the experts independently reviewed the two reference apps without knowing about UX-LLM, starting with the Quiz app. They navigated through the interfaces, vocalising their observations and identifying usability issues, similar to the think-aloud protocol. The identified usability issues are listed in the replication package [19].

TABLE II: Usability issues of UX-LLM (see replication package [19]) annotated with assessments from UX experts.

ID	E1	E2	ID	E1	E2	ID	E1	E2	ID	E1	E2
C1	A	A	C14	A	A	C27	B	A	C40	B	B
C2	C	B	C15	A	A	C28	D	D	C41	B	A
C3	D	D	C16	A	A	C29	B	B	C42	A	A
C4	A	A	C17	C	B	C30	B	B	C43	A	A
C5	B	B	C18	A	B	C31	D	D	C44	A	A
C6	A	B	C19	B	A	C32	B	A	C45	A	A
C7	A	A	C20	A	A	C33	A	A	C46	A	A
C8	A	A	C21	A	C	C34	D	D	C47	A	A
C9	B	A	C22	A	A	C35	A	A	C48	B	C
C10	C	A	C23	A	A	C36	A	A	C49	C	A
C11	A	B	C24	A	A	C37	A	A			
C12	A	A	C25	C	A	C38	B	B			
C13	A	B	C26	B	A	C39	B	B			

Legend: ID= Issue ID, E1= UX Expert 1, E2= UX Expert 2, A = Usability Issue, B = No Usability Issue, C = Uncertain, D = Irrelevant/Incorrect Statement.

The **second phase** was dedicated to assessing the usability issues identified by UX-LLM listed in details in the replication package [19]. The experts completed a questionnaire, where, for each usability issue identified by UX-LLM, they had to to classify it into one of the following categories: "Usability Issue", "No Usability Issue", "Uncertain", or "Irrelevant/ Incorrect Statement". Their assessments are listed in Table II.

B. Perception of a Development Team (RQ3)

To address RQ3, we conducted an in-depth focus group as part of a post-graduate university capstone project to develop an app for real clients from industry. The team was developing a native iOS transit app focusing on visually impaired users for a large public transport company. At the time of the focus group, the students had one month remaining to complete their project (out of five). They already had performed extensive requirements, design, and development work with several prototypes. The six-person team was organised into three pairs: each responsible for a project part. The focus group included three participants, one from each pair. The participating graduate students had at least two years of development experience working part time.

Before meeting with the group, we gathered their app code and evaluated it with UX-LLM, preparing usability issues for its seven main views (included in [19]). Then, we ran the focus group for approx. 2 hours in a university collaboration space along four steps: First, we interviewed the group about the project details, their roles, and their usability evaluation work so far. Second, we demonstrated UX-LLM and asked about initial impression and preliminary thoughts. Third, we presented the identified usability issues, asked the group to rate their usefulness and discussed them briefly one by one. Last, we reflected with the group about integrating UX-LLM into their workflows as well as any remaining concerns they had. We took notes of the session for later analysis. The entire list of questions is included in the replication package [19].

When answering semantic Scale questions, we used an approach similar to the planning poker from agile practices [29].

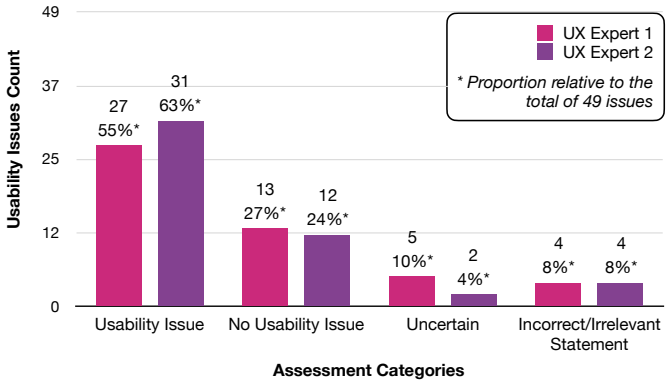

Fig. 5: Experts assessment of the issues identified by UX-LLM.

Participants simultaneously displayed their chosen numerical rating between 1 and 4 using hand gestures after a countdown from three, ensuring unbiased and independent responses.

IV. Evaluation Results

A. Performance Analysis

Figure 5 presents a bar chart of the UX experts assessments of the usability issues generated by UX-LLM. The two UX experts provided their evaluations in the four categories mentioned above. Expert 1 labelled 27 samples as actual usability issues, 13 as non-usability issues, 5 as uncertain, and 4 as incorrect/irrelevant statements. Expert 2 labelled 31 samples as usability issues, 12 as non-usability issues, 2 as uncertain, and 4 as incorrect/irrelevant statements. The top of each bar shows the number of issues identified by each expert in each category, with percentages indicating the relative ratio to the total of 49 issues assessed. This indicates that around 60% of usability issues identified by UX-LLM are valid.

However, we noted some discrepancies in the experts evaluations. For instance, the experts collectively identified 37 issues as usability issues, while individually they reported 27 and 31 issues. This is highlighted by Cohen's Kappa measure, a widely accepted metric for assessing evaluator agreement [30], which was $\kappa = 0.53$. Taking into account established benchmarks for interpreting Cohen's Kappa, such as the one proposed by Landis and Koch [31] or Altman [32], a κ value of 0.53 suggests "Moderate" agreement between the UX experts.

Looking at the differences, the main reason for disagreement seems to be the subjective nature of usability evaluation and a wide room for interpretation [9]. For example, the experts disagreed on issue C19 about the absence of a description label on the progress bar during the quiz. One labelled it as an issue, while the other argued that it should be intuitive to users what a progress bar in a quiz app represents. Similarly, C27 mentions that after the quiz, a "more detailed feedback [despite the score] could enhance the learning experience", which one expert viewed as a suggestion for a new feature rather than an existing usability issue. Likewise, issue C32 denotes the lack of clear separation between tasks, since tasks were listed without distinct dividers or spacing in the overview of the To-Do app. This was seen differently by experts: one classified it as a usability issue that could overwhelm users, while another considered it a minor aesthetic choice, assuming that users can navigate through lists regardless of visual grouping.

Next, we calculated the precision and recall [24] for UX-LLM. Due to variations in the expert assessments, different metrics for each expert are calculated. To calculate these metrics, we need a definition for true positives (TP), false positives (FP) and false negatives (FN) in the context of UX-LLM. True positives are all usability issues generated by UX-LLM that an expert has identified as a "Usability Issue" during their evaluation. Samples identified as "No Usability Issue" and "Incorrect/Irrelevant Statement" are defined as false positives. The issues labeled as "Uncertain" are excluded from the calculations because the experts said that they might provide valuable insights, even if they only offer a change in perspective. For example, issue C2 critiques the dark mode colour scheme, which was difficult for the experts to assess, but both agreed on the positive impact of highlighting it to developers. False negatives are defined as all usability issues not identified by UX-LLM but found during the usability testings or expert reviews. To avoid counting the same usability issue multiple times, we manually matched the same issues from each method, as presented in Table III. From this, the precision and recall values for Expert 1 (E1) and Expert 1 (E2) are as follows:

$$Precision_{E1} = \frac{TP_{E1}}{TP_{E1} + FP_{E1}} = \frac{27}{27+17} \approx 0.61$$

$$Recall_{E1} = \frac{TP_{E1}}{TP_{E1} + FN} = \frac{27}{27+51} \approx 0.35$$

$$Precision_{E2} = \frac{TP_{E2}}{TP_{E2} + FP_{E2}} = \frac{31}{31+16} \approx 0.66$$

$$Recall_{E2} = \frac{TP_{E2}}{TP_{E2} + FN} = \frac{31}{31+51} \approx 0.38$$

To directly address research question one (RQ1) the precision scores, which range from 0.61 to 0.66, are encouraging. This suggests that the usability issues generated by UX-LLM are generally valid while maintaining false positives at a manageable level. The recall scores, being on the lower side, ranging from 0.35 to 0.38, were expected given the complexity of detecting usability issues. This was highlighted by Molich et al. [9], who demonstrated that even simple websites can contain more than a hundred usability issues.

B. Comparison of Usability Evaluation Methods

To compare UX-LLM with usability testing and expert review, we matched the issues capturing the same problem possibly with a varying level of details as shown on Table III. Looking at the issues, there is a clear difference in

TABLE III: Matching of issues in reference apps identified by usability testing (A), expert review (B), and UX-LLM (C).

View	U. Testing	Expert 1	Expert 2	UX-LLM
Category View (Quiz App)	A1	B4	B4	/
	A2	B5	/	/
	A3	/	/	C6
	/	/	B2	C1, C12, C20, C23
	/	/	B6	C4
Setup View (Quiz App)	A5	B10, B11, B14	B9	C8, C14
	A6	/	B13	/
	A7	/	/	C11
Quiz View (Quiz App)	A9, A10	B21	/	C15, C21
	A14	B20	/	C22
	/	B18	B18	C19
Score View (Quiz App)	A15, A16	B29	/	C25
	/	B23, B26	/	C24
List View (To-Do App)	A17	/	B35	/
	A18	B37	B37	C37
	A19	B36	/	/
	A21	/	/	C33
	A22	/	B43	C35
	/	B38	/	C32
	/	B39	/	C36
Task Detail View (To-Do App)	A25	B57	/	C44, C47
	A26	B54	/	C46
	A27	B58	/	/
	/	B44	/	C42
	/	B51	B51	C41
	/	B53	/	C43

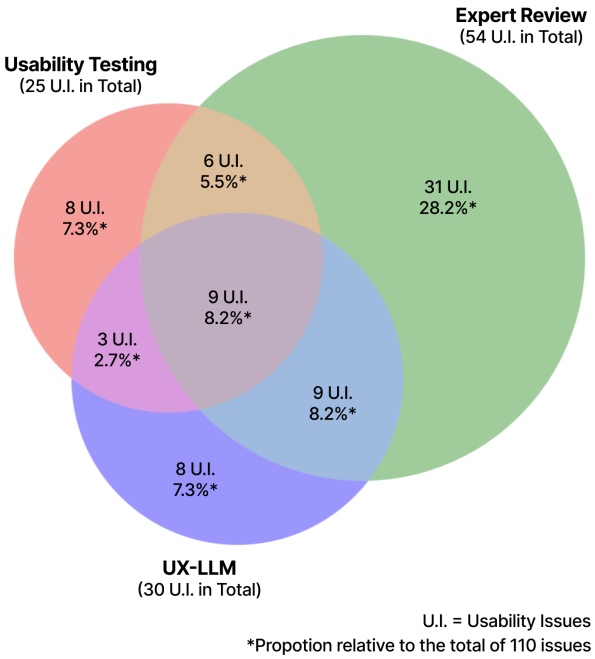

Fig. 6: Venn Diagram showing the overlap of usability issues identified by usability testings, expert reviews, and UX-LLM.

the detail captured. Issues identified from usability testings generally provide a broad impression, while those from expert reviews are sharper with more precise focus. For instance, issue A5 identified in a usability test, describes that users were overwhelmed by a chaotic-looking screen. This is matched with four issues from the expert reviews: B9, B10, B11, and B12 that provide more details: revealing reasons why the screen appears overwhelming such as "overload of content", "inconsistent grid layout", and "texts [...] inconsistent [...] use of capitalisation".

The Venn diagram shown in Figure 6 is obtained by matching duplicate/similar issues. In the issue set of UX-LLM, all samples were included where *at least one expert* identified them as actual usability issues. The diagram shows the overlap and unique contributions of the usability testing, the expert review, and UX-LLM. Of the total 110 issues, the usability testings uncovered 25 issues, with 8 unique to it. The expert review pointed out 54, including 31 unique issues. UX-LLM identified 30 issues, contributing 8 unique insights. The diagram shows that only 9 issues were identified by all three methods. There are 6 issues that both the testings and the experts identified but UX-LLM did not. The usability testing and UX-LLM together identified 3 issues that the expert review missed. Similarly, the expert review and UX-LLM together found 9 issues that were not detected through the usability testing.

It is noteworthy that the **expert review** identified a greater number of distinct issues. They break down overarching usability issues into several smaller usability issues, as in the example above (B9-B12). This allows for more targeted improvements, as more focused issues make resolution more actionable [33], [34]. Further, they point out minor design issues, such as a label being slightly misaligned vertically (B24), or minor copy-writing inconsistencies, where the same feature is referred to with different words (B30). These issues likely went unnoticed in the usability testing as they were too minor to significantly impact the user experience for an evaluator to notice (but are certainly still relevant).

The intersection between UX-LLM and the expert review, seems to contain issues that are difficult to extract from usability testings due to the absence of specific user groups or testing conditions. For instance, issue C1 refers to a text that is hard to read for "users with visual impairments or when viewing in bright light". Another type of issues identified are those that could not be stimulated with the the usability testing tasks. For example, issue B38 notes that "Completed tasks do not visually stand out from pending tasks". This might have gone unnoticed in the testings, as the maximum number of tasks displayed on the overview was four and the problem might become significant only as the list expands.

Finally, looking at the issues **only UX-LLM** has identified, it seems to spot issues other methods might miss by analysing code. It identifies, e.g., problems on less common user paths, such as the app performance under slow internet conditions. For example, issue C7 states: "When the app is fetching categories, there is no visual indication to the user that data is being loaded, which could lead to confusion". This scenario was neither encountered in the usability testing nor the expert reviews, as they had fast internet connections. However, it is valid for

users with a limited internet connection. Another example is issue C18, which notes: "The question text does not have a maximum line limit, which may cause issues if the question is too long, leading to the text being cut off or the layout looking cluttered". This issue was also not encountered because the test questions were not overly long. With different longer questions, this could become a usability issue. However, UX-LLM did not identify broad context or navigation-related issues, likely since it analyses only one view at a time. Issue B30, e.g., that involves inconsistent wording of the same feature across different screens, was beyond UX-LLM capability.

C. Perception of a Development Team (RQ3)

1) Current Practices of the Team: The focus group started with a waterfall-like approach, initially creating a high-fidelity prototype, which then served as the foundation for development. Later, the prototype was discarded and design changes were directly implemented in the app, bypassing adjustments to the design files. In addition, they conducted several usability evaluations through testings and walk-through, engaging directly with representatives of the visually impaired community.

2) Predicting Usability Issues with UX-LLM: During the interview, we briefly demonstrated UX-LLM to the team by using it on one of their app views. Their initial impression of its potential to enhance the user experience of an app was moderately positive, with an average rating of $2.3/4$. One participant stated: "Some issues feel a bit generic and some don't make sense, since they are addressed in previous screens". More specifically, UX-LLM criticised the text size for being too small, which was just personalised in the onboarding before. The other participant said: "I appreciate the fresh perspectives it offers. Even incorrect usability issues can be valuable as they make me reevaluate design decisions" (referring to an issue where UX-LLM criticise the colour scheme as not accessible).

We examined all issues identified by UX-LLM within the seven main app views (removing results that were clearly false positives). The students evaluated the **usefulness** of these issues, resulting in a positive average rating of $3.3/4$. They found some of the issues particularly insightful, even uncovering bugs they had not recognised before. One participant said: "The feedback on the button bug was spot on; it's not something we would have thought about by ourselves". The other acknowledged that the issues are properly phrased so that they can "directly be imported as to-dos on our task board". Also, "On some screens we assumed something is not ideal, but we did not know what the problem was, these issues are very helpful". Overall, they felt that pre-filtered results significantly improved the usefulness and appeal of UX-LLM.

3) Integrating UX-LLM into Development Workflows: Regarding the ease of integrating UX-LLM, the team gave it a mildly positive average rating of $2.6/4$. A developer mentioned that it would be a further burden for him to fill out UX-LLM required fields. He stated: "I'm a laid-back person, so it would annoy me to have to use another application beside my IDE". In addition, he said: "I'm not sure if I'd regularly go through these $\sim$10 points", indicating that it could be tedious to go over the issues for every app view over the course of development.

The group considered having a team member consistently review the app views with UX-LLM and share the feedback with the rest. One of them said, "Personally, I wouldn't mind using the tool, as I, being the project manager, could regularly use UX-LLM and incorporate its findings into our task board". Although doable with their team size, they recognised the challenge of adopting another task given their current time pressure. The UX designer stated, "It's great to see an overview of what's available; you can quickly eliminate unnecessary issues and reflect on them. In the end, it saves a lot of time as it is easier than conducting usability evaluations ourselves", indicating her excitement about the tool and that she feels like she can easily filter out the false positives.

4) Additional concerns and improvement suggestions: For applying UX-LLM in daily work, the group proposed embedding it as a plugin in the IDE or in a continuous integration (CI) pipeline to allow for automatic operation in the background. They emphasised that an IDE plugin "just makes sense, as the Xcode IDE[9] already has the code and view's preview side by side". In addition, the team expressed the desire to not only identify issues but also provide solutions, including alternative designs of their input image. The designer said: "When it criticised the accessibility of the colours, it would be nice if it could also show what colours to use instead". Interestingly, they also stated a limitation that could result from the policy of the LLM used. A discussed example is when creating an app about wine, as some LLMs might not be allowed to answer queries about alcohol. Furthermore, the presentation of usability issues was relevant to the team, as they found it demotivating to receive a lengthy list pointing out their product deficiencies. Also, the developer was sceptical about UX-LLM capability to identify usability issues in features reliant on hardware interactions involving user input or external factors, such as camera functions and voice input. Lastly, they emphasised the importance of a holistic analysis to detect broader inconsistencies and navigation issues. This approach would also reduce predicted usability issues, which are already addressed in other sections of the app.

Overall, the team perspective on UX-LLM was positive, ending the session saying: "It has identified issues that we overlooked, and not just a few". The tool was credited with uncovering valuable issues that had previously gone unnoticed, besides conducting usability evaluation methods.

V. Related Work

There is a huge body of knowledge around usability [35]–[37] and usability engineering [8], [38], [39], including a recent comprehensive review of usability for mobile apps [10]. The closest areas to our work are software analysis for usability, user data and feedback analytics, and GenAI for GUI development.

[9]https://developer.apple.com/xcode/

1) Code Analysis for Usability: Research has proposed static and dynamic code analysis to identify usability issues. For example, Mathur et al. [40] proposed an analysis framework to check the source code of Android apps against (pre-configured) usability guidelines and validation cases (e.g. all password fields should provide an option to reveal the clear text). Similarly, UX-LLM only uses development artefacts, notably source code, but focuses on iOS apps and does not require pre-configuration. In a study with 16 mobile app companies, Mathur et al. found that developers often skip usability evaluation due to needed effort, lack of resources, and know-how [40]. This is a main motivation for our AI-assisted usability evaluation. The authors also found their framework to be effective and helpful but developers were sceptical about the extra work to build own validation cases. Similar methods [41], [42] are capable of detecting pre-defined issue (categories), which can be beneficial, e.g., when evaluating accessibility requirements [43]–[45]. However, they require initial setup and maintenance as development libraries, guidelines, and usage scenarios evolve over time [46]. As UX-LLM the main advantage of code analysis approaches is their applicability during the development phase to predict and prevent usability issues, before such issues impact real users.

2) Usage Data Analytics: App usage logs deliver great value for developers as they enable analysing the user behaviour [47]–[49]. For example, they can identify which app features are used most and focus usability evaluation on those areas [50], [51]. Mobile tracking [51], [52] aka analytics tools like *Firebase* [10] or *TelemetryDeck* [11] enable exploring runtime, engagement, and navigation data, or tracking actions such as button presses. However, such tracking remains exploratory and raises privacy concerns depending on its implementation [53], [54].

To detect issues, Park et al. [55] generated a representative "real user activity model" from execution traces and an "expected activity model" from developers' execution of intended usage scenarios. The authors matched both models to reveal discrepancies between intended and actual usage, uncovering four types of usability issues: "unexpected action sequence", "unexpected gesture", "repeated gesture", and "exceeded elapsed time". Similarly, Jeong et al. [56] presented a graph-based approach to identify dissimilarities in user behaviour, assuming that when users face usability issues, they show different behavioural patterns. Evaluating the approach on issues identified from usability testings of two Android apps, the authors found a correlation between areas of high behavioural dissimilarity and identified usability issues. Unlike UX-LLM, usage data analytics often does not require access to the source code, but actual users executing the app. This provides a strong evidence about the user issues but prevents the applicability to earlier development stages, e.g. before releases. Moreover, tracking user activity could raise privacy concerns and only some user segment might participate [54].

3) User Feedback Analysis: Research has shown that, for many developers, App Store reviews represent a primary source of user feedback [57], [58]. These reviews enable gathering issues (including on usability) and help developers improve their apps, enabling user-driven quality assessment [47]. Users also report issues in other feedback channels too as social media [34] and product forums [59]. Thus, monitoring and processing feedback looking for usability related reports is an active research field that has received a strong boost with the availability of recent LLMs [60], [61]. While certainly an important complementary source for monitoring usability issues, actual user priorities, and novel ideas, user feedback analytics also requires releasing the app and having a significant user base—whereas UX-LLM can be used during the development as it simulates common (user) knowledge coded in foundation models. UX-LLM is also beneficial for less popular apps, which do not have enough users for a meaningful analysis [58]. Moreover, feedback analytics requires manual tuning and interpretation to extract more valuable insights [60], [62], [63]. Feedback can vary greatly in quality [60] with up to 75% of app store reviews consist of praise rather than constructive feedback [34], [58].

4) Generative AI for GUI Engineering: Recent advance in AI has also benefited usability research and UI design in general. Beltramelli [64] presented "pix2code", which generates code from a GUI image input. This approach uses Convolutional Neural Networks to identify distinct elements, positions, and poses in GUI images. It then employs Recurrent Neural Networks and a decoder to generate executable code. The rapid raise of Generative AI brought significant improvements in how models solve the "Design2Code" task. Si et al. [65]. conducted a benchmarking with state-of-the-art models and showed that GPT-4 Turbo with Vision performs best on this task, compared to other models like Gemini Pro. Their study annotators claimed that GPT-4 generated websites which can replace the original reference in terms of visual appearance and content in 49% of cases. In 64% of cases, the annotators stated that GPT-4 even generated better websites than the original reference.

Other recent approaches has suggested to generate GUI images. Lee et al. [66] proposed a hybrid neural network to generate design layouts that meet user-specified constraints (e.g., a design with one headline and two images). In their experiments, they demonstrated that the generated layouts are visually similar to real design layouts. Wei et al. [67] took a step further to generate the designs using UI-Diffuser and LayoutDM [68], a Transformer-based model for layouts generation. Users provide a simple prompt of what the GUI should represent, e.g., a music player. Their preliminary results indicate that this could be a cost-effective approach for creating mobile GUI designs. However, the GUIs are far from being directly directly reusable, need improvements, and usability evaluation. Uizard[12] is a similar recent commercial product, which generates mobile GUI designs with a simple prompt. Mattisson et al. [69] demonstrated that Uizard can significantly enhance designers efficiency, boost creativity in idea generation,

[10]https://firebase.google.com

[11]https://telemetrydeck.com

[12]https://uizard.io

and improve communication with stakeholders. Particularly the editable designs with nested components allow for continuous customisation and adjustments, e.g. after usability evaluation. In another recent study, Wei et al. [70] tuned a Vision-Language Model to retrieve relevant GUI designs from a large constructed dataset. This approach seems to be more effective yielding more relevant and higher quality screens—highlighting the importance of the human-in-the-loop for GUI recommendation.

The works are are perhaps closest to ours are of Kuang et al. [71] and of Kocaballi [72]. To explore how UX evaluators use AI assistant Kuang et al. [71] conducted a study with 20 participants using a simulated AI assistants. They found that participants asked the bot for five categories of information such as user actions and mental model and observed other trends. The authors finally derive design considerations for future conversational AI assistants for UX evaluation. In hypothetical design project, Kocaballi used ChatGPT to generate personas, simulate interviews with fictional users, and create new design ideas. The work highlights drawbacks such as forgotten information, partial responses, and a lack of output diversity, suggesting the importance of human oversight in the design process. To the best of our knowledge, this work is the first to use Generative AI for predicting actual usability issues in apps while comparing the performance with conventional usability testing and evaluation by humans.

VI. Threats to Validity and Limitations

1) Internal and External Validity: For the usability testing, we aimed to recruit diverse participants, with ages ranging from 24 to 60 years and various occupations. Although this represents a broad spectrum, some participants have a technical affinity, which could potentially limit the range of usability issues identified. Furthermore, only three participants were female and three had only limited iOS experience, which could potentially obscure the results, as unfamiliarity with platform features might be mistakenly perceived as usability issues [73]. In addition, no participant had impairments, potentially missing some issues. Replicating exact real-world conditions is a perpetual challenge. Although usability tests (also ours) are designed to match real-world conditions (than laboratory tests), they are never entirely the same [74]. Some usability issues might have been overlooked due to artificial usage environment or participants motivation to use the apps with pre-defined tasks.

Another potential threat to internal validity is that only two UX professionals were recruited for the expert review, Although the evaluations were thoroughly conducted as the result details show, involving additional experts might lead to more diverse perspectives on UX-LLM and its results.

The selection of two reference apps restricts a broader reliability of our findings. The chosen Quiz and To-Do app are simpler compared to the wide range of more complex available apps. As we primarily aimed to support smaller app teams, it made more sense to focus the evaluation on simpler apps. Evaluating UX-LLM for niche domains like “Ireland - COVID Tracker” would rather limit the results. The two apps used in our study provided the best compromise between:

- Broadly used and maintained app.
- Clean SwiftUI Model-View-ViewModel Architecture to inject into UX-LLM.
- General apps, not too niche domain (also to enable the usability testing).

. Nevertheless, to increase the generalisability, it is important to replicate our study with apps from various categories, embodying varying levels of complexity and incorporating different types of user interactions [49], [75]. The amount and quality of code entered into UX-LLM can also have a major impact on its performance.

We chose expert reviews and usability testings as comparative usability evaluation methods. While these methods are well-established [76], for a comprehensive analysis, UX-LLM should also be compared with additional usability evaluation methods such as questionnaires or feedback, usage data [49], and code analysis techniques discussed in Section V.

In addition, the focus group involved a small student development team working on a capstone university project app. The insights gained do not fully represent the experiences of more diverse professional development teams working on a commercial app, affecting the perceived utility and integration potential of UX-LLM in various development contexts. For more in-depth results, it would be beneficial to have developers use the tool over time to gather more evaluation experiences.

2) Construct Validity: Unlike the usability testing and the expert assessment which followed strict protocols, during the expert reviews we refrained from instructing the experts to follow a certain process. A more structured review would likely increase the replicability and agreement rate. In fact, we considered using the more structured approach called “Heuristic Expert Reviews” where evaluators are aided by a checklist. However, we decided for the general review form: mainly to not interfere or direct the experts to a certain direction [77]. Moreover, heuristic evaluations tend to miss context-specific issues [77].

During the expert assessment, the survey had four options to categorise UX-LLM’s usability issues: “Usability Issue”, “No Usability Issue”, “Uncertain”, and “Incorrect/Irrelevant Statement”. The appropriateness of these categories could also be questioned, considering that they might not capture all possible reactions. To mitigate this potential threat, we took detailed notes of all expert comments.

Furthermore, during performance evaluation, the recall was calculated by defining false negatives as all usability issues not identified by UX-LLM but found during usability testings and expert reviews. There remains ambiguity about whether all usability issues have been uncovered, with a strong likelihood that some issues have gone unnoticed. This was highlighted by Molich et al. [9], who demonstrated that even simple websites can contain more than a hundred usability issues. Similar to Park et al. multiple studies skipped the recall calculations entirely, stating that, “it is nearly impossible to define a totally complete set of [usability issues].” [55] Accordingly, we combined issues from both expert evaluation and the testings

suggest using precision as main metric to demonstrate the method accuracy and recall rather as indicative.

To compare and match the overlap of usability issues found by the three methods, we constructed Table III. Although we took special care and applied a strict guideline that only the same or sub/super issues are matched, other UX experts might come to slightly different matching [9], [78], which we encourage assessing by share our evaluation data. Only usability issues between the methods were matched. Matching issues within each method could also lead to additional insights.

Finally, for the studies on UX-LLM only OpenAI's GPT-4 Turbo with Vision was used. Using different LLMs and different prompts could significantly change the performance. Moreover, during all studies, we ignored the severity of issues, Comparing the issue severity would likely lead to additional insights but requires additional studies, due to its subjective variability [78]. We also did not examine the determinism of UX-LLM results. Generating different outputs from the same input could impact performance and usefulness [16].

VII. Discussion and Conclusion

Our results show that—with fairly low effort and widely-available models—GenAI can predict in source code valid usability issues, that can easily be reviewed and fixed before releasing the app, avoiding to dissatisfy users and compromise their experience. This saves time and resources particularly for smaller app teams and freelancers, who tend to ignore usability evaluation [40]. However, our study also shows that relevant issues particularly identified by usability experts were missed by UX-LLM. This suggests that traditional usability evaluation can certainly be boosted with GenAI but not completely replaced, still requiring a **UX expert in the loop** [70].

Usability evaluation is a rather continuous, iterative, subjective process [8], [79]. It is thus unlikely that one method or one person can identify "all" usability issues [9]. This should be kept in mind when interpreting the rather limited recall values of UX-LLM, calculated conservatively. Relaxing the interpretation, e.g., including issues rated "undecided" or confirmed by one expert would lead to higher performance values. Our qualitative analysis, e.g., from the focus group, also shows that "debatable" issues or "rather feature ideas" can also be insightful for developers to re-evaluate their designs.

We think that **method triangulation** or hybrid approaches are particularly appealing to gather various perspectives and increase evaluation effectiveness and efficiency. UX-LLM seems to outperform when access to certain users is difficult (e.g. impaired users) or rare scenarios and tasks are not specified (Section IV-B). Experts are particularly able to reason about the broader pictures, e.g., issues across multiple views or inconsistent terminology, which is a limitation of UX-LLM. Automated analysis of user feedback, usage data, and code discussed in Section V represent additional complementary techniques. However, a central question remains unanswered: how and when to effectively combine the various usability evaluation techniques. Our comparative study and the focus group contributed only preliminary insights. This need to be re-evaluated at a broader scale and including multiple apps, domains, programming frameworks, and numerous issues.

At the time of our research, there was **no public datasets** available with source code and actual validated usability issues from various evaluation methods. This was a major rationale to focus our work on an in-depth study creating such data for 3 apps, rather than a benchmarking different Foundation Models and prompt engineering techniques. We hope that by replicating our study and augmenting our dataset with additional apps and issues, studies on model-tuning and advance prompt engineering (including, e.g. more / custom issue examples or techniques as Chain of Thoughts) will be become possible. This will likely lead to more precise GenAI-assisted usability evaluation tools and will enable meta-studies of hybrid approaches. We think that the research community should focus on this in near future. Finally, integrating usability issue prediction in the IDE (as a "usability debugger") as well as suggesting potential fixes are promising directions for a higher acceptance of the tool in daily development work.

Acknowledgement

This work was partly supported by MaibornWolff GmbH. We thank all participants in the usability testing, the UX experts, as well as the participants of the focus group for their time and feedback.

References

[1] J. Huang, Y. Shen, L. Zhang, and Y. Zhuang, "Research on trend, similarity and difference of chinese and oversea' s smartphone brand——take apple and xiaomi as an example," *BCP Business & Management*, 2022.

[2] R. P. Da Costa, E. D. Canedo, R. T. De Sousa, R. D. O. Albuquerque, and L. J. G. Villalba, "Set of usability heuristics for quality assessment of mobile applications on smartphones," *IEEE Access*, vol. 7, pp. 116 145–116 161, 2019.

[3] A. de Lima Salgado and A. P. Freire, "Heuristic evaluation of mobile usability: A mapping study," in *Human-Computer Interaction. Applications and Services: 16th International Conference, HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part III 16*. Springer, 2014, pp. 178–188.

[4] Y. G. Ji, J. H. Park, C. Lee, and M. H. Yun, "A usability checklist for the usability evaluation of mobile phone user interface," *International journal of human-computer interaction*, vol. 20, no. 3, pp. 207–231, 2006.

[5] M. S. Bashir and A. Farooq, "Euhsa: Extending usability heuristics for smartphone application," *IEEE Access*, vol. 7, pp. 100 838–100 859, 2019.

[6] J. Nielsen, "Usability inspection methods," in *Conference companion on Human factors in computing systems*, 1994, pp. 413–414.

[7] D. Lavery, G. Cockton, and M. P. Atkinson, "Comparison of evaluation methods using structured usability problem reports," *Behaviour & Information Technology*, vol. 16, no. 4-5, pp. 246–266, 1997.

[8] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 1994.

[9] R. Molich and J. S. Dumas, "Comparative usability evaluation (cue-4)," *Behaviour & Information Technology*, vol. 27, no. 3, pp. 263–281, 2008.

[10] P. Weichbroth, "Usability of mobile applications: a systematic literature study," *Ieee Access*, vol. 8, pp. 55 563–55 577, 2020.

[11] N. Bornoe and J. Stage, "Supporting usability engineering in small software development organizations," in *Proceedings of the The 36th Information Systems Research Conference in Scandinavia (IRIS 36)*, 2013, pp. 1–12.

[12] N. AlDahoul, J. Hong, M. Varvello, and Y. Zaki, "Exploring the potential of generative ai for the world wide web," *arXiv preprint arXiv:2310.17370*, 2023.

[13] W. H. Pinaya, M. S. Graham, E. Kerfoot, P.-D. Tudosiu, J. Dafflon, V. Fernandez, P. Sanchez, J. Wolleb, P. F. da Costa, A. Patel *et al.*, "Generative ai for medical imaging: extending the monai framework," *arXiv preprint arXiv:2307.15208*, 2023.

[14] M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, "Advances in neural information processing systems 34," 2021.

[15] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4. arxiv," *arXiv preprint arXiv:2303.12712*, 2023.

[16] H. Tian, W. Lu, T. O. Li, X. Tang, S.-C. Cheung, J. Klein, and T. F. Bissyandé, "Is chatgpt the ultimate programming assistant–how far is it?" *arXiv preprint arXiv:2304.11938*, 2023.

[17] D. Sobania, M. Briesch, C. Hanna, and J. Petke, "An analysis of the automatic bug fixing performance of chatgpt," in *2023 IEEE/ACM International Workshop on Automated Program Repair (APR)*. IEEE, 2023, pp. 23–30.

[18] M. A. Haque and S. Li, "The potential use of chatgpt for debugging and bug fixing," 2023.

[19] Anonymised, "(Replication Package) Does GenAI make usability testing obsolete?" Tech. Rep., 2024. [Online]. Available: https://figshare.com/s/154fd5d2fe948a1ab3de

[20] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

[21] X. Ye and G. Durrett, "The unreliability of explanations in few-shot prompting for textual reasoning," *Advances in neural information processing systems*, vol. 35, pp. 30 378–30 392, 2022.

[22] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[23] V. Sarcar, "Bridge pattern," in *Java Design Patterns: A Hands-On Experience with Real-World Examples*. Springer, 2022, pp. 301–325.

[24] I. D. Melamed, R. Green, and J. Turian, "Precision and recall of machine translation," in *Companion volume of the proceedings of HLT-NAACL 2003-short papers*, 2003, pp. 61–63.

[25] H. Söbke, "Space for seriousness? player behavior and motivation in quiz apps," in *Entertainment Computing-ICEC 2015: 14th International Conference, ICEC 2015, Trondheim, Norway, September 29-Ocotober 2, 2015, Proceedings 14*. Springer, 2015, pp. 482–489.

[26] Apple, "Use reminders on your iphone, ipad, or ipod touch," Apple Support, 2024, accessed: 2024-04-19. [Online]. Available: https://support.apple.com/en-us/HT205890

[27] J. Nielsen, "Why you only need to test with 5 users," Nielsen Norman Group, Tech. Rep., 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[28] L. Faulkner, "Beyond the five-user assumption: Benefits of increased sample sizes in usability testing," *Behavior Research Methods, Instruments, & Computers*, vol. 35, pp. 379–383, 2003.

[29] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, pp. 22–23, 2002.

[30] S. Sun, "Meta-analysis of cohen's kappa," *Health Services and Outcomes Research Methodology*, vol. 11, pp. 145–163, 2011.

[31] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[32] D. G. Altman, *Practical statistics for medical research*. Chapman and Hall/CRC, 1990.

[33] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?" in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 308–318.

[34] D. Martens and W. Maalej, "Extracting and analyzing context information in user-support conversations on twitter," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 131–141.

[35] E. Frøkjær, M. Hertzum, and K. Hornbæk, "Measuring usability: are effectiveness, efficiency, and satisfaction really correlated?" in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 2000, pp. 345–352.

[36] "Ergonomic requirements for office work with visual display terminals (vdts)—part 11: Guidance on usability," International Organization for Standardization (ISO), 1998, accessed: 11.04.2024. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-1:v1:en

[37] "Ergonomics of human-system interaction—part 11: Usability: Definitions and concepts," International Organization for Standardization (ISO), 2018, accessed: 11.04.2024. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en

[38] F. Arnesson and J. Bengtsson, "Usability evaluation of a production system development framework," *School of Engineering*, 2012.

[39] R. Harrison, D. Flood, and D. Duce, "Usability of mobile applications: literature review and rationale for a new usability model," *Journal of Interaction Science*, vol. 1, pp. 1–16, 2013.

[40] N. Mathur, S. A. Karre, and Y. R. Reddy, "Usability evaluation framework for mobile apps using code analysis," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, 2018, pp. 187–192.

[41] H. Y. Abuaddous, A. M. Saleh, O. Enaizan, F. Ghabban, and A. B. Al-Badareen, "Automated user experience (ux) testing for mobile application: Strengths and limitations." *International Journal of Interactive Mobile Technologies*, vol. 16, no. 4, 2022.

[42] R. Coppola, L. Ardito, M. Torchiano, and E. Alégroth, "Translation from layout-based to visual android test scripts: An empirical evaluation," *Journal of Systems and Software*, vol. 171, p. 110845, 2021.

[43] M. Bajammal and A. Mesbah, "Semantic web accessibility testing via hierarchical visual analysis," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 1610–1621.

[44] A. Alshayban and S. Malek, "Accessitext: automated detection of text accessibility issues in android apps," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 984–995. [Online]. Available: https://doi.org/10.1145/3540250.3549118

[45] H. N. da Silva, A. T. Endo, M. M. Eler, S. R. Vergilio, and V. H. S. Durelli, "On the relation between code elements and accessibility issues in android apps," in *Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing*, ser. SAST '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 40–49. [Online]. Available: https://doi.org/10.1145/3425174.3425209

[46] A. Rastogi and G. Gousios, "How does software change?" *CoRR*, vol. abs/2106.01885, 2021. [Online]. Available: https://arxiv.org/abs/2106.01885

[47] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE Software*, vol. 33, no. 1, pp. 48–54, 2016.

[48] M. Gómez, B. Adams, W. Maalej, M. Monperrus, and R. Rouvoy, "App store 2.0: From crowdsourced information to actionable feedback in mobile ecosystems," *IEEE Software*, vol. 34, no. 2, pp. 81–89, 2017.

[49] T. Roehm, N. Gurbanova, B. Bruegge, C. Joubert, and W. Maalej, "Monitoring user interactions for supporting failure reproduction," in *2013 21st International Conference on Program Comprehension (ICPC)*, 2013, pp. 73–82.

[50] C. Stanik, M. Haering, C. Jesdabodi, and W. Maalej, "Which app features are being used? learning app feature usages from interaction data," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020, pp. 66–77.

[51] J. Harty, H. Zhang, L. Wei, L. Pascarella, M. Aniche, and W. Shang, "Logging practices with mobile analytics: An empirical study on firebase," in *2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft)*. IEEE, 2021, pp. 56–60.

[52] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *2013 10th working conference on mining software repositories (MSR)*. IEEE, 2013, pp. 41–44.

[53] Y. Tang, X. Zhan, H. Zhou, X. Luo, Z. Xu, Y. Zhou, and Q. Yan, "Demystifying application performance management libraries for android," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 682–685.

[54] Y. S. Van Der Sype and W. Maalej, "On lawful disclosure of personal user data: What should app developers do?" in *2014 IEEE 7th International Workshop on Requirements Engineering and Law (RELAW)*, 2014, pp. 25–34.

[55] S. Park, S. Park, and K. Ma, "An automatic user activity analysis method for discovering latent requirements: usability issue detection on mobile applications," *Sensors*, vol. 18, no. 9, p. 2963, 2018.

[56] J. Jeong, N. Kim, and H. P. In, "Detecting usability problems in mobile applications on the basis of dissimilarity in user behavior," *International Journal of Human-Computer Studies*, vol. 139, p. 102364, 2020.

[57] S. Hassan, C. Tantithamthavorn, C.-P. Bezemer, and A. E. Hassan, "Studying the dialogue between users and developers of free apps in the google play store," *Empirical Software Engineering*, vol. 23, pp. 1275–1312, 2018.

[58] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE international requirements engineering conference (RE)*. IEEE, 2013, pp. 125–134.

[59] J. Tizard, H. Wang, L. Yohannes, and K. Blincoe, "Can a conversation paint a picture? mining requirements in software forums," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 17–27.

[60] W. Maalej, V. Biryuk, J. Wei, and F. Panse, "On the automated processing of user feedback," in *Handbook of Natural Language Processing for Requirements Engineering*. Springer, 2025.

[61] J. Wei, A.-L. Courbis, T. Lambolais, B. Xu, P. L. Bernard, and G. Dray, "Zero-shot bilingual app reviews mining with large language models," in *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2023, pp. 898–904.

[62] C. Zhi, J. Yin, S. Deng, M. Ye, M. Fu, and T. Xie, "An exploratory study of logging configuration practice in java," in *2019 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2019, pp. 459–469.

[63] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?" *Information and Software Technology*, vol. 76, pp. 135–146, 2016.

[64] T. Beltramelli, "pix2code: Generating code from a graphical user interface screenshot," in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2018, pp. 1–6.

[65] C. Si, Y. Zhang, Z. Yang, R. Liu, and D. Yang, "Design2code: How far are we from automating front-end engineering?" *arXiv preprint arXiv:2403.03163*, 2024.

[66] H.-Y. Lee, L. Jiang, I. Essa, P. B. Le, H. Gong, M.-H. Yang, and W. Yang, "Neural design network: Graphic layout generation with constraints," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 491–506.

[67] J. Wei, A.-L. Courbis, T. Lambolais, B. Xu, P. L. Bernard, and G. Dray, "Boosting gui prototyping with diffusion models," in *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, 2023, pp. 275–280.

[68] S. Chai, L. Zhuang, and F. Yan, "Layoutdm: Transformer-based diffusion model for layout generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 18 349–18 358.

[69] O. Mattisson and O. Hamik, "Vad kan ai-baserade verktyg innebära för ux-designers?: En explorativ användarstudie om ux-designers upplevelser med det generativa ai-prototypverktyget uizard," 2022.

[70] J. Wei, A.-L. Courbis, T. Lambolais, B. Xu, P. L. Bernard, G. Dray, and W. Maalej, "Guing: A mobile gui search engine using a vision-language model," *ACM Trans. Softw. Eng. Methodol.*, Nov. 2024, just Accepted. [Online]. Available: https://doi.org/10.1145/3702993

[71] E. Kuang, M. Li, M. Fan, and K. Shinohara, "Enhancing ux evaluation through collaboration with conversational ai assistants: Effects of proactive dialogue and timing," in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, ser. CHI '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3613904.3642168

[72] A. B. Kocaballi, "Conversational ai-powered design: Chatgpt as designer, user, and product," *arXiv preprint arXiv:2302.07406*, 2023.

[73] N. K. C. Pong and S. Malacria, "Awareness, usage and discovery of swipe-revealed hidden widgets in ios," in *Proceedings of the 2019 ACM international conference on interactive surfaces and spaces*, 2019, pp. 193–204.

[74] T. Kallio, A. Kaikkonen *et al.*, "Usability testing of mobile applications: A comparison between laboratory and field testing," *Journal of Usability studies*, vol. 1, no. 4-16, pp. 23–28, 2005.

[75] W. Maalej, T. Fritz, and R. Robbes, *Collecting and Processing Interaction Data for Recommendation Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 173–197. [Online]. Available: https://doi.org/10.1007/978-3-642-45135-5_7

[76] F. Paz and J. A. Pow-Sang, "A systematic mapping review of usability evaluation methods for software development process," *International Journal of Software Engineering and Its Applications*, vol. 10, no. 1, pp. 165–178, 2016.

[77] A. Harley, "Ux expert reviews," Nielsen Norman Group, Tech. Rep., 2018. [Online]. Available: https://www.nngroup.com/articles/ux-expert-reviews/

[78] N. E. Jacobsen, M. Hertzum, and B. E. John, "The evaluator effect in usability studies: Problem detection and severity judgments," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 42, no. 19. SAGE Publications Sage CA: Los Angeles, CA, 1998, pp. 1336–1340.

[79] J. Nielsen, "Usability 101: Introduction to usability," Nielsen Norman Group, Tech. Rep., 2012. [Online]. Available: https://www.nngroup.com/articles/usability-101-introduction-to-usability/