How Should I Improve the UI of My App?: A Study of User Reviews of Popular Apps in the Google Play

QIUYUAN CHEN, College of Computer Science and Technology, Zhejiang University, China
CHUNYANG CHEN, Faculty of Information Technology, Monash University, Victoria, Australia
SAFWAT HASSAN, Department of Engineering, Thompson Rivers University, Canada
ZHENGCHANG XING, College of Engineering & Computer Science, National University, Australian
XIN XIA, Faculty of Information Technology, Monash University, Victoria, Australia
AHMED E. HASSAN, Software Analysis and Intelligence Lab (SAIL), Queen's University, Canada

UI (User Interface) is an essential factor influencing users' perception of an app. However, it is hard for even professional designers to determine if the UI is good or not for end-users. Users' feedback (e.g., user reviews in the Google Play) provides a way for app owners to understand how the users perceive the UI. In this article, we conduct an in-depth empirical study to analyze the UI issues of mobile apps. In particular, we analyze more than 3M UI-related reviews from 22,199 top free-to-download apps and 9,380 top non-free apps in the Google Play Store. By comparing the rating of UI-related reviews and other reviews of an app, we observe that UI-related reviews have lower ratings than other reviews. By manually analyzing a random sample of 1,447 UI-related reviews with a 95% confidence level and a 5% interval, we identify 17 UI-related issues types that belong to four categories (i.e., *"Appearance," "Interaction," "Experience,"* and *"Others"*). In these issue types, we find *"Generic Review"* is the most occurring one. *"Comparative Review"* and *"Advertisement"* are the most negative two UI issue types. Faced with these UI issues, we explore the patterns of interaction between app owners and users. We identify eight patterns of how app owners dialogue with users about UI issues by the review-response mechanism. We find *"Apology or Appreciation"* and *"Information Request"* are the most two frequent patterns. We find updating UI timely according to feedback is essential to satisfy users. Besides, app owners could also fix UI issues without updating UI, especially for issue types belonging to *"Interaction"* category. Our findings show that there exists a positive impact if app owners could actively interact with users to improve UI quality and boost users' satisfactoriness about the UIs.

CCS Concepts: • **Software and its engineering** → *Software creation and management*; *Designing software*;

Additional Key Words and Phrases: Mobile app reviews, the Google Play Store, user interface

This research was partially supported by the Australian Research Council's Discovery Early Career Researcher Award (DECRA) funding scheme (DE200100021) and ARC Discovery grant (DP200100020).
Authors' addresses: Q. Chen, College of Computer Science and Technology, Zhejiang University, 38 Zheda Rd, 310000, Hangzhou, Zhejiang, China; email: chenqiuyuan@zju.edu.cn; C. Chen and X. Xia (corresponding author), Faculty of Information Technology, Monash University, Wellington Rd, Melbourne, Victoria, Australia; email: {chunyang.chen, xin.xia}@monash.edu; S. Hassan, Department of Engineering, Thompson Rivers University, Kamloops, British Columbia, Canada; email: shassan@tru.ca; Z. Xing, College of Engineering & Computer Science, National University, Canberra, Australian; email: ahmed@cs.queensu.ca; A. E. Hassan, Software Analysis and Intelligence Lab (SAIL), Queen's University, Kingston, Ontario, Canada; email: ahmed@cs.queensu.ca.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
1049-331X/2021/04-ART37 $15.00
https://doi.org/10.1145/3447808

ACM Reference format:
Qiuyuan Chen, Chunyang Chen, Safwat Hassan, Zhengchang Xing, Xin Xia, and Ahmed E. Hassan. 2021. How Should I Improve the UI of My App?: A Study of User Reviews of Popular Apps in the Google Play. *ACM Trans. Softw. Eng. Methodol.* 30, 3, Article 37 (April 2021), 38 pages.
https://doi.org/10.1145/3447808

1 INTRODUCTION

User Interface (UI) provides a visual bridge between mobile applications (apps) and end-users through which they can interact with each other. A good UI design makes an app easy, practical, and efficient to use, which significantly affects the success of the app and the loyalty of its users [30, 40, 43]. However, app updates with UI issues can lead to a high rate of user complaints [30] and may result in deploying emergency fixes to the raised UI issues [31]. Different from other functionalities that can be easily validated by inputting test cases [83], it is hard for designers to guarantee the success of UI design, which is often influenced by subjective factors of users. App stores (such as the Google Play Store[1] and the Apple App Store[2]) provide a feedback mechanism that enables users to post their feedback about the app in the reviews. For example, users can express their experience with the UI design of an app. Hence, app owners can leverage reviews of their apps to understand how to improve the UI of their apps.

However, there are two challenges for app owners to improve the quality of the UI when utilizing users' reviews. On the one hand, the difficulty lies in *how to understand user reviews concerning UI issues.* As users are often not professional designers, they may not explicitly describe how the UI is unsatisfactory (e.g., *"the UI is ugly"* or *"the color contrast should be consistent"*). Hence, it would be beneficial to summarize user complaints about UI in the reviews, which could give app owners a better guide to solve UI issues. On the other hand, the difficulty lies in how to encourage app users to provide more details about the raised issues and provide feedback about whether the raised issues are fixed. For example, when users raise UI issues in reviews, app owners often need more details for improvement; after app owners fix an issue, they often need further feedback to validate the results.

Faced with the mentioned challenges, we conduct an in-depth analysis by studying more than 3M UI-related reviews from 22,199 top free-to-download and 9,380 top non-free apps in the Google Play Store. Our initial analysis shows that the mean rating of apps with all reviews is 4.17, while the mean rating of apps with UI-related reviews gets a lower rating of 3.68, which reflects the severity of UI issues. Hence, we perform an in-depth analysis of the UI issues along with the following research questions (RQ)s:

RQ1: What do users complain about in UI-related reviews?

A category of UI issues can help app owners to have a better understanding of user perception towards UI. To this end, we manually analyze a random sample of 1,447 reviews out of the 3.3M UI-related reviews for the studied free apps and non-free apps. We identify four categories of UI issues: (1) *"Appearance"*—users comment on the main appearance and visual elements of the app; (2) *"Interaction"*—users comment on the inconvenience or failures of mutual actions with UI; (3) *"Experience"*—users comment on the ill-considered design that lower users' satisfactoriness of UI and (4) *"Others"*—users comment on the UI using generic words (e.g., *"The UI is ugly."*).

The four categories contain 17 UI-related issue types that users concern about in reviews. In these issue types, we find *"Generic Review"* is the most occurring one in which the reviews of

[1] https://play.google.com/store.
[2] https://www.apple.com/ios/app-store/.

free-to-download apps are more generic than non-free apps. We find issue type *"Comparative Review"* and issue type *"Advertisement"* contain the highest percentage of low rating (1- and 2-star) reviews (0.90 and 0.66, respectively). Besides, we find *"Redundancy"* is the most frequently raised UI issue type (15.9%) in the reviews of non-free apps. *"Feedback"* issue is the most occurring UI issue type in the reviews of game apps (17.4%), and users of non-game apps are highly impacted by the UI changes and UIs of competitor apps.

RQ2: How do app owners interact with users concerning UI issues?

When app owners respond to the posted reviews, they start a dialogue with users, and we find that the first of such dialogue occurs within a week on average. To study how the dialogues between app owners and users could help handle UI issues, we select UI-related reviews with responses of app owners and manually analyzed a random sample of 764 dialogues to explore the interactions. We identify eight patterns (e.g., *"Information Request"* and *"Justify the UI Issue"*) of how app owners dialogue with users about UI issues by the review-response mechanism. We find patterns of *"Apology or Appreciation"* and *"Information Request"* are the most frequent patterns (39.0% and 17.9%). Besides, we explore how app owners interact with users for different UI issues. We find app owners will reconcile with users on particular UI issues and the most frequent issue types belong to *"Interaction"* category.

RQ3: Will the interaction improve users' satisfactoriness?

We calculate the review ratings before and after the interaction. Our findings show that there exists a positive impact if app owners could actively interact with users to improve UI quality, and the essential purpose of dialogue is getting sufficient information about the UI issues. According to our analysis, such interactions between users and app owners indeed help the app get a better UI and boost users' satisfactoriness with the updated UI.

In summary, we make the following contributions in this article:

- Our study is the first to explain in detail how the textual complaints in user reviews are linked to visual UI issues and contribute a taxonomy of UI issues. Our study identifies four categories, including 17 UI issue types that are mentioned by users in reviews.
- We explore the patterns of interaction between app owners and users concerning UI issues. We find such interactions have a positive impact on the app to get better UI and boost users' satisfactoriness with the updated UI. Furthermore, we explore pattern distributions in each UI issue to understand how the interaction differs for different issues. We find app owners can satisfy users by promising and timely fulfilling the requested features in the following updates, and they can also fix issues without UI updates by proactive communication.

The rest of this article is organized as follows: Section 2 describes how we collect user reviews and the rating analysis of the dataset. Section 3 discusses the UI issues in reviews and the dialogue patterns between users and app owners concerning these UI issues. Section 4 discusses the implications of our study. Section 5 discusses the limitations and threats to the validity of our findings. Section 6 discusses the related work and describes the difference between our work and theirs that is presented in this article. Section 7 presents our conclusion and future work.

2 SETUP

2.1 Research Questions

In this article, we aim to understand UI issues perceived by end-users of apps and how app owners solve these issues. Specifically, we seek to understand UI issue categories and the interaction of the users and the app owners concerning how to solve these issues. To achieve these goals, we study the following three research questions:

RQ1. What do users complain about in the UI-related reviews? We perform a qualitative analysis to explore users' complaints about the UI and summarize a taxonomy of the underlying issues. There are two sub-questions in RQ1:

RQ1.1 What is the taxonomy of UI issues? We sample UI-related reviews in free and non-free apps and perform an open coding procedure to get a taxonomy of the UI issues in the user reviews.

RQ1.2 What are the characteristics of UI issues? For different UI issues, we calculate the *"Low Rating Ratio"* (1- and 2-star reviews) as the severity of the UI issue type and analyze the different characteristics among the UI issues.

The answer to RQ1 helps to gain a better understanding of the situation of UI issues from the perspective of the app users.

RQ2. How do app owners interact with users concerning UI issues? We explore how app owners interact with users regarding the identified UI issues. There are three sub-questions in RQ2:

RQ2.1 How many and how frequent are the interactions between the users and the app owners concerning UI issues? We calculate the frequency and the numbers of the interactions and report the overall statistic. In this way, we can obtain a quantitative overview of the interactions.

RQ2.2 What are the patterns of the interactions between the users and the app owners? We perform a qualitative analysis and summarize the patterns of the interactions regarding the UI issues.

RQ2.3 What are the distributions of response patterns on each UI issue? To explore how app owners interact and collaborate with users, we further combine dialogue patterns with UI issues and study how the patterns correspond to different UI issues.

The answer to RQ2 helps to understand better the interactions between the users and the app owners and how these interactions are related to different UI issues.

RQ3. Will the interaction improve users' satisfactoriness? Since app owners have different patterns to interact with users, we go one step further to investigate whether and how these interactions can impact the review ratings.

RQ3.1 What are the rating changes before and after the interactions between the users and the app owners? We calculate the changes in the rating before and after the dialogues to investigate the interactions' effect.

RQ3.2 How do the app owners interact with the users to improve the UI quality? We perform a qualitative study to investigate the positive interactions that satisfy users and get the rating increases concerning improving UI quality.

The answer to RQ3 helps to gain a better understanding of the effectiveness of the interaction. It also gives app owners the implications of improving UI quality via interaction with end-users.

2.2 Research Approaches

2.2.1 Approach for RQ1. UI Issues are defined as the complaints about the UI of an app in user reviews. Note that high-rank UI-related reviews are also considered, because it can give a positive guide for app owners. For example, a user would describe how the new UI is improved to be user-friendly in a high-rank review (e.g., *"I'm giving it 5/5 (stars) again because recent changes restored old filter icons (opposed to previews which were terrible to work with). Additionally, fine tuning option has been finally introduced to let us manually set the parameters for low-colour filters."*).

Boundary of UI issue. Many factors (e.g., back-end crash) may lead to issues that reflect on the screen. In this article, we consider the explicitly mentioned UI elements [26]. Besides, we also consider complaints about a particular functionality design (i.e., a feature), which is also considered as UI elements in previous work [67].

We manually analyze a random sample in the dataset with a 95% confidence level and a 5% confidence interval following the previous work [30, 32]. We calculate the size using the Sample Size Calculator,[3] which calculates how many reviews we need to investigate to get results that reflect the target number of reviews as precisely as needed with a particular confidence level and confidence interval. Our qualitative approach is based on the open coding method, a procedure of identifying underlying issues with individual labeling and discussion involving two coders (the first and the second author of this article) [84]. We coupled our methodology with deductive thematic analysis [10, 81, 92]. The steps to categorize UI issues are as follows:

Step 1: Identify UI complaints. Two authors (coders) manually read a random sample of 384 reviews (with 95% confidence level 5% confidence interval) selected from the UI-related reviews. We employed a process called *theoretical sampling* [17, 55], which can ensure that the samples capture as many aspects as possible. We analyze each sentence in every selected review and transform each sentence into *what* users complain about in this review. The use of formatting and transforming sentences to model knowledge and information has been successfully used by prior researchers to capture and reason about types of information independently of irrelevant context [22, 46, 55]. We reconciled and reworded the complaints to achieve consistency in style and decide issue type names. At the end of this step, we identified 17 issue types (e.g., *"Accessibility"* and *"Advertisement"*). Table 3 shows the list of the identified issue types.

Step 2: Label user reviews using the identified issue types. We examine samples of 1,447 reviews (with 99% confidence level and 5% confidence interval) of free and non-free apps (792 and 655). There are 22 (1.50%) false positives and they are not counted. At last, we label user reviews using the identified issue types. Then, we measure the frequency and the rating of the reviews that belong to every issue type. The process is labor-intensive. On average, one person could label about 40 reviews per hour (not counting the time for discussion). After completing the labeling process, the two authors discuss their disagreements to reach a common decision. During the discussion, each author gives a rationale for every disagreement. Two authors adopt reasonable results for most of the disagreements. For the few remaining disagreements, we negotiate a standard or invite a third person who has UI design experience into the discussion to reach a final agreement. For example, visual issues (e.g., *"color"*) impact not only the normal but also the disabled, so for these disagreements, we negotiate a standard to identify *"accessibility"* issues only when the reviews explicitly mention they are with accessibility troubles. Such a discuss-and-decide process to reach a consensus is also adopted in previous work [89, 95]. Finally, we calculated the agreement between two authors using Fleiss Kappa [21]. Fleiss Kappa measures the agreement between the two authors and the value of [0.01, 0.20], (0.20, 0.40], (0.40, 0.60], (0.60, 0.80], (0.80, 1] is considered as slight, fair, moderate, substantial, and almost perfect agreement, respectively [21]. We calculated the Fleiss Kappa, and the result is 0.76, which indicates the agreement of the identified category is considered to be substantial.

2.2.2 Approach for RQ2. Prior study [32] showed that users and app owners leverage the user-developer dialogues as a user support mechanism. In this section, we conduct a qualitative study of the dialogue between a user and app owner to understand how app owners communicate with users to solve UI issues.

In the Google Play Store, if the app owners have dialogues with the users in a particular review, their replies to the review and users' further updates are all attached to the same review. Therefore, the dialogue consists of the original review, the replies of the app owner, and the updates of the

[3] https://www.surveysystem.com/sscalc.htm.

user. We refer to one **"Review-Response Iteration"** as one time the users are responded by app owners and update their reviews. We refer to **"Review Update Time"** as the duration until the users update the reviews after their reviews getting responses.

We examine a random sample of 764 reviews (with a 95% confidence level and a 5% confidence interval [30, 32]) of free-to-download apps and non-free apps reviews with replies of app owners (383 and 381, respectively). So, there are 764 dialogues (a dialogue consists of the original review, app owner replies, and user updates). There are 12 (1.55%) false positives in the sample, and they are not counted.

We adopt the thematic deductive approach described in Section 2.2.1 and open coding [84] approach. The coders (the first and the second author of this article) independently followed the approach to identify patterns of how app owners interact with users and handle UI issues. For every studied dialogue, the coders identify the dialogue pattern (e.g., requesting information from users). If the pattern is not in the identified patterns, then the coder extends the list and revisits all dialogues using the newly identified patterns. The open coding method terminates when there are no newer patterns identified, and all dialogues are studied. Then both coders compared and marked the difference of all patterns. At last, the two coders revisit all the dialogues and decide ultimate patterns. We calculated the Fleiss Kappa, and the result is 0.79, which indicates that the agreement of the identified category is considered substantial.

2.2.3 Approach for RQ3. We calculated the changes in the rating before and after the dialogues. Furthermore, we give a strict condition that only when users and app development teams update the reviews/responses at least once (instead of only updating response or review) could be regarded as an interaction. The dialogues are attached to particular reviews. Each time the users update the review (i.e., interact with app owners), they can also choose to update the rating of this review (keep, increase, or decrease the rating). We calculate the rating differences before (i.e., the first rating) and after (i.e., the last updated rating) the dialogue. In this way, we identified 4,169 interactions. The Wilcoxon signed-rank test [25] indicates a significant difference (p-value <0.05) in the review rating before and after the responses, with a small effect size.

To explore the reasons for rating increases, we manually examine a random sample of 94 (a confidence level of 95% and a confidence interval of 10%) of reviews with rating increases. There is no false positive in the sampled reviews that get rating increases.

2.3 Data Collection

In this section, as in Figure 1, we describe our approach for collecting user reviews (including responses if they exist) of apps in the Google Play Store and identifying the studied UI-related reviews. There are many other sources to derive UI-related issues (e.g., scanning UI screenshots or analyzing UI-related bugs), but they lack end-users' feedback towards UIs, which could be observed in user reviews. So, we mainly study the UI issues by analyzing a large-scale dataset of user reviews.

2.3.1 Collecting Reviews of the Top Free-to-download and Non-free Apps. We collected the top popular 500 apps in each app category in the Google Play Store using App Annie Report in 2018 [7]. In 2018, the Google Play Store had 33 categories and 17 sub-categories in the Games apps, and 9 sub-categories in the Family apps. As the sub-categories can reflect the complexity inside a category (e.g., UIs of Game apps), we also take these sub-categories into consideration, which can help follow the real-world situation. In total, we collected a list of 29,500 apps belonging to 59 categories and sub-categories (i.e., 33 app categories, 17 Game app sub-categories, and 9 Family app sub-categories). We noticed that 2,776 apps appear in multiple categories in App Annie Report. For example, the "Typing Games Master" app appears in both the "Word" Games subcategory

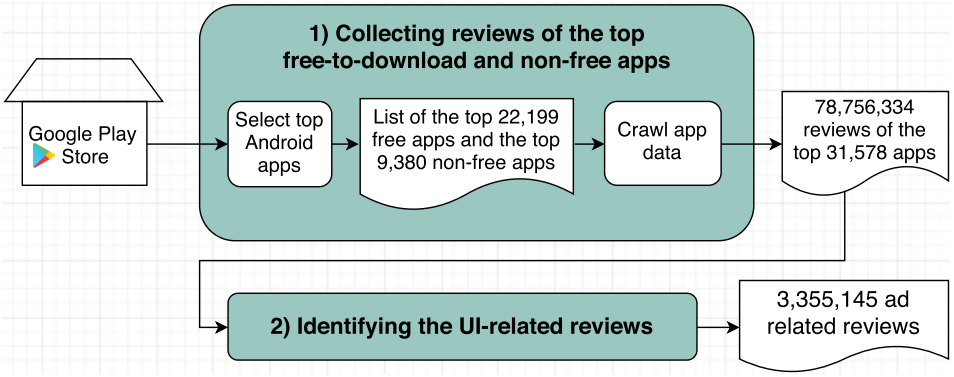


Fig. 1. An overview of our approach for collecting and identifying studied UI-related issues in the Google Play Store.

Table 1. Data Description of Studied Apps and Identified UI-related Reviews

	Free apps	Non-free apps	Total
Number of studied apps	22,199	9,380	31,578
Number of UI-related reviews	3,035,518	319,627	3,355,145

and the "Action & Adventure" Family sub-category. Hence, we obtained a list of 26,724 unique free apps. We could not collect the apps' general data (e.g., the release notes of updates) for 4,525 free apps. In the end, we could collect the reviews and the app details (e.g., the ratings) of 22,199 free apps. Similarly, for non-free apps, we collected the top 500 non-free apps belonging to 59 categories and sub-categories. We noticed that most of the sub-categories in the non-free apps have less than 500 apps. For example, the "Card" Games sub-category contains 124 non-free apps. In total, we identified 10,930 unique non-free apps. We could not collect the apps' general data for 1,550 non-free apps. Hence, we could collect the reviews and the app details of 9,380 non-free apps in total. In total, as in Table 1, we collected data of 22,199 top free-to-download apps and 9,380 top non-free apps in the Google Play Store. We studied popular apps, because popular apps have a large user-base, which enables us to analyze a rich dataset of UI-related reviews. In addition, there are more reviews in popular apps, so we can analyze how the app owners resolve the raised UI issues. Popular apps are used by a large user-base, which enables us to analyze a large number of user reviews and identify a wide taxonomy of UI-related issues that are raised by multiple users.

We use a Google Play Crawler [3] to crawl data from October 16th, 2018, to March 11th, 2019. For each studied app, we collected the following data:

- **App metadata:** app title, app description, number of downloads, app rating, and app owners.
- **App reviews:** review title, review text, review time, reviewer name, rating, the time it took an app owner to respond, and the text in the app owner response (also referred to as dialogues).

The date of collected reviews ranges from February 11th, 2009, until March 11th, 2019. In total, we collected 75,422,963 and 3,333,371 reviews for 22,199 free-to-download and 9,380 non-free apps, respectively.

Table 2. Mean, Standard Deviation and Five-number Summary of Rating of Studied Apps

	Mean	Std	Min	1st Qu.	Median	3rd Qu.	Max
App rating with all reviews	4.17	0.53	1.00	3.98	4.29	4.52	5.00
App rating with UI-related reviews	3.68	1.08	1.00	3.00	3.96	4.53	5.00

2.3.2 Identifying UI-Related Reviews. To identify reviews that are related to UI, we use keywords extraction combining with a manual check to examine the collected data. Previous work shows that word2vec, a neural network implementation for converting words to vector representations, can be effective for keywords extraction [56, 91]. Man et al. retrieve similar keywords to determine the keywords list [56]. Based on the previous work, we not only adopt word2vec similarity but also interactively retrieve keywords for multiple rounds. In summary, compared with the previous work, we make three improvements: First, we adopt n-grams (n equals two) instead of uni-gram; second, we perform similarity retrieval for every keyword in multiple rounds, which can help identify as many keywords as possible; third, we determine the keywords in an interactive way for every iteration, which can help identify most relevant keywords with human evaluation. See Appendix A for more details about how we performed keywords extraction and identify UI-related reviews.

Finally, we utilize the final refined list of UI-related keywords and the heuristic rules to identify the UI-related reviews (see Appendix B for the final keywords list and the heuristics descriptions). In our study, we identified 3,355,145 UI-related reviews, as shown in Table 1.

2.4 Rating Analysis of the Dataset

In this section, we quantitatively analyze the collected data from the perspective of the review rating. To analyze the UI-related reviews, we start with a hypothesis that the rating of an app with the only UI-related reviews (referred to as *UI rating*) reflects user satisfactoriness towards the UI of the app while the rating of an app with all reviews (referred to as *overall rating*) reflects the overall user satisfactoriness.

Significance Test. To quantify the differences in the distributions of the ratings, we use the Wilcoxon signed-rank test [25]. We calculated the p-value and Cliff's delta (d) effect size between UI ratings and overall rating. A p-value of fewer than 0.05 means that the difference between the distribution of the two samples is statistically significant. We use the following thresholds for interpreting d, as provided by Romano et al. [80]:

$$\text{Effect size} = \begin{cases} negligible(N), & \text{if } |d| \leqslant 0.147. \\ small(S), & \text{if } 0.147 < |d| \leqslant 0.33. \\ medium(M), & \text{if } 0.33 < |d| \leqslant 0.474. \\ large(L), & \text{if } 0.474 < |d| \leqslant 1. \end{cases}$$

The significant test results show that there is a significant difference (i.e., p-value <0.05) between UI ratings and overall rating, with a small effect size.

Rating Analysis. We calculated the mean, standard deviation, and five-number summary (i.e., minimum, the first quarter, median, the third quarter, and maximum number) of the overall ratings and the UI ratings to show the differences between them. The results are shown in Table 2.

Based on the hypothesis, a lower mean UI rating (3.68 versus 4.17) indicates that user satisfactoriness towards UI has a negative impact on the overall evaluation of apps in the Google Play Store. The app store is highly competitive [70], and the success of an app is closely tied to the reviews and ratings that it receives [30]. A 2015 survey shows that 69% of the users consider the app

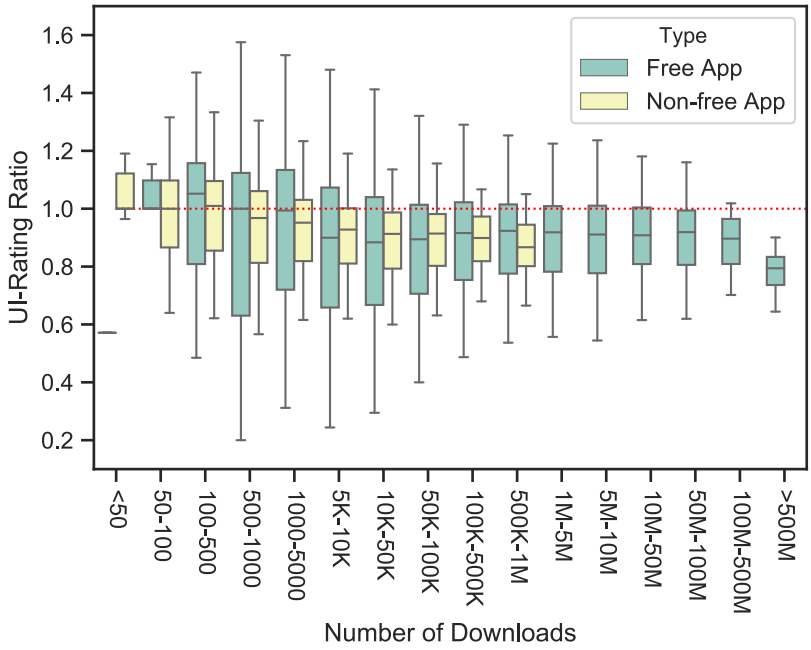


Fig. 2. Boxplot of the UI rating ratio in different ranges of the number of downloads of the free and the non-free apps. The horizontal dotted line means the ratio equal to one. The ratio above the line means that UI rating is better than app rating, and vice versa.

rating as an important or very important deciding factor when downloading an app [30]. In addition, 77% of the users will not download an app that has a rating that is lower than three stars [77]. Hence, lower ratings of UI-related reviews weaken the competitiveness of the apps. A higher standard deviation UI rating (1.08 versus 0.53) indicates that the user satisfactoriness towards UI in the Google Play Store has a relatively large fluctuation compared with the overall evaluation of apps.

Based on this observation (i.e., fluctuation of UI quality), we proceed further with a detailed analysis, comparing the distributions of different UI ratings in the different subpopulations of apps. In particular, we analyze the UI ratings from two perspectives:

(1) *Number of downloads:* number of downloads can represent the scale of the user-base. The more times an app is downloaded, the more users will interact with its UI and give feedback.
(2) *Price:* whether the app is free or not may influence user expectations.

We observe that from the perspective of the *number of downloads*, which represents the popularity, the standard deviation increases when the popularity decreases, which represents a more drastic fluctuation of user satisfactoriness towards UI.

Furthermore, Figure 2 shows the UI rating ratio of free and non-free apps. The UI rating ratio is calculated as the ratio of the UI rating of an app to the overall app rating, which is calculated as the mean rating of all reviews of the app. The horizontal dotted line means the ratio equal to one. The ratio above the line means that UI rating is better than app rating, and vice versa. For both free and non-free apps, the UI rating ratio follows a decreasing trend when the popularity decreases, which represents a decreasing UI quality. For free apps, the UI rating ratio is lower, which can be explained as: The complaints about UI in the reviews of free apps are more severe than in non-free

apps. While for non-free apps, the UI rating ratio follows a monotone decreasing trend when the popularity increases, which can be explained as: The UI rating is more likely to be influenced when the user-diversity and user-base increases.

Summary of Rating Analysis

The relatively low UI rating reflects the severity of UI issues concerning user feedback on apps, and the fluctuation of UI rating indicates the diversity of UI issues in app reviews. The UI rating ratios of both free-to-download and non-free apps follow a decreasing trend when the number of users increases.

Since the quantitative study of rating analysis can only reveal the macro characteristics of the data, we further study what the UI issues are complained about and how these issues could be fixed in the remainder of the article.

3 AN EMPIRICAL STUDY OF UI-RELATED REVIEWS AND INTERACTIONS

In this section, we present three research questions (RQs) and our observations.

3.1 RQ1 What Do Users Complain about in the UI-related Reviews?

3.1.1 RQ1.1 What is the Taxonomy of UI Issues? As in Table 3, we identified 17 UI issue types considering the granularity fitness and meaning of UI issues and categorized UI issues into four categories, namely, *"Appearance," "Interaction," "Experience,"* and *"Others,"* from the perspective of UI design. Then, we introduce each category and the corresponding UI issue types.

3.1.1.1 Appearance. The category focuses on the flaws of the visual effect of a UI. We categorized appearance issues into UI issue types of *"layout," "legibility and color," "typography and font," "iconography,"* and *"image."*

(A-1) Layout refers to the arrangement of visual elements on a UI. The layout issue indicates the layout is disordered or lacks consistency. For example, a user complains that the *"screen is split and looks to be compressed,"* which makes the content of apps unwatchable. Bad layouts lack consistency among different UIs. We observe users complain they are confused by a different layout when shifting to a new UI. Without consistency, users have to waste extra efforts to understand pages.

(A-2) Legibility and Color issue refers to low legibility or inconsistent color use of UI elements and components. The term legibility indicates how prominent the visual elements can be presented on the UI. First, over-design (e.g., *"I am confused by too many colors in the main page"*) and under-design (e.g., *"My status bar went full dark. Even those labels below the quick toggles are gone dark. I am not able to see anything"*) of colors will confuse users. Second, unexpected color use contradictory to common usage will influence the consistency of UIs (e.g., *"We get a persistently orange status bar that clashes with the color of virtually every card. Design inconsistencies like this are present everywhere in this app!"*).

(A-3) Typography and Font issue represents inappropriate typography and font of textual content. The term typography refers to the art and technique of arranging type to make written language readable and appealing when displayed [11]. Typography and font of text should express the hierarchy and brand presence of products [26]. The typography should be consistent with the UI; otherwise, some users will complain the font is not appropriate in the situation (e.g., *"the font is childish"*).

Table 3. Categories of UI Issue Types

(Index) Category	**(D)** Description - **(E)** Example
(A) Appearance	
(A-1) Layout	**D:** Disordered layout and inconsistent visual elements. **E:** *"It is a better experience than the old layout but still **lacking a list tab**."*
(A-2) Legibility and Color	**D:** Low legibility or inconsistent color of elements and components. **E:** *"Information on notification bar is not visible in apps **background and text are same in colour (white)**. Android 6.0// Redmi Note 4 // MIUI Global 8.5// Please fix. Thank you."*
(A-3) Typography and Font	**D:** Inappropriate typography and font of textual content. **E:** *"It's so eye bothersome with the **huge characters and sharp font**."*
(A-4) Iconography	**D:** Inaesthetics and meaningless product icon, system icon, or animated icon. **E:** *"**This new icon is so ugly** I literally puked when I saw it. It's not worth having the app if I have to look at this before I open it."*
(A-5) Image	**D:** Inappropriate use of the image, which includes illustration and photography. **E:** *"**The UI always shows the blurred images**. Please help."*
(I) Interaction	
(I-1) Navigation	**D:** UI design makes users feel difficult or even fails to move through the app. **E:** (1) *"It would be nice to **have a guide that explains every option**on UI."* (2) *"I find the UI clunky and very common buttons are **hidden instead of easily accessible**."*
(I-2) Notification	**D:** Absence or abuse of notification. **E:** *"It rings when I do not get notifications and when I do, it doesn't. I would like you to fix this because it is very **annoying** hearing that sound to open my phone to an empty homescreen."*
(I-3) Motion	**D:** Design of motions lacks fluency or meaning in which UI motion refers to the move, reshape, and transition of UI elements. **E:** *"I needs improvement. Vibrates incessantly and unnecessarily as you scroll through tickets for an event."*
(I-4) Gesture	**D:** Interaction issues with the screen when using touch gestures. **E:** *"Not as satisfying to me on account for the **imprecise controls of touch screens**. Please continue to further improve touch controls."*
(I-5) Accessibility	**D:** UI is not user-friendly to the disabled. **E:** *"Magnifer picture is jerkey and makes it diffucult to focus. **Not recommended for people with sight problems**."*
(E) Experience	
(E-1) Redundancy	**D:** Redundant design that makes UI bloated and less user-friendly. **E:** *"**Bundled with bloatware and features that are repeated**."*
(E-2) Customization	**D:** Inflexible design that restricts users to customize their UI. **E:** *"I hope you provide **more themes for the picture background and different font styles and colors**. I'll give you 5 stars then."*
(E-3) Advertisement	**D:** Inappropriate way the advertisements are presented. **E:** *"Slow and now with **1/3 of the screen blocked by advertising**. Really SUCKS NOW."*
(E-4) Feedback	**D:** UI fails to provide appropriate, clear, and timely feedback. **E:** *"The playing button can take about 20 seconds to switch status before it really starts playing songs."*
(O) Others	
(O-1) Generic Review	**D:** Generic evaluation and subjective perception of the UI. **E:** *"This app is PAINFULLY UGLY please redesign it."*
(O-2) Comparative Review	**D:** Compare current UI with UI of other versions or of other apps. **E:** *"After the previous update this app **become ugly and less accurate**."*
(O-3) Design Specification	**D:** Underuse, misuse, and abuse of current design specification. **E:** (1) *"This is not IOS it is a Google product. Bring back the material design."* (2) *"Hope for **material design for action bar** and setting page! And make the height of action bar larger says 56dp just like other normal apps."*

Each contains several issue types with description and example reviews.

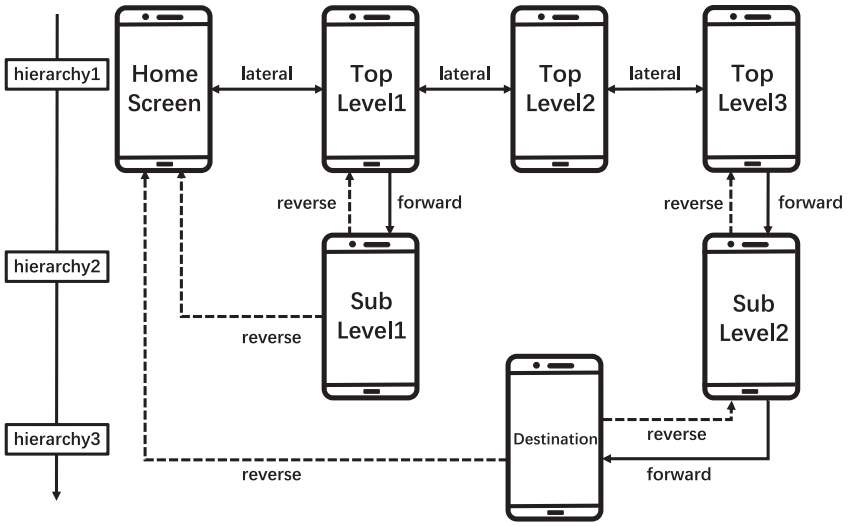


Fig. 3. An example of UI navigation design with lateral navigation (the solid line), forward navigation (the solid line), and reverse navigation (the dotted line), which has three hierarchies. UI navigates users to explore the app and fluently reach the destination. The navigation design should be clear with proper hints and avoid being too complex (e.g., avoid too many hierarchies).

(A-4) Iconography includes *product icons* (i.e., logo) that give visual expression of a brand, *system icons* that symbolize common actions, and *animated icons* that express certain status shifting [26]. According to our observation, the frequencies of the three icons are 34.3%, 56.2%, and 9.6%. Issues with (1) *product icons* are the most frequent complaints (56.2%). *Product icons* (app logos) give the first impression to users and even determine if users will select the apps. The importance of *product icons* in the app competition is highlighted by Miniukovich et al. [61]. We observe the outdated *product icon* without modern design is often complained about. (2) *System icons* are meant to be simple, visual elements that are recognized and understood immediately. So, UIs that fail to make *system icons* intuitive will induce complaints. For example, wrong icons will confuse users (e.g., *"The icon shifting songs instead of playing is so confused."*). So they cannot get the right information from the graphic meaning, which degrades the rating in the review. (3) *Animated icons* reflect the action an icon performs. However, some animation is annoying, so the users will be irritated. For example, we observe a user complaining that *"Floating icon is the most annoying thing ever,"* as the animation of floating will distract the user.

(A-5) Image issue refers to inappropriate use of image, including *illustration* and *photography*. The illustration helps describe abstract concepts (e.g., a map app uses illustration that contains several arrows to guide users to find destination step-by-step.) while photographs can better represent specific concepts. We observe that over 70% of issues lie in size, that is, the photos are too small to be watchable, and others are too large, so they lead to pixelation of the image.

3.1.1.2 App Interaction. The category mainly involves bad designs that lead to inconvenience or failures of mutual actions with UI. We categorized interaction issues into UI issue types of *"navigation," "notification," "motion," "gesture,"* and *"accessibility."*

(I-1) Navigation issue refers to the dedicated design to guide users to move through an app. As shown in Figure 3, the app can navigate users with methods of *Lateral Navigation* (solid lines), *Forward Navigation* (solid lines), *Reverse Navigation* (dotted lines), and *Search Navigation* (direct access to the destination page) [26]. The issue of navigation occurs when users have difficulty in

moving through the UI or even fail to find their destinations (e.g., the target button to open an article). According to our observation, users will complain about the ambiguous *Forward navigation* (e.g., *"guidelines should be provided in detail"*), absence of *Reverse Navigation* (e.g., *"I cannot come back to home screen"*), and malfunction of *Search Navigation* (i.e., cannot retrieve required UI page). The complex hierarchy of UI leads to navigation issues. For example, users are likely to be lost in too many hierarchies. Sometimes, users want to locate and switch to the destinations directly (e.g., use the search function to navigate to a certain UI like the homepage), but when the position is wrong (i.e., cannot retrieve required content), they will degrade the rating.

(I-2) Notification issue refers to the absence of necessary notification and the abuse of irritating notifications. We observe users complain when the notification fails to inform them of the correct status of the app. For example, users require to be informed when the app is running in the background (*"It always runs in the background and runs out my battery, but I'm not notified"*). Besides, more complaints happen when app owners abuse notifications (e.g., too frequent pop-up notifications on banners). Intense notifications will also increase the energy consumption, as they require extra resource usage (e.g., network usage to keep synchronizing the data) [48, 49, 76], hence may reduce the fluency of the app and influence user experience.

(I-3) Motion refers to the move, reshape, and transition of UI elements. The issue of motion includes counter-intuitive motion design and meaningless motion design. Counter-intuitive motion is not consistent with the actions of users (e.g., *"It is slow to move items"*) and fails to be informative (show what will happen if the action is taken, e.g., *"can not feel the process of transition"*). Meaningless motion fails to be focused (facilitate drawing users' attention on important content) and expressive (express spatial relationships between elements). For example, in a review, a user is *"often distracted by disturbing transition animation when browsing and shifting the pages."*

(I-4) Gesture of UI interaction refers to the special touch actions that enable users to interact with the screen. The issue of gesture refers to the design of gesture is not consistent with user expectation. Table 4 shows nine main basic gestures describing how users can interact with the screen [26] in which seven gestures get complaints. Some of these gestures malfunction and fail to meet the need. The other complaints are incurred because of the oblivion of the gestures design. For example, when the user wants to shift the page by swiping but finds this intuitive gesture is disabled, the user feels inconvenienced and degrades the rating.

(I-5) Accessibility issue refers to the unfriendly use of UI, especially by people with disabilities [33], including those with low vision, blindness, hearing impairments, cognitive impairments, motor impairments (e.g., partial paralysis or lack of coordination have difficulty typing on the screen). Improving accessibility is an important aspect in designing an app, as 15.3% of the world population has a prevalence of a moderate disability, and 2.9% have a severe impairment [93], and these users need to use the apps with the barrier-free design. As shown in Table 5, we reference the work of Yesilada et al. and Krainz et al. [47, 96] and evaluate issues on our collected app reviews. These aspects are only very basic principles of accessibility, but there are still many issues not user-friendly to people with disabilities. Considering there are many reviews not explicitly mentioning accessibility even impede normal users from normal use, the problems can only be much more severe for the disabled.

3.1.1.3 User Experience. The category *experience* mainly involves ill-considered design that lowers users' satisfactoriness of UI. We categorized experience issues into *"redundancy," "customization," "limitation," "advertisement,"* and *"feedback."*

(E-1) Redundancy issue refers to a redundant UI design that makes UI bloated and user-unfriendly. We identify redundancy issue when users explicitly complain about the UI as bloated

Table 4. Gesture Types Provided by Google Design [26] and the Number of Reviews of Each Issue Type That Are Mentioned

Gesture	Review Number	Example
Tap	25	*"I followed the instructions and **tapped the screen** to focus on an area and it refused to do it."* [t]
Scroll and Pan	6	*"It is confusing and more painful to **scroll on screen**!"*
Drag	10	*"The screen **drag need improvement**."*
Swipe	7	*"Outlook calender doesn't interface with the native calendar **unable to swipe** through emails."*
Pinch	7	*"**I had problem with pinch to zoom out** before the tutorial prompt which left the instruction at the bottom on the screen."*
Pick up and Move	6	*"After choosing the image I want to edit but **just hangs there**."*
Compound Gestures	15	*"I have a problem with the controls every time I **tap** and **drag**. It dosent register it like on the long notes I **hold on the screen** and I'm the middle it stops."*
Long press	-	-
Double Tap	-	- [b]

"-" indicates the gesture type is not mentioned.

Table 5. List of App Accessibility Issues [47, 96] and the Number of Reviews of Each Issue That Are Mentioned

Accessibility issues	Review #	Example
Functional Images Lacking Text	7	*"Voice guidance is very lame. **No image caption of left or right**. Other times if you need to make a right turn it will say bear right."*
Image Contrast	8	*"The new design is very distracting and I can't read some buttons because they have **the same color with the background**."*
Screen Brightness	7	*"Need better **sunlight visibility** compared to stock brightness."*
Cascading Menu	4	*"Tapbox overlap to somewhere."*
Touch Target Size	5	*"The buttons are **too small to people with sight problem**."*
Text Legibility	12	*"All of my song names and their brief stats (on the main page) are colored in highlighter yellow. It's almost **impossible to see the letters and numbers**."*
Magnificantion Issue	9	*"It may be my Droid MAXX but magnifer picture is jerkey and makes it difficult to focus. **Not recommended for people with sight problems**."*
Position of UI Elements	-	-
Rich Images Lacking Equivalent Text	-	-
Missing Layout Clues	-	-
Double Label	-	-
Images Used as Titles	-	-

"-" indicates the issue is not mentioned.

(41 reviews) or complain about unnecessary UI components (71 reviews). We observe that feature enhancements often cause redundancy. Though new designs bring new troubles [30], at the same time, UI evolution is inevitable and could be useful. Prior research shows that to win the market, app owners have to add features [70] frequently, and we observe even though some apps do have redundancy issues, there still are users who express appreciation and benefit from their UIs.

(E-2) Customization issue refers to an inflexible design that restricts users to customize their UI. Users will complain about the inflexible design if they have no options for customization. Some users will ask for adding extra modes of the UI for the particular need (e.g., an intense contrast mode under intense light) and for the aesthetics (e.g., choose their preferred background). By customizing according to personal preference, users can improve the usability and aesthetics of UI.

(E-3) Advertisement issue refers to the inappropriate way the advertisements are presented.

We observe that app owners embed advertisements to the UI in a crude way, which prevents users from normal use (e.g., pop-up ads). For example, some advertisements show up in an unexpected way (e.g., *"In-app advertisements pop up and misguide me"*) or block users from using the app (e.g., a user complains he/she cannot even close the ads). It is hard to say the user-unfriendly advertisements are not intentionally designed, because the more users click the advertisements, the more the app development team could get paid [78]. Some advertisements are designed to be put on the places where users are more likely to touch (e.g., open screen ads, eye-catching banner, or even full-screen ads).

(E-4) Feedback issue refers to the lack of appropriate feedback, especially when the apps malfunction.

We observe two kinds of cases in which the feedback is absent or unsynchronized. The first is missing status feedback, which means UI does not inform users of the status of the app. Without being informed of the status, users will be confused, then angry, and degrade the rating at last. The second is failing to consider exceptions and designing backup UIs to give feedback. A typical example is that the unsynchronized UI fails to give feedback when issues between UI and back-end leading to an unexpected page.

3.1.1.4 Other UI Issues. Users often use subjective and ambiguous words to complain about UI design. We categorized these issues into *"generic," "comparative,"* and *"design specification."*

(O-1) Generic Reviews are those reviews in which users just express their generic evaluation and subjective perception of UI without providing specific information.

(O-2) Comparative Reviews are those reviews in which users compare the current UI with the UI of prior versions of the app (91 reviews) or compare the current UI with the UI of other apps (51 reviews).

(O-3) Design Specification issue are those reviews in which users think the UI design fails to follow design specification, which violates the commonly expected behavior or look.

3.1.2 RQ1.2 What are the Characteristics of UI Issues? Khalid et al. examined the low rating (1- and 2-star) reviews to study how these reviews negatively reflect on the quality of an app [43]. *"Low Rating Ratio"* is the percentage of low rating reviews in all reviews in a particular UI issue type. Hence, as shown in Table 6, we calculate the *"Low Rating Ratio"* in our studied dataset, which can reflect how negative a UI issue type is. We use *"Low Rating Ratio"* to indicate the severity of an issue type, since low ratings largely influence the app reputation [82]. The observations and explanations are as follows:

The low rating ratio of free apps is higher than the low rating ratio of non-free apps. The significant test shows the difference is significant (p-value <0.05) with medium size. The higher value of the low rating ratio indicates a high percentage of negative reviews. It can be

Table 6. The Number of Reviews and the Low Rating (1- and 2-star) Review Ratios of Each Issue Type in Free, Non-free, and All Apps

UI issue type	Free app reviews	Low rating ratio	Non-free app reviews	Low rating ratio	All app review	Low rating ratio
Layout	25 (3.2%)	0.44	37 (5.6%)	0.43	62 (4.3%)	0.44
Legibility and Color	25 (3.2%)	0.16	25 (3.8%)	0.12	50 (3.5%)	0.14
Typography and Font	57 (7.2%)	0.33	16 (2.4%)	0.25	73 (5.0%)	0.32
Iconography	32 (4.0%)	0.50	41 (6.3%)	0.20	73 (5.0%)	0.33
Image	36 (4.5%)	0.36	37 (5.6%)	0.35	73 (5.0%)	0.36
Navigation	39 (4.9%)	0.28	36 (5.5%)	0.42	75 (5.2%)	0.35
Notification	36 (4.5%)	0.69	81 (12.4%)	0.25	117 (8.1%)	0.38
Motion	19 (2.4%)	0.42	18 (2.7%)	0.22	37 (2.6%)	0.32
Gesture	41 (5.2%)	0.63	29 (4.4%)	0.48	70 (4.8%)	0.57
Accessibility	12 (1.5%)	0.33	10 (1.5%)	0.40	22 (1.5%)	0.36
Redundancy	8 (1.0%)	1.00	104 (15.9%)	0.38	112 (7.7%)	0.43
Customization	64 (8.1%)	0.25	43 (6.6%)	0.19	107 (7.4%)	0.22
Advertisement	48 (6.1%)	0.67	5 (0.8%)	0.60	53 (3.7%)	0.66
Feedback	44 (5.6%)	0.57	50 (7.6%)	0.50	94 (6.5%)	0.53
Generic Review	183 (23.1%)	0.17	49 (7.5%)	0.10	232 (16.0%)	0.16
Comparative Review	133 (16.8%)	0.92	9 (1.4%)	0.56	142 (9.8%)	0.90
Design Specification	7 (0.9%)	0.71	88 (13.4%)	0.26	95 (6.6%)	0.29
Total	792	0.47	655	0.31	1,447	0.39

We use *"Low Rating Ratio"* to indicate the severity of an issue type. A high value of *"Low Rating Ratio"* indicates a high percentage of negative reviews.

explained that the UI quality of free apps is lower than the non-free apps. This result is consistent with the observation in our quantitative analysis in which the app rating with UI-related reviews is lower than the app rating with all reviews.

The low rating ratio of *"Comparative"* reviews is nearly 50% higher than the second. This observation reflects that users are sensitive to the differences between UIs, especially the changed UI and the competitor apps' UI. Users are likely to give low ratings once they find the shortage of the UIs compared with other UIs. This observation also reflects the intensive competition in the app store [4].

We calculate the Spearman correlation (the rank correlation, ρ) of free and non-free apps. The Spearman correlation ranges from -1 to 1. It will be high when the UI issues have similar ranks (i.e., the relative position of UI issue types) between the free and non-free apps in terms of severity (i.e., the *"Low Rating Ratio"*). For example, the correlation coefficient is 1 if the severity ranks are the same (e.g., both ranks are 1st *"Layout,"* 2nd *"Image,"* 3rd *"Gesture"*). As a result, the rank correlation coefficient is 0.51, indicating there is a moderate correlation between free and non-free apps in terms of issue severity (i.e., *the "Low Rating Ratio"*). The p-value is less than 0.05, indicating such a correlation is significant. Our interpretation of ρ is based on Hinkle et al.'s scheme [34]: low correlation ($0.3 \leq |\rho| < 0.5$), moderate correlation ($0.5 \leq |\rho| < 0.7$), high correlation ($0.7 \leq |\rho| < 0.9$), and very high correlation ($0.9 \leq |\rho| \leq 1$).

Reviews about UI issues of the free-to-download apps are more negative and more generic than the reviews of the non-free apps. The overall *"Low Rating Ratio"* of free apps is 0.47, which is higher than the *"Low Rating Ratio"* of non-free apps, 0.31. As the higher the ratio, the more severe the UI issue, it can reflect an overall better UI quality of non-free apps, which is consistent with the

Table 7. The Number of Reviews of Each Issue Type in Game and Non-game Apps

UI issue type	Game review	Non-game reviews
Layout	9 (5.1%)	53 (4.0%)
Legibility and Color	12 (6.7%)	38 (2.9%)
Typography and Font	10 (5.6%)	63 (4.8%)
Iconography	3 (1.7%)	70 (5.3%)
Image	4 (2.2%)	69 (5.3%)
Navigation	7 (3.9%)	68 (5.2%)
Notification	15 (8.4%)	102 (7.8%)
Motion	6 (3.4%)	31 (2.4%)
Gesture	12 (6.7%)	58 (4.4%)
Accessibility	4 (2.2%)	18 (1.4%)
Redundancy	17 (9.6%)	95 (7.3%)
Customization	7 (3.9%)	100 (7.6%)
Advertisement	5 (2.8%)	48 (3.7%)
Feedback	31 (17.4%)	63 (4.8%)
Generic Review	21 (11.8%)	211 (16.1%)
Comparative Review	1 (0.6%)	141 (10.8%)
Design Specification	14 (7.9%)	81 (6.2%)
Total	178	1,309

observation that the mean UI rating ratio for free apps is lower than non-free apps (Section 2.4). In particular, there are much more *"Generic Reviews"* in free apps (23.1%) than non-free apps (7.5%) and the *"Low Rating Ratio"* is higher (0.17 vs. 0.10). A possible explanation is that users of non-free apps are more likely to leave effective reviews and give specific information for the app owners than apps that are free.

UI issue type *"Color"* has the lowest percentage of negative reviews (0.14). The result indicates that even though users pay attention to the *"Color"* issues, they still give relatively high ratings compared with other issue types. We can explain the observation as: The *"Color"* issue influences less on users' satisfactoriness, and users are more tolerant of the *"Color"* issue compared to other UI issues.

The UI of Game apps is complicated compared with non-Game apps, and Game apps range across 18 app sub-categories. Prior work shows that reviews of Games are different from regular apps (non-Game apps) [50]. Therefore, as shown in Table 7, we also explore the differences in UI issue types of Game and non-Game apps. The low rating ratio of Game apps is relatively lower than the overall reviews.

In the Game apps, the *"Feedback"* issue is the most occurring UI issue type (17.4%). In non-Game apps, *"Generic Review"* and *"Comparative Review"* are the top two frequent UI issue types.

Summary of RQ1

We identify 17 issue types (e.g., *"Accessibility"* and *"Advertisement"*) that belong to four main categories: *"Appearance"*, *"Interaction"*, *"Experience"* and *"Others"*. We find that reviews about UI issues of the free-to-download apps are more negative and more generic than the reviews of the non-free apps.

Table 8. Proportion of Replies to UI-related Issues

Statistics	Free	Non-Free
App number	22,199	9,380
Number of reviews with reply	338,156 (11.14%)	43,149 (13.50%)
Average rating	3.53	3.29

Table 9. The Length and Updating Frequency of Dialogues on UI Issues

Review number	Review-response iteration number	Avg. update time (days)
275,391	0	6.39
31,134	1	63.02
2,845	2	65.40
487	3	61.47
86	4	2.19
37	5	7.70
37	>5	5.50

"Avg. update time" refers to the average update time of reviews in days.

3.2 RQ2 How Do App Owners Interact with Users Concerning UI Issues?

3.2.1 RQ2.1 How Many and How Frequent Are the Interactions between the Users and App Owners Concerning UI Issues? App owners of non-free apps have a slightly higher probability of responding to UI-related reviews than app owners of free apps. We are hesitant to interpret it as users' payments motivate app owners of non-free apps to respond more actively. Still, a possible explanation we can give according to our observation in this study (Section 3.1.2) is that users give less *"Generic Reviews"* to non-free apps (7.5%) than free apps (23.1%), so app owners are able to respond to reviews with specific information. We identify the UI-related reviews with responses and count the statistics of UI-related reviews that get responses from app owners and their ratings. As shown in Table 8, app owners of non-free apps respond more frequently than app owners of free-to-download apps.

Table 9 shows the distribution of the number of iterations within a dialogue between app owners and users. During our study period, we find that 98.8% of the dialogues end after zero or one iteration. On average, the first iteration is within a week, indicating that the app owners are willing to respond to users. It will take users up to two months to further contact app owners (i.e., the first round of dialogue). The second round of dialogue (i.e., two or three iterations of reviews) also takes a long time for several reasons: (1) app owners need time to update the UI and then notify users; (2) users enjoy the new UI and express appreciation after a certain amount of time; (3) for a long time users find the UI still the same and complain again. However, when the number of iterations is more than three, the average review update time gets faster. It indicates that once app owners build a tight bond with users, the dialogue will be more active, and end-users are involved in the rapid iteration process to improve the UI.

We observe some app owners will automatically respond to user reviews. These automatic responses are often templates that express appreciation or apology and ask users to contact them via support system or email (e.g., *"Thank you for your feedback! please contact via XXX@XXX.com if you have further feedback"*). We regard identical responses in one app as automatic responses and find that in our dataset, there are 73,456 automatic responses (21.72% of free reviews with

Table 10. Dialogue Patterns and Statistics of the Patterns in Free-to-download Apps and Non-free Apps

Dialogue Pattern	Description	Free apps	Non-free apps	Total
Apology or Appreciation	App owners express apology or appreciation.	155 (39.0%)	103 (26.6%)	258 (32.91%)
Give Specific Advice	App owners help users to solve UI issues.	46 (11.6%)	58 (15.0%)	104 (13.27%)
Information Request	App owners ask for more details of UI issues.	71 (17.9%)	56 (14.5%)	127 (16.2%)
Justify the UI Issues	App owners give a rationale to the UI design.	30 (7.6%)	44 (11.4%)	74 (9.44%)
Make Promises	App owners promise users to improve the UI.	33 (8.3%)	44 (11.4%)	77 (9.82%)
Update Notification	App owners notify the users the issues are fixed.	32 (8.1%)	47 (12.1%)	79 (10.08%)
Unspecified	App owners respond irrelevant information.	30 (7.6%)	35 (9.0%)	65 (8.29%)
Automatically Reply	App owners use template-based reply.	-	-	-

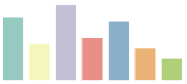
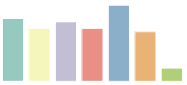
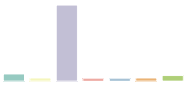
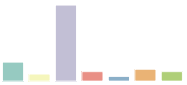
replies) in free apps and 5,022 automatic responses (11.64% of non-free review with replies). As the automatic responses are all identical, we regard it as one pattern and study the rest reviews with responses. Even though they are automatic template-based responses, it can also help trigger further communication of users.

3.2.2 RQ2.2 What Are the Patterns of the Interactions between the Users and App Owners? As described in the Approach section, we manually study a statistically representative random sample of 764 dialogues. We exclude the template responses, as they are not informative. Finally, we identify eight dialogue patterns that app owners attempt to communicate and collaborate with users to solve UI issues. Table 10 shows the identified patterns. In the study, the pattern *"Apology or Appreciation"* counts most in both free and non-free apps, because it is a basic polite response. In free apps, the pattern of *"Information Request"* is the second most frequent. This can be explained as: The details of the UI issue in the review are often not informative enough, which is consistent with our previous observation in identifying UI issues in Section 3.1.1 (i.e., 16.0% of the UI-related reviews are generic user reviews, which is the highest percentage among all UI issues).

3.2.3 RQ2.3 What Are the Distributions of Response Patterns on each UI Issue? To explore how app owners interact and collaborate with users, we further combine dialogue patterns with UI issues. Table 11 shows the distribution of response patterns on each UI issue in which the bars of the histogram in each UI issue, respectively, correspond to the patterns in Table 10 except for the pattern of *"Automatic Reply."* The observations and explanations of the study are as follows:

In the free apps, app owners tend more to justify the UI (red bar) and promise the UI improvement (orange bar) for *"Experience"* issue than *"Appearance"* issue and interaction issue. In our observation, the main increase of *"Justify"* pattern comes from **(E-3)** advertisement issue. In the dialogues, app owners explain they need the revenues from the app and ask for understanding but often ignore to design a less intrusive UI with advertisements. The main increase

Table 11. Distribution of Response Patterns on Each UI Issue

UI Issue	Patterns in free apps	Score	Patterns in non-free apps	Score
Appearance		2.96		3.46
Interaction		2.56		3.30
Experience		2.88		3.15
Others		4.51		4.24

The bars of the histogram in each UI issue, respectively, correspond to the patterns in Table 10. From the left to the right: "Information Request," "Update Notification," "Apology or Appreciation," "Justify the UI Issue," "Give Specific Advice," "Make Promise," and "Unspecified," except for the pattern of "Automatic Reply," as it was eliminated before.

of *"Promise"* pattern comes from **(E-2)** customization issue in which app owners promise users to meet their customized requirements on UI.

As in Table 11, in both free and non-free apps, there are more *"Give Specific Advice"* responses (the blue bar) for the *"Interaction"* issue than the *"Appearance"* issue. In these *"Interaction"* issues, most responses that give advice for users are the *"Navigation"* issue (46.51%) and the *"Gesture"* issue (37.21%). For example, app owners guide users to find the required functions or advise users to use appropriate gestures. Besides, we observe that in 48.39% of the responses to *"Experience"* issues, app owners do design customizations for the apps and recommend users to choose the preferred UI.

In both free and non-free apps, the pattern of *"Apology or Appreciation"* accounts for the overwhelming proportion. It indicates that app owners do not have good approaches for responding to generic UI issues (especially **(O-1)** *"Generic Review"*) except for attempting to trigger further communication with users. Many app owners attempt to ask users to contact via other tunnels (e.g., email or within the app) so we cannot observe further communication. However, we still observe the cases that the *"Apology or appreciation"* dialogues trigger *"Give specific"* dialogues. For example, there is a dialogue as follows:

- **Review**: *"Very good audio and video quality but have a little UI issue."*
- **Response 1**: *"Thanks for the review! There are no good applications without user feedback!"*
- **Review update 1**: *"The 360/180 interface is very close to the top of the screen and easily triggered because it is exactly in the top center and you tend to hold your head on the center more rather than the sides."*
- **Response 2**: *"Have you tried changing the vertical position of the UI for 360/180 videos in the settings?"*
- **Review update 2**: *"It works, thank you very much."*

Table 12. Rating Changes before and after a Dialogue between App Owners and Users

	Review rating after the dialogue				
Before the dialogue	Rating 1	Rating 2	Rating 3	Rating 4	Rating 5
Rating 1	24.61%	**1.25%**	**2.16%**	**2.40%**	**3.02%**
Rating 2	4.70%	5.90%	**1.42%**	**1.92%**	**2.52%**
Rating 3	2.69%	2.97%	7.03%	**2.71%**	**3.60%**
Rating 4	1.10%	1.18%	1.94%	5.95%	**3.84%**
Rating 5	1.03%	0.60%	1.06%	1.32%	13.10%

The values above the diagonal mean that the rating value is increased and values below the diagonal (i.e., bold values) mean that the rating value is decreased. The sum of values of the upper triangle of the matrix is larger than the lower triangle.

- **Response 3**: *"Thank you very much for the wonderful review! We are so glad you are loving the app!"*

In this dialogue, the app owner expresses appreciation when a user reports a UI issue abstractly, which motivates the user to give a more specific description, and the app owners help to solve the issue.

Summary of RQ2

We identify eight dialogue patterns that app owners interact with users via the review-response mechanism. We find that *"Apology or Appreciation"* is the most frequent response pattern. In both free and non-free apps, there are more *"Give Specific Advice"* responses for the *"Interaction"* issue than the *"Appearance"* issue.

3.3 RQ3 Will the Interaction Improve Users' Satisfactoriness?

3.3.1 RQ3.1 What Are the Rating Changes before and after the Interactions between the Users and App Owners? There is a positive impact if app owners could actively interact with users to improve UI quality. We observe that responding to users will positively affect users, which is also observed by Hassan et al. [30]. Table 12 shows the detailed percentage of the change in the user review rating after an interaction happens between the app owners and users. As illustrated in Table 12, 24.82% (i.e., the sum of the values above the diagonal) of the interactions get an increase in the review ratings. The ratio of rating increase (24.82%) is more than the ratio of rating decrease (18.58%). The reason for most rating decrease is that app owners do not fully meet users' needs or respond too late. We examine a statistical representative sample to understand why users increase their ratings, and our observations are as follows:

In cases of star raising from three and four (i.e., the bold values in the third and the fourth columns in Table 12), 43.3% cases raise the rating by justifying the benefit of displaying advertisements. An interesting phenomenon in reviews of free apps is that users would complain about advertisements but give a rating of five stars (e.g., *"Good app, fewer ads is better"*). It reveals a tolerance of moderate advertisements.

Implementing the requested UI features could improve the satisfactoriness and lead to a rating increase of the low rating (1- and 2-star) reviews. In cases of star raising from one and two (i.e., the bold values in the fourth and the fifth column in Table 12), the app owner promises users to improve the UI (i.e., *"Make Promises"* pattern). In these cases, we observe when app owners update the requested UI, users are satisfied with the new design and increase the rating, which indicates the importance of active updates. However, in our survey for real app

Table 13. An Example of Dialogue for the "Nuttri" App

Rating	Reviews	Date	Response	Date
4 Star	It's nice and simple which is great but I would love to have something that I can export the data and get **a list view of what I fed baby and when**. Currently I have to click into the date to see what I fed baby.	2018-2-7	Thank you for your review *user*! We will keep your comment in mind while working on future versions of our app!	2018-2-7
5 Star (Raised by user from 4 Star)	**UPDATE**: The latest update offers a week by week view. While I appreciate the fact that the meals for the day is visible on the same screen the inability to switch between month view and weekly view makes it difficult to find a date. I would really appreciate a list view of all the meals and an export option.	2018-2-9	**UPDATE**: Hi *user*! alright let us bring the monthly view back! On the export option would you be able to contact us at *developer@email.com* and tell us more about it?	2018-2-9

App owners satisfy the need of the user, update the UI timely, and notify the user. The user expresses satisfaction and raises the rating.

owners (Section 3.4), they think satisfying all users cannot always work because userss' requests may contradict each other (e.g., an app owner says, *"Users tell me that they like a feature that another user does not like."*).

3.3.2 RQ3.2 How Do the App Owners Interact with the Users to Improve the UI Quality? **The positive impact of interaction to improve users' satisfactoriness is mainly based on active updating of UI.** In our observation, most rating increases (51 reviews) are obtained when users are informed of the UI updates and then raise the rating. Table 13 shows an example of dialogue in reviews of the "Nuttri" app. In the dialogue, a *user* presented a customization issue **(E-2)**, and app owners promised to improve the UI. Then they fulfilled the promise and redesigned the UI. At last, the update satisfied *user* and app owner obtained a rise of rating. In particular, we observe a review in which the app development team insists on informing the users of the new improvement of UI for up to 10 times and finally obtained an increase of 5 stars of users. On the contrary, the decreases in ratings are caused by ignoring users' complaints or failing to fulfill the promise (the 6th dialogue pattern). For example, we observe a user required changeable icons (**(E-2)** customization issue) for nine times and gradually decreased the rating from 5 to 1 when finding the UI was not improved.

By giving advice, app owners can help users to get used to UI changes. In the cases of rating increases, 15.96% of the issues are *"Give Specific Advice."* In these dialogues, we observe that the advice is mainly for *"Interaction"* issues in which responses to *"Navigation"* issues obtain most of the rating increase (73.34%). In contrast, it is also possible to get specific suggestions in user reviews to improve UI quality. We observe in many cases app owners incorporate the suggestions for *"Interaction"* issues and get the rating increases.

Communication with users can give a rationale for UI design and satisfy users without changing the UI design. In our sampled data, 29.79% of issues do not really make a change on the UI. For example, in 53.3% of these cases, we observe app owners explain to users the advantages of a new UI design to alleviate users' inconsistent perception, and users raise the stars. We also observe that in 33.3% of these cases, app owners explain to users why they need to design advertisements to get revenues, and users accept the explanation and raise the rating. Besides, we observed the users appreciate that app owners can respond and raise the rating (two cases).

Summary of RQ3

There is a positive impact if app owners could actively interact with users to improve UI quality. Implementing the requested UI features could improve the satisfactoriness and lead to rating increases of low rating reviews. We observe that app owners can increase the user ratings without deploying new updates by giving specific advice or giving rationale to the UI design.

3.4 A Survey of the App Owners

3.4.1 Motivation. In this article, we identify UI issues in user reviews and summarize them using a taxonomy. We also explore the interactions between users and app owners in the Google Play Store. In this section, we report our user study that aims to investigate how the taxonomy can help app owners in practice.

3.4.2 Validation Survey. In this article, we utilize the methods of Kichenham et al. [45] and Chen et al. [14] to design a survey for collecting app owners' opinions concerning the UI issues. To increase the response rate, we adopt an anonymous survey. Besides, we provide a raffle for the app owners who participate in our survey.

3.4.3 Survey Design. First, we collect demographic information to better understand the background of the respondents. The following five questions can help us have an overall understanding of the respondents:

a. Demographic.

- Professional mobile application developer?: Yes / No
- Involved in the development of app UI?: Yes / No
- The main role in the UI development?: Design / Test / Development / Management / Others
- Experience in years (decimals OK)?
- Current country of residence?

Then, we give an example of a user review that discusses the UI issues in the Google Play Store. After that, we show the summarized UI issues as well as the corresponding descriptions and examples. Then, we ask respondents about the difficulty in identifying and solving UI issues in the user reviews. Last, we ask about the usefulness of our UI issue taxonomy. The following are the main questions of our survey:

b. Questions about the UI issue taxonomy.

- How difficult is it for you to identify the following UI issue types in the user reviews? (17 issue types in four UI issue categories)
- In the four categories, which kinds of UI issues do you think are difficult to fix in order to satisfy users? (*"Appearance," "Interaction," "Interaction,"* and *"Others"*)

- To what extent do you think categorizing the UI issues in user reviews can help you better identify the true intentions of users and improve the app UI?
- To what extent can the categories of UI issues serve as a guideline to help you better communicate with team members? (individual developers OK)

To increase the response rate, we prepare two versions of the survey (the Chinese version[4] and the English version[5]), as Chinese is the most spoken language in the world, and English is an international language. The Chinese version survey is carefully translated to ensure that the content between the two versions is the same. Since some questions or the issue types may not be easy to understand, we add an option "not sure" to ensure reliable results. We also give each question a textbox to enable respondents to give their opinions.

3.4.4 Recruitment of Respondents. To get a sufficient number of respondents from different backgrounds, we first piloted our survey with some partners who are working in world-famous companies. We then sent our email to 534 app owners who contribute to open-source Android app projects on GitHub according to the list of F-droid (we also ensure that these apps are also on the Google Play Store). All respondents could enter their email to participate in a raffle to win two $50 Amazon gift cards.

3.4.5 Results of the Study. In total, we received 43 responses (the response rate is about 8.05%). Our respondents come from 19 different countries or regions. The top three countries in which the respondents reside are China (18.60%), Italy (16.28%), and German (11.63%). In our collected surveys, 63% of the app owners are professional developers and 90.70% of them are involved in the development of the UI. Among these respondents, 1 (2.33%), 12 (12.32%), 2 (4.65%), 1 (2.33%) described their job roles as design, development, test, and management, respectively. The other 27 (62.79%) responses said they have multiple roles (full-stack).

Table 14 shows the respondents' ratings for the difficulty in identifying UI issue types. The ratings range from 1 (very easy) to 5 (very hard). The "average" rating evaluates the average difficulty of the issue type. Compared with the *"Appearance"* and *"Experience"* issues, *"Interaction"* issues are relatively difficult to identify, as the average rates of *"Motion," "Gesture,"* and *"Accessibility"* are 3.19, 3.23, and 3.16, respectively. Besides, *"Generic"* UI issues are also relatively hard to identify (3.19).

Considering fixing the four kinds of UI issues, though the *"Interaction"* issue is relatively hard to identify, the respondents do not think it is the most difficult to fix (16, 37.21%). In contrast, the respondents think it is hard to fix the *"Appearance"* issue (22, 51.16%) and the *"Experience"* issue (25, 58.14%) (the question allows multiple choices).

In total, we received 32 replies to the free-form questions. Some replies give very valuable information on the handling of UI issues in practice. To fix UI issues, respondents often need more specific information from users (e.g., *"I often have to contact the user again by email to clarify the issue,"* and *"I read the reviews and try to understand what the user is referring to, but often users just say something like 'the user interface is bad' without giving specifics."*). They also agree it is hard to satisfy all users (e.g., *"Sometimes I disagree with the user assessment, or other users tell me that they like a feature that another user doesn't like."*). Besides, several respondents say that they prefer to fix UI issues with specific information or UI issues that require fewer efforts (e.g., *"Only the reviews are prior (or urgent) and clear, I will try to fix them,"* and *"Accept the proposal if it improves the UI, and it is not a major change to app structure."*).

[4]Content of the survey: https://osf.io/yh7w5.
[5]Content of the survey: https://osf.io/x4u5e.

Table 14. Results of the Ratings for the Difficulty in Identifying UI Issue Types in the User Reviews

	1 (Very easy)	2 (Easy)	3 (Neutral)	4 (Difficult)	5 (Very difficult)	Not sure	Average
Layout	3	19	8	7	4	2	2.63
Legibility and Color	8	16	6	7	4	2	2.47
Typography and Font	3	18	10	8	2	2	2.58
Icon	3	13	13	7	3	4	2.58
Image	3	17	6	10	2	5	2.44
Navigation	6	16	4	12	3	2	2.63
Notification	7	13	13	4	5	1	2.63
Motion	3	4	12	15	6	3	3.19
Gesture	3	8	8	14	8	2	3.23
Accessibility	4	9	5	11	11	3	3.16
Redundancy	6	19	6	5	3	4	2.26
Customization	4	17	8	8	3	3	2.53
Advertisement	12	12	7	3	1	8	1.72
Feedback	4	19	13	2	3	2	2.42
Generic Review	4	9	7	11	10	2	3.19
Comparative Review	7	7	13	10	4	2	2.79
Design Specification	4	14	10	8	1	6	2.30

In summary, our taxonomy gets positive feedback because 25 (58.13%) respondents think that our UI issue taxonomy is useful (18, 41.86%) or very useful (7, 16.28%) in helping them map UI issues from the user reviews; 14 (32.56%) respondents agree the taxonomy is likely to be useful and no respondent thinks that our taxonomy is not useful. 37 (84.09%) respondents think the taxonomy can serve as a guideline in the development team (7, 15, and 15 respondents think most, many, and some of the categories can be a guideline, respectively; 3 respondents are individual app owners). We also receive positive comments on our work. For example, a respondent says, *"I do feel a better system of categorizing UI issues would be good."* Besides, a respondent emphasizes the usefulness for individual app owners, *"I am more skillful in back-end development, but understanding and remembering your code (i.e., our taxonomy) is important (at least you should know which part to look at when users are complaining), especially for self-employed developers."*

4 DISCUSSION

In this section, we discuss the implications of UI design in software engineering and application markets.

4.1 Implication on UI-related Review Analysis and UI Design

App owners can leverage the reviews of different apps to solve subjective-oriented issues (e.g., UI issues), which are often not enough in a single app. We observe that there are many (16.0%) non-informative subjective reviews (e.g., *"The UI is ugly"*) from which it is hard for the designer to figure out specific problems.

A possible explanation of this phenomenon is that users are unable or unwilling to describe issues accurately when they are unsatisfied, since how to evaluate a visual UI using natural language is subjective. Besides, we observe that the user reviews are often casual and oral, as there is no limitation for the users on the writing content. The phenomenon that most of the UI-related

reviews are *"generic"* is consistent with the observation of Chen et al. [15], which found that most of the user reviews are non-informative.

Reviews of one single app are often not enough to find hints. Therefore, our findings can be used as a checklist to improve the UI, because our study is based on a large-scale app reviews dataset and can, to a degree, reflect most concerned UI-related issues.

Automated approaches to tag reviews may help to identify UI-related reviews. However, existing automated tagging approaches are not very successful at tagging issues raised in reviews even for general categories [60, 74]. As the UI issue is a very focused and specific area of app issues, from the perspective of summarizing issues and investigating causes, automated techniques are not comparable to manual analysis. Hence, to achieve better accuracy while providing specific issue information to researchers, we decided to tag reviews manually, as the manual process is considerably more resource-intensive than the automated approaches.

In the next subsections, we discuss the implications of each UI issue type in categories of *"Appearance," "Interaction,"* and *"Experience."*

4.1.1 Appearance. **Improving Layout.** Layouts of UI **(A-1)** should keep consistency across the spatial organization and across devices. The low rating ratio of this issue is 0.44. App owners should make sure the content is organized logically and important content is prominent. The consistency of spatial organization should be guaranteed, because it can help users get accustomed to the layout of UI [26].

Typography and Font. Font style **(A-3)** of text has to make sense in different contexts (e.g., serious versus playful), and typography should be consistent with the theme of the app. This issue's low rating ratio is 0.32, but it is often related to *"Accessibility"* issue **(I-5)**. Designers should not only make important text prominent but also allow users to highlight the content of their interest [26, 29].

Improving Icons. Though icons **(A-4)** are very little, its graphics and visual meaning will also largely influence user perception, especially for *product icon,* because the low rating ratio of *product icon* is 52%. We observe regularly updating *product icons* could often obtain positive reviews, and the update itself is also a common method to prevent users from being tired of the apps and make the product more competitive [43, 70].

4.1.2 Interaction. **UI Notifications.** The percentage of reviews that mention of notification issue **(I-2)** is 8.1%. Future research needs to be done to help app owners design appropriate notifications (avoid the absence or abuse of the notification).

Accessibility Issue. For accessibility issue **(I-5)**, there is a long way for app owners to go to realize the necessity of improving the accessibility of UI to help the disabled [86]. App owners have to keep in mind the principles of accessibility when designing UI. For example, at least meet the basic need of people with sight disability to add captions to images (so they can "hear" what the images are) and guarantee the legibility of UIs [94].

4.1.3 App Experience. **Redundancy is inevitable**. The critical point of the contradiction between users' preference of simple UI and redundant design requirements is that feature enhancement does not equal to UI enhancement. However, UI should be carefully designed to keep its succinctness and avoid frustrating users (e.g., to avoid the review like *"I am confused by too many things on the homepage. The UI is bloated"*).

Advertisement Issue. There is a contradiction for designing advertisements (**(E-2)** issue) on UI: advertisements are profitable, but they are naturally not useful for users (from the perspective of app usability). However, the contradictions are reconcilable. As we observe that users, to some degree, can accept reasonable advertisements, it would be beneficial if app owners could design

self-restrained advertisements with as little influence on users as possible to reduce the resistant emotion of users and gain the understanding of users, especially in free apps. The advertisements' location and form should be taken into careful consideration so the advertisement and the app could be naturally integrated without irritating users. For example, advertisement library developers provide guidance and best practices for displaying advertisements [63, 64]. App owners can apply these guidelines to improve the UI experience while getting revenue from displaying advertisements.

The feedback of UI. Though unexpected processes are inevitable, the feedback of UI **(E-4)** still could be designed to alleviate the negative feeling of users. What is essential is that the attitude to solve the problem must be reflected in the UI design to give appropriate feedback. For example, typical feedback is that when a page is lost, showing "Not Found" and offering help is much better than a blank page. App owners can even provide alternative content to distract the attention of users (e.g., relevant content to attract users) and alleviate the discontent [26].

4.2 Implication on Application Markets

There lacks a tunnel that supports app owners to get sufficient information about UI issues. There are many replies based on a particular template (i.e., *automatic reply* pattern) asking users to contact app owners further (e.g., via email). A possible explanation of this phenomenon is that the dialogues in the app store cannot solve the problem. A typical example is that we observe the most required details are screenshots of the UI. Only based on the text dialogue, there will be an information loss from the natural language of text reviews to visual information of UI. Hence a tunnel to convey visual information (e.g., screenshot image) is required to solve UI issues. However, current app stores do not support the image review, so app owners can only choose to turn to other ways. However, this will hide much information when the dialogue is out of the app store. On the one hand, other users lose a reference to evaluate the app objectively; on the other hand, the user is likely not willing to waste time to cooperate with app owners in an inconvenient way (e.g., capture a screenshot and send it via email) so app owners cannot get suggestive feedback.

Store owners could provide detailed aspects of the rating of an app and build better communication tunnels to help improve UI quality. Two major store owners (the Google Play Store and the Apple App Store) offer only the overall app rating. The overall app rating hides useful information about the aspects influenced by subjective factors, such as whether the UI is inaesthetic or aesthetic. Detailed aspects of rating can give users more reference to evaluate an app before downloading it. For example, a useful app may be degraded by not well-designed UI and get low overall ratings, making the app less competitive in the store. We hypothesize that the fact that the users do not provide good enough information with the first review is an indication that the way to report this kind of error is lacking, thus suggesting a potential communication breakdown. However, a more structured form to provide feedback on this kind of problem could reduce the interaction between the app owners and their users with unforeseeable effects. Hence, store owners (such as Google) should provide both aspects of ratings so users can have the ability to decide whether to download the app and app owners could obtain a clear target to improve their apps.

4.3 Triangulation Evidence from GitHub Issue Tracker

In this article, we study UI issues from the user reviews' perspective and the perspective of the interactions between the users and the app owners. However, it lacks triangulated evidence from the perspective of the real development processes. Therefore, to support our conclusions, we also find evidence from the issue reports of apps in GitHub (a.k.a., GitHub issues). The GitHub issues are mainly for the developers, which are different from the aforementioned UI-related issues in

user reviews. We use the term ***"UI-related GitHub issues"*** to describe issue reports in GitHub that are related to UI.

Identifying available apps. Because many apps in the Google Play Store are proprietary and their issue trackers are not available, we identify open-source apps according to a list provided by F-droid[6] and collect their GitHub issues. The F-droid is a store that keeps open-source Android apps, and many apps on the F-droid store are also on the Google Play Store. In total, there are 5,192 apps in F-droid,[7] and 2,496 out of the 5,192 are hosted on GitHub. We then identify two stores' overlapping apps by filtering apps that are not in Google Play Store and not in our collected dataset. In this way, we identify 1,046 apps and collect their GitHub issues.

Identifying *"UI-related GitHub issues"* and sampling the studied data. In total, there are 78,565 GitHub issues in our collected apps. We use the same keywords set as mentioned in Section 2.3 to perform the searching process, and we retrieve 22,380 *"UI-related GitHub issues."* Note that they are all *"closed GitHub issues,"* so we can observe the final status of the UI issues. We sample 646 *"UI-related GitHub issues"* (with 99% confidence level and 5% confidence interval) and investigate the UI issues in the GitHub issues and the how developers solve them. In the sample, there are 96.44% true positives (i.e., GitHub issues that discuss UI).[8]

Differences of UI issues in the user reviews and in the GitHub issues. We read the sampled *"UI-related GitHub issues"* with a second consideration of the research questions for user reviews. In RQ1, we summarize four main categories (i.e., *"Appearance,"* *"Interaction,"* *"Experience,"* and *"Others"*) in user reviews. Hence, we study and count the sampled *"UI-related GitHub issues."* The studied user reviews and the *"GitHub issues"* are from the same collection of apps. We find that most of them are related to *"Appearance"* (61.61%) and *"Interaction"* (21.20%). Only 11.30% and 2.32% *"GitHub issues"* are related to *"Experience"* and *"Others,"* respectively. In particular, most GitHub issues in *"Appearance"* are *"Layout"* (21.52%), *"Color"* (20.74%), and *"Image"* (13.31%), while most GitHub issues in *"Interaction"* are *"Gesture"* (13.93%) and *"Navigation"* (7.58%). Besides, there are 15 GitHub issues that are related to *"Material design."* As the main audience of GitHub are developers instead of end-users, app developers mainly discuss and improve UIs from aspects of *"Appearance"* and *"Interaction."* Compared with user reviews, *"GitHub issues"* related to UI *"Experience"* is lacking (11.30%), and more requests about end-users' experience are encouraged to improve the UI quality.

How do app owners improve UI according to end-users' feedback in the GitHub issues? In RQ2 and RQ3, we study the interactions between the users and the app owners. Hence, we focus on what motivates developers to improve UI in GitHub issues. From this perspective, we observe that end-users' feedback can motivate the developers to update UI but not all GitHub issues are adopted. Specifically, in 19 GitHub issues, app owners update the UI according to the users' feedback (they mention users' requirements from Google Play Store when proposing GitHub issues to update UI), which is consistent with our finding that *"it is possible to get specific suggestions in user reviews to improve UI quality"* (Section 3.3). However, we also observe in 25 GitHub issues that the developers appreciate the users' feedback but decide not to adopt it. The reasons are *"difficult to implement"* (7), *"leave it to the future"* (5), or unknown (13). Besides feedback from end-users, we find UI guidelines (e.g., material design) play the role of guiding developers to improve the UI in 15 GitHub issues.

In summary, we find that the developers' community is relatively isolated from the end-users, since we can only observe a small number of interactions between the end-users and the app

[6] https://f-droid.org/.

[7] We use the tools provided by F-droid to obtain the app list: https://github.com/f-droid/fdroiddata/tree/master/tools. Last used: 2020/9/15.

[8] There are 12 false positives discussing incorrect image URLs, 7 discussing the interface issues, and 2 discussing the source code style is ugly.

owners, e.g., the rest GitHub issues are often proposed by team members or other developers who follow the project, and the app owners cannot get feedback from the end-users efficiently. For example, we observe an app owner seeks end-users' feedback by randomly reading a little proportion of user reviews but is still agnostic to how users are unhappy with the UI. Better tunnels connected with end-users or a guideline could help developers to improve the UI quality. Our work, especially the summarized taxonomy, can potentially play the role of a guideline to help app owners understand the end-users' perception towards app UI.

5 THREATS TO VALIDITY

5.1 Construct Validity

There are several ways to understand how users perceive the UI of an app. For example, interviewing and surveying users might be one way. In this article, we choose to study UI-related reviews instead. Both approaches have their own limitations. For example, with surveys, users may miss reporting on some instant feedback to UI, depending on their recollection. Nevertheless, the mining approach has its limitation as well. For example, the collected data cannot represent all reasons for complaining about the UI. To avoid the possibility of missing UI issues type, we examined a statistical representative sample of 384 reviews (with a 95% confidence level and a 5% confidence interval) to double-check the issue types.

Though user reviews are often precise, some complicated reviews may contain multiple intentions. We record all such multiple-label reviews (cf., Table 6), and they account for no more than 3%. Besides, multi-label issues are often related and do not contradict with each other, so it does not influence the coding procedure. Therefore, we believe multiple labels do not influence much on the categorization results.

5.2 Internal Validity

We analyze over 3M reviews of 31,578 apps and manually labeled 1,447 reviews. While manual labeling is a tedious and time-consuming task, we go through the manual analysis process to identify the UI issue types and get detailed insights about the characteristics of the identified issue types. Further work can extend our study by providing approaches for identifying the raised UI issue type in user reviews and extend our work on a larger dataset.

It is possible that the rating changes in the dialogues are influenced by other factors. To exclude such influence, if the review updates do not lead to explicit UI changes, then we do not consider these dialogues. In this way, we focus on the rating changes that explicitly mention UI issues in the text.

In this article, we aim to identify UI-related reviews to perform our empirical study. Thus, our proposed approach is not fully automatic and involves much manual analysis effort. In the future, we plan to extend our study by proposing automated approaches for UI-related review identification.

The results of our manual studies are impacted by the experience of the coders and the amount of the collected data. To reduce the errors, two coders who have experience in analyzing mobile UI design issues participated in the manual analysis process. We calculated the inter-rater agreement using Fleiss Kappa. The agreement results are considered to be substantial for the UI taxonomy and dialogue patterns. Hence, while we put considerable effort into mitigating the bias, the extracted reasons may be biased by our experience and intuition. The summarized taxonomy might not be complete due to a limited number of user reviews, considering that end-users are not professional in UI design usually. To mitigate this threat, we incorporate knowledge of previous work and

professional design guidelines in our study and compare these works with ours in the Related Work section.

5.3 External Validity

The Google Play Store shows only the most recent 500 reviews per app, which means that previously posted reviews or changes in the existing reviews will not be accessible. In our study, we needed to collect as many reviews related to UI as possible to conduct an in-depth study of UI issues. Martin et al. discussed the sampling error in analyzing store data [58]. To minimize the sampling error (i.e., collecting as many reviews as possible), we adjusted our crawler to visit the store every day, and we collected data from the store over four months. Hence, we collected a large dataset of 78M reviews that are sufficient to support our analysis of UI-related issues.

The generalizability of our results is also a threat to validity. Although we have collected and identified a large number of UI-related reviews (i.e., 3.3M reviews), these reviews were sampled from a limited number (i.e., 1,447) of reviews. Though we calculate the sample size with a particular confidence level and a confidence interval, the limited generalizability of the sample might still introduce bias to our experiment results. We provide the first step towards this direction. We encourage future studies to explore these things in a deeper manner through several more in-depth and focused studies on various aspects.

A threat to validity is the sampling strategy. In this article, we collect category-balance numbers of the app, which may not represent the app distribution in real-world. To mitigate such a threat, we collect as many apps as possible (i.e., top 500) to cover most of the mainstream popular apps. Besides, we also consider sub-categories provided by the Google Play Store to follow the app distribution in real-world.

Another threat to validity is that there may exist false positives (i.e., non-UI-related reviews are identified as UI-related reviews) and the false negatives (i.e., UI-related reviews are identified as non-UI-related reviews). To investigate the integrity of the data, we perform another examination with a tighter confidence level and a smaller confidence interval. Specifically, we sample another 1,848 reviews from the retrieved UI-related reviews (with 99% confidence level and 3% confidence interval) and 4,160 reviews from the rest non-UI-related reviews (with 99% confidence level and 2% confidence interval), respectively. We perform a validity check of the sampled reviews and find 1.95% false positives (i.e., 36 out of 1,848 reviews) in the UI-related reviews and 0.65% false negatives (i.e., 27 out of 4,160 reviews) in the non-UI-related reviews. We believe that such proportions of false positives and false negatives do not impact the reliability of the dataset largely. Besides, we use manual analysis in this article, which can also identify and exclude the impact of false positives.

6 RELATED WORK

In this section, we describe the work related to analyzing the characteristics of successful UI and analyzing user reviews.

6.1 Mobile GUI Development

GUI provides a visual bridge between apps and users through which they can interact with each other. Developing the GUI of a mobile app involves two separate but related activities: design the UI and implement the UI. To assist UI implementation, Nguyen and Csallner [69] reverse-engineer the UI screenshots by image-processing techniques. More powerful deep-learning-based algorithms [9, 13, 65] are further proposed to leverage the existing big data of Android apps. Retrieval-based methods [8, 79] are also used to develop user interfaces. Reiss et al. parse developers' sketch into structured queries to search related UIs of Java-based desktop software in the database [79]. GUIfetch [8] customizes Reiss's method [79] into the Android app UI search

by considering the transitions between UIs. Deka et al. use auto-encoder to support UI searching (i.e., search design images and interaction traces) by inputting the rough sketch [18]. To render inspirations to the designer, Chen et al. propose a program-analysis method to generate the storyboard with UI screenshots efficiently, given one app executable file [16]. Linares-Vasquez et al. and Wan et al. proposed approaches to revamp the UI design to reduce the brightness of the displayed components while still maintaining the good look and feel of the design [51, 88]. Swearngin et al. adopt the image-processing method to help designs with converting the mobile UI screenshots into editable files in Photoshop, so designers can take it as a starting point for further customization [85]. Unlike these works, which target the UI implementation, our work lies more in UI design, especially about the GUI design issues.

Testing for UI is also an essential aspect to improve UI quality. Studies related to **Visual GUI Testing (VGT)** aim to test certain visual aspects of a software application's GUI and the underlying functional properties. Moran et al. propose an approach for Android in a tool called **Gvt (Gui Verification sysTem)**, which resolves GUI-related information from both implemented apps and mock-ups and uses computer vision techniques to identify common errors in the implementations of mobile [66]. Alegroth et al. summarize the limitation, challenges, and opportunities of the GUI testing approaches [6]. Then they perform an empirical study to investigate the applicability of automated component-based GUI testing and VGT in the GUITAR tool and a prototype tool called VGT GUITAR. They find that GUITAR is applicable in practice, while VGT GUITAR is not [5]. These works focus on the UI testing or the inconsistency between the UI scratch and the implementation. Different from their works, we study the UI issues from the perspectives of users instead of the perspectives of the UI's scratch implementation and testing.

Many prior works focus on one or several UI issues, including layout [62], typography [90], colors [38, 39], icons [61], gestures [90, 94], in-app advertisements [1, 2], and accessibility [12, 62, 94]. Wang et al. explore the user satisfaction with typography design by mining touch interaction based on a hypothesis that users' touch behaviors in reading can reflect their satisfaction with the typography design [90]. Jahanian et al. explore the associations between the colors and linguistic concepts. They adopt an LDA-based model to infer color-keyword over a corpus of 2,654 magazines [39]. They then present a curated dataset of 21 years of web design in the internet archive concerning the color design's evolution, based on which they propose a deep-learning-based model to predict the year of the web design [38]. Miniukovich et al. review the visual qualities of icons that could make them noticeable and likable and computationally measure the saliency and complexity for 930 icons and link the computed scores to app popularity [61]. Ahasanuzzaman et al. study advertisement integration practices by analyzing 1,837 free-to-download apps of the Google Play Store and summarize four common strategies for integrating multiple advertisement libraries [2]. The accessibility for the disabled and UI usability for diverse people are also concerned [12, 94]. Casadei et al. investigate 18 virtual communities of mobile design and development to identify issues on the accessibility of Android mobile UI design patterns and analyze 127 documents to propose recommendations to improve the accessibility of mobile interfaces [12]. Wong et al. perform a user study to examine the usability of smartphone user interface and mobile apps among 80 older adults. They use four tasks, including "making and retrieving voice calls," "using phone book," "installing a mobile app from Google Play Store," and "using WhatsApp" to evaluate the usability. They find that there is still room to improve mobile UI design issues, especially for the older adults' cohort [94]. Miraz et al. explore the design issues that are specifically relevant for multilingual users. They use principal component analysis to identify five decorrelated components (i.e., text, layout and navigation, readability, graphics, and cross-browser compatibility) in UIs and then use hypothesis tests to assess the components. They find that the English versions of the website have superior usability and there is a need to improve UIs of other

translated versions [62]. In this article, we also discuss the aspects of these UI issues. Different from their work, we summarize a taxonomy of UI issues from the perspectives of the user reviews.

There are works summarizing guidelines that help the app owners build their UI [26, 27, 52, 68]. Neil et al. summarize the UI design patterns according to their experience as a professional designer during practice to help guide junior designers [68]. Liu et al. follow the design rules from Material Design to annotate the mobile GUI design to represent its semantics [52]. Google provides the Android developer documentation [27] and a design system called Material Design [26] for the app owners. They introduce the pre-built UI components in Android [27] and recommend tips of the best practice when constructing the UI [26, 27]. The Android developer documentation emphasizes the UI implementation and recommends tips from the perspectives of the source code in practice. In contrast, we do not investigate the implementation but study users' perception of UI issues and how the app owners interact with users to solve these issues. Material Design provides a popular style (not all styles) that summarizes design patterns to guide app owners to build a consistent UI; hence, it mainly focuses on the appearance and the interaction. In contrast, we do not propose a design system but summarize UI issues in the user reviews. Apart from the *"Appearance"* issue and the *"Interaction"* issue, we also summarize *"Experience"* issues (e.g., *"Advertisement"*), which emerges after the UIs are exposed to real users. Besides, our study focuses on users' complaints. In our user study, a respondent uses Material Design, and he/she also says, *"Understanding and remember your code (i.e., our taxonomy) is important (at least you should know which part to look at when user complaining)."*

Besides, many prior studies categorize the mobile apps' issues in which their works involve UI issues from various perspectives. Ferreira et al. perform a qualitative study of real agile projects involving significant UI design, and they find that the agile iterations can significantly improve the quality of the relationship between UI designers and software developers [20]. Gould et al. propose that mobile devices must map fundamental dimensions of worldwide cultures to components of UIs to cope with global product and service development, and they claim that tools may emerge to facilitate tuning designs per culture [28]. Llanos et al. study the differences between the integrated and separated UIs of games. They claim that there is no necessary connection between a transparent interface (i.e., integrated UI) and involvement, and separated UIs are preferred due to the clear information they present [53]. Man et al. propose a framework called CrossMiner to analyze the essential app issue across different platforms. CrossMiner generates keywords for seven issues, including UI issues, and prioritizes the user reviews corresponding to the issues to help developers gain understanding and design test cases. [56]. Marcus et al. study four elements of UIs regarding communication with human beings, including metaphor (i.e., how well the elements are recognized), mental model (i.e., how well the elements are organized), navigation model (i.e., how fluently a user can move through the UI), and look (i.e., the appearance) [57]. Nayebi et al. study the functionality deletion of the mobile apps by analyzing 213,866 commits from 1,519 open-source Android apps from a total of 14,238 releases in which they report 29.22% deletion of UI elements. They found most deletions are motivated by unneeded functionality (i.e., redundancy issues) [67].

6.2 User Reviews of Mobile Apps

As one important information source for mobile apps, the app reviews have been extensively investigated by researchers. Prior work analyzes user reviews to extract useful information such as complaints and feature requests [23, 35–37, 41–43, 54, 59, 60, 71, 87]. To identify useful and valuable information, Chen et al. proposed **AR-Miner (App Review Miner)**, which uses data mining and ranking techniques and minimal human efforts to extract valuable information (e.g., feature requests) from raw user review [15]. Then Palomba et al. leverage the approach of Chen et al. to filter reviews with little information and proposed CRISTAL, which links user reviews to

the corresponding code changes by utilizing text similarity [72]. Later, Palomba et al. proposed ChangeAdvisor to group user reviews that request similar features and map these reviews to the corresponding source code [73]. Iacob et al. proposed **MARA (Mobile App Review Analyzer)** to identify bug-related and feature-request-related reviews by leveraging linguistic rules [35, 36]. To automatically classify useful feedback contained in user reviews, Panichella et al. proposed **ARdoc (App Reviews Development Orient Classifier)** [74, 75], which combines three techniques: (1) **Natural Language Parsing (NLP)**, (2) Text Analysis, and (3) Sentiment Analysis. Di Sorbo et al. proposed **SURF (Summarizer of User Reviews Feedback)** [19], which is based on Panichella et al.'s approach [74, 75]. Fu et al. propose a system called WisCom that analyzes tens of millions user reviews in mobile app markets to identify reasons why users like or dislike given apps and report the top 10 causes found by the model. However, only 2 causes are related to the UIs, and they do not perform more in-depth study on these causes [23]. Our study differs from prior work, as we are more concerned with specific users' reviews about the UI issues, which is not touched by previous works.

Furthermore, the studies not only utilize review text but also connect the review with updates and responses of reviews. Gao et al. proposed the **IDEA (IDentifying Emerging App issues)** approach that identifies the emerging topics in every update of an app [24]. Hassan et al. studied the app reviews at the update-level, rather than at the app-level, so researchers can study an app at a fine granularity [30]. They also studied the dialogue between app users and developers in the Google Play Store to help developers better perfect apps [32]. Different from them, we analyze the dialogues between the development team and users to understand how app owners can get useful UI-related information to improve the UI.

Many research works aim to build a taxonomy for particular purposes, like helping the developer identify bugs or figure out user requirements. Maalej et al. used several probabilistic techniques to classify app reviews [54] into four types: (1) bug reports, (2) feature requests, (3) user experiences, and (4) unspecified based on the extracted features. Khalid et al. built a more detailed taxonomy [41, 42]. They studied user complaints in mobile apps and identified 13 issue types that were raised in user reviews. Then they surveyed the existing mobile reviews and proposed an approach for presenting and grouping user reviews [44]. Mcilroy et al. improve the taxonomy of issue types identified by Khalid et al. and proposed an approach to classify reviews into the corresponding issue type [44, 60] automatically. Our work differs from prior research, because we are the first to perform an in-depth analysis of app reviews for UI designers rather than developers. In particular, we focus our study on UI-related reviews and study how users perceive the UI and how the reviews can help UI designers.

7 CONCLUSION

In this article, we conduct an empirical study, analyzing more than 3M UI-related reviews from 22,199 free-to-download apps and 9,380 non-free apps in the Google Play Store and summarize how users complain about the UI. Our findings highlight that the interactions and collaborations between app owners and users can help the app get better UI and boost users' satisfactoriness about the updated UI. The most important findings of our study are:

(1) The mean rating of the apps with UI-related reviews is lower than the rating of the apps with all reviews, which reflects the severity of UI issues.
(2) We identify 17 issue types that belong to four main categories: *"Appearance," "Interaction," "Experience,"* and *"Others."* We find that reviews about UI issues of the free-to-download apps are more negative and more generic than the reviews of the non-free apps.

(3) We identify eight dialogue patterns that app owners interact with users via the review-response mechanism in which the pattern *"Apology or Appreciation"* is the most frequency response pattern.
(4) We find that improvement of UI can be achieved by active interaction with users, as we observe that there is a larger percentage of rating increases before and after the responses to the user reviews.
(5) We find app owners can satisfy users by promising and timely fulfilling the requested features and can also fix issues and satisfy users without UI updates by proactive communication.

Our study is useful for app owners who wish to evolve their UI and satisfy users. In particular, these app owners can leverage our taxonomy as a checklist and our derived patterns as a reference to improve UI and interact with users.

In the future, we would like to extend our works both vertically and horizontally. First, based on the current work, which focuses on the broad category, we will extract more fine-grained UI issues from the reviews and organize them into a better structure to assist designers. Second, apart from the UI issues, we will also customize our approach to other issues such as app performance and app functionalities.

REFERENCES

[1] Md Ahasanuzzaman, Safwat Hassan, Cor-Paul Bezemer, and Ahmed E. Hassan. 2020. A longitudinal study of popular ad libraries in the Google Play Store. *Empir. Softw. Eng.* 25, 1 (Jan. 2020), 824–858. DOI: https://doi.org/10.1007/s10664-019-09766-x.
[2] Md Ahasanuzzaman, Safwat Hassan, and Ahmed E. Hassan. 2020. Studying Ad library integration strategies of top free-to-download apps. *IEEE Trans. Softw. Eng.* (2020), 1–1. DOI: https://doi.org/10.1109/TSE.2020.2983399.
[3] Akdeniz. 2018. Google Play Crawler. Retrieved from: https://github.com/Akdeniz/google-play-crawler.
[4] Afnan AlSubaihin, Federica Sarro, Sue Black, Licia Capra, and Mark Harman. 2019. App store effects on software engineering practices. *IEEE Trans. Softw. Eng.* 47, 2 (2019). https://ieeexplore.ieee.org/abstract/document/8606261.
[5] Emil Alégroth, Robert Feldt, and Lisa Ryrholm. 2015. Visual GUI testing in practice: Challenges, problems and limitations. *Empir. Softw. Eng.* 20, 3 (2015), 694–744. Springer.
[6] Emil Alégroth, Zebao Gao, Rafael Oliveira, and Atif Memon. 2015. Conceptualization and evaluation of component-based testing unified with visual GUI testing: An empirical study. In *Proceedings of the IEEE 8th International Conference on Software Testing, Verification and Validation (ICST'15)*. IEEE, 1–10.
[7] App Annie. 2018. The App Analytics and App Data Industry Standard: Google Play Store. Retrieved from: https://www.appannie.com/cn/.
[8] Farnaz Behrang, Steven P. Reiss, and Alessandro Orso. 2018. GUIfetch: Supporting app design and development through GUI search. In *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*. ACM, 236–246.
[9] Tony Beltramelli. 2018. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, 3.
[10] Richard E. Boyatzis. 1998. *Transforming Qualitative Information: Thematic Analysis and Code Development*. Sage.
[11] Robert Bringhurst. 2004. *The Elements of Typographic Style*. Hartley & Marks, Point Roberts, WA.
[12] Vitor Casadei, Toni Granollers, and Luciana Zaina. 2017. Investigating accessibility issues of UI mobile design patterns in online communities: A virtual ethnographic study. In *Proceedings of the Brazilian Symposium on Human Factors in Computing Systems*. 1–10.
[13] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI design image to GUI skeleton: A neural machine translator to bootstrap mobile GUI implementation. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, 665–676.
[14] Jiachi Chen, Xin Xia, David Lo, John Grundy, Xiapu Luo, and Ting Chen. 2020. Defining smart contract defects on ethereum. *IEEE Trans. Softw. Eng.* (2020). https://ieeexplore.ieee.org/abstract/document/9072659.
[15] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 767–778.

[16] Sen Chen, Lingling Fan, Chunyang Chen, Ting Su, Wenhe Li, Yang Liu, and Lihua Xu. 2019. StoryDroid: Automated generation of storyboard for Android apps. *arXiv preprint arXiv:1902.00476* (2019).
[17] Juliet Corbin and Anselm Strauss. 2008. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage, Thousand Oaks, CA.
[18] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th ACM Symposium on User Interface Software and Technology*. ACM, 845–854.
[19] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2016. What would users change in my app? Summarizing app reviews for recommending software changes. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'16)*. ACM Press, Seattle, WA, 499–510. Retrieved from: http://dl.acm.org/citation.cfm?doid=2950290.2950299.
[20] Jennifer Ferreira, James Noble, and Robert Biddle. 2007. Agile development iterations and UI design. In *Proceedings of the Agile Development Conference (AGILE'07)*. IEEE, 50–58.
[21] Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters.*Psychol. Bull.* 76, 5 (1971), 378.
[22] Thomas Fritz and Gail C. Murphy. 2010. Using information fragments to answer the questions developers ask. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*. ACM, 175–184.
[23] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh. 2013. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1276–1284.
[24] Cuiyun Gao, Jichuan Zeng, Michael R. Lyu, and Irwin King. 2018. Online app review analysis for identifying emerging issues. In *Proceedings of the IEEE/ACM 40th International Conference on Software Engineering (ICSE'18)*. IEEE, 48–58.
[25] Edmund A. Gehan. 1965. A generalized Wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika* 52, 1-2 (1965), 203–224.
[26] Google. 2018. Material Design. Retrieved from: https://material.io/.
[27] Google. 2020. User Interface & Navigation | Android Developers. Retrieved from: https://developer.android.com/guide/topics/ui.
[28] Emilie W. Gould, Pia Honold, Masaaki Kurosu, Jay Melican, Aaron Marcus, and Li Anne Yu. 2003. Culture issues and mobile UI design. In *CHI'03 Extended Abstracts on Human Factors in Computing Systems*. ACM, 702–703.
[29] Jerzy Grobelny and Rafal Michalski. 2015. The role of background color, interletter spacing, and font size on preferences in the digital presentation of a product. *Comput. Hum. Behav.* 43 (2015), 85–100. Elsevier.
[30] Safwat Hassan, Cor-Paul Bezemer, and Ahmed E. Hassan. 2018. Studying bad updates of top free-to-download apps in the Google Play Store. *IEEE Trans. Softw. Eng.* 46, 7 (2018). https://ieeexplore.ieee.org/abstract/document/8457304.
[31] Safwat Hassan, Weiyi Shang, and Ahmed E. Hassan. 2017. An empirical study of emergency updates for top Android mobile apps. *Empir. Softw. Eng.* 22, 1 (2017), 505–546.
[32] Safwat Hassan, Chakkrit Tantithamthavorn, Cor-Paul Bezemer, and Ahmed E. Hassan. 2018. EMSE2018-studying the dialogue between users and developers of free apps in the Google Play Store. *Empir. Softw. Eng.* 23, 3 (June 2018), 1275–1312. Retrieved from: http://link.springer.com/10.1007/s10664-017-9538-9.
[33] Shawn Lawton Henry, Shadi Abou-Zahra, and Judy Brewer. 2014. The role of accessibility in a universal web. In *Proceedings of the 11th Web for All Conference*. ACM, 17.
[34] Dennis E. Hinkle, William Wiersma, and Stephen G. Jurs. 2003. *Applied Statistics for the Behavioral Sciences*. Vol. 663. Houghton Mifflin College Division.
[35] Claudia Iacob and Rachel Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 41–44.
[36] Claudia Iacob, Rachel Harrison, and Shamal Faily. 2013. Online reviews as first class artifacts in mobile app development. In *Proceedings of the International Conference on Mobile Computing, Applications, and Services*. Springer, 47–53.
[37] Claudia Iacob, Varsha Veerappa, and Rachel Harrison. 2013. What are you complaining about?: A study of online reviews of mobile applications. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*. British Computer Society, 29.
[38] Ali Jahanian, Phillip Isola, and Donglai Wei. 2017. Mining visual evolution in 21 years of web design. In *Proceedings of the CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2676–2682.
[39] Ali Jahanian, Shaiyan Keshvari, S. V. N. Vishwanathan, and Jan P. Allebach. 2017. Colors—Messengers of concepts: Visual design mining for learning color semantics. *ACM Trans. Comput.-human Interact.* 24, 1 (2017), 2.
[40] Bernard J. Jansen. 1998. The graphical user interface. *ACM SIGCHI Bull.* 30, 2 (1998), 22–26.
[41] Hammad Khalid. 2013. On identifying user complaints of iOS apps. In *Proceedings of the International Conference on Software Engineering*. IEEE Press, 1474–1476.

[42] Hammad Khalid, Meiyappan Nagappan, Emad Shihab, and Ahmed E. Hassan. 2014. Prioritizing the devices to test your app on: A case study of Android game apps. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 610–620.

[43] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E. Hassan. 2015. What do mobile app users complain about?*IEEE Softw.* 32, 3 (2015), 70–77.

[44] Mubasher Khalid, Muhammad Asif, and Usman Shehzaib. 2015. Towards improving the quality of mobile app reviews. *Int. J. Inf. Technol. Comput. Sci.* 7, 10 (2015), 35.

[45] Barbara A. Kitchenham and Shari L. Pfleeger. 2008. Personal opinion surveys. In *Guide to Advanced Empir. Softw. Eng.* Springer, 63–92.

[46] Andrew J. Ko, Robert DeLine, and Gina Venolia. 2007. Information needs in collocated software development teams. In *Proceedings of the 29th International Conference on Software Engineering*. IEEE Computer Society, 344–353.

[47] Elmar Krainz, Klaus Miesenberger, and Johannes Feiner. 2018. Can we improve app accessibility with advanced development methods? In *Proceedings of the International Conference on Computers Helping People with Special Needs*. Springer, 64–70.

[48] Ding Li, Shuai Hao, Jiaping Gui, and William G. J. Halfond. 2014. An empirical study of the energy consumption of android applications. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*. IEEE, 121–130.

[49] Ding Li, Yingjun Lyu, Jiaping Gui, and William G. J. Halfond. 2016. Automated energy optimization of http requests for mobile applications. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 249–260.

[50] Dayi Lin, Cor-Paul Bezemer, Ying Zou, and Ahmed E. Hassan. 2019. An empirical study of game reviews on the Steam platform. *Empir. Softw. Eng.* 24, 1 (2019), 170–207.

[51] Mario Linares-Vásquez, Gabriele Bavota, Carlos Eduardo Bernal Cárdenas, Rocco Oliveto, Massimiliano Di Penta, and Denys Poshyvanyk. 2015. Optimizing energy consumption of GUIs in Android apps: A multi-objective approach. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*. ACM, 143–154.

[52] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning design semantics for mobile apps. In *Proceedings of the 31st ACM Symposium on User Interface Software and Technology*. ACM, 569–579.

[53] Stein C. Llanos and Kristine Jürgensen. 2011. Do players prefer integrated user interfaces? A qualitative study of game UI design issues. In *Proceedings of the DiGRA Conference: Think Design Play*.

[54] Walid Maalej and Hadeer Nabil. 2015. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *Proceedings of the IEEE 23rd International Requirements Engineering Conference (RE'15)*. IEEE, 116–125.

[55] Walid Maalej and Martin P. Robillard. 2013. Patterns of knowledge in API reference documentation. *IEEE Trans. Softw. Eng.* 39, 9 (2013), 1264–1282.

[56] Yichuan Man, Cuiyun Gao, Michael R. Lyu, and Jiuchun Jiang. 2016. Experience report: Understanding cross-platform app issues from user reviews. In *Proceedings of the IEEE 27th International Symposium on Software Reliability Engineering (ISSRE'16)*. IEEE, Ottawa, ON, Canada, 138–149. DOI: https://doi.org/10.1109/ISSRE.2016.27.

[57] Aaron Marcus. 1993. Human communications issues in advanced UIs. *Commun. ACM* 36, 4 (1993), 100–109. ACM New York, NY.

[58] William Martin, Mark Harman, Yue Jia, Federica Sarro, and Yuanyuan Zhang. 2015. The app sampling problem for app store mining. In *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 123–133.

[59] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. 2017. A survey of app store analysis for software engineering. *IEEE Trans. Softw. Eng.* 43, 9 (2017), 817–847.

[60] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. 2016. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empir. Softw. Eng.* 21, 3 (2016), 1067–1106.

[61] Aliaksei Miniukovich and Antonella De Angeli. 2016. Pick me!: Getting noticed on Google Play. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 4622–4633.

[62] Mahdi H. Miraz, Peter S. Excell, and Maaruf Ali. 2016. User interface (UI) design issues for multilingual users: A case study. *Univ. Access Inf. Soc.* 15, 3 (2016), 431–444. Springer.

[63] MoPub. 2019. Interstitial Ads Best Practices | MoPub Publisher Best Practices | MoPub Developers. Retrieved from: https://developers.mopub.com/publishers/best-practices/interstitial-ads/.

[64] MoPub. 2019. Native ads Best Practices for App Publishers | MoPub Blog. Retrieved from: https://www.mopub.com/2019/06/28/native-advertising-best-practices.

[65] Kevin Moran, Carlos Bernal-Cárdenas, Michael Curcio, Richard Bonett, and Denys Poshyvanyk. 2018. Machine learning-based prototyping of graphical user interfaces for Mobile apps. *arXiv preprint arXiv:1802.02312* (2018).

[66] Kevin Moran, Boyang Li, Carlos Bernal-Cárdenas, Dan Jelf, and Denys Poshyvanyk. 2018. Automated reporting of GUI design violations for mobile apps. In *Proceedings of the 40th International Conference on Software Engineering*. 165–175.

[67] Maleknaz Nayebi, Konstantin Kuznetsov, Paul Chen, Andreas Zeller, and Guenther Ruhe. 2018. Anatomy of functionality deletion: An exploratory study on mobile apps. In *Proceedings of the 15th International Conference on Mining Software Repositories*. 243–253.

[68] Theresa Neil. 2014. *Mobile Design Pattern Gallery: UI Patterns for Smartphone Apps*. O'Reilly Media, Inc.

[69] Tuan Anh Nguyen and Christoph Csallner. 2015. Reverse engineering mobile application user interfaces with REMAUI (T). In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE'15)*. IEEE, 248–259.

[70] Ehsan Noei, Daniel Alencar Da Costa, and Ying Zou. 2018. Winning the app production rally. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18)*. ACM Press, 283–294. Retrieved from: http://dl.acm.org/citation.cfm?doid=3236024.3236044.

[71] Jeungmin Oh, Daehoon Kim, Uichin Lee, Jae-Gil Lee, and Junehwa Song. 2013. Facilitating developer-user interactions with mobile app review digests. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM, 1809–1814.

[72] Fabio Palomba, Mario Linares-Vasquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. 2015. User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME'15)*. IEEE, 291–300.

[73] Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. 2017. Recommending and localizing change requests for mobile apps based on user reviews. In *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 106–117.

[74] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME'15)*. IEEE, 281–290.

[75] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2016. ARdoc: App reviews development oriented classifier. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 1023–1027.

[76] Gustavo Pinto, Fernando Castor, and Yu David Liu. 2014. Mining questions about software energy consumption. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 22–31.

[77] Martin Poschenrieder. 2015. 77% will Not Download a Retail App Rated Lower Than 3 Stars.Retrieved from: http://blog.testmunk.com/77-will-not-download-a-retail-app-rated-lower-than-3-stars/.

[78] Yogesh Chunilal Rathod. 2011. Method and System for Customized, Contextual, Dynamic and Unified Communication, Zero Click Advertisement and Prospective Customers Search Engine. https://patents.google.com/patent/US20110208822A1/en.

[79] Steven P. Reiss, Yun Miao, and Qi Xin. 2018. Seeking the user interface. *Autom. Softw. Eng.* 25, 1 (2018), 157–193.

[80] Jeanine Romano, Jeffrey D. Kromrey, Jesse Coraggio, Jeff Skowronek, and Linda Devine. 2006. Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen's d indices the most appropriate choices? In *Proceedings of the Meeting of the Southern Association for Institutional Research*. Citeseer.

[81] Kathryn Roulston. 2001. Data analysis and "theorizing as ideology." *Qualit. Res.* 1, 3 (2001), 279–302.

[82] Israel J. Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Thorsten Berger, Steffen Dienst, and Ahmed E. Hassan. 2015. Examining the rating system used in mobile-app stores. *IEEE Softw.* 33, 6 (2015), 86–92. IEEE.

[83] Walt Scacchi. 2002. Process models in software engineering. *Encycl. Softw. Eng.* (2002). https://onlinelibrary.wiley.com/doi/abs/10.1002/0471028959.sof250.

[84] Carolyn B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Trans. Softw. Eng.*4 (1999), 557–572.

[85] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Andrew J. Ko. 2018. Rewire: Interface design assistance from examples. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 504.

[86] Christopher Vendome, Diana Solano, Santiago Liñán, and Mario Linares-Vásquez. 2019. Can everyone use my app? An empirical study on accessibility in Android apps. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME'19)*. IEEE, 41–52.

[87] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 14–24.

[88] Mian Wan, Yuchen Jin, Ding Li, Jiaping Gui, Sonal Mahajan, and William G. J. Halfond. 2017. Detecting display energy hotspots in Android apps. *Softw. Test., Verif. Reliab.* 27, 6 (2017), e1635.

[89] Zhiyuan Wan, Xin Xia, Ahmed E. Hassan, David Lo, Jianwei Yin, and Xiaohu Yang. 2018. Perceptions, expectations, and challenges in defect prediction. *IEEE Trans. Softw. Eng.* 46, 11 (2018). https://ieeexplore.ieee.org/abstract/document/8502824.

[90] Junxiang Wang, Jianwei Yin, Shuiguang Deng, Ying Li, Calton Pu, Yan Tang, and Zhiling Luo. 2018. Evaluating user satisfaction with typography designs via mining touch interaction data in mobile reading. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 113.

[91] Yujun Wen, Hui Yuan, and Pengzhou Zhang. 2016. Research on keyword extraction based on word2vec weighted textrank. In *Proceedings of the 2nd IEEE International Conference on Computer and Communications (ICCC'16)*. IEEE, 2109–2113.

[92] Bernard E. Whitley, Mary E. Kite, and Heather L. Adams. 2013. *Principles of Research in Behavioral Science*. Routledge.

[93] WB WHO. 2011. World report on disability. *Geneva: WHO* (2011). https://www.who.int/disabilities/world_report/2011/chapter2.pdf.

[94] Chui Yin Wong, Rahimah Ibrahim, Tengku Aizan Hamid, and Evi Indriasari Mansor. 2018. Usability and design issues of smartphone user interface and mobile apps for older adults. In *Proceedings of the International Conference on User Science and Engineering*. Springer, 93–104.

[95] Xin Xia, Zhiyuan Wan, Pavneet Singh Kochhar, and David Lo. 2019. How practitioners perceive coding proficiency. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering (ICSE'19)*. IEEE, 924–935.

[96] Yeliz Yesilada, Giorgio Brajnik, and Simon Harper. 2011. Barriers common to mobile and disabled web users. *Interact. Comput.* 23, 5 (2011), 525–542.

Received May 2020; revised January 2021; accepted January 2021