Automatic App Review Using Android Application

HISHAM*[1], KUNA VIDYA SAGAR*[2], J. JABIR*[3], Mrs. L. Dharani *[4], Dr. V. Sai Shanmuga Raja*[5], and Dr. G. Gunasekaran*[6].

Mail id: aswaakh2012@gmail.com, sagarkuna9001@gmail.com, jabir.05.2002@gmail.com, dharanil.cse@drmgrdu.ac.in, saishanmugaraja@gmail.com, gunasekaran.cse@drmgrdu.ac.in

*[1,2,3] IV Year B. Tech Students, Dept of Computer science and Engineering, *[4,5,6]Professors, Dept of CSE DR. M.G.R EDUCATIONAL AND RESEARCH INSTITUTE, Maduravoyal, Chennai-95, Tamil Nadu, India

***Abstract*— DeepMatcher comes with a tool that connects usage data from app analytics to error reports in the monitored data. DeepMatcher uses advanced techniques to automate this process, enabling developers to quickly spot bugs and dynamically improve bug reporting. In this Paper we have got Simple Mail Transfer protocol (SMTP) which can send and receive Mail to concerned app developers for the improvement in app experience. As our abstract aims to make a standalone app that can make the user give feedback directly to developer so that the app satisfies the user as NLP required in DeepMatcher to report the SMTP used is simplified and make the developer easily rectify the issue with regards to this research we propose DeepMatcher, a tool to automate bug detection and enhance bug reporting in app development .We contrast our solution to an alternative, SMTP (Simple Mail Transfer Protocol), featuring distinct advantages. Whilst DeepMatcher illustrates automation, SMTP provides user-to-developer direct and immediate communication channels, simplifying feedback submission and bug reports. This direct approach provides timely and detailed feedback, faster bug resolution and higher app quality resulting in improved user satisfaction.**

Keywords*— *SMTP, CoRe, Standalone, Feedback, Developer, Generated

1. INTRODUCTION

When it comes to creating apps, the smooth incorporation of user feedback is highly valuable in terms of ensuring user satisfaction and improving the software quality. DeepMatcher has usually been used to solve this problem of user feedback obtained through app reviews and bug reports stored in issue trackers. DeepMatcher uses the latest developments in the field to programmatically carry out issues matching from user reviews to those in bug reports, thus making it easy for developers to quickly identify and fix issues in software.

Nevertheless, our investigation instigates a paradigm change by offering an alternative path for user feedback integration, stressing the exploitation of Simple Mail Transfer Protocol (SMTP) as direct communication line between users and developers. In contrast to DeepMatcher that depends on the automatic algorithms to interpret user feedback, the SMTP approach provides a convenient way of expressing one's concerns and suggestions in the form of emails to developers.

Feedback loop augmented with the SMTP allows users to make their voice heard on the matter they are concerned with in a more personal way thus, engaging them directly to the development. This direct line of communication allows the developers to get immediate and thorough feedback which enables them to spot

bugs in a more timely manner and make the bug fixing be more prompt. In addition, the simplicity of SMTP presents less of a barrier for users to provide feedback which would lead to a greater volume and improved quality of inputs solicited.

This article investigates the comparative advantages of SMTP over traditional automated tools like DeepMatcher in integrating user feedback in app development. We show how SMTP allows feedback collection with a more direct and immediate approach, enabling developers to speedily and user-focused enhance the app experience. Through evaluation and case studies we illustrate the power of SMTP as an essential instrument to improve user appreciation and software quality when developing an application.

Moreover, expanding the framework for user feedback integration to utilize SMTP grants developers a channel for a more personalized and interactive relationship with their user-base. Unlike the automated system like DeepMatcher which creates a sense of disconnect between users and developers because of the use on algorithmic matching, in SMTP there are created direct communication lines that encourage user interaction and collaboration. This direct interaction not only improves user satisfaction but also creates a sense of ownership and loyalty among the users; they see their feedback being valued by the team of developers. Therefore, the incorporation of SMTP is not merely a technical upgrade but a strategic move making the user a co-partner in the app development process aiming to enhance the user engagement and evolve the app development practices.

2. EXISTING SYSTEM

The system, described in the paper, is the existing one which is centered around the creation of DeepMatcher - a solution created to address the issue of feedback integration into the

bug fixing process in app development. The traditional model gives developers a headache because of the required manual analysis of thousands of app reviews that are submitted by the users every day, a task that is further complicated by the widespread use of minor expressions like “I love this app” or “I hate it”.

For this purpose, DeepMatcher automatically extracts problem reports from user reviews and matches them with the bug reports existing in the development team's issue tracker. The proposed automated approach greatly relieves the pain of manual analysis giving developers the opportunity to easily find and fix software problems.

DeepMatcher works in a manner where they first filter problem reports from user reviews predominantly. This is done exploiting recently done related work for this particular purpose. Upon completion of the manual validation of the filtered reports, DeepMatcher employes context-sensitive embeddings to obtain text representations which cosine similarity is used to find potential matches between problem reports and bug reports.

The performance of DeepMatcher showcases its efficiency in finding relevant matches as evidenced by 167 out of 200 issue reports being able to be matched with bug reports. Further analysis reveals cases where DeepMatcher missed potential candidates, or no bug report are available. Sequentially, one can say that the presented results show that DeepMatcher can help developers find bugs earlier, improve bug reports with user feedback, and refine the techniques for detecting duplicate or similar bugs.

3. PROPOSED SYSTEM

Considering the issues mentioned in the abstract, we put forward a standalone app that serves as a single point that users can use to input feedback, and report bugs directly to the app developers. Our application is going to bring in Simple Mail Transfer Protocol (SMTP) that is going to consolidate user and developer communication, thus streamlining the reporting and feedback gathering process.

Our proposed app shall include a user-interface which will be user-friendly and will list the top apps along with the developer’s contact information which can be reached via email. The users may either encounter an issue or wish to provide feedback; such cases can be handled by selecting the app from the app and a feedback form shall prompt with the area to describe the problem or provide suggestions.

On submission, the feedback will be automatically routed to the email address `Supplied` of the respective app developer via SMTP. This two-way communication channel allows developers to get immediate and detailed responses from users on which they can act immediately and improve the overall experience the app gives.

With the utilization of SMTP within the proposed app, we plan to simplify the process of bug reporting and feedback collection, thereby promoting a more collaborative relationship between users and developers. This method does not only enable users to participate in the improvement of their preferred apps but also speeds up the bug fixes and iterative improvements which results in better user experience and app quality. Here are the advantages.

- Streamlined Communication
- User-Friendly Interface
- Immediate Feedback
- Enhanced Collaboration
- Faster Bug Fixes and Iterative Improvements
- Improved User Experience

3.1 SYSTEM ARCHITECTURE

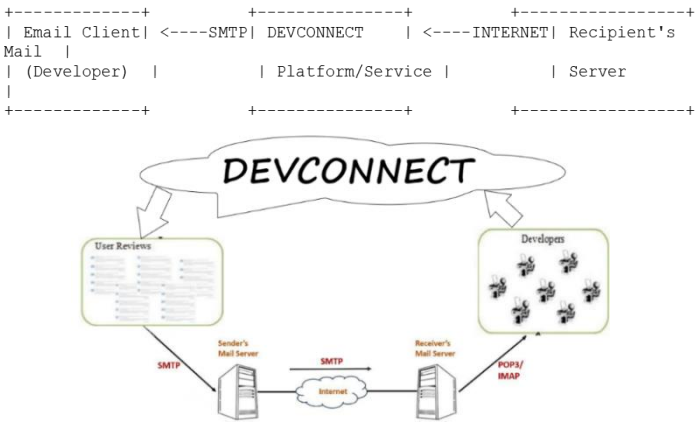

Fig 3 Devconnect app integration with SMTP

In Fig 3 it describes the implementation of the architecture, developers have complete control over submitting requests and providing recommended adjustments for the email client. Utilizing the SMTP protocol, the client transmits the email to DEVCONNECT where any feedback or issue reports are handled, before delivering the message to the recipient's mail server via SMTP. This seamless system allows developers to seamlessly give feedback and report bugs within the email client itself, streamlining the user experience and simplifying the feedback and bug reporting process with this email of the user is registered in the app and while the email is sent for feedback on app the developer gets the email received and the dev can knows who have sent the mail. The following are the integration with SMTP,

1. Email Client: This is the app where developers can leave feedback as well as report issues. It can be a custom-built app or a third-party app like Outlook/Gmail

2. DEVCONNECT: The platform or the service is integrated with the email client. It gives a developer-friendly interface for submitting feedback or reporting bugs.

3. SMTP: This protocol is germane to the email clients and DEVCONNECT in email sending to recipients.

4. RELATED WORK

These are the works that can be done with SMTP,
Bug Tracking Systems: Many existing bug tracking systems e.g., Jira, Bugzilla, and GitHub Issues aid bug reporting and tracking within development teams. Also, such systems are mostly designed for internal usage, if intended for users to submit bugs they might not provide a user-friendly interface.

Feedback Collection Apps: There are some other feedback collecting apps out there like UserVoice, SurveyMonkey, and Gather from Zendesk which provide platforms for gathering user feedback. Although these tools are good in gathering feedback, they do not have integration with functionalities for bug reporting and direct communication channels with developers.

Email-Based Feedback Systems: Certain applications and services have feedback systems based on emails whereby users can email developers their feedback directly. However, this approach can be bulky, since is not automated and lacks response immediacy, having also chances of feedback loss.

In-App Feedback Tools: Most mobile applications are equipped with integrating in-app feedback tools that enable the users to provide feedback right within the app interface. For instance: Instabug, Appsee, Apptentive. Although these tools ease user operations, they might fail to smoothly synchronize with developer communication channels such as email.

Research on User Feedback and Bug Reporting: Research from academia has investigated the multifarious facets of users feedback and bug reporting in software development. Recent works focus on the examination of alternative feedback collection methods, effect of user feedback on software quality and issues of managing and prioritizing bug reports during development.

5. METHODOLOGY DESCRIPTION

Research Design: An integrated approach using mixed methods mixes qualitative methods (the survey, interview) for assessing user needs and tastes, and quantitative methods (such as the analysis of the app usage data) for app evaluation.

Participants: The study includes participants with different backgrounds in terms of app users and developers from among various demographics and developer communities.

Data Collection: Data is collected via surveys to capture user feedback and preferences, interviews to gain better insight and automatic tracking of app usage to measure effectiveness over time.

Instruments/Tools: The instruments and interview protocols of surveys are created to get a complete picture of users' experiences as well as developers' perceptions. App usage tracking mechanisms are embedded in the app under consideration.

Procedures: Participants are sourced online via platforms and developer communities. Data collection includes sending surveys electronically, conducting either in-person or video conferencing interviews, and additionally collecting app usage data.

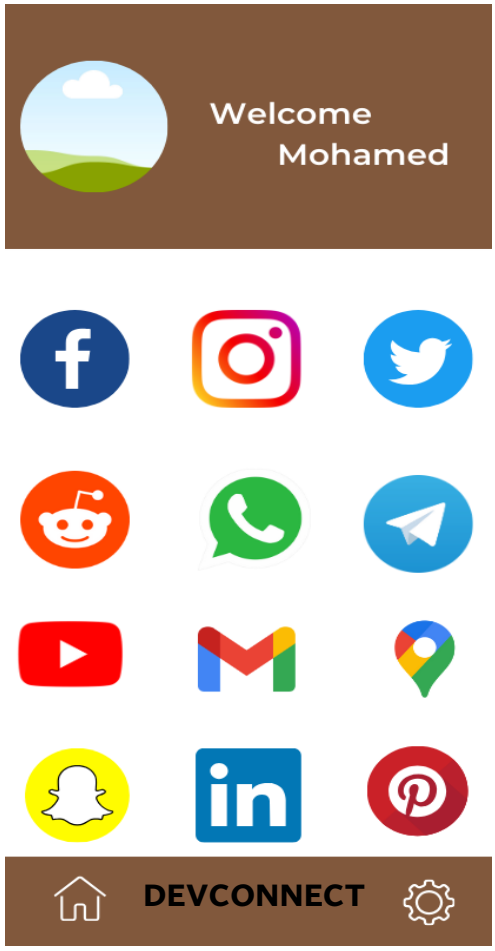

Fig 1 Architecture of proposed System

App logo: the logos of the app are visible on Fig **1**. The logo of the app is its primary visual component that helps users recognize the app quickly. "

Rating icons: The image may depict various icons, i.e., stars represent different ratings, which give users the possibility to rate their experience with the app. Users can simply tap on these icons to give their ratings.

Feedback form: The image can also have a text box or a feedback form for the users to write a message to the app's developer directly. This is the avenue for the users to give constructive criticism on behalf of bugs or with them being able to talk about new features and functionality of the app.

Navigation buttons: The image might have navigation buttons allowing users to move between different pages/sections of the app. These buttons may have labels including "Back," "Home," or "Next" enabling users to navigate the app easily.

App categories: This image could also contain icons or labels representing the app categories, for example, "Games," "social media," or "Productivity". These categories could help users to find the app easier in app stores, or on the developer's

website.

Social proof: The image could feature social proof elements e.g. awards or badges, indicating the app's popularity and/or awards it has won. These features promote trust with users and also motivate them to download and use the app.

Call-to-action: In conclusion, the photo is expected to feature a prominent CTA inviting users to rate and review the app or linking to the app store where they can download it. The CTA should stand out and be eye catching to motivate users to take action.

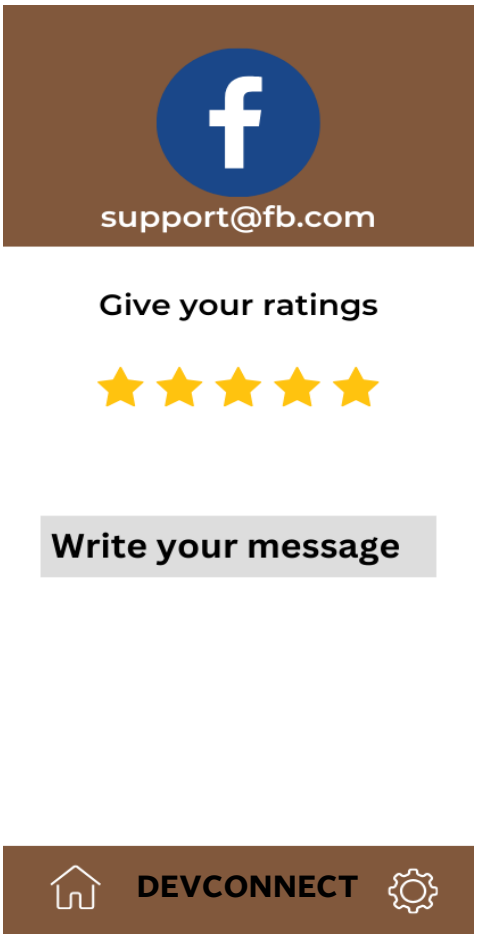

Fig 2 Ratings and Feedback

Open the app: The user can start the app on any of their devices.

Navigate to the feedback section: The user can go to the feedback or support side of the application, which is likely in the settings or help menu.

Provide ratings: The user can then enter their ratings by selecting the right number of stars or through any other rating system that is included within the app.

Write a message: The user can type the message in the feedback form, or the text box made available within the application. They can also voice their ideas, views, or proposals about the app or service.

Send the feedback: The user can give the feedback by clicking on the "Submit" or "Send" button which are available within the app. The app will then send the email via SMTP to the appropriate app developer**.**

6. TECHNOLOGY USED FOR APP RESPONSE

Standalone App: Create a separate app where the users can submit feedback and bugs directly to developers,

Simple Mail Transfer Protocol (SMTP): Seamlessly incorporate SMTP to facilitate communication between users and developers, ensuring they receive timely feedback.

User-Friendly Interface: Introduces and easy to use UI that provides top apps with their respective developer contact information, for easy management of user feedback submission.

Automatic Routing: Automate the feedback's direct routing to developers' mailboxes which should be taken care of as soon as possible.

Two-Way Communication: Bring in users' direct responses towards user-user collaboration and faster bugs outcomes.

Advantages:

Streamlined Communication: Make feedback process straightforward for users and developers.

User-Friendly Interface: Humanize the interface with an easy UI.

Immediate Feedback: Launch prompt responses and actions implemented off user feedback.

Enhanced Collaboration: Develop a user-developer cooperation relationship.

Faster Bug Fixes: Accelerate bug fixing and iterative enhancements.

Improved User Experience: Lead to the improvement of the app quality and user satisfaction.

Enhancing Bug Reports with User Feedback

Detailed Descriptions: Urge the users to give a detailed description of the problem that occurred, how to reproduce it as well as the error messages seen. This helps developers find the problem better and assists speedy resolution.

User Context: Allow users to supply contextual data such device type, OS version, app version and network condition. This the can the contextual data help in identifying potential causes for the bug and rank them in terms of priority.

Screenshots and Screen Recordings**: Let users attach screenshots or record screen videos illustrating the problem. Visual aids can provide extremely helpful context and facilitate our seeing the problem better.

User Feedback Integration**: Embed user feedback channels straight into the bug reporting process. Permit users to raise general suggestions or bug reports so that developers will

work on usability issues or feature requests simultaneously.

Feedback Prioritization: In place of user voting, introduce mechanisms that allow them to prioritize reported bugs depending on their severity or impact to their experience. This helps developers to prioritize critical issues first and response to user concerns quickly.

Two-Way Communication: Establish a communication channel between users and developers to clarify or reopen the cases of reported bugs. This enables collaboration and developers can get more details if necessary.

Feedback Analysis: Analyze user feedback and bug reports methodically and search for the patterns and problems frequently occurring among these inputs. This can also guide quality assurance activities and determine which bugs to take care of first according to user impact and rate of occurrence.

Feedback Acknowledgment: Act on the user comments in a timely manner to assure the users that their reports have been taken notice of and are being acted upon. Keeping the users informed of the current state of the reported bugs will maintain transparency and trust.

Thanks to user feedback on bugs, developers can improve their troubleshooting skills, prioritize bug fixes accurately and end up with a better product overall.

7. IMPLEMENTATION

App Development:

- Design and develop the single- purpose application with the convenient user interface.
- Provide functions to specify top apps and developer contact.
- Implement a feedback form for users to report problems or give ideas.

SMTP Integration:

- Add SMTP to the app to automate email communication.
- Set up SMTP servers and configure the app to send emails to developers email addresses.

Feedback Routing:

- Set up the app to send the feedback submissions to the email addresses of respective app developers.
- Make sure that feedback submissions include such details as app name, user details and issue description.

Two-Way Communication:

- Allow developers to respond to feedback directly through email, thus, setting up a two-way communication channel.
- Include the implementation of a system that notifies users of developer responses in the app.

Testing and Quality Assurance:

- Conduct comprehensive testing of the app to guarantee functionality and usability.
- Verify email delivery and routing via SMTP.
- Resolve all the bugs and/or problems discovered during testing.

Deployment:

- Deploy your app to the applicable app stores or distribution platforms for your users to get the app and install it.
- Make sure that proper documentation and instructions are given to users for feedback submission.

User Engagement and Support:

- Publish the app to users and get them to send comments and bug reports.
- The support of the user and help should be provided for any problems or questions connected to the app.

Feedback Analysis and Iterative Improvements:

- Repeatedly analyze feedback from users and prioritize bug fixes and improvements according to the input of users.
- Updates release to the app address reported issues and improve user experience.

Monitoring and Maintenance:

- Deploy performance monitoring such as submission rate and response time.
- Resolve user or technical concerns by regular updates and maintenance.

7.1 Challenges and Constraints

Technical Complexity: To implement SMTP into the app and to set up a mail server may be technically difficult, in particular, for those novice developers when it comes to the email communication protocols.

Email Deliverability: Making sure emails coming from the application are received at developers email addresses without being flagged as spam or blocked by email filters can prove to be a Challenge which affects the reliability of feedback routing.

User Engagement: The task of convincing users to provide feedback and report bugs may prove to be difficult because most users may fail to get the motivation to spend their time carrying out such task, thereby leading to low utilization of the platform and as a result incomplete feedback.

Response Time: Maintaining prompt responses to user feedback and bug reports can be challenging, more so for developers handling multiple apps or with a high volume of feedback. Lags in response time cause user dissatisfaction and frustration.

Feedback Analysis: Analysis and prioritization of the feedback from the users can be laborious and contextual, which necessitates developers to sort through a significant amount of feedback to determine valuable insights and prioritize bug fixes and improvements.

User Support: It is difficult to support users when the app encounters issues, even for small development teams with fewer resources. Timely and satisfactory solving of user

problems is paramount to have happy users.

Quality Assurance: Ensuring the app quality and reliability through comprehensive testing and bug fixing is tough due to the fact that bugs tend to appear with each new app release and require constant maintenance and updates.

Performance Monitoring: Ensuring processor speed and delivery time can be difficult in deploying performance monitoring tools. In this case continuous monitoring and adjustment must be done to optimize the app's performance and user experience.

8. CONCLUSION

In this paper we put forward an individual application in this paper, the function of which is to simplify the process of user feedback receipt and bug reporting for the mobile applications. Integrating SMTP into the app, we meant to bring together the communication between the users and the developers so that these requests are prompt and complete. The app provides a simple interface that has the list of the top apps and the developer contact information as well as also the feedback form for the users to state if there are any issues and give suggestions. Feedback submissions are directly sent to the concerned app developers via email encouraging a very collaborative working relationship hence the faster bug fixes and also iterative improvements.

Our system is characterized by simplified communication, a user-friendly interface, near instantaneous feedback, increased collaboration, quick bugs fixes, and a better user experience. Developers can obtain more good insights into the app issues when the bug reports are supplemented with the user feedback and prioritize the bug fixes, accordingly, thus resulting in better app quality and user satisfaction.

Briefly, the implementation of the proposed scheme supplies a unified solution to the problem of feedback and also bug reporting in the mobile application development. Using technology and user feedback, the developers amplify their debugging capacities, focus on fixing the apparent bugs and consequently end up developing a very pleasant and nice user experience.

REFERENCES

1. Haering, M., Stanik, C., & Maalej, W. (2021, May). Automatically matching bug reports with related app reviews. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 970-981). IEEE.
2. Aljedaani, W., Mkaouer, M. W., Ludi, S., & Javed, Y. (2022, March). Automatic classification of accessibility user reviews in android apps. In 2022 7th international conference on data science and machine learning applications (CDMA) (pp. 133-138). IEEE
3. Araujo, A. F., Gôlo, M. P., & Marcacini, R. M. (2022). Opinion mining for app reviews: an analysis of textual representation and predictive models. Automated Software Engineering, 29, 1-30.
4. Gao, C., Zhou, W., Xia, X., Lo, D., Xie, Q., & Lyu, M. R. (2021). Automating app review response generation based on contextual knowledge. ACM Transactions on Software Engineering and Methodology (TOSEM), 31(1), 1-36.
5. Hassan, S., Tantithamthavorn, C., Bezemer, C. P., & Hassan, A. E. (2018). Studying the dialogue between users and developers of free apps in the google play store. Empirical Software Engineering, 23, 1275-1312
6. Kamonphop Srisopha, Devendra Swami, Daniel Link, and Barry W. Boehm. 2020. How features in iOSApp Store Reviews can Predict Developer Responses. In EASE '20: Evaluation and Assessment in Software Engineering, Trondheim,Norway, April 15-17, 2020, Jingyue Li, Letizia Jaccheri, Torgeir Dingsøyr, and Ruzanna Chitchyan (Eds.). ACM, 336–341
7. Phong, M. V., Nguyen, T. T., Pham, H. V., & Nguyen, T. T. (2015, November). Mining user opinions in mobile app reviews: A keyword-based approach (t). In 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 749-759). IEEE
8. Stuart McIlroy, Weiyi Shang, Nasir Ali, and Ahmed E. Hassan. 2017. Is It Worth Responding to Reviews? Studying the Top Free Apps in Google Play. IEEE Software 34, 3 (2017), 64–71.
9. Yang, T., Gao, C., Zang, J., Lo, D., & Lyu, M. (2021, April). Tour: Dynamic topic and sentiment analysis of user reviews for assisting app release. In Companion Proceedings of the Web Conference 2021 (pp. 708-712).
10. Liu, H., Shen, M., Jin, J., & Jiang, Y. (2020, July). Automated classification of actions in bug reports of mobile apps. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 128-140).
11. Fazzini, M., Moran, K., Bernal-Cardenas, C., Wendland, T., Orso, A., & Poshyvanyk, D. (2022). Enhancing mobile app bug reporting via real-time understanding of reproduction steps. *IEEE Transactions on Software Engineering*, *49*(3), 1246-1272.
12. Tan, S. H., & Li, Z. (2020, June). Collaborative bug finding for android apps. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (pp. 1335-1347).
13. Malgaonkar, S., Licorish, S. A., & Savarimuthu, B. T. R. (2022). Prioritizing user concerns in app reviews–A study of requests for new features, enhancements and bug fixes. *Information and Software Technology*, *144*, 106798.
14. Mazuera-Rozo, A., Trubiani, C., Linares-Vásquez, M., & Bavota, G. (2020). Investigating types and survivability of performance bugs in mobile apps. *Empirical Software Engineering*, *25*, 1644-1686.
15. Su, T., Fan, L., Chen, S., Liu, Y., Xu, L., Pu, G., & Su, Z. (2020). Why my app crashes? understanding and benchmarking framework-specific exceptions of android apps. *IEEE Transactions on Software Engineering*, *48*(4), 1115-1137.