Hybrid Crowd-powered Approach for Compatibility Testing of Mobile Devices and Applications

Qamar Naith[1] and Fabio Ciravegna[2]
[12] The University of Sheffield
[1] University of Jeddah
[1] {qhnaith1@sheffield.ac.uk, qnaith@uj.edu.sa}, [2] {f.ciravegna@sheffield.ac.uk}

ABSTRACT

Testing mobile applications (apps) to ensure they work seamlessly on all devices can be difficult and expensive, especially for small development teams or companies due to limited available resources. There is a need for methods to outsource testing on all these models and OS versions. In this paper, we propose a crowdsourced testing approach that leverages the power of the crowd to perform mobile device compatibility testing in a novel way. This approach aims to provide support for testing code, features, or hardware characteristics of mobile devices which is hardly investigated. This testing will enable developers to ensure that features and hardware characteristics of any device model or features of a specific OS version will work correctly and will not cause any problems with their apps. It empowers developers to find a solution for issues they may face during the development of an app by asking testers to perform a test or searching a knowledge base provided with the platform. It will offer the ability to add a new issue or add a solution to existing issues. We expect that these capabilities will improve the testing and development of mobile apps by considering variant mobile devices and OS versions on the crowd.

KEYWORDS

Crowdtesting, Compatibility, Knowledge base, Mobile Devices

1 INTRODUCTION

Testing mobile apps on different mobile device models and OS versions is an expensive and complicated process. The main reason is the compatibility issues (mobile fragmentation) which comes from the complexity of mobile technologies and variety of existing mobile device models and OS versions [19]. Developers are not sure whether their apps behave and work as expected on all mobile devices. This issue requires developers to perform mobile device compatibility testing on a variety of mobile device platforms, models, and OS versions in the least time possible to ensure the quality of the app.

The issue of compatibility testing is attributed to several causes: (a) Mobile apps (especially, sensing apps) behave slightly differently, not only on mobile devices from different manufacturers but also on mobile devices from the same manufacturer[17, 29]; (b) Most of the mobile app developers are small teams or individual developers, it is often too expensive and difficult for them to have a variety of mobile devices models and OS versions for testing. (c) The automated testing approach is insufficient to test all real-life scenarios or captures all aspects of real mobile devices [23]. (d) The lack of relevant knowledge and experience[33]. Recently, several testing approaches like Automated testing [28], Cloud testing services [15, 19, 20, 30] and industrial crowdtesting platforms or companies [5, 7, 11, 13, 19, 20, 26, 30] have been built to address the compatibility issue. However, the issue remains [16].

Due to the need for a generic solution to address compatibility issues mentioned above, we consider the following requirements in our proposed solution: (1) A new testing approach is required to support a global-scale mobile device compatibility testing [12]; (2) The individual developers and small teams cannot have all different types of devices, so there is a need for outsourcing to obtain enough mobile devices models with different OS versions; (3) An efficient method for sharing relevant compatibility testing knowledge among developers and testers. (4) The need for manual compatibility testing on a real mobile phone to address the automated testing issue. (5) Guidelines to assist developers in the mobile apps development lifecycle.

This paper proposes a new mobile devices compatibility testing approach based on a new crowdsourcing method *" Hybrid Crowdsourcing"*(see section 4). The proposed hybrid crowdsourced compatibility testing approach is defined as a web-based crowdtesting platform. It provides the ability to perform compatibility testing of mobile devices at two different levels, low-level (as code) or high-level (as physical features of API level related to OS versions or mobile device) against mobile apps requirements. These two levels of compatibility testing will be achieved through the participation of public crowd testers using real mobile devices with different OS versions. This platform not only provides testing services but also a knowledge base for storing the collected results (incompatibility issues and their reasons) from different human and hardware resources in a structured manner, which enable developers to search for the specific issues they may face.

The overall content of this paper is organized as follows.Section 2 describes existing approaches for mobile app compatibility testing. Section 3 discusses the limitations of all these approaches. Section 4 presents our proposed hybrid crowdsourced compatibility testing approach. Section 5 presents the working mechanism of our crowdsourced compatibility testing approach. Section 6 provides a list of all our contribution in this paper. Finally, Section 7 presents our conclusions.

2 RELATED WORKS

Several solutions have been proposed in literature to address the compatibility testing issues of mobile apps [2, 4–7, 10, 11, 13, 14, 19, 26, 30].

ICCSE, 28-31 July 2018, Singapore
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Industrial Crowdsourced Testing Platforms

Industrial crowdtesting platforms or companies like Mob4Hire [4], Pay4Bugs [7], uTest (Applause) [10], 99Tests [1], Testbirds [8], Passbrains [6], Global App Testing (GAP) [2] etc., are considered as initial solutions to address computability issues. Most of these platforms and companies are not directly targeting compatibility testing in their testing solutions. These platforms and companies focus more on other types of testing such as functional, security, usability, load, localization, and automation [13]. The crowdtesting method used in these aforementioned platforms and companies is mainly focused on testing the app as a whole to make sure that the app is free of errors. They have designed based on "On-demand matching and competition" [26], which means that crowd testers are selected based on matched requirements (their availability, demographic information, rate, and testing type preferences).

Automated and Cloud-based Testing Services

Prathibhan et al. [28] designed an automated testing framework over the cloud for testing mobile app on different Android devices. This framework can provide performance, functional and compatibility testing. Testdroid [21]is an online platform for automated UI testing. It enables developers to execute automated scripted tests on physical Android or iOS mobile devices with different OS versions and screen resolutions. These devices are physically connected to online servers which manage the test queue for the individual physical device. This platform has limited testing service, i.e. it does not support for testing applications that depend on gesture, voice, or movement input.

Cloud-based testing services are another type of solution that is used to address the compatibility issue since different mobile devices and OS versions are available in the cloud. Developers can access these devices and use them via various cloud services. Huang and Gong [20] proposed *RTMS* which is a cloud of mobile devices for testing mobile apps. In the *RTMS* the users could remotely access a pool of mobile devices that exist in a local lab and perform app testing of uploading, startup, and testing by clicking and swiping actions.

Besides, the authors of [19] extended the *RTMS* and they proposed *AppACTS* that provides automated scripting service to perform compatibility testing of apps on different users real Android devices and collecting the results of testing through a web server. The compatibility testing of the proposed method covers installation process, startup, random key, screen actions, and the removal process of the mobile app on real remote devices. The most important feature of the *AppACTS* is its geographical distribution and scalability, and unlike *RTMS* it will not use a cloud of mobile devices, and it uses real remote devices. The MyCrowd platform [11] provides an automated compatibility testing service for mobile apps. The service proposed is considered as SaaS (Software as a Service) and different from the service provided in AppACTS [19]. This service help developers to upload their apps through the web interface, choose the target device models and OS version versions, perform UI testing remotely, and then review testing report after finishing testing.

Starov et al. [30] proposed *CTOMS* framework for cloud-based testing of mobile systems that perform the testing on the remote real mobile device through the cloud (each pool of real mobile devices connect to a mobile server, and then all mobile servers connected to one master server). This cloud testing is carried out by the participation of their own crowd community. This framework provided the concept and the prototype of cloud testing of mobile systems together with multi-directional testing that tests the app on various Android devices only with various OS versions and new device models for their compatibility testing.

Knowledge base development for Compatibility Testing

StackOverflow is a service based on open crowdsourcing (volunteer work) for addressing technical programming issues [14, 27]. This platform has documentation that focuses on storing questions and issues related to computer programming only (coding issue). It is also used by mobile apps developers to assist them with the issues they are facing within the implementation process (programming) of mobile apps, but it is clear that rarely they used for testing (see section 3). The authors of [18] addressed Android fragmentation problem in mobile application compatibility automated testing. Also, it could analyze the fragmentation both in code level as well and API level based on the provided knowledge base from previous tests. The proposed crowdsourcing platform will involve a knowledge base (documentation), especially for documenting issues related to mobile device compatibility testing.

3 THE LIMITATIONS OF EXISTING APPROACHES

In our view, the solutions above do not provide enough support to address issues of mobile device compatibility testing for the following reasons:

- ***Test Distribution:*** Most of these platforms, companies, and cloud services described previously have a limited distribution of the test. Since they do not provide opportunities to other public crowd testers with limited resources in worldwide to contribute. Consequently, these solutions will be unable to study enough users' behaviors or interactions with the app.
- ***Variety of Mobile Devices:*** Most of the mentioned approaches may not cover a large variety of mobile devices or OS versions. The main reason is the lack of availability of all different versions of mobile phones models or OS versions in one place like a company headquarter or its specific crowd community. To our best knowledge, there is not a complete set of mobile phones with different OS versions in a laboratory or a registered cloud system to be accessible by developers as discussed at RTMS [20], AppACTS [19], MyCrowd [11], as well as CTOMS [30].
- ***Coordinated Scheduling:*** It can be also an important issue for all mentioned approaches when crowd testers need to share a single pool of mobile devices in different cases such as: (1) When they required performing the test across various locations at the same time. (2) When some of the devices need to be tested with more than one OS versions for different tasks at the same time. This could lead to issues

in the coordination process between testers and delay [32] as well as it may not covering all the OS version.

- ***Knowledge Accessibility:*** Most of the mentioned testing methods do not have a public place to store and share the results of compatibility testing with public developers to access. The results are stored in a private space of a company, only developer teams within the company will be able to access. This will not improve the experiences and knowledge of both testers and developers as well as domain knowledge of the mobile app development.
- ***Compatibility Testing Type:*** Most of these approaches focus on testing mobile apps itself to be sure there is not any failure, not for checking and understanding the compatibility of various mobile devices with the app requirement. This way of testing will not help app developers to broaden their level of knowledge and experiences.
- ***Impossibility of Testing:*** In Stack Overflow, there is a documentation for programming purposes, and is not designed for testing. Also, there is no space where developers can ask to test a specific issue (test case, feature, or code) on different mobile phone models. Human-based tagging system might be another drawback found in StackOverflow, where users select the tag based on their feeling or potentially communities surrounding those tags. This might lead to tag the information in a wrong classification.

The proposed crowdtesting platform and compatibility testing approach are designed to satisfy this need.

4 PROPOSED HYBRID CROWDSOURCED COMPATIBILITY TESTING PLATFORM

Manual compatibility testing is the suggested method for testing the mobile apps during the development process [22]. We design our approach as a web-based platform specially designed to support a fully mobile device compatibility testing for Android and iOS through crowdsourcing methods. This platform involves two main parts: Manual compatibility testing services based on proposed hybrid crowdsourcing model, and a crowd-powered knowledge base.

4.1 The Concept of Proposed Compatibility Testing Approach

The concept of our compatibility testing approach focuses on testing whether features or hardware components of mobile devices and features of OS versions compatible with functionalities of different types mobile apps (e.g., health, banking, activity tracking, social, or any other apps). It considers testing the compatibility for all various dimensions of the mobile device and its criteria with the app like (1) Hardware dimensions such as the camera (type and resolution), screen (rotation, size, resolution), sensors, CPU speed, GPS, Bluetooth, RAM, etc.); (2) Software dimensions such as different APIs level (minimum supported Application Programming Interfaces), Multimedia supported, etc.; (3) Human behaviors and interaction with the app also considered as another dimension to reduce compatibility testing issue such as localization, languages, style, accessibility requirements, target user of the app, type of environment (e.g., medical, finance, education, etc.); Therefore, our approach possibly could answer the following questions: (1) Is there any missing hardware components within mobile device models or features within API levels related to OS versions that the app needs it to work correctly? (2) How much the hardware component is compatible with a specific functionality of a specific app for different mobile device models and OS versions?

In our approach, the compatibility testing could be achieved in two ways:

(1) Low-level testing: A developer can publicly test the compatibility by examining a piece of code to identify whether the code testing results comply with the expected results from a variety of mobile devices model and OS versions. By proposed approach, this type of test could be carried out by a large number of testers with both adequate and very high level of experience.
(2) High-level testing: A developer can privately test the compatibility by testing the hardware features of the device itself under different OS versions. This type of test may help testers with a limited level of experience to perform multiple tests and improve their experience.

These two ways of testing with all mobile device dimensions allow for achieving full compatibility testing service through the use of new crowdsourcing method.

4.2 Proposed Hybrid Crowdsourcing Method for Testing

In the literature, two basic crowdsourcing methods are used in software development and testing process: Traditional Crowdsourcing method (used by most of the testing companies [26, 31] and Open Crowdsourcing as used by in [27]. The traditional crowdsourcing and open crowdsourcing method are similar in that both of them do not support replication of the tasks which means that they specify a different job for each crowd in order to complete the testing process quickly within a limited time. These two methods are different in the contribution form. Traditional crowdsourcing normally used based on online competition and on-demand matching [24, 26] where the crowd is selected from the registrants' crowd list based on their high experience to perform specific tasks. On the other hand, open crowdsourcing method is used based on open collaboration [26] where the public crowd participate (as volunteers) based on their interests in tasks.

To avoid these constraints, we proposed an alternative crowdsourced testing method, called "Hybrid Crowdsourcing" method. This method uses the power of the public crowd testers to enhance and facilitate full mobile device and mobile app compatibility testing process. By employing the hybrid crowdsourcing method in proposed compatibility testing approach that described in section 4.1, we will enable a large number of anonymous and public crowd testers to participate based on their interests without time constraints. Such open participation of public crowd testers will assist developers in: (1) Testing more mobile devices with different OS versions and gathering more accurate results. (2) Covering most of the possible compatibility issues in early stages of the mobile app testing process. This method will support flexible incentives mechanism which is based on the negotiations between the parties and the agreement for the incentives they willing to provide or

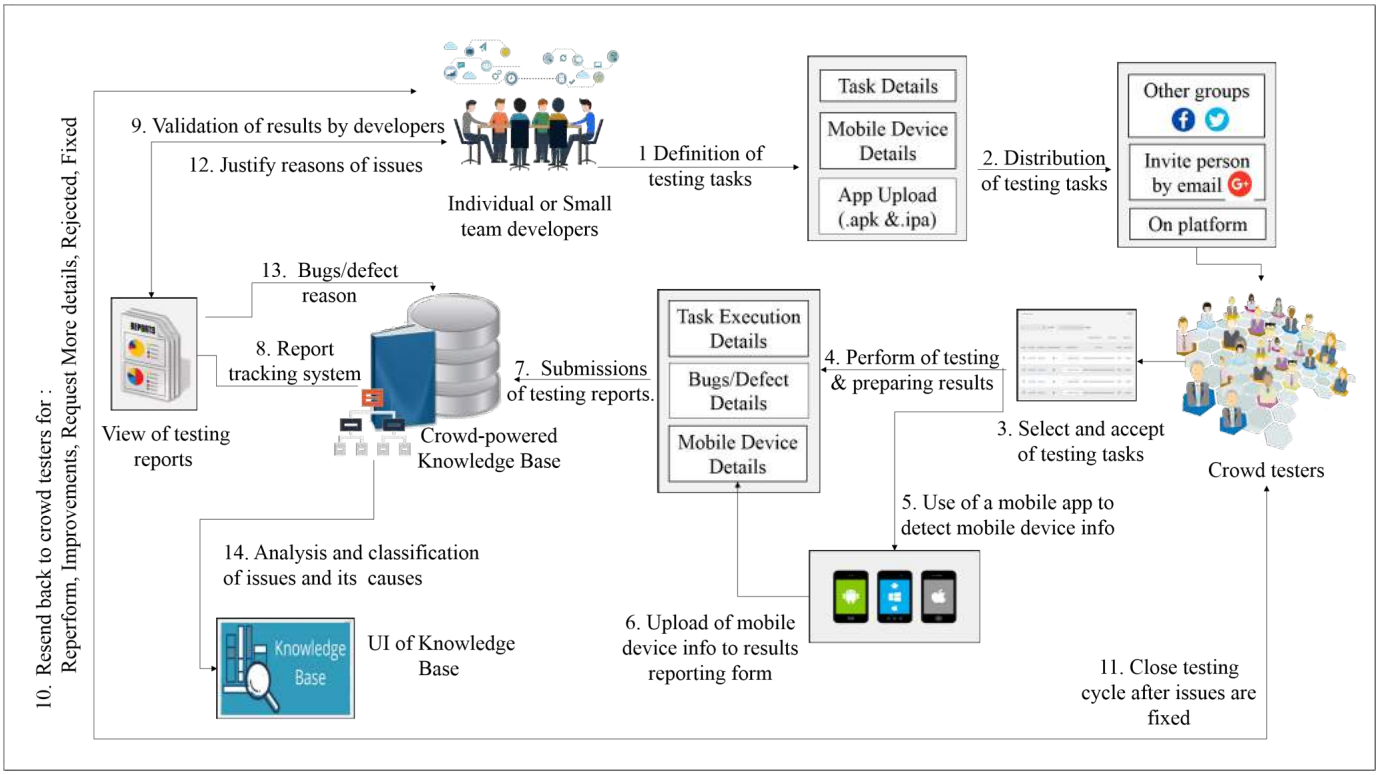

Figure 1: The working mechanism of proposed hybrid crowdsourced compatibility testing approach

earn. Such incentive mechanism will help small teams or individual developers with limited resources to test their apps. Table I shows a comparison between Traditional Crowdsourcing, Open Crowdsourcing, and Hybrid Crowdsourcing methods based on different dimensions. While Table II describes the advantages and disadvantages of the three Crowdsourcing methods.

5 PROPOSED WORKING MECHANISM

Based on the concept of the proposed compatibility testing approach described in Section 4.1 and hybrid crowdsourcing method explained in Section 4.2, we will describe in this section a set of process that is implemented in the working mechanism of proposed crowdsourced compatibility testing. Fig 1 provides an overview of the proposed working mechanism, and the set of the process are listed below:

(1) ***Definition of testing tasks:*** Developers define their projects for testing, the individual project can include more than one task. At this stage, developers need to complete three main steps to clearly define the project: (a) *Task information:* developers are able to define more than one task with the required details for each task. First, they need to specify the hardware components or feature category of the mobile device or OS versions they would like to test. Next, developers need to select the way they would like to test their task with. testing as code by providing the title of the task and source code or testing as a feature by providing the title and testing steps. They can use both ways at the same time to test a specific task. Finally, developers need to specify the results they expected to get from each testing task will be performed. (b) *Device information:* developers need to define the required mobile device platforms, OS versions, brand, and device models for testing. It is possible for developers to define mobile devices information for the project as a whole or for each individual task within the project. (c) *Application information:* At the beginning of this step, developers need to select the type of app they would like to ensure it works seamlessly without any errors against the tested mobile device's or OS version's feature(e.g., Social, Business, Education, Finance, Lifestyle, Medical, etc.). Then developers will need to upload the targeted app to the platform. Our approach provides an easy way that enables crowd testers to download the app on their device without any constraints. In case of uploading Android apps, developers need to upload the.apk file on the platform. In contrast, for uploading the iOS app, developers need achieving four main steps in order:

- Download ".cer" file from the platform and click to install it on their mac device.
- Download ".provision" file and click to install it on their mac device.
- Build their application with the identifier.
- Generate ".ipa" file and Upload it to the platform.

Table 1: The comparison of the three crowdsourcing methods

Method	Participation Form	Expertise Demand	Crowd Size	Task Length	Task replication	Incentive Type	Incentive example
Traditional Crowdsourcing	On-demand Matching On-line Competition	Extensive	Limited	limited time	None	Extrinsic (Physical Incentive)	Compensation/Reward (Mostly small amount of money)
Open Crowdsourcing	Open Collaboration	Extensive	Open	limited time	None	Extrinsic (Social Incentive)	Name Recognition and Satisfaction
Hybrid Crowdsourcing	Open Collaboration	Extensive Moderate Minimal	Open	Flexible time	Several	Extrinsic as (Physical & Social) Intrinsic as (Self knowledge & learning)	- Different types of reward not only money - Name Recognition & Satisfaction -knowledge& Experience

Table 2: The advantages and disadvantages of the three crowdsourcing methods

Method	Advantages	Disadvantages
Traditional Crowdsourcing	– All the selected crowd testers have a high level of experiences. – Lower cost compared to non-crowdsourced testing companies.	– Inherent constraints of human resources and hardware resources. – Persuading someone who is not interested to perform a specific task might be difficult, this will probably lead them to perform the task quickly to get the reward only, this may cause inaccurate results. – The use of extrinsic incentives only will be weakened the crowd efforts to perform tasks correctly over time Liang et al [32].
Open Crowdsourcing	– The public crowd is participating based on their interests which they could do more effort to get very good results. – Ability to study more human behavior and interaction with mobile devices.	– Non-physical incentives can be attributed to the lack of more crowd participation or laziness in executing the tasks in some cases over time.
Hybrid Crowdsourcing	– The ability for studying more of human behavior and interaction with mobile devices. – The anticipation of unknown reward may attract more people to take a part in the tasks to get different types of reward. – The use of both extrinsic and intrinsic incentive mechanism together to motivate the crowd will significantly improve the effort of the crowd in performing tasks as proved recently in 2018 by Liang et al [25].	– The expertise of crowd participating is unknown unless the previous experiments have been taken, the outcome would be unpredictable

(2) ***Announcement and distribution of testing tasks:*** Once the developers finished the process of defining the new testing project with its tasks requirements, an announcement will be sent to all crowd testers registered on the platform to execute the test rather than selecting specific crowd testers based on requirements matched (e.g., available hardware and software resources, demographic information, or geographic

location) as in most existing crowdtesting platforms. In addition, developers will be able to use the random URL link that automatically generated after finishing defining the process to distribute the test on a large scale: (a) Send the link to specific users by email. (b) Publish (copy/paste) the URL link in an online space such as Facebook, Twitter, LinkedIn, blogs, or any testers' groups on the internet. As we mentioned before, purchasing a big list of mobile devices only to test for a specific time is really expensive. Further, it is also difficult for companies dealing with specific communities, testing labs, or cloud services to always keep the devices up to date. Our global distribution of tests will allow public crowd testers to participate and use their own device for testing. In this case, it is very much possible to cover newly released devices that probably are not still available with crowd tester community in existing crowdtesting method or still not registered in the could-testing system. In addition, it may also cover the old devices that rarely became available in public.

(3) ***Review project and its tasks specification by crowd testers:*** When registered crowd testers receive the announcement of the test, they will be able to review the requirements of the project. On the other hand, when non-registered crowd testers see the URL link anywhere on the internet, they will first need to register to the platform and then review test requirements. After crowd testers review the requirements of the announced project, they will have an option to accept the project for testing or not. If they are not interested in performing a specific project, the proposed approach gives them an opportunity to review requirements for all other projects that can be tested using their device/s platform. So that they can accept more than one project simultaneously and then perform the test in order.

(4) ***Execute testing:*** After crowd testers select and accept the project, they will review all testing requirements before start performing the test to avoid âĂIJout of scopeâĂİ issues. Then they will start testing all the tasks associated with the project using required types of devices and OS versions. In our approach, both crowd testers and developers will be able to view all tested devices for each task so they can know whether there are specific devices or OS version need to be covered. Further, due to our global distribution of test, crowd testers that are participating to execute the test can be from different levels of experience. This diversity of experience can help to find more issues quickly. This is because experienced testers follow specific testing steps and they always use them for testing any mobile app. As a human behavior, the crowd testers with different levels of experience may try to do more effort to perform the test and provide better testing results. This will also help crowd testers to find more issues and accurately reproduce the behavior of an app that runs on exact mobile.

(5) ***Prepare and submit testing reports:*** Before submitting testing report, crowd testers need to notice that if they do not accept the project they will not be able to report their feedback even if they performed the test. As we mentioned previously one project may include more than one task. When the crowd testers accept the whole project, all the tasks belong to this project will be added directly to their file; each task within the project will have their own status (submitted or in progress). This means that crowd testers can execute each task separately and then submit a single report to developers for each task executed. Each report includes crowd tester's feedback, this feedback includes the following information:

   (a) *Device information:* It is essential for crowd testers to enter the details of mobile devices (platform, brand, model, and OS version) that they used in testing correctly in the report, especially if they performed the test on many devices. In this approach, if crowd testers would like to enter device information for the first time, they require selecting "new device" from the options then using our mobile app to detect device information. Once tester is accessing the app from the device that has tested, the information will directly be detected and sent to the server and then upload into reporting form during the reporting process. This method will provide accurate inputting of device details and avoid the errors that may occur while entering the details manually in the reporting form, which can confuse the developer as in most crowdsourced testing approach existing. On the other hand, if one crowd tester would like to enter device information that already tested and registered before, the tester will require selecting "registered device" from the options then the tester can choose from his list of tested devices.

   (b) *Task information:* At this step, crowd tester needs to enter how many times they repeated the test and the estimated time for each test cycle, as well as actual results.

   (c) *Issue information:* It is the responsibility of crowd testers to provide a report with correct information for issues. The information must compose of the following:(i) Issue ID, title, priority, and description. (ii) Actions and steps to reproduce the issue clearly written, showing how the tester arrived at the location of the issue. (iii) Attachments showing the issue either in a picture or in a video. (iv) Any suggestion or idea they would like to share with developers in terms of improving or solving the issue as well as how much they are sure about the result? In this approach, crowd testers can enter more than the issue related to a particular task in one report instead of submitting multiple reports for the same task so this can reduce the time needed for reporting process.

(6) ***Collect, track, and organize testing reports*** After many crowd testers are executing different tasks for different projects at different times, developers need to collect and organize the report to view all issues that have been reported. To avoid the limitations of collecting and organizing of testing report manually by crowd manager or crowd leader as in most existing crowdsourced testing method, our approach will implement a report tracker system slightly similar to JIRA [3] and Zoho [9] for collecting, tracking and organizing all testing reports submitted by crowd testers automatically. This tracking system will assist developers to: (a) *Task Status Tracking:* Track the status of each task within each specific project; (b) *Detailed Report:* Track master details regarding

the issues Id, type, description, priority, etc.; (c) *Testers Identity:* Track the crowd testers' details who report the results; (d) *Scheduled Reports:* Track recent update on the results based on Daily, Weekly, or Hourly; (e) *Modification Method or issue tracking life cycle:* Ability to modify an existing issue status, open, in progress, under review, Invalid/rejected, accepted, or fixed; (f) *Notification Method:* To notify developers about any new bug reported, also testers about any change on the bug they have reported; (g) *Results visualization:* To provide charts and analysis of reported results during the time; (h) *Efficient Classification:* To classify the reported issues based on the type of app;

(7) ***View testing reports and validate results:*** After crowd testers are submitted their reports, and after collected and organized the reports by our tracking system, the responsibility of the developers to view and validate the results in every single report. At this step, developers starting validate results in one report after another. Based on the crowd testers' results and validation results of developer they will be able to "Approved" or "Rejected" the reports. In case of the results is not accurate the developer will directly reject the report and will not reward the crowd testers. On the other hand, if the results are correct and developer has approved the results, they will have the option to change the status of the report to deferred (they will try to fix the issues after finishing current work), in progress (I am trying to fix the issues), or fixed (I have finished fixing issues). Once the developer changes the status of a specific report, the status of the report will directly be updated on the profile of crowd tester's who has submitted the report. Note, the priority of fixing issues based on the priority of issues for all reports. After developer fixing the issue, they will send the report back with a notification to crowd tester to retest and make sure that the error is fixed and not appear again. Once finishing retest process, crowd tester will send the report back to developers with a notice that the issue has totally fixed and then asks for closing the test task cycle. After developer finishing validation of all tasks within a specific project, a notification is sent to the tester to close any activities related to the project.

(8) ***Update crowd testers rating by developers:*** Based on the good preparation of reports and accuracy of crowd testers' feedbacks (including full issues details reported), developers will rate crowd testers. In addition, accepting and performing as much as possible of testing projects will also positively influence developers rating for crowd testers. On the other side, rejected reports will negatively effect on crowd testers' rating. In our approach, due to dealing with public and anonymous crowd testers, the rating (collected points) of crowd testers can reduce developers' concerns for working with public crowd testers and increase their confidence in crowd testers' works performance and results they will report.

(9) ***Encourage and reward of crowd testers:*** After developers finish validation and evaluation of crowd testers, they will need to send a notification to crowd testers to close the testing project. After that, our system will monitor and calculate all closed project for different testers and send a notification with a list of all crowd testers need to reward. This process will minimize the problem of consuming a long time by crowd leader or crowd manager to monitor and identify a list of crowd testers that need to be rewarded and then send the list to developers to perform the rewarding process. According to results proved by Liang et al [25] that using extrinsic incentives only (rewarding) weaken crowd performance and effort to perform tasks correctly, so there is a need for using intrinsic incentive as well. In our approach, the two incentive types will be implemented to encourage crowd testers: Extrinsic and Intrinsic. *The extrinsic incentive"* is the rewarding step which depends on the agreement among developers and testers. *In extrinsic method*, developers are free to choose the type of reward they are willing to offer to the testers (e.g., some money, free app, gift card, certificate, job, etc.), it is possible to provide multiple options for crowd testers. At this stage, developer sends an offer including reward options to crowd testers. Once the crowd tester receives this offer the crowd tester will choose one option, and then send his approval to the offer again to the developer. *The intrinsic incentive* includes self-knowledge and learning. In our approach, crowd testers can get benefits and learn more from other crowd testers' knowledge and testing scenarios that have been stored in our knowledge base. This incentive mechanism could motivate the testers more and more, and at the same time serve developers with limited resources, unlike to other crowdtesting platforms that are restricted developers into the only money.

(10) ***Justify reasons of issues by developers:*** In this step, after developer finish validation of all submitted reports related to testing a particular task on a variety of mobile device, the developer will justify reasons of issues on these tested mobile devices and OS version to be stored in our knowledge base. This step is not available in any of current crowdsourcing methods. Implementing this step can assist developers in the future to understand incompatibility reasons of a specific mobile app with mobile devices and OS versions while developing a new mobile app.

5.1 A crowd-powered knowledge base

Our approach includes a knowledge base that documenting collected compatibility results (including issues) through the crowd testers within the platform. This knowledge base will bring supporting evidence enable developers to understand the reasons for specific issues appeared. Also, it will allow developers to get useful guidelines; for example, some of the mobile devices will not support a specific functionality within a specific type of app for any reason such as if some hardware component or characteristics necessary for the proper running of an app is missed in any device model, or not supported by any API framework related to specific OS versions. Documenting such this results will help developers a lot in the future when they need developing a new app that may have some similar functionality of the apps that have been developed before and tested on many devices. Of course, this will reduce the issues and time taken for compatibility testing on different devices,

especially during early stages of the development cycle. In addition, it provides ability for the developers and testers to vote, rate, and comment to evaluate the quality of other published results related to specific compatibility issue of the mobile device.

6 CONTRIBUTION

The main contributions in this paper are:

(1) A crowdsourced testing platform that supports the distribution of full mobile device compatibility testing, which is suitable for both professional and beginner developers and testers.
(2) It lays out a new crowdsourced testing framework, that can be used to execute other mobile apps testing types such as GUI, Usability, Localization, etc. for all supported platforms.
(3) A novel direct interaction mechanism between the developer to provide better coverage of testing in specific software or hardware that are in need of testing by the crowdtesting platform.
(4) A public knowledge base allows for a deep understanding of the internal complexity of testing the features or hardware characteristics of mobile device models or features of API levels related to OS versions for all supported platform. This can improve developers' knowledge and experiences and helps them during the app development process.

7 CONCLUSIONS

This paper narrows the research gap related to studying causes of compatibility testing issues. It reviewed the current approaches that address the compatibility testing issues. Since compatibility testing requires to cover many mobile device models and OS versions, the hybrid crowdsourced testing method has been proposed to use the power of public crowd testers and their own mobile devices to perform testing on a variety of devices. This paper focuses on testing the features and hardware characteristics of mobile devices with different OS versions in two different ways, which are code or features, against mobile apps requirements. The proposed approach shows that the use of public crowd testers and processes used in the hybrid crowdtesting method might be significantly strong enough for several reasons. The ability to cover a large number of mobile device model with different OS versions. To improve the testing and development of mobile apps through direct interaction between developers and crowd testers. The ability to discover more issues through different experience levels of crowd testers. To reduce the delay in the coordination process among crowd testers for the availability of using a specific device. To avoid mistakes that may occur by crowd testers when reporting information about tested mobile device by using a mobile app as automated data detection method. To reduce developers' concerns of working with anonymous crowd testers. To attract more crowd testers to participate through the use of both extrinsic and intrinsic incentive method. To reduce delay in waiting for either the crowd manager or crowd leader to collect and organize testing reports. As a future work, we will evaluate our proposed approach "hybrid crowdsourced compatibility testing" to measure its effectiveness and efficiency in addressing the compatibility testing issues.

REFERENCES

[1] 99tests news. https://99tests.com/news. [Online; accessed 29-July-2017].
[2] Global app testing. https://www.globalapptesting.com/. [Online; accessed 20-Jun-2017].
[3] Jira. https://www.atlassian.com/software/jira. [Online; accessed 18-April-2018].
[4] Mob4hire. http://www.mob4hire.com/. [Online; accessed 8-Aug-2017].
[5] Most popular crowdsourced testing companies of 2018. http://www.softwaretestinghelp.com/crowdsourced-testing-companies/. [Online; accessed 28-Mar-2018].
[6] Passbrains. https://www.passbrains.com/. [Online; accessed 18-Nov-2017].
[7] Pay4bugs.crowdsource software testing. https://www.pay4bugs.com/. [Online; accessed 20-Jun-2017].
[8] Testflight. https://developer.apple.com/testflight/. [Online; accessed 18-Nov-2017].
[9] Zoho bug tracker. https://www.zoho.com/bugtracker/features.html. [Online; accessed 18-April-2018].
[10] utest. https://www.utest.com/, 2007. [Online; accessed 8-Aug-2017].
[11] Mycrowd qa. https://mycrowd.com/, 2015. [Online; accessed 1-Nov-2017].
[12] M. Akour, A. A. Al-Zyoud, B. Falah, S. Bouriat, and K. Alemerien. Mobile software testing: Thoughts, strategies, challenges, and experimental study. *International Journal of Advanced Computer Science and Applications*, 7(6):12–19, 2016.
[13] S. Alyahya and D. Alrugebh. Process improvements for crowdsourced software testing. 2017.
[14] A. Bacchelli, L. Ponzanelli, and M. Lanza. Harnessing stack overflow for the ide. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, pages 26–30. IEEE Press, 2012.
[15] R. J. Bhojan, K. Vivekanandan, S. Ganesan, and P. M. Monickaraj. Service based mobile test automation framework for automotive hmi. *Indian Journal of Science and Technology*, 8(15), 2015.
[16] H. K. Ham and Y. B. Park. Mobile application compatibility test system design for android fragmentation. In *International Conference on Advanced Software Engineering and Its Applications*, pages 314–320. Springer, 2011.
[17] H. K. Ham and Y. B. Park. Designing knowledge base mobile application compatibility test system for android fragmentation. 2014.
[18] H. K. Ham and Y. B. Park. Designing knowledge base mobile application compatibility test system for android fragmentation. *IJSEIA*, 8(1):303–314, 2014.
[19] J.-f. Huang. Appacts: Mobile app automated compatibility testing service. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*, pages 85–90. IEEE, 2014.
[20] J.-f. Huang and Y.-z. Gong. Remote mobile test system: a mobile phone cloud for application testing. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 1–4. IEEE, 2012.
[21] J. Kaasila, D. Ferreira, V. Kostakos, and T. Ojala. Testdroid: automated remote ui testing on android. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, page 28. ACM, 2012.
[22] M. Kamran, J. Rashid, and M. W. Nisar. Android fragmentation classification, causes, problems and solutions. *International Journal of Computer Science and Information Security*, 14(9):992, 2016.
[23] D. Knott. Hands-on mobile app testing. *Indiana: Pearson education Inc*, 2015.
[24] T. D. LaToza and A. van der Hoek. Crowdsourcing in software engineering: Models, motivations, and challenges. *IEEE software*, 33(1):74–80, 2016.
[25] H. Liang, M.-M. Wang, J.-J. Wang, and Y. Xue. How intrinsic motivation and extrinsic incentives affect task effort in crowdsourcing contests: A mediated moderation model. *Computers in Human Behavior*, 81:168–176, 2018.
[26] K. Mao, L. Capra, M. Harman, and Y. Jia. A survey of the use of crowdsourcing in software engineering. *Journal of Systems and Software*, 126:57–84, 2017.
[27] D. Phair. *Open crowdsourcing: Leveraging community software developers for IT projects*. PhD thesis, Colorado Technical University, 2012.
[28] C. M. Prathibhan, A. Malini, N. Venkatesh, and K. Sundarakantham. An automated testing framework for testing android mobile applications in the cloud. In *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on*, pages 1216–1219. IEEE, 2014.
[29] T. Samuel and D. Pfahl. Problems and solutions in mobile application testing. In *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17*, pages 249–267. Springer, 2016.
[30] O. Starov. *Cloud platform for research crowdsourcing in mobile testing*. East Carolina University, 2013.
[31] K.-J. Stol, T. D. LaToza, and C. Bird. Crowdsourcing for software engineering. *IEEE Software*, 34(2):30–36, 2017.
[32] TestObject. Mobile app testing main challenges, different approaches, one solution. 2017.
[33] K. Wnuk and T. Garrepalli. Knowledge management in software testing: A systematic snowball literature review. *e-Informatica Software Engineering Journal*, 12(1):51–78, 2018.