International Journal of Crowd Science

Mobile devices compatibility testing strategy via crowdsourcing
Qamar Naith, Fabio Ciravegna,

Mobile devices compatibility testing strategy via crowdsourcing

Qamar Naith and Fabio Ciravegna
University of Sheffield, Sheffield, UK

Received 15 September 2018
Revised 15 October 2018
Accepted 15 October 2018

Abstract

Purpose – This paper aims to support small mobile application development teams or companies performing testing on a large variety of operating systems versions and mobile devices to ensure their seamless working.

Design/methodology/approach – This paper proposes a "hybrid crowdsourcing" method that leverages the power of public crowd testers. This leads to generating a novel crowdtesting workflow Developer/Tester- Crowdtesting (DT-CT) that focuses on developers and crowd testers as key elements in the testing process without the need for intermediate as managers or leaders. This workflow has been used in a novel crowdtesting platform (AskCrowd2Test). This platform enables testing the compatibility of mobile devices and applications at two different levels, high-level (device characteristics) or low-level (code). Additionally, a "crowd-powered knowledge base" has been developed that stores testing results, relevant issues and their solutions.

Findings – The comparison of the presented DT-CT workflow with the common and most recent crowdtesting workflows showed that DT-CT may positively impact the testing process by reducing time-consuming and budget spend because of the direct interaction of developers and crowd testers.

Originality/value – To authors' knowledge, this paper is the first to propose crowdtesting workflow based on developers and public crowd testers without crowd managers or leaders, which light the beacon for the future research in this field. Additionally, this work is the first that authorizes crowd testers with a limited level of experience to participate in the testing process, which helps in studying the behaviors and interaction of end-users with apps and obtains more concrete results.

Keywords Harnessing the crowd in human-computer interaction,
Tools and platforms to support crowd science and engineering,
Models and methods for crowd science and engineering,
Crowdsourced testing, Crowdtesting for mobile apps, Industrial crowdsourcing

Paper type Research paper

1. Introduction

Testing mobile applications (apps) on various mobile device models and operating system (OS) versions is an expensive and complex process because of compatibility issues (mobile fragmentation) (Huang, 2014b). Developers are not always sure if their apps are behaving and running as expected on all mobile devices. This issue requires mobile app developers to

The first author would like to thank the University of Jeddah and the Ministry of High Education in Saudi Arabia for the PhD research funding. The first author would also like to thank Professor Fabio Ciravegna at Sheffield University for his support on her PhD journey.

International Journal of Crowd Science
Vol. 2 No. 3, 2018
pp. 225-246
Emerald Publishing Limited
2398-7294
DOI 10.1108/IJCS-09-2018-0024

perform mobile device compatibility testing on a variety of mobile device platforms, models and OS versions in the shortest time possible to ensure the app quality. Lately, we started to ask ourselves "what else could be achieved to improve mobile app developers' and testers' lives on the internet and reduce the mobile app compatibility testing issue?" Too many important reasons exist regarding mobile app development and testing, which cause compatibility testing issue:

- Some architectures of mobile devices platforms, like Android, are not standardized across mobile device manufacturers to date. The differences in the architectures and the characteristics of the mobile devices components are largely undocumented and not yet available for the public[1] (Wnuk and Garrepalli, 2018).
- Mobile apps (especially sensing, banking, tracking apps) behave slightly differently in various of mobile devices (Samuel and Pfahl, 2016).
- Most of the app developers are small teams or individual developers; it is often too expensive for them to have a variety of mobile models and OS versions for testing.
- The automated testing approach is insufficient to test all real-life scenarios or to capture all aspects of real mobile devices (Knott, 2015).

Recently, several testing approaches such as Automated testing (Prathibhan *et al.*, 2014), Cloud testing services (Huang, 2014b; Bhojan *et al.*, 2015; Huang, 2012a; Starov, 2013) and industrial crowdtesting platforms or companies (Huang, 2014b; Alyahya and Alrugebh, 2017; Mao *et al.*, 2017; Starov, 2013; Huang, 2012a), MyCrowd QA[2] and Pay4Bug[3] have been built to address the compatibility issue. However, the issue still remains.

All the reasons of the compatibility testing issue mentioned above have attracted the attention of authors of this paper and motivated them to address that issue, the following two questions have been investigated:

Q1. How can the crowdsourcing approach and human computation play an important role in software testing and in a way that is different from that which currently exists?

Q2. Which crowdsourced testing workflow will now be used to construct a complete and effective testing environment?

To answer the questions above, the following requirements have been considered:

- a novel compatibility testing approach on global-scale (Akour *et al.*, 2016);
- the individual developers and small teams cannot have all the different types of devices, so there is a need for outsourcing to obtain enough mobile devices models with different OS versions;
- manual compatibility testing method on a real mobile phone to address the automated testing issue; and
- an online public space that documenting good Android development knowledge and experiences to understand the overall architecture of Android.

This paper is aimed at proposing a novel compatibility testing approach for mobile devices and apps that is demonstrated as a web-based crowdtesting platform, named AskCrowd2Test. This platform is constructed using two novel methods: A new crowdsourcing method known as Hybrid Crowdsourcing (Section 6.2) that depend on the participation of unknown public crowd testers to use their real mobile devices with running

various OS versions to achieve the compatibility testing process. In addition, a new crowdtesting workflow known as Developer/Tester- Crowdtesting (DT-CT) (Section 7) that mainly relies on the direct interaction between the developer and crowd testers and disposes with the services of crowd managers and crowd leaders as well as distributing their work to both developers and crowd testers in the testing process. To understand the external complexity of mobile technology, AskCrowd2Test platform has mainly targeted the mobile devices and apps' compatibility testing in two distinct levels: high-level (as physical properties of API level related to OS versions or mobile device) or low-level (code properties) to ensure that these properties are compatible with a mobile apps specification and requirements. Besides, the compatibility testing service, AskCrowd2Test platform also builds a crowd-powered knowledge base (CPKB) (Section 7.1) regarding storing incompatibility issues and relevant solutions that were acquired while performing the test with different hardware and human resources.

Consequently, building such system that mainly focuses to test the hardware components of the mobile device to demonstrate the issues discover because of differences of the mobile device architecture and documenting such these results including differences and issues will have a significant impact on both academic and industrial domain, in which will help developers to obtain more insight about the suitable environments required for developing a specific app with specific functionalities.

The structure of this paper is given as follows. In Section 2, the authors provide a description of the existing approaches regarding the compatibility testing of mobile applications. In Section 3, drawbacks/limitations of the existing approaches are given. In Section 4, the current crowdtesting working mechanism is presented, and Section 5 highlights observations on current working mechanisms. In Section 6, the authors present the proposed hybrid crowdsourced compatibility testing approach. In Section 7, the authors explain the overall working mechanism of the DT-CT. In Section 8, conclusions of this paper are presented. In Section 9, the authors provide the list of all their contributions.

2. Related works: existing compatibility testing approaches

Several solutions have been proposed in the literature to address the compatibility testing issues of mobile apps; these solutions have been divided into three categories (Section 2.1, 2.2 and 2.3).

2.1 Industrial crowdsourced testing platforms

Industrial crowdtesting platforms or companies such as Mob4hire[4], Pay4Bugs[5], uTest[6], Passbrains[7], Global App Testing[8] and Applause[9] are considered as initial solutions to address computability issues. Most of these testing platforms and/or companies are not directly targeting compatibility testing in their testing solutions. These testing platforms and companies have a more focus on other types of testing such as functional, security, usability, load, localization and automation (Alyahya and Alrugebh, 2017). The crowdtesting method used in these aforementioned testing platforms and companies mainly concentrates on testing the mobile app as a whole to make sure that the app is free of errors. They have designed based on "On-demand matching and competition" (Mao *et al.*, 2017), which means that crowd testers are selected based on matched requirements (their availability, demographic information, rate, and testing type preferences).

2.2 Automated and cloud-based testing services

Prathibhan *et al.* (2014) designed an automated testing framework over the cloud for testing mobile apps on different Android devices. This framework can provide performance,

functional and compatibility testing. Kaasila *et al.* (2012) presented Testdroid as 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 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 (2012a) proposed remote mobile test system (*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 Huang (2014b) extended the *RTMS* and they proposed *AppACTS* that provides an 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 the 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, unlike *RTMS*, it will not use a cloud of mobile devices, and it uses real remote devices. The MyCrowd QA[10] platform 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 (Huang, 2014b). This service helps 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 the testing report after testing is completed.

Starov (2013) proposed the *CTOMS* framework for cloud-based testing of mobile systems that performs the testing on the remote real mobile device through the cloud (each pool of real mobile devices connects 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.

2.3 Knowledge base development for compatibility testing

StackOverflow is a service based on open crowdsourcing (volunteer work) for addressing technical programing issues (Phair, 2012; Bacchelli *et al.*, 2012). This platform has documentation that focuses on storing questions and issues related to computer programing only (coding issue). It is also used by mobile app developers to assist them with the issues they are facing within the implementation process (programming) of mobile apps, but it is clear they are rarely used for testing (Section 3). The authors in Ham and Park (2014) addressed Android fragmentation problems in mobile application compatibility automated testing. In addition, 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 platform AskCrowd2Test will involve a knowledge base (documentation), especially for documenting issues related to mobile device compatibility testing.

3. The limitations of compatibility testing approaches

In the authors view, several limitations exist in the current testing solutions, which need to be addressed to enhance the mobile device compatibility testing process. These limitations are:

- *Test distribution*: most of the testing platforms or companies, and cloud testing services described previously have a limited distribution of tests, as they target group-specific selection of crowd testers based on general profile characteristics. Thus, they do not provide opportunities to public crowd testers around the world who have limited experience in contributing to testing process and discover more issues. Consequently, these solutions will not be able to study sufficient users' behaviors or interactions with the app.
- *Variety of mobile devices*: the majority of mentioned testing approaches may not cover a large variety of mobile devices or OS versions. This is because of the lack of availability of a very large number of different mobile device models or OS versions in one place like a companys headquarters or its particular crowd community. To the authors' best knowledge, there is not a complete set of different mobile device models with different OS versions in a tests laboratory or a registered cloud testing system to be accessible by mobile app developers as discussed at RTMS (Huang, 2012a), AppACTS (Huang, 2014b), MyCrowd QA[10], as well as CTOMS (Starov, 2013).
- *Coordinated scheduling*: this is an important issue which exists in all mentioned approaches since crowd testers need to share a single pool of mobile devices in various cases; for example, when crowd testers are required to conduct the test across various geographical locations at the same time. Another example is when a particular mobile device needs to be tested with more than one OS versions for different tasks simultaneously. This could lead to issues in the coordination process amongst testers and also to delays in the testing process as it may not cover all the mobile models and OS versions.
- *Knowledge accessibility*: most of the mentioned testing approaches do not have a public place where testing results (including issues and their solutions) can be stored, and which public mobile app developers can access quickly without any constraints. Currently, these results are stored in a private space of each company, which only development teams within the company are able to access. This will not enhance the experience and knowledge of both developers and testers as well as domain knowledge of the mobile app development.
- *Compatibility testing type*: the majority of these testing approaches focus on testing mobile apps themselves to ensure there is not any failure but not for checking and understanding the compatibility of various mobile devices' characteristics with the app requirements. Certainly, this way of testing will not assist mobile app developers to broaden their knowledge and experience level.
- *Impossibility of testing*: stack overflow is an example of an online public library for storing knowledge and experience for programing purposes, not designed for testing. In addition, there is no online space mainly designed for testing purpose where developers can ask to test a specific issue (test case, feature or code) on different mobile device models. Human-based tagging systems might be another drawback found in StackOverflow, where users select the tag based on their feelings or potentially communities surrounding those tags. This might lead to tagging the information in a wrong classification.

AskCrowd2Test platform is designed to satisfy this need.

4. Current crowdtesting working mechanisms

The first crowdsourcing technology followed the common crowdtesting processes (C-CT) shown in Figure 1. At that time, crowdtesting included only four key elements: developer, crowd manager, crowd leader, and crowd testers. Alyahya and Alrugebh (2017) emphasized that C-CT workflow has several limitations. They addressed these limitations by proposing certain automated processes as illustrated in Figure 1:

- In C-CT workflow, there is a lack of a mechanism for monitoring the availability of crowd managers or crowd leaders to supervise new testing projects. They are both assigned manually. This process is not efficient as several projects might be allocated to the same crowd managers or leaders while others are available. They address this

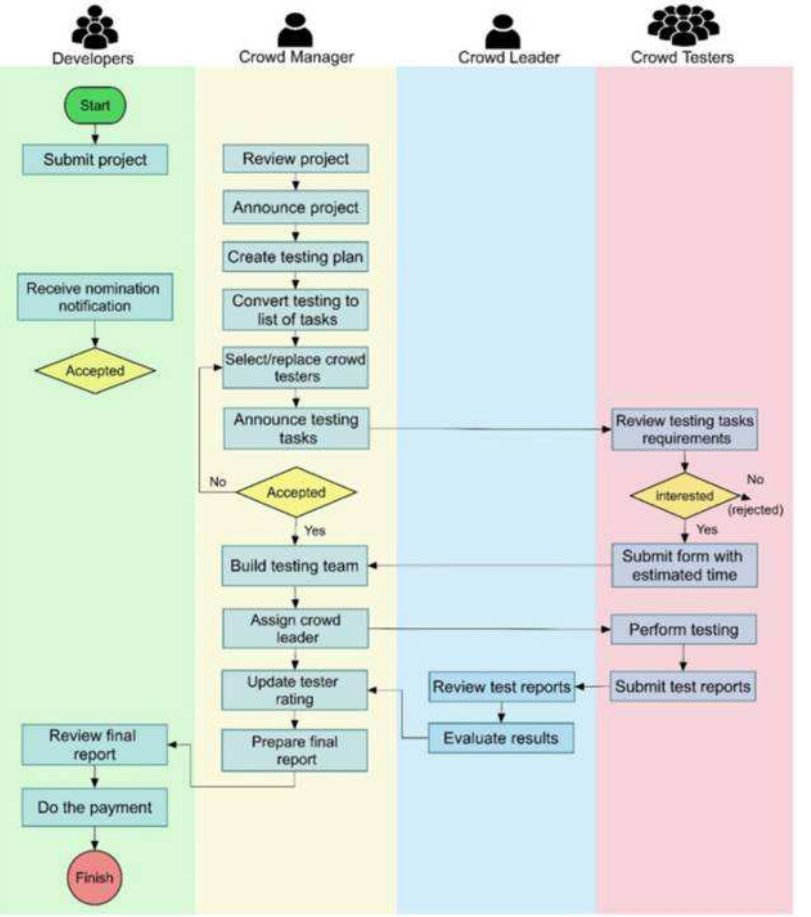

Figure 1.
The common crowdsourced testing workflow (C-CT)

Source: Adapted from Alyahya and Alrugebh (2017)

issue by proposing an automated assigning of crowd managers and leaders to new coming projects based on the best availability of a manager or leader.

- The testing team for any project is created manually. As suitable crowd testers are selected by matching the project requirements with their experience and geographical location from a large list of crowd testers this can be time consuming. This issue was addressed by enabling an automated allocation method based on best fit with project requirements.
- There is also a lack of a method for monitoring the status of the crowd testers for the accepted projects. They may agree to contribute in the testing process but then disappear or fail to submit their test report. An automated method has been proposed to address this problem and avoid sending invitations to crowd testers who are not active for a short period of a workday (e.g. 12 h).

Recently, some of the testing organizations and/or platforms started following the most recent crowdtesting workflow Automated Crowdtesting (Auto-CT) by Alyahya and Alrugebh (2017). The processes of the Auto-CT are illustrated in Figure 2 and described below in details.

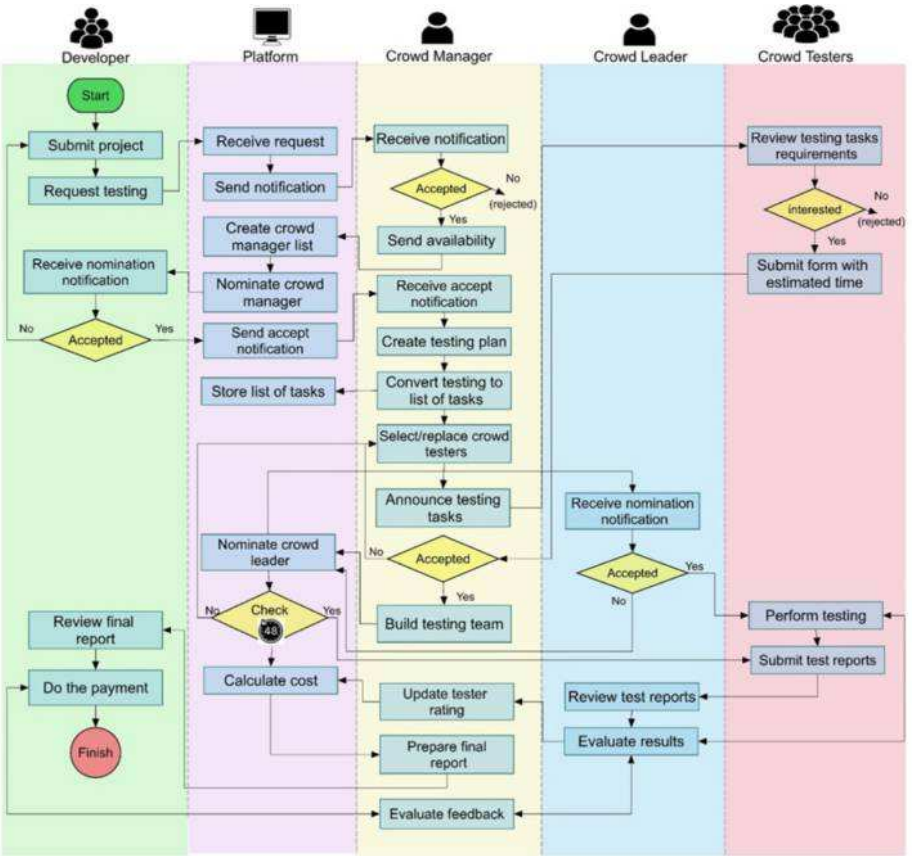

Figure 2.
The most recent crowdsourced testing workflow (Auto-CT)

4.1 Submitting the project

The developers define the project (whole app) for testing automatically or manually and identify the business needs and goals. At this stage, the developer sends a testing request through the platform and specifies the required information: the targeted app, types of testing required, OS versions, mobile platforms and required testing automation tools.

4.2 Selecting crowd manager

The platform will check the availability of crowd managers to supervise the new project. A list of all available crowd managers will be created and an automatic notification will be sent to available crowd managers based on testing requirements like testing type/environment. Once they receive the notification, they should send their availability and agreement to work on the new project.

4.3 Nominating crowd manager

Once crowd managers accept the testing task, they must provide an estimation of when they will be ready to start testing the project. Then, a nominate message (notification) will be sent through the platform to the developers which includes the names of all managers who have accepted the test. The most appropriate (available) crowd manager will be selected by the developer to be the primary crowd manager of the new project.

4.4 Review the submitted project by crowd manager

The selected crowd manager will receive a notification from the developer to supervise the project. The crowd manager reviews the project requirements and then designs the testing plan for the project which includes an estimation of testing effort and testing cost.

4.5 Dividing project to a set of tasks

Once the crowd manager reviews the project requirements and designs the testing plan, they will convert the testing project to a set of tasks to facilitate the testing process. These tasks are then stored under the project name in the platform.

4.6 Selecting crowd testers from the community and announcing testing tasks

After the crowd manager has divided the project into a set of tasks, the platform will automatically check the availability of crowd testers and send an invitation to most the appropriate crowd testers based on several factors such as the quality of the bugs they submit, the crowd's available mobile device type and geographic location.

4.7 Review project and its tasks specification by crowd testers

After crowd testers receive the invitation, they will review the tasks' specifications and send an acceptance form with the estimated time of achieving the test and submitting the reports.

4.8 Build test team

The crowd manager will review the list of crowd testers interested in participating in the test cycle and verify their information to make sure that they satisfy the required qualifications. After that, the crowd manager will build a testing team and recruit the suitable crowd testers for specific tasks.

4.9 Assign crowd test team leader
After the crowd manager has built a testing team, a team leader will be selected by the crowd manager from the list of most available crowd leaders to work under the guidance of a crowd manager. The selected crowd leader will receive a list with all selected crowd testers to start executing the test.

4.10 Execute testing and submit reports
After crowd testers performed the test automatically or manually, they will submit the testing reports to crowd leader using the management tool (e.g. JIRA[11]) specified by the leader and crowd manager.

4.11 Review and validate testing reports by crowd leader
After crowd testers have submitted the testing reports, the crowd leader will review and evaluate the severity of the bugs and then modify the status to one of the following two options pending approval or pending rejection.

4.12 Update crowd testers rating by crowd manager
After crowd leader evaluate testing reports of each crowd tester and validate the bugs, they will refer the bugs to one of these classifications: Exceptionally valuable, very valuable or somewhat valuable, rejected bugs. Based on the classification of bug, the crowd manager will rate crowd testers and store the rating of all crowd testers in the platform.

4.13 Calculate cost
The platform will calculate the cost of all crowd testers that submitted the reports based on their rating (sensitivity of bugs) and price specified by developers for the whole project.

4.14 Prepare final report by crowd manager
Depending on the rating and calculated cost of all crowd testers the crowd manager prepares a final report that includes the name of all crowd testers, results of each tester (e.g. #.of discovered bugs, the sensitivity of bug) crowd testers' rating, and the appropriate cost for each tester.

4.15 Review final report and providing incentives by developers
After the developer receives the final report, they will check and then send feedback to the crowd manager to change the status of each report into approved or rejected. Once the crowd manager updates the status of each report, the approved report will be combined and organized again in a new report with the cost of all approved report. The developer will then check the report, action the payment and finish the testing lifecycle.

5. Observations on the auto-CT workflow

The study of the recent workflow of crowdsourced testing Auto-CT proposed by Alyahya and Alrugebh (2017) showed several significant observations that the authors of this paper believe could negatively impact the crowdtesting process. These observations are:

- The general concept of crowdsourcing is dividing big projects into small tasks that can be easily achieved by crowds. This process takes several steps in Auto-WF. To announce the testing tasks to crowd testers, it is first necessary to convert the submitted project to multiple tasks to be easily performed by the crowd tester. This process needs to be achieved by crowd managers, which requires more effort since

they need to divide the project to small tasks (e.g. divide the app in to small tasks, then to smaller tasks) as well as being time consuming, especially for large projects. This issue has been eliminated in the proposed workflow DT-CT by forcing developers to define and announce their project in the form of a set of small tasks, instead of submitting the whole project and then waiting for an available crowd manager to divide the project to small tasks.

- Partitioning any of the projects into multiple small tasks by crowd managers may lead to forgetting to complete some of these small tasks (e.g. test cases) that might be important for developers to be tested. It may also lead to perform test cases that are not important for the developers and do not affect the apps quality. To improve that, there is a need for a method to avoid testing unnecessary tasks (test cases) that may sometimes be announced by crowd managers. The DT-CT workflow considered this requirement by forcing developers to decide the tasks/test cases by themselves before announcing them. This method will avoid unnecessary testing tasks for developers.
- In Auto-WF, in performing test process, high-level testing (black box testing) can only be conducted by crowd testers where developers occasionally need to test a specific code. The DT-CT workflow addresses this issue by enabling developers to enter and define their code in the task defining process and request crowd testers to test.
- The Auto-CT may be insufficient to distribute the test to a very large number of crowd testers. The testers who have high level of experience are only selected from their community, while in some case it is very good to perform the test by crowd testers who have little-to-no experience. AskCrowd2Test platform and the DT-CT workflow are compatible with all levels of experience and allow unknown public crowd testers with different levels of experience to participate and use their own device for testing. 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 put in more effort to perform the test and provide better testing results. This will also help developers to find more issues and accurately reproduce the behavior of an app that runs on an exact mobile. This may also give the developer excellent insight about how end-users, who do not understand the internal specification of the app, will interact with it.
- Assigning a crowd manager or crowd leader to each submitted project requires several steps. This process can be inefficient and may cause more delay while the developers need to wait for crowd manager to be selected as well as crowd leader. In addition, this process might require developers to spend more budget because of their role in the test cycle, which, in turn may be expensive for the small developer's teams. The DT-CT workflow improves the crowdtesting process by eliminating the delay time taken for assigning both crowd manager and leader; in addition, it decreases the amount of the budgets to be paid. This is achieved by presenting a direct-testing process between developers and crowd testers. The idea of that process is dispensing the role of both crowd manager and crowd leader. Thus, distributing their roles on both developers and crowd testers.
- There is a lack of an internal mechanism to automatically track and organize the submitted reports. The reports are usually collected and organized manually by the crowd leader and reviewed by crowd testers, or through the use of external

management software to organize them such as JIRA, which may be time-consuming. The DT-CT workflow has considered the report tracking steps as one of the most important processes in the crowdtesting workflow between crowd testers and developers. It has supported tracking processes slightly similar to JIRA[11] software.

- There is a lack of a mechanism for monitoring and tracking the status of testing tasks. For example, the project might be international and need to be achieved by large-scale end-users over the world. Another example is the tasks might require testing on specific devices that run specific OS versions. Consequently, the developers need to ensure that the test really covers the requirements. In Auto-CT workflow, the developers will not be sure whether everything is going well during the testing cycle. Crowd managers need to open each report, check it, and then prepare the final report to send to developers. This is can be time-consuming. To investigate this issue an automated process designed in DT-CT that allow developers to easily check their tasks status (e.g. distribution over countries or list of tested mobile devices) during the testing cycle and to make a quick decision when they would like to close the test.
- The mechanisms for monitoring the progress of crowd testers and for avoiding those who are non-productive are insufficient and not equitable. The Auto-CT excludes the crowd testers who are not active for several days or hours after accepting the task without knowing the main reasons. This situation may anger the crowd testers and reduce their enthusiasm to participate in any new projects in the future. To minimize the problem of having non-productive crowd testers, additional financial rewards (bonuses) will be added to the total amount to be paid for active crowd testers in the rewarding process. This could delight crowd testers and motivate them to be more active and to submit their reports on time.
- In Auto-CT, to provide a feedback to crowd testers, many processes need to be completed. The developer first needs to contact the crowd manager and then the crowd manager contacts the crowd leader to reach the crowd tester and provide feedback about the report. To minimize the delay time and effort taken to provide the feed-back, a direct interaction process between developers and crowd testers has been implemented in the proposed DT-CT workflow, which allows the developer to contact crowd testers immediately during the testing cycle or at the end to provide the feedback.
- The selecting mechanism of crowd testers based on their profile matching (e.g. available hardware and software resources, demographic information, geographic location, experiences) with the testing task requirements are not good enough. Each task is different from the other in complexity level or type of test. The proposed selection mechanism improves this process by considering other variables such as reliability, complexity, level of task, type of task, quality of crowd testers and consistency in their work to apply a fair and good selection.

6. Proposed hybrid crowdsourced compatibility testing platform AskCrowd2Test

Manual compatibility testing is the suggested method for testing the mobile apps during the development process (Kamran *et al.*, 2016). The proposed approach has been designed as a web-based platform known as AskCrowd2Test that is 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 the hybrid crowdsourcing method (Section 6.2) and the DT-CT workflow (Section 7) and a CPKB (Section 7.1).

6.1 The concept of a proposed compatibility testing approach

The concept of a proposed compatibility testing approach implemented in AskCrowd2Test platform focuses on testing whether the hardware components of mobile devices and features of OS versions are compatible with the functionalities of different types of mobile apps (e.g. banking, health, social, activity tracking or any other apps). It considers testing the compatibility for all different dimensions of the mobile device and its features with the app such as:

- hardware dimensions such as the camera (type and resolution), sensors, CPU speed, screen (rotation, size, resolution), GPS, Bluetooth, RAM, etc.);
- software dimensions such as different APIs level (minimum supported Application programing Interfaces), multimedia supported, etc.; and
- human behaviors and interaction with the app also considered as a third important dimension to reduce compatibility testing issues such as localization, accessibility requirements, languages, target user of the app, type of environment (e.g. finance, education, medical, business etc.).

Therefore, the proposed approach possibly could answer the following questions:

Q3. 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?

Q4. How much the hardware component is compatible with a specific functionality of a specific app for different mobile device models and OS versions?

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

6.1.1 Low-level testing. A developer can publicly test 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 adopting this 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.

6.1.2 High-level testing. A developer can privately test 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 a full compatibility testing service through the use of the new crowdsourcing method.

6.2 Proposed hybrid crowdsourcing method for testing

In the literature, two basic crowdsourcing methods are used in the software development and testing process: the "traditional crowdsourcing" method (used by most of the testing companies (Mao *et al.*, 2017; Stol *et al.*, 2017)) and the "open crowdsourcing" (Phair, 2012). The traditional crowdsourcing and open crowdsourcing method are similar in that both do not support replication of the tasks which means that they specify a different job for each crowd to complete the testing process quickly within a limited time. These two methods are different in the contribution form. The traditional crowdsourcing normally used is based on

online competition and on-demand matching (Mao *et al.*, 2017; LaToza and Van der Hoek, 2016) 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 (Mao *et al.*, 2017) where the public crowd participate (as volunteers) based on their interests in tasks (Table I).

To avoid these constraints, the authors in this study proposed an alternative crowdsourced testing method, called "hybrid crowdsourcing" method. This method uses the power of the public crowd testers to enhance and facilitate the full mobile device and mobile app compatibility testing process. By employing the hybrid crowdsourcing method in the proposed compatibility testing approach that is described in Section 6.1, 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:

- testing more mobile devices with different OS versions and gathering more accurate results; and
- covering most of the possible compatibility issues in early stages of the mobile app testing process.

This method will support the flexible incentives mechanism which is based on the negotiations between the parties and the agreement for the incentives they are willing to provide or earn. Such an incentive mechanism will help small teams or individual developers with limited resources to test their apps. Table I shows a comparison between the 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.

7. Proposed working mechanism DT-CT

To overcome the observations of the Auto-CT crowdtesting workflow listed in Section 5, a set of improvements have implemented in the DT-CT. These process improvements are listed below:

7.1 Defining and announcing testing tasks to crowd testers

The general concept of crowdtesting is dividing large projects into small tasks that can be easily achieved by crowd testers. In Auto-CT, this process requires several steps by the crowd manager. The DT-CT workflow eliminated these several steps. Developers define their project as a set of small tasks and clarify the requirements for each individual task at the beginning of the test cycle rather than by submitting the whole project and waiting for an available crowd manager to divide the project. These small tasks will be stored on the tasks page with previously defined tasks in the platform, to be visible to all crowd testers. This can be an iterative process, every time developers need to test a new set of tasks, they will be able to add them under the same project. This improvement will no longer require the crowd manager to review the whole project and complete this division process.

7.2 Distribute the test on large-scale and selecting crowd testers

Once the developers finished the process of defining the new testing project with its tasks requirements, they will need to distribute tasks to the public. As shown in Figure 3, there are two main methods for distributing the task and selecting crowd testers in the proposed mechanism:

Table I.
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 and social) Intrinsic as (self-knowledge and learning)	Different types of re-ward not only money Name recognition and satisfaction Knowledge and experience

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.* (2018)
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.* (2018)	The expertise of crowd participating is unknown unless the previous experiments have been taken, the outcome would be unpredictable

Table II.
The advantages and disadvantages of the three crowdsourcing methods

- *Send tasks to external public crowd testers*: developers will need to publish (copy/paste) the random URL link that is automatically generated after finishing defining the process in an online space such as Facebook, Twitter, LinkedIn, blogs or any testers' groups on the internet.
- *Send tasks to internal public crowd testers*: developers here need either to share the task with all registered crowd testers in the platform or to access a crowd testers' dashboard to select specific crowd testers to perform the test based on different selection criteria: consistency of crowd testers work, reliability and high-quality performance in a specific type of testing task or complexity level of the task.

In both internal and external cases, once the developer distributes the tasks a notification/invitation will send to crowd testers. Notice that developers can use both if they would like.

7.3 Review tasks specifications by crowdtesters

Once registered (internal) crowd testers receive the announcement of the test (UR, invitation or notification), as displayed in Section 3, they need to review the specification and requirements of the project to avoid out of scope issues. On the other hand, when non-registered (external) crowd testers get the URL link, they will first need to register to the platform and then review test specifications and requirements. After reviewing the requirements of the announced project, crowd testers will have the option to either accept or reject the project. If they are not interested in carrying out a specific project, the platform

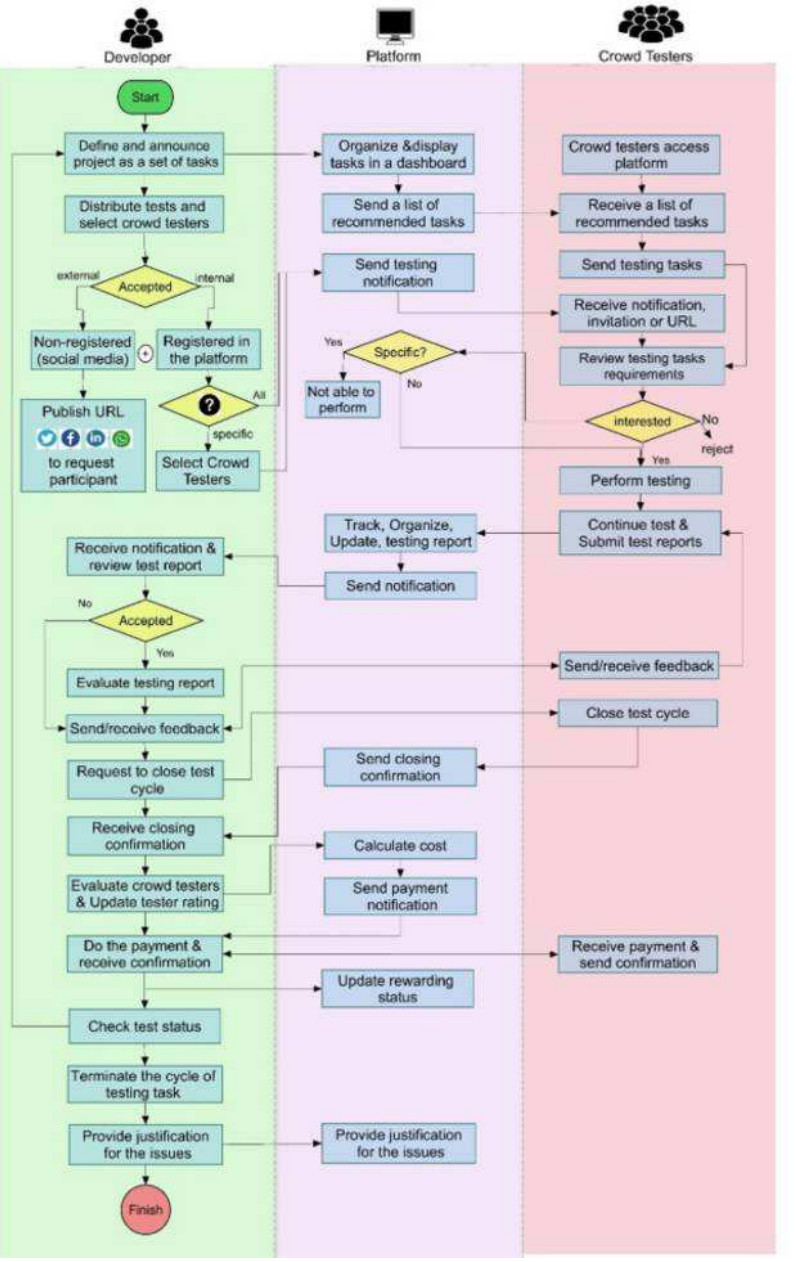

Figure 3.
The proposed crowdsourced testing workflow DT-CT

gives them an opportunity to review the specifications and requirements for all other projects that can be tested. Thus, they could accept more than one project simultaneously and perform them in order; this step can be done in next section D.

7.4 Selecting task by crowd testers

When crowd testers access the platform, a list of recommended tasks will be displayed for them based on matching some important properties such model of the device, type of task, running OS version, complexity level of the task. If the crowd tester is interested in performing a specific test from a list of tasks published in the platform. At this stage, they will not be able to accept the task immediately (Section 3). The platform required first to check if this specific task has been chosen before as a private task (inviting specific crowd testers to perform) or it has been distributed publicly to all crowd testers. If it is public then they will be able to review the requirements, accept and conduct the project (Figure 3).

7.5 Execute testing tasks and submit reports

Crowd testers will start testing the tasks associated with the project through the use of required types of device models and OS versions. As we mentioned previously one project may include more than one task. When the crowd testers accept the whole project, all the tasks belonging to this project will be added directly to their file. This means that crowd testers can execute each task separately. After finishing the test, they must submit a single report to developers for each task executed through the submission form within the platform.

7.6 Tracking testing reports

After crowd testers are executing different tasks for different projects at different times, developers need to review all reports including issues that have been reported. To avoid the limitations of collecting and organizing testing report manually or automatically using external tools by crowd manager or crowd leader as in Auto- CT that used by most of the current platforms. The DT-CT implements a report tracker system slightly similar to JIRA[11], ZOHO[12]. This system collecting, tracking and organizing all testing reports submitted by crowd testers is automatically based on two main factors: last submitted time and sensitivity of issues. Each time new report is submitted the tracking system will notify developers it is available for review. This tracking system will achieve the following:

- *Task status tracking*: track the status of each task within each specific project;
- *Detailed report*: track master details regarding the issues Id, type, description, priority, etc.;
- *Testers identity*: track the crowd testers' details who report the results;
- *Modification method or issue tracking life cycle*: ability to modify an existing issue status, open, in progress, under review, Invalid/rejected, accepted or fixed;
- *Scheduled reports*: track recent updates on the results based on daily, weekly, or hourly; and
- *Notification method*: notify developers about any new issues reported.

7.7 Review and validate testing reports by developers

After tracking and storing reports in the platform, the developers must review and validate the results provided in every single report. If the results in the report are not accurate, the

developer will directly reject the report and send evaluation feedback to crowd testers to retest or update the report. In that case, crowd testers will receive the developer's feedback and then review it, to update the report and send it again to the developer. On the other hand, if the results are correct and a developer has approved the results, then a notification will be sent to crowd testers to inform them of the status of their report, that it is deferred, in progress or fixed.

7.8 Finish testing cycle by crowd testers

Once the crowd tester receives the validation feedback that the issue has been fixed, they will retest the project to check that, and send a notification to the developers that the issue has totally fixed. After the developer receives the notification, they will request the crowd tester to close the test. Once the crowd tester closes the test, the platform will send a confirmation to the developer that the test has been closed.

7.9 Evaluate crowd testers by developers

The developers evaluate crowd testers based on the quality of the report (correctness of results, how is distinct from other results reported, and whether there is any solution or suggestion provided), on time delivery, complexity level of task, and type of testing task instead of evaluating crowd testers based on working hours, the number of discovered bugs, and sensitivity of bug as in the previous workflows C-CT and Auto-CT. In addition, in proposed work, accepting and performing as many as possible of testing projects will also positively influence the developer's evaluation for crowd testers. On the other side, rejected reports will negatively effect on the crowd testers' evaluation. In the proposed crowdtesting approach, because of dealing with public crowd testers, quality and accuracy of test results and reliability level of crowd testers have been considered significant characteristics. These characteristics are important factors for reducing developers' concerns (working with unknown testers) and increasing their confidence in crowd testers' performance and the results produced.

7.10 Calculate cost

The platform will monitor and calculate the cost of each report submitted by the crowd tester based on the report quality level, on time delivery, and the crowd tester's reliability. Once the calculation process is finished, a payment notification will be sent to the developer with the final price for each tester. At this stage, the platform will check the status of all tasks added by a specific developer, which have been closed by crowd testers, and then sends a notification with a list of all crowd testers who have not yet been paid.

This process will minimize the time-consuming problem of the crowd leader or crowd manager to monitor and identify a list of crowd testers that need to be rewarded, and sending the list to developers to perform the reward process.

7.11 Provide incentives by developers

After the developer receives the payment notification from the platform for each crowd tester who has finished and closed their test cycle, they will achieve the payment process immediately. Later, the platform will update the incentive status from a non-paid task to a paid. This will facilitate the testing process, minimizing the waiting time of crowd testers to receive the reward which may motivate the testers more and more to perform other tests. Unlike other workflows that require crowd testers to wait for other crowd testers to finish

their test and for the crowd manager to prepare, the final report and submit it to developers to get paid.

7.12 Checking task status and terminate project or task by developers

During the testing process, the developers may need to check the status of each task. For instance, if there are specific device models with a specific OS version that has not yet been tested. In that case, the developer will be able to go to the first stage "defining and announcing task" and change the required information to the devices not yet covered. Whereas, if all the devices have been covered then the developer will end the task so that no crowd testers can test it later.

7.13 Justify reasons of issues by developers

Once the developer finishes the validation process of all submitted reports related to testing a particular task, the developer should have justified reasons for all discovered issues on these tested mobile devices and OS version to be stored in the knowledge base. This process is not available in current crowdsourcing workflow. Implementing this process could 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.

7.14 A crowd-powered knowledge base

The proposed approach includes a knowledge base that documents collected compatibility testing results (including issues and their solution) through the crowd testers within the platform. This knowledge base will bring supporting evidence to enable developers to understand the reasons why specific issues appeared. Also, it will allow developers to obtain 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 a certain hardware component or characteristics necessary for the proper running of an app is missed in any device model, or is not supported by any API framework related to specific OS versions. Documenting these results will help developers substantially in the future when they need to develop 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 the early stages of the development cycle. In addition, it provides the 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.

8. Conclusions

In this paper, the research gap is narrowed regarding studying the reasons behind issues of compatibility testing. This paper has a review of current compatibility testing approaches and their imitations. To address these limitations, AskCrowd2Test has been presented as a new crowdsourced compatibility testing platform for testing the hardware characteristics of mobile devices, piece of code, or features of OS versions, against the specifications of mobile applications. This paper has described the development procedures, which has been achieved to build AskCrowd2Test platform. First, the proposed Hybrid Crowdsourcing method has been discussed with a comprehensive comparison carried out to distinguish between the proposed method and other crowdsourcing methods such as Traditional crowdsourcing and open crowdsourcing. Second, a review of current crowdtesting workflows has been provided in Section 4. In addition, a clear discussion of the proposed

crowdtesting workflow DT-CT has been presented. Further, a brief comparison between the proposed and previous crowdtesting workflows has been elucidated with the illustration of observations that the authors of this study believe can be the source of weaknesses (Section 5).

The results of this paper show that the use of AskCrowd2Test with the proposed hybrid crowdsourcing method and the new DT-CT workflow has numerous advantages. There is sufficient improvement in testing and developing of mobile apps via participation of unknown public crowd testers. It enables the research community to cover a large variety of devices and various OS versions. It authorizes unknown public crowd testers with different levels of experience to contribute and carry out compatibility testing, which can help in discovering more concrete results. AskCrowd2Test platform may facilitate the app development process by studying end users' interaction and the compatibility of mobile devices and OS properties with applications. The proposed approach also avoids mistakes amongst crowd testers when they report about testing mobile devices through a mobile application and an automated data detection technique. Further, there are also improvements in the testing process of mobile apps via the direct interaction between crowd testers and developers, this facilitates the tasks announcement process and reduces the required time for segmenting the project to small tasks, avoiding testing unnecessary tasks. In addition, it enables developers to directly monitor their task status at any time. Delay in the reduction during communication among crowd testers and developers has been achieved. This work also reduces the delay in assigning the crowd manager or crowd leader to collect while they wait for collecting and organizing testing reports. The authors expect that AskCrowd2Test platform may facilitate the app development process by studying end-users interaction and the compatibility of mobile devices and operating system properties with applications.

The potential downside of the proposed crowdtesting methodology is that while it reduces the cost on the one side it increases the workload of the developers on the other side. While this may increase the developers' workload, for small development teams it is still far more important, because of the limited income, is to reduce the time needed to complete the tests and reduce the cost paid to crowd testers. The developers increased workload is partly because of the sharing of the work, normally done by the tests managers and team leaders, with the testers and it may not be very significant. An experimental test will be carried out in the future to evaluate the impact on the total cost. The authors of this paper believe, in principle, that this methodology will have positive impacts. For future work, the authors also plan to evaluate the proposed crowdtesting method and DT-CT workflow to demonstrate its advantages and effectiveness in addressing the issues of compatibility testing in a significant way.

9. Contribution

The main contributions in this paper are:

- A crowdsourced testing platform AskCrowd2Test that supports the distribution of full mobile device compatibility testing, which is suitable for both professional and beginner developers and testers.
- It lays out a new crowdsourced testing workflow DT-CT that can be used to execute other mobile apps testing types for all supported platforms.
- Proposed a hybrid crowdsourcing method that can be used in any field not only mobile app testing or software testing.

- A novel direct interaction mechanism between the developer to provide better coverage of testing in specific software or hardware that is in need of testing by the crowdtesting platform.
- A public CPKB that is documenting a good development knowledge foundation required for an understanding of the overall architecture of mobile devices, especially for Android devices.

Notes

1. Available at: https://source.android.com/devices/sensors/index.html
2. Available at: https://mycrowd.com
3. Available at: www.pay4bugs.com
4. URL: www.mob4hire.com
5. URL: www.pay4bugs.com
6. URL: www.utest.com/
7. URL: www.passbrains.com/
8. Available at: https://go.globalapptesting.com/
9. URL: www.applause.com/
10. Available at: https://mycrowd.com
11. Available at: www.atlassian.com/software/jira/
12. Available at: www.zoho.com/crm/

References

Akour, M., Al-Zyoud, A.A., Falah, B., Bouriat, S. and Alemerien, K. (2016), "Mobile software testing: thoughts, strategies, challenges, and experimental study", *International Journal of Advanced Computer Science and Applications*, Vol. 7 No. 6, pp. 12-19.

Alyahya, S. and Alrugebh, D. (2017), "Process improvements for crowdsourced software testing", *International Journal of Advanced Computer Science and Applications*.

Bacchelli, A., Ponzanelli, L. and Lanza, M. (2012), "Harnessing stack overflow for the ide", *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, IEEE Press., pp. 26-30.

Bhojan, R.J., Vivekanandan, K., Ganesan, S. and Monickaraj, P.M. (2015), "Service based mobile test automation framework for automotive HMI", *Indian Journal of Science and Technology*, Vol. 8 No. 15.

Ham, H.K. and Park, Y.B. (2014), "Designing knowledge base mobile application compatibility test system for android fragmentation", *International Journal of Software Engineering and Its Applications*, Vol. 8 No. 1, pp. 303-314.

Huang, J.F. (2012a), "Remote mobile test system: a mobile phone cloud for application testing", *IEEE 4th International Conference Cloud Computing Technology and Science (CloudCom)*, pp. 1-4.

Huang, J.F. (2014b), "AppACTS: mobile app automated compatibility testing service", *2nd IEEE International Conference in Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 85-90.

Kaasila, J., Ferreira, D., Kostakos, V. and Ojala, T. (2012), "Testdroid: automated remote UI testing on android", *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia, ACM*, p. 28.

Kamran, M., Rashid, J. and Nisar, M.W. (2016), "Android fragmentation classification, causes, problems and solutions", *International Journal of Computer Science and Information Security*, Vol. 14 No. 9, p. 992.

Knott, D. (2015), *Hands-on Mobile App Testing, Pearson education Inc, IN*.

LaToza, T.D. and Van der Hoek, A. (2016), "Crowdsourcing in software engineering: models, motivations, and challenges", *IEEE Software*, Vol. 33 No. 1, pp. 74-80.

Liang, H., Wang, M.M., Wang, J.J. and Xue, Y. (2018), "How intrinsic motivation and extrinsic incentives affect task effort in crowdsourcing contests: a mediated moderation model", *Computers in Human Behavior*, Vol. 81, pp. 168-176.

Mao, K., Capra, L., Harman, M. and Jia, Y. (2017), "A survey of the use of crowdsourcing in software engineering", *Journal of Systems and Software*, Vol. 126, pp. 57-84.

Phair, D. (2012), "Open crowdsourcing: leveraging community software developers for IT projects", *Doctoral dissertation, Colorado Technical University*.

Prathibhan, C.M., Malini, A., Venkatesh, N. and Sundarakantham, K. (2014), "An automated testing framework for testing android mobile applications in the cloud", *Advanced Communication Control and Computing Technologies (ICACCCT), International Conference on IEEE*, pp. 1216-1219.

Samuel, T. and Pfahl, D. (2016), "Problems and solutions in mobile application testing", *International Conference on Product-Focused Software Process Improvement*, Springer, Cham, pp. 249-267.

Starov, O. (2013), *Cloud Platform for Research Crowdsourcing in Mobile Testing, East Carolina University*.

Stol, K.J., LaToza, T.D. and Bird, C. (2017), "Crowdsourcing for software engineering", *IEEE Software*, Vol. 34 No. 2, pp. 30-36.

Wnuk, K. and Garrepalli, T. (2018), "Knowledge management in software testing: a systematic snowball literature review", *e-Informatica Software Engineering Journal*, Vol. 12 No. 1, pp. 51-78.

Corresponding authors
Qamar Naith can be contacted at: qhnaith1@sheffield.ac.uk and Fabio Ciravegna can be contacted at: f.ciravegna@sheffield.ac.uk