ZIPT: Zero-Integration Performance Testing of Mobile App Designs

Biplab Deka[1] **Zifeng Huang**[1] **Chad Franzen**[1] **Jeffrey Nichols**[2] **Yang Li**[2] **Ranjitha Kumar**[1]
[1]University of Illinois at Urbana-Champaign [2]Google Inc.
{deka2,zhuang45,cdfranz2,ranjitha}@illinois.edu, jwnichols@google.com,yangli@acm.org

ABSTRACT
To evaluate the performance of mobile app designs, designers and researchers employ techniques such as A/B, usability, and analytics-driven testing. While these are all useful strategies for evaluating known designs, comparing many divergent solutions to identify the most performant remains a costly and difficult problem. This paper introduces a design performance testing approach that leverages existing app implementations and crowd workers to enable *comparative* testing at scale. This approach is manifest in ZIPT, a zero-integration performance testing platform that allows designers to collect detailed design and interaction data over *any* Android app — including apps they do not own and did not build. Designers can deploy scripted tests via ZIPT to collect aggregate user performance metrics (e.g., completion rate, time on task) and qualitative feedback over third-party apps. Through case studies, we demonstrate that designers can use ZIPT's aggregate data and visualizations to understand the relative performance of interaction patterns found in the wild, and identify usability issues in existing Android apps.

ACM Classification Keywords
D.2.2 Software Engineering: Design Tools and Techniques

Author Keywords
App design; design support tools; zero-integration performance testing

INTRODUCTION
Throughout the mobile app design process, designers seek to understand the artifacts they build and the experiences those artifacts confer on users. During ideation, designers attempt to evaluate the relative merits of potential satisficing designs. Before design specifications are sent to the engineering team, designers endeavor to detect usability issues. Once an app is implemented, designers attempt to optimize user performance metrics and benchmark them against competitors.

To evaluate the performance of mobile designs, designers employ a number of testing techniques which generally fall into one of three categories: A/B [16], usability [21], and analytics-driven testing [7]. While these techniques all provide value to designers and elevate the practice of design beyond guesswork and intuition, the costs associated with optimizing design performance can be steep. A/B and analytics-driven testing require substantial engineering resources to build out alternative solutions and instrument them for testing. Manually aggregating unstructured usage data (e.g., video and audio streams) acquired during usability testing is costly and time-consuming [22].

This paper presents a design performance testing approach that makes it economically feasible to answer the fundamental question of app design: "given the space of possibilities, which solution performs best?" This approach is based on the key observation that, given the millions of mobile apps available today, it is likely that *any* UX problem a designer encounters has been tackled many ways by a number of existing apps. In this paper, we leverage existing app implementations to enable *comparative* testing at scale.

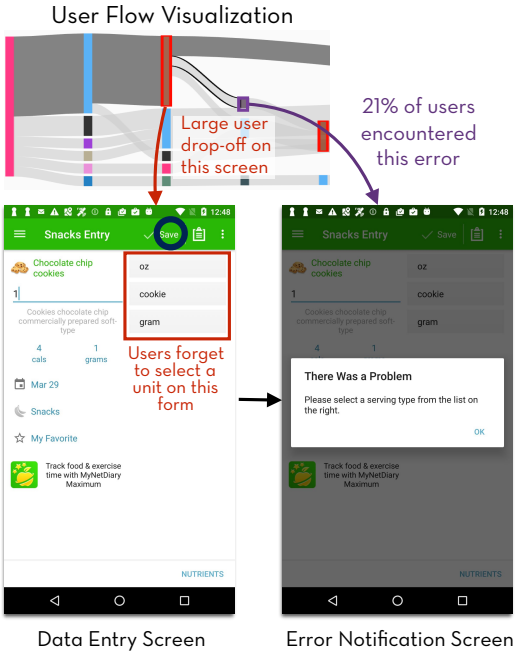

Figure 1: ZIPT helps designers discover usability issues in existing Android apps. The user flow visualization shown here summarizes the paths crowd workers took while trying to "add a cookie to their daily food log" in *MyNetDiary*, a calorie counter app. Using the flow visualization, designers were able to quickly identify a UI screen with a large user drop-off and its associated usability issues.

UIST 2017, October 22–25, 2017, Quebec City, QC, Canada

ISBN 978-1-4503-4981-9/17/10…$15.00
DOI: https://doi.org/10.1145/3126594.3126647

This approach is manifest in ZIPT, a zero-integration performance testing platform that allows designers to collect detailed design and interaction data over *any* Android app — including apps they do not own and did not build — and correlate this data with quantitative metrics and qualitative feedback. Analytics and A/B testing tools require access to an app's source: at a minimum, a developer must instrument the app with tracking code to begin collecting data. ZIPT, in contrast, provides comparable functionality with *zero* integration: it requires no access to code, and can be deployed by any user over any app in the Android store. Unlike usability testing platforms, ZIPT *automatically* captures and aggregates structured interaction data from app usage: therefore, it can provide user performance data for third-party apps at a tiny fraction of the cost (around thirty cents a user versus $50 on a usability testing platform like usertesting.com).

ZIPT builds upon prior work by Komarov et al. demonstrating that crowdsourcing is a viable option for conducting performance evaluations over user interfaces [18]. To evaluate design performance via ZIPT, designers use a web interface to specify apps to test, tasks they want crowd workers to perform on those apps (e.g., "search for a product, and add it to the shopping cart"), and survey questions for workers to answer. ZIPT then launches the tests on a crowdsourcing platform, installs and runs the specified apps on a mobile device farm, and streams the app screens to workers browsers, allowing workers to interact with the app while collecting detailed interaction and design data in the background. Once data for a test has been collected, ZIPT computes visualizations that allow designers to quickly understand aggregate interaction and performance metrics (e.g., completion rate, time on task), as well as inspect individual user traces and screens to identify usability issues.

To understand the types of design insights that this approach can generate, we used ZIPT to collect and analyze performance data for tasks from 10 different apps, crowdsourcing 15–50 user traces for each one. We used ZIPT to identify usability issues in three apps by examining both common and unusual user traces. For three remaining tasks, we performed a comparative analysis between two popular Android apps whose implementations had significant design differences. Results from an informal evaluation with five app designers demonstrate that ZIPT can be used to understand the relative performance of interaction patterns found in the wild, and identify usability issues in existing Android apps (Figure 1).

BACKGROUND & FORMATIVE STUDY

This work builds on prior work on example-based design. Examples are used early in the design process for inspiration, understanding the space of possible solutions, and identifying reusable design patterns [12, 13, 20]. Researchers have built search engines to help designers find relevant extant designs in a number of domains, including web design [25, 19] and mobile app flows [10].

Prior to building ZIPT, we conducted interviews with five Google employees — two interaction designers, two UX researchers, and one UX Engineer — to understand how practitioners leverage examples in their current work, and to discuss how they might incorporate crowdsourced performance data on extant mobile designs. Echoing the literature, all participants referenced examples early in the design process, especially for competitive analyses. The interaction designers, in particular, reported frequently studying existing flows — logical sequences of screens for performing a task — in competitors' apps and manually comparing them.

Although mere knowledge of the existence of a design example can be useful, the practitioners we interviewed often expressed a desire to go further and understand how well an example performs: whether users can understand a design and use it as intended to accomplish their goals. When choosing examples to build upon, participants also considered how easy it would be to communicate the benefits of a particular design pattern to the rest of their team. All participants indicated that they do not turn to examples enough and wished to do so more frequently.

All participants expressed interest in crowdsourcing user performance data to evaluate existing design patterns. Participants offered different ways in which this data might be useful in their design process. P3 observed *"if getting such data was easy, I would ask team members to generate it for patterns they see in the wild before they bring it to me to build a prototype."* P2 said *"I would use such a process to quickly pinpoint the 'what' (what works or does not work) and use other methods to find out the 'why'."* P4 said *"I can think of two ways I would use such data. One is for finding unexpected behavior and the other is for storytelling."*

To conduct remote usability tests, participants wished to collect performance and interaction data over specific tasks in apps. They requested aggregate usability metrics such as completion rate, as well as the average time and number of interactions required to finish the specified task. To identify common usage patterns and unexpected behavior, designers also wished to examine the interaction paths users took to accomplish a task. At an app level, designers hypothesized that it would be useful to collect general user perceptions around usability and branding.

While participants were generally positive about crowdsourced usability testing, they expressed concerns around ensuring uniform device setup across workers: they were wary of trusting crowd workers to follow configuration steps such as software installation. Although participants indicated that they might use crowdsourcing to test their own apps, several expressed reservations about inadvertently exposing features that were not yet public.

INTRODUCING ZIPT

The ZIPT platform comprises three stages (Figure 2). In the first stage, designers define the *scripted* usability tests they want to run over a set of apps. In the second stage, ZIPT deploys the user-defined tests to crowd workers. Finally, in the third stage, ZIPT computes and presents designers with aggregate performance metrics and visualizations based on the data collected from crowd workers.

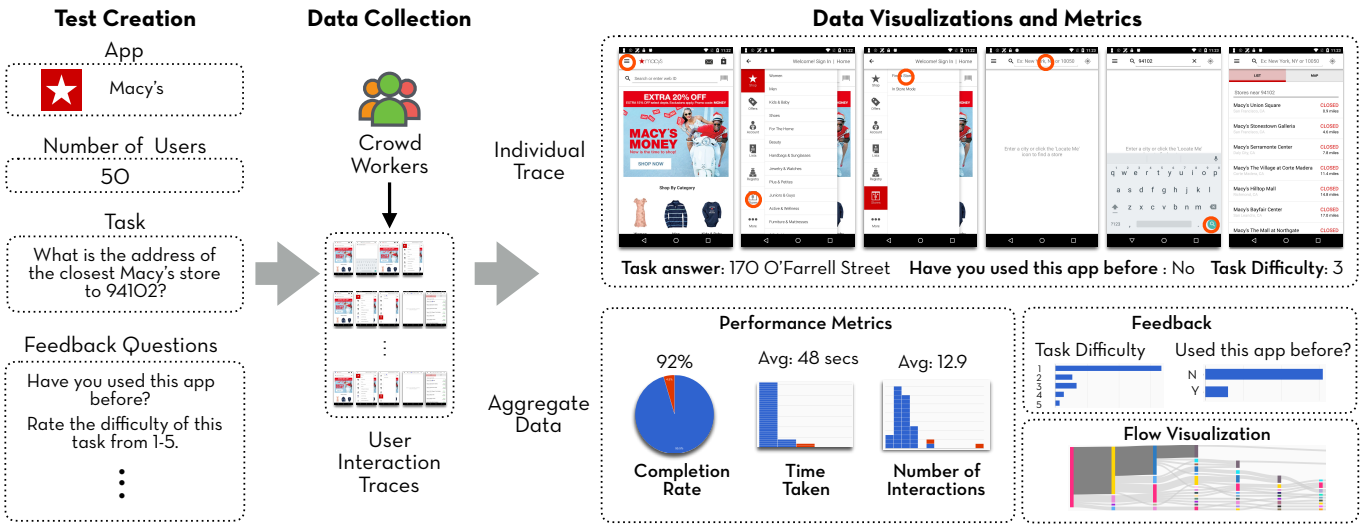

Figure 2: An overview of ZIPT: designers define usability tests (left), which are deployed to crowd workers (center). ZIPT computes and presents designers with aggregate performance metrics and visualizations based on the data collected from crowd workers (right).

Usability Test Creation

ZIPT allows designers to define a usability test by uploading a set of Android APK files, specifying the number of traces to be collected, and providing a description of the task for crowd workers to perform. The description can be open-ended (e.g., "browse for music"), or very specific (e.g., "create a playlist with three songs"). It can be written in plain text or in the Jade templating language if designers want to customize its presentation.

Designers can customize tasks in two additional ways. First, they can construct tasks with built-in verification, requiring a user to provide an answer to a question in addition to performing a task in the app (e.g., "what is the address of the store closest to the 94102 zip code?"). Phrasing tasks as questions can be useful for quickly identifying traces where users could not complete the task. Second, designers can provide a specific starting point for crowd workers within the app other than the home screen. This feature is useful for testing specific flows that a user would encounter deeper within an app. Designers specify custom starting points by demonstration. ZIPT runs the app within the browser, records interaction paths taken by designers, and replays those interactions before presenting the app to a crowd worker.

ZIPT requires each crowd worker to self-report whether or not they successfully completed the specified task. It also allows researchers to add other pre- and post-task questions, which can be used to collect demographic data for cohort analysis and qualitative feedback to understand usability and brand perception. ZIPT supports questions with Likert-scale, multiple choice, and free-form text answers.

Visualizations and Metrics

Once designers submit a test, ZIPT deploys it on Amazon Mechanical Turk. As crowd workers use the apps, the platform automatically captures design and interaction data streams. After collecting the requisite number of user traces, ZIPT computes aggregate metrics and visualizations, and presents a results dashboard comprising four types of data:

Individual Traces. ZIPT allows designers to inspect each crowd worker's interaction path through the app. User traces are presented as sequences of screenshots with user interaction annotations. Each trace also contains a user's answers to task and feedback questions, if any were specified by the designer.

Performance Metrics. ZIPT computes and presents three performance metrics: completion rate, time on task, and number of interactions performed. The performance dashboard visualizes completion rates as pie charts, and the other two metrics as histograms. Researchers can interactively select different performance segments in these visualizations to view their associated traces.

Qualitative Feedback. The dashboard also collates crowd workers' answers submitted in the pre- and post-task questionnaires. It uses bar charts for summarizing Likert-scale and multiple-choice answers. Like the performance visualizations, designers can select segments in these charts to inspect corresponding traces. For example, designers can quickly identify and inspect all traces where crowd workers reported high task difficulty to uncover potential usability bugs.

Flow Visualization. ZIPT presents interactive flow visualizations to help designers compare and contrast the different paths taken by users to accomplish tasks. These visualizations build on prior work on representing user flows in mobile apps with Sankey flow diagrams [14]. Color-coded nodes representing different screens in a user interaction trace are arranged sequentially along the horizontal axis. The nodes in the visualization are connected by bands whose thickness is directly proportional to the number of users who took the path defined by its node endpoints. These diagrams can allow

designers to quickly understand aggregate interaction data, as well as inspect individual screens to identify usability issues. For example, designers can interact with the nodes to view screenshots, and with the bands to see user interaction between two screens. Additionally, a designer can create a flow by demonstration in an app and use it to highlight paths of interest in the visualization. For example, by defining a *golden trace* for a task — the intended path a user should take to complete a task — and highlighting this path in the diagram, a designer can quickly determine the screens that cause the most confusion and are most likely to prevent users from completing the task.

Implementation

To ensure that all crowd workers use apps under identical conditions without having to install any software on their devices, ZIPT uses a web-based streaming approach for collecting data from mobile apps. Apps run in a controlled environment on a mobile device farm, and are streamed to client devices through the browser. This streaming approach has been used in recent systems including ERICA for mining mobile app designs [10] and MobiPlay for implementing record and replay systems for app testing [23], and been piloted by Google for mainstream Android app usage [2].

We modeled ZIPT's web-based app streaming implementation after ERICA [10]. Apps are run on a set of Android devices connected to a server that hosts the ZIPT web application. The web application continuously streams the phone screens as compressed JPEG images. When users interact with these images in their browser, the front-end web application captures these interactions and sends them to the phones via the server. When an app's screen changes as a result of these interactions, users see the updated screens in their browser with a slight delay. ZIPT supports diverse interactions including tapping, scrolling, two finger pinch-and-zoom, and keyboard text entry.

In contrast to ERICA and MobiPlay, where app streaming occurs over a local network, ZIPT must be performant over the public Internet. As a result, ZIPT uses a low frame rate (5–10 frames per second) and high-compression ratio to minimize bandwidth requirements. To produce acceptable latencies, ZIPT only recruits crowd workers within the US (where the server is located). In addition, ZIPT logs the incurred latency for each user interaction, and can filter out traces impeded by network performance.

Like ERICA [10], ZIPT logs screenshots, view hierarchies, and interactions produced during usage, and combines them to produce interaction traces. ZIPT computes completion rates based on user-reported data, and the other two metrics — time on task and number of user interactions — directly from the interaction traces. When examining individual traces, a designer can fix any mislabeled traces, where a user incorrectly self-reported the completion of a task. ZIPT factors in network and server-to-phone communication latencies when computing time on task for a more accurate estimate. In the future, ZIPT can be extended with mechanisms for outlier detection [18] or data validation [8, 6, 24, 24, 17, 11] as necessary.

To construct flow visualizations, ZIPT automatically computes the set of screens that represent the same UI state and merges them together into a single node. To do so, ZIPT employs the content-agnostic similarity heuristic presented in Deka et al. [9], which compares screens based both on their structural and visual properties. Designers can also manually merge and split nodes to fine-tune the visualization.

ZIPT IN ACTION

We used ZIPT to identify usability issues in — and compare the performance of — a number of popular apps. In this section, we present a series of case studies to demonstrate how the data collected and visualized by ZIPT can provide novel design insights for apps used by millions of people. For these case studies, we selected 10 popular Android apps that target a diverse set of day-to-day consumer activities. Via ZIPT, we crowdsourced performance data over tasks central to these apps, collecting between 15–50 user traces for each task. We paid Amazon Mechanical Turk workers $0.30 for each trace and used a server backed by three Android devices, which allowed us to collect approximately 15 traces per hour.

Identifying Usability Issues

ZIPT demonstrates that combining aggregate flow and individual trace visualizations allows designers to quickly identify potential usability problems in apps. Researchers can detect usability problems by examining both common and unusual user traces. For example, if many users drop off at a common node in the flow visualization, this may signify a usability problem in the critical path of the task. On the other hand, if most users successful follow the golden path but a small number are unable to complete the task, this may indicate a usability issue in a part of the app that users are not able to easily recover from. We encountered both cases in the user traces we crowdsourced with ZIPT.

In one test, we asked crowd workers to find the next train's departure time from a specific station on the *Transit* app, which provides real-time information updates on public transportation systems. The easiest way to complete the task is to search for the station using the search bar at the top of the app's home screen. From the flow visualization, we observed that only 21% of users (6 out of 28) initially attempted to search for the answer (Figure 3). We hypothesize that more users would have tried the search bar if it was more visually prominent: many search bars have a box around them or a high contrast line underneath them.

In another test, we used ZIPT to ask crowd workers to "add a cookie to your food log" on *MyNetDiary*, an app used to track calories. From the resultant flow visualization, we noticed a large user drop-off on the data entry UI (Figure 1), and that 21% (5 out of 24) of users encountered an error on the next screen. We discovered that the error occurs when a unit (oz, cookie, gram) is not selected on the data entry screen. We hypothesize that the unit options are not easily recognizable as radio buttons, and positioning them to the side of the quantity field may have exacerbated the issue since users are accustomed to scanning forms vertically rather than horizontally [1].

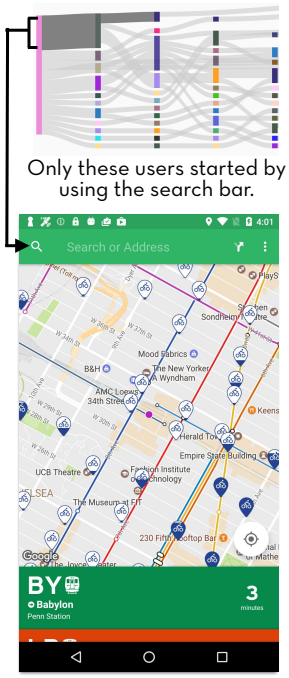

Figure 3: 76% percent of *Transit* app users did not initially use the search bar to find the departure time of the next train.

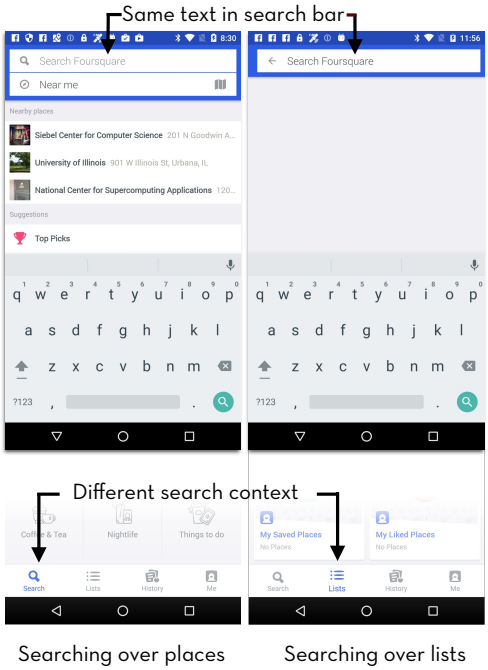

Figure 4: The *Foursquare* app presents search bars with different *scope* based on which tab is selected. However, the placeholder text does not change when tabs are switched to indicate a scope change, which can confuse users.

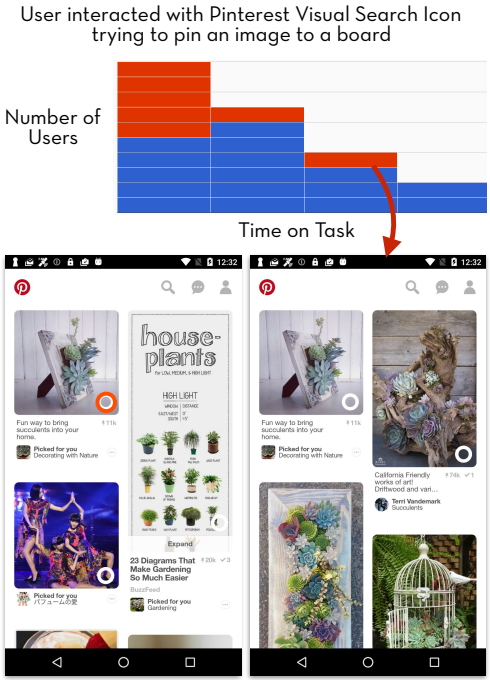

Figure 5: We discovered a potential usability issue on *Pinterest* by inspecting an outlier's trace. After spending considerable time on the task, this user was unable to add an image to a board because the new visual search icon (white circle) behaved in an unexpected way.

By examining *outliers* — users who fail to complete a task after spending considerable time on it — ZIPT makes it possible to uncover usability issues that are not directly related to the specified task. For instance, we asked crowd workers to find a specific restaurant's phone number on *Foursquare*, a social app for discovering places. Most crowd workers easily completed the task; however, we noticed that one user became stuck searching for the specified restaurant in the "lists" tab. Foursquare uses *scoped search*, a design pattern where search bars on different screens search over different types of entities. On Foursquare, the "search" tab supports search over places, while the "lists" tab supports search over user lists (Figure 4). Although scoped search is a popular design pattern, when the scope is not communicated clearly, it can lead to usability problems [3]. In this app, even though the scope changed between tabs, the placeholder text in the search box remained the same.

In another test, ZIPT helped uncover a usability issue on *Pinterest*, a social, visual inspiration app. Pinterest recently added a visual discovery feature that allows users to quickly search for similar images by clicking on a white circular icon on the bottom right corner of an image (Figure 5) [5]. We asked crowd workers to "pin an image to a board," and then identified and inspected the trace of an outlier who had fixated on the visual discovery icon, tapping it several times on different images before abandoning the task. Commonly, an icon placed on an entity such as an image brings up a menu for additional actions that can be performed on it: we suspect that other users will have to "unlearn" their expected behavior on the Pinterest app.

Analyzing Comparative Performance

ZIPT affords designers — for the first time — the ability to perform *comparative* analysis at scale. During ideation, designers can leverage existing implementations to understand the relative performance of different design patterns. After implementation, ZIPT allows designers to benchmark against competitors' apps. We demonstrate how ZIPT's quantitative and qualitative performance data can be combined to conduct comparative performance analyses. Moreover, researchers can inspect flow visualizations and individual traces to better understand *why* these performance differences exist, uncovering usability issues, user expectations, and general trends in user behavior.

Finding Store Locations in Shopping Apps

Big-box retailer apps use different navigation patterns such as *hamburger icons with flyout menus* and *tabbed navigation* to expose their main capabilities. To understand the relative performance of these patterns, we analyzed how users perform store location searches on two popular retail apps — *Macy's* and *Best Buy*. Finding a specific store location is an important feature of these apps, since it directly impacts revenue.

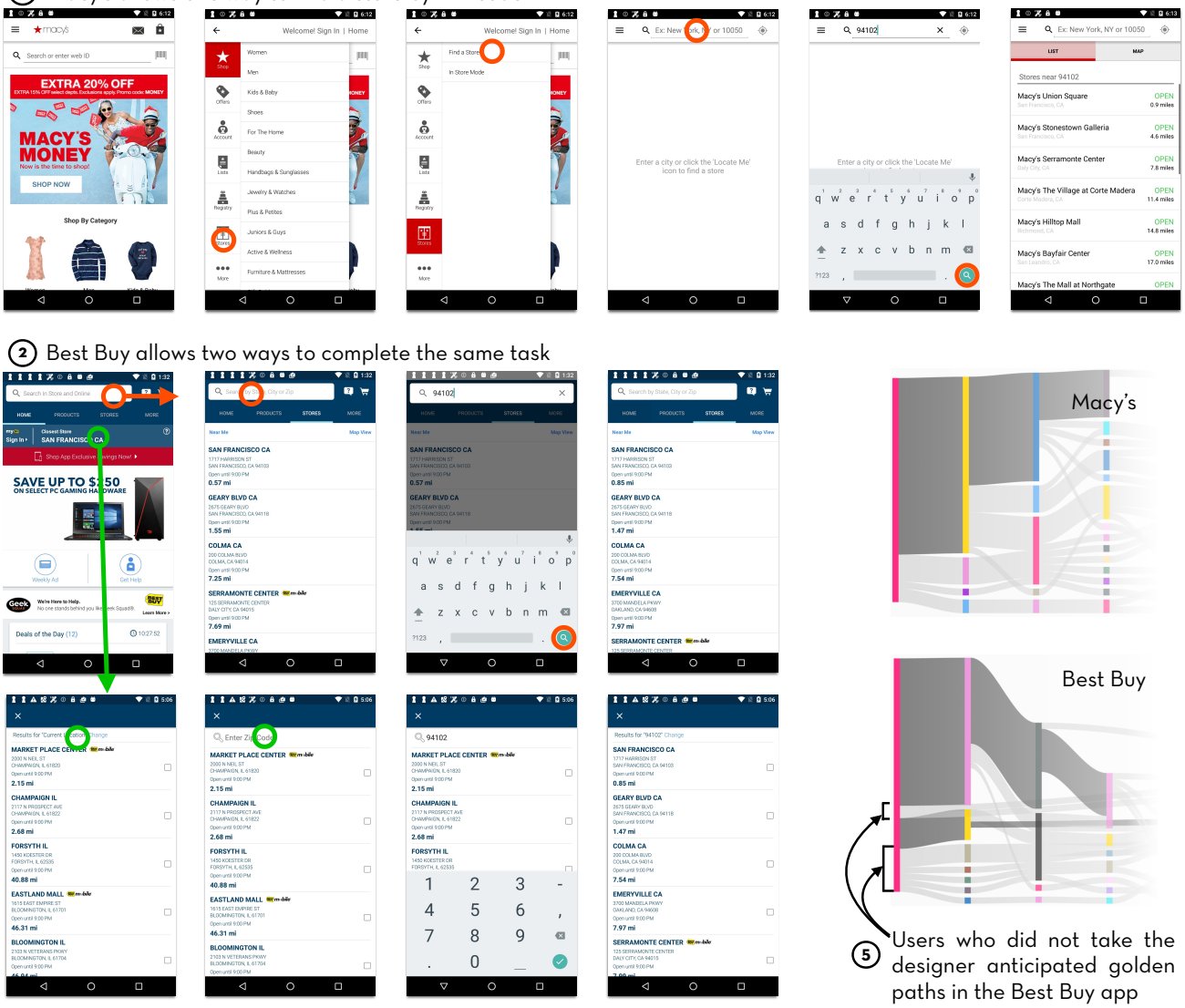

Figure 6: We asked crowd workers to find the address of a store location closest to a specified zip code on the *Macy's* and *Best Buy* apps. The Macy's app uses a hamburger icon and flyout menu navigation, whose sub-menu contains the store locator. The Best Buy app uses tabbed navigation and offers two ways to accomplish the task. On average, users reported that the task was more difficult to perform on the Best Buy app than on Macy's. The hamburger icon was a more familiar design idiom than tabbed navigation with scoped search.

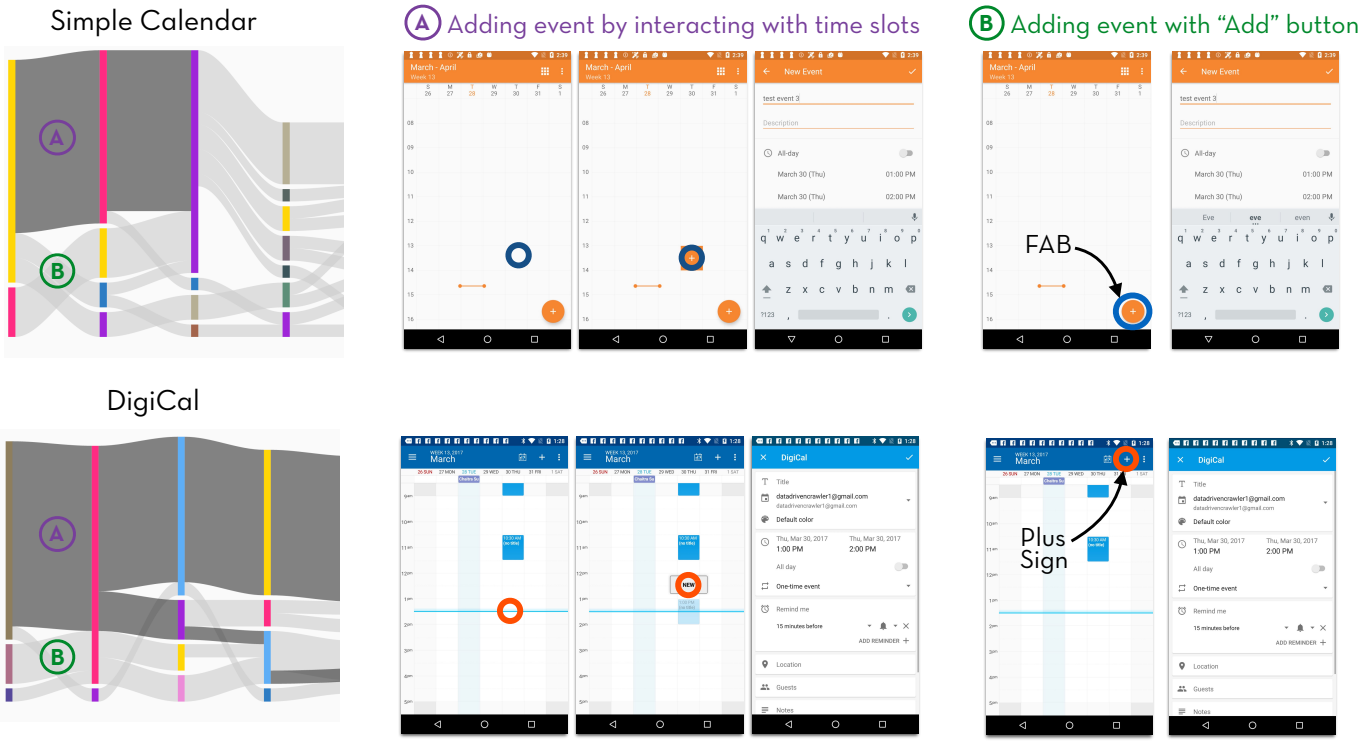

Figure 7: Both *Simple Calendar* and *DigiCal* provide two ways for users to add an event to their calendar: directly interacting with slots on the calendar or through an action button. Although the former strategy is slower than the latter, a larger percentage of users in both apps started the task using direct time slot manipulation.

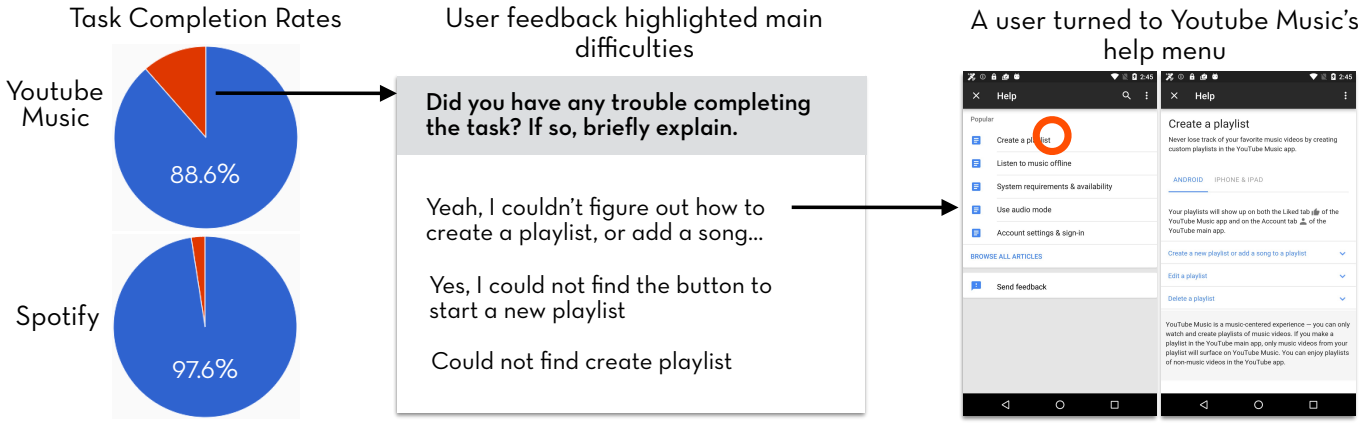

Figure 8: Feedback from the crowd explained why the completion rate of "adding two songs to a new playlist" was lower for *Youtube Music* than for *Spotify*. Youtube Music did not support creation of empty playlists, a capability expected by users.

We used ZIPT to ask crowd workers "what is the address of the store to closest to the 94102 zip code?" We deployed the test for both the Macy's and Best Buy apps and collected 31 and 35 traces, respectively. In the Macy's app, there is only one golden path for answering the question: through a sub-menu in the flyout menu triggered by the hamburger icon (Figure 6). The Best Buy app, in contrast, offers two ways to perform this task. From the tabbed navigation, users can select the "Stores" tab and search for the specified ZIP code. Users can also click on the "Closest Store" location prominently displayed on the home screen and change the ZIP code to find the same information.

When we examined the data collected by ZIPT, we noticed that the task completion rate on the Macy's app (95.7%) was significantly higher than on Best Buy (82.5%). Moreover, on average, users reported that the task was harder to perform on the Best Buy app than on Macy's. By examining the user traces where workers submitted incorrect answers to the task question, we noticed one possible cause of confusion: users did not understand the Best Buy search bar's scope, which changes depending on the selected tab. Although Best Buy does change the search bar's placeholder text, some users still tried searching for stores on the "Home" tab, which returned

zero results. Interestingly, 77% (24 out of 31) of Macy's app users took the correct first step of opening the flyout menu and following the design scent to find the store locator. Perhaps sticking to a familiar design idiom — the hamburger icon — is more effective than having custom navigation.

Adding Events in Calendar Apps

Creating and adding events to a calendar is one of the most common tasks performed by users on scheduling apps. These apps generally provide action buttons on the home screen to support this essential feature. We picked two calendar apps — *Simple Calendar* and *DigiCal* — that offer slightly different action buttons. To study their relative performance, we created and deployed a ZIPT test for both apps, asking crowd workers to "add a new event to a calendar." We collected 24 traces for Simple Calendar app, and 18 traces for DigiCal.

Simple Calendar uses a distinct *floating action button* (FAB) that appears on the bottom right of the main screen (Figure 7). FABs are a popular design pattern and are part of the Android material design library [4]. They are recommended for highlighting promoted actions in an app and are frequently used to create new items (e.g., when composing a new email). DigiCal, in contrast, uses a less distinct +-shaped icon in the top action bar. Both apps allow users to complete the task by directly interacting with a time slot on the calendar, which requires two interactions to reach the screen where event details can be entered. By interacting with the action button, users can reach this screen in one step.

In both apps, we observed that most users attempted the task by directly manipulating the time slots on the calendar. This behavior is easy to spot in flow visualizations (Figure 7). Even though the FAB is high contrast and a popular pattern, only 17% (4 out of 24) of users interacted with it in Simple Calendar. The differences in action buttons had little effect on behavior, since users have a strong habitude of manipulating calendar events directly.

Creating Playlists in Music Apps

Most music service apps support the creation of playlists for curating collections of songs. We studied the relative user performance of playlist creation flows in two popular music apps: *Spotify* and *YouTube Music*. We deployed a task on ZIPT asking crowd workers to "add two songs to a new playlist that you create," and collected 41 user traces for Spotify and 31 for YouTube Music.

We observed a higher task completion rate on Spotify than Youtube Music (92.9% versus 81.5% with p=0.17) (Figure 8). Worker feedback on the task indicated that users were confused by being unable to create empty playlists on YouTube Music. One individual trace revealed that a crowd worker even used the help functionality to learn how to create playlists in YouTube Music.

By inspecting the apps and the collected traces, we discovered that Spotify allows users to add a song to a *new* playlist in two ways. Users can "add to playlist" on a song and then specify that it is going to be part of a new playlist. Alternatively, they can "create new playlist" from a screen for managing playlists, and then navigate to a song and add it. Youtube Music does not support the creation of empty playlists, which seems to be an expected capability.

INFORMAL EVALUATION

We conducted user interviews to gauge the usefulness of ZIPT and to understand how the system might be integrated into the design process. We recruited four Google employees for the study: three of them were participants in our formative study (P3, P4, and P5), and the fourth (P6) was another UX researcher. The interviews lasted one hour and participants were compensated $25.

Each interview comprised a system overview (30 minutes), system walkthrough (10 minutes), and discussion (20 minutes). During the system overview, we introduced the goals of the system and then demonstrated its features using examples from the case studies described in the previous sections. During the system walkthrough, we let each participant use the system to explore the data collected for the *MyNetDiary* calorie counter app. We then started a discussion around how each participant might incorporate the ZIPT system and data into their workflow.

After the interview, we sent participants a survey with 11 questions addressing the overall usefulness of the system and data presentation. The questions elicited responses on a 7-point Likert-scale from strongly disagree (1) to strongly agree (7). Three of the four participants completed the survey. We summarize the questions and their responses in Table 1.

	Question	Avg (S.D.)
Q1	The data produced by this system would be helpful in the app design process.	6.0 (0.0)
Q2	This system would help me understand what works well for design patterns that I see in existing apps.	6.0 (1.0)
Q3	This system would help me find unexpected user behavior.	6.0 (1.0)
Q4	This system would help me find usability bugs.	6.3 (1.2)
Q5	The data produced by this system would help me communicate the merits and demerits of a design pattern to my team members.	6.3 (0.6)
Q6	The metrics overview was helpful.	5.7 (1.2)
Q7	The flow visualization was helpful.	6.3 (1.2)
Q8	The user feedback was helpful.	6.0 (1.0)
Q9	The user interface was easy to use and learn.	6.0 (0.0)
Q10	I believe that the app "MyNetDiary - Calorie Counter" has a usability issue.	7.0 (0.0)
Q11	I would gather performance data for existing apps more often with this system.	6.7 (0.6)

Table 1: Participant feedback about the overall usefulness of ZIPT and its different features.

Potential Uses Cases

Overall, participants reported that ZIPT would be helpful in their design process (Q1–Q5). From the discussions, it was clear that participants valued the potential comparative use cases. P3 would use ZIPT to understand "*if reusable components in different apps share similar usability issues.*" P5 was interested in using ZIPT to benchmark his own apps against competitors': "*I think it fits nicely with how designers think about problems...we think a lot about critical user journeys...this is a way for us to focus on those journeys and benchmark them against similar apps.*" He added, "*if adding a record for food takes ten seconds on my competitor's app, that would be my target...if I have a set of core tasks that can be matched up with those in other apps, then I can understand how well I am doing.*"

Participants found the aggregate data views were useful (Q6-Q8) for both identifying user problems, and for understanding *why* they occur. P3 commented "*I really like that you can state your hypothesis about what the golden paths are and then you check to see if you were correct...and the ability to drill down into the flows where it didn't happen and see in the traces where the derailing was, is very powerful.*" Similarly, during the demonstration of the feedback view, P4 remarked, "*when you first showed me the task creation interface, I was dismissive of the open text fields, but when you are trying to come up with a theory of why things are going wrong, just skimming those things can be very useful.*"

Perceived Benefits

Participants reported that they would use a system like ZIPT to collect performance data for existing apps more often (Q11). P4 commented that "*even small improvements that reduce the cost of user research have a big impact on peoples willingness to do research.*" Participants appreciated that ZIPT reduced the monetary and engineering cost of testing and user research. P5 remarked that "*it is way cheaper than usertesting.com.*" He added, "*the benefit (compared to building prototypes) is that you don't have to create parts of the app that do not relate to the features that you want to test...like creating logins and realistic content...you can save time by using existing applications since all those things are already there.*"

Participants realized that the structured data collection made the testing platform flexible and general. P6 stated, "*you could do a UI evaluation; or an interaction evaluation; or a component evaluation; each of those could benefit from an approach like this, especially once you have all the data, you can ask arbitrary questions of that data.*" P5 viewed the ability to produce a "*flow visualization using the view hierarchy*" as ZIPT's unique advantage. Participants also felt that the structured representation of data was complete. While viewing an individual trace, P4 remarked, "*it seems there is very little that you lose from having an over-the-shoulder video camera.*"

Generated Design Insights

After using ZIPT for ten minutes to explore the user data for MyNetDiary, all participants reported that it had at least one usability issue (Q10, $\mu = 7.0$). P3 and P6 noticed the same issue we identified: the visual design of the data entry screen (Figure 1) did not clearly communicate that "*something else was needed*" (P6).

Participants identified additional usability issues as well. P4 and P5 mentioned that it was an "anti-pattern" that the error message was a global pop-up and not attached to the offending elements. After inspecting multiple traces, P3 noted, "*there are a number of people here who expect that the flow should go to the bottom right when they are done instead of the save button in the top right.*" Participants also spontaneously generated solutions to address some of these issues, including preselecting one of the options (P6, P3), restructuring the data entry as a two step process (P5), changing the component used for selection (P5), and modifying the layout of the screen to be diagonal (P3).

Participants felt comfortable generalizing from specific examples to broader design insights. P3 explained, "*I am not a trained interaction designer, but even with the experience I have, I can look at the one problem case and I can infer from that what the general problem is*". Similarly, when we mentioned the case of users trying to create empty playlists in Youtube Music, P4 generalized that "*people like to think of containers as things*." He named two specific products where he had seen similar user behavior in the past and added, "*you want to be able to have an empty thing to put things in...its how we think of stuff. You want to support the object-first use case where you start the collection from an object, but that's not the first way people think of when you ask them to create the thing.*"

DISCUSSION AND FUTURE WORK

The case studies described in this paper illustrate that ZIPT can provide both the detailed data of usability testing and the scale of analytics-driven and A/B testing. ZIPT allows designers to combine quantitative and qualitative data to identify problems, quantify their magnitude, and gain insight into their causes. In the future, extending ZIPT to support more sophisticated statistical techniques could allow designers to specify a maximum budget and desired confidence interval for key performance indicators, instead of requiring them to mandate the number of user traces to collect.

More broadly, we envision ZIPT becoming part of an integrated mobile design exploration platform, where designers can search for examples using tools like the flow search system from Deka et al. [10], and then invoke ZIPT to understand their performance through a series of inexpensive, targeted experiments. Observing how designers use ZIPT under realistic conditions would also help improve its task specification and data-analysis interfaces.

Although it was not designed to support this use case, ZIPT could also be an interesting platform for evaluating one's own apps. Initially, many app development teams do not have the resources to conduct usability testing, or the users to spend on A/B testing [15]. ZIPT could afford these teams the ability to iterate on their app and get low-cost, low-overhead usability feedback. In the words of one of our participants, "*if it was just me, and it was my startup, and this was my application,*

after seeing this data, I would at least change the position of the save button and see if that makes the flow any better. I would be quite eager to do a couple of iterations until I get something that is as frictionless as possible."

ACKNOWLEDGMENTS

We thank the reviewers for their helpful comments and suggestions; the Google designers, researchers, and engineers who participated in our studies; and the crowd workers who completed the ZIPT tasks. This work was supported in part by a Google Faculty Research Award.

REFERENCES

1. Web form design guidelines: an eyetracking study, 2009. https://www.cxpartners.co.uk/our-thinking/web_forms_design_guidelines_an_eyetracking_study/.
2. Introducing Android Instant Apps, 2010. http://android-developers.blogspot.com/2016/05/android-instant-apps-evolving-apps.html.
3. 20 Best Practices for Mobile App Search, 2017. https://www.raywenderlich.com/153260/20-best-practices-for-mobile-app-search.
4. Material design for Android, 2017. https://developer.android.com/design/material/.
5. Search outside the box with new Pinterest visual discovery tools, 2017. https://blog.pinterest.com/en/search-outside-box-new-pinterest-visual-discovery-tools.
6. Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. *Communications of the ACM 58*, 8 (2015), 85–94.
7. Cardello, J. Usability Metrics, 2013. https://www.nngroup.com/articles/usability-metrics/.
8. Chang, S., Dai, P., Hong, L., Sheng, C., Zhang, T., and Chi, E. H. Appgrouper: Knowledge-based interactive clustering tool for app search results. In *Proc. IUI* (2016).
9. Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., and Kumar, R. Rico: A mobile app dataset for building data-driven design applications. In *Proc. UIST* (2017).
10. Deka, B., Huang, Z., and Kumar, R. ERICA: Interaction mining mobile apps. In *Proc. UIST* (2016).
11. Dow, S., Kulkarni, A., Klemmer, S., and Hartmann, B. Shepherding the crowd yields better work. In *Proc. CSCW* (2012).
12. Eckert, C., and Stacey, M. Sources of inspiration: A language of design. *Design Studies 21*, 5 (2000), 523–538.
13. Eckert, C., Stacey, M., and Earl, C. References to past designs. *Studying Designers 5* (2005), 3–21.
14. Grossauer, C., Holzmann, C., Steiner, D., and Guetz, A. Interaction visualization and analysis in automation industry. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, ACM (2015), 407–411.
15. Iitsuka, S., and Matsuo, Y. Website optimization problem and its solutions. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM (2015), 447–456.
16. King, R., Churchill, E., and Tan, C. *Designing with Data*. OReilly Media, 2017.
17. Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. The future of crowd work. In *Proc. CSCW* (2013).
18. Komarov, S., Reinecke, K., and Gajos, K. Z. Crowdsourcing performance evaluations of user interfaces. In *Proc. CHI* (2013).
19. Kumar, R., Satyanarayan, A., Torres, C., Lim, M., Ahmad, S., Klemmer, S. R., and Talton, J. O. Webzeitgeist: Design mining the web. In *Proc. CHI* (2013).
20. Miller, S. R., and Bailey, B. P. Searching for inspiration: An in-depth look at designers example finding practices. In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2014).
21. Nielsen, J. Finding usability problems through heuristic evaluation. In *Proc. CHI* (1992), 373–380.
22. Nielsen, J. Usability Metrics, 2001. https://www.nngroup.com/articles/usability-metrics/.
23. Qin, Z., Tang, Y., Novak, E., and Li, Q. Mobiplay: A remote execution based record-and-replay tool for mobile applications. In *Proceedings of the 38th International Conference on Software Engineering*, ACM (2016), 571–582.
24. Quinn, A. J., and Bederson, B. B. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI conference on human factors in computing systems* (2011).
25. Ritchie, D., Kejriwal, A. A., and Klemmer, S. R. d. tour: Style-based exploration of design example galleries. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (2011), 165–174.