User-Centered Design of a Mobile Java Practice App: A Comparison of Question Formats

Mohona Ahmed
mohonaahmed@ufl.edu
University of Florida
Gainesville, Florida

Kimberly Michelle Ying
kimying@ufl.edu
University of Florida
Gainesville, Florida

Kristy Elizabeth Boyer
keboyer@ufl.edu
University of Florida
Gainesville, Florida

ABSTRACT

Learning computer science presents many challenges to students, and providing resources for meaningful practice is recognized as a way to support rigorous learning and diverse student participation. At the same time, mobile phones are increasingly ubiquitous, creating an underutilized opportunity for practice outside of traditional methods. This experience report presents a user-centered approach to designing a practice app for introductory Java. We investigated user preferences through a series of small studies, first conducting think-aloud sessions and focus groups, and finally conducting a usability study comparing two prototype versions. The initial studies suggested how to leverage the affordances of small screens, ruling out free-response practice problems in favor of either fill-in-the-blank (FB) or multiple-choice (MC) questions. The comparison study revealed statistically significant differences in students' survey responses: (1) usability scores were significantly higher for the MC version than the FB version; (2) students reported significantly greater satisfaction and desire to learn for the MC version; and (3) students reported enjoying and being more comfortable with the MC version compared to the FB version. We contextualize this observation within related research on question formats. Takeaways from this experience report can provide guidance on designing mobile applications that give students opportunities for meaningful practice.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; **Computer science education**; *Adult education*; • **Human-centered computing**;

KEYWORDS

Question formats; introductory programming; user-centered design; mobile practice application

ACM Reference Format:
Mohona Ahmed, Kimberly Michelle Ying, and Kristy Elizabeth Boyer. 2020. User-Centered Design of a Mobile Java Practice App: A Comparison of Question Formats. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366942

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366942

1 INTRODUCTION

Novices learning computer science may find it challenging to solidify coding concepts. Providing meaningful practice opportunities to engage students can strengthen their grasp of computer science knowledge [11]. Many tools have been developed to aid students in learning computer science; however, an underutilized platform for practice problems is mobile applications. Although research has been done on incorporating mobile applications for practice within the classroom [6], offering a practice tool that can be used outside the classroom could be especially useful for students.

Mobile applications are ubiquitous and are used often in the day-to-day lives of young adults. In 2018, 95% of teens in the U.S. reported having access to a smartphone [2]. The Pew Research Center also reported that smartphone ownership is "universal," as access is not significantly different across gender, race/ethnicity, and socioeconomic backgrounds, in contrast to a computer, for which access varies by level of education among parents and household income. Additionally, many students in American colleges reported using their phone "very often" or "fairly often" to get information quickly (73%) or when they are bored (77%) [9]. Given their widespread use, mobile applications are a promising platform for educational initiatives. Mobile tools can provide meaningful opportunities for students to practice and reinforce concepts in a way that fits their lifestyle.

This experience report outlines the user-centered process we followed for designing a mobile practice app for Java. It discusses key takeaways and design decisions resulting from the small studies we conducted. User input from think-aloud sessions and focus groups informed the development of two versions of an initial prototype. We focused largely on the decision for an appropriate question format, specifically comparing multiple-choice (MC) and fill-in-the-blank (FB) versions of an otherwise identical mobile app in the A/B testing phase. Seventeen participants from collegiate computer science courses were recruited to use each version and report their experience through completion of a three-pronged survey that evaluated usability, motivation to learn, and perceived effectiveness. After the participants used both versions of the application, they were given an additional survey of questions that asked them to compare the two versions directly.

This paper provides novel contributions to the computer science education community, including an overview of the user-centered design approach and insights on design decisions for mobile practice applications. The design process used to create the prototype can be followed to create other applications and potentially make

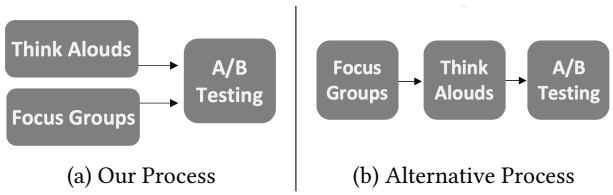
them more motivating for students to use. Additionally, this paper gives insight on the benefits and shortcomings of implementing two different question formats for a mobile application used for practice. The findings in this paper can inform those interested in developing or using mobile practice tools to support student learning.

2 BACKGROUND

There has been considerable effort invested in creating learning tools for students in computer science such as coding environments [21, 29], virtual tutors [5], serious games [27], in-class assessment tools [20], web applications [8], and feedback tools [28]. Most of these learning tools were created to be used on a traditional computer or laptop. While the latter two learning tools are web-based, only Zheng and colleagues emphasize their in-class feedback tool for use on mobile devices [28]. This avenue could be underinvestigated due to the challenges of adapting to a small screen size. Despite this limitation, mobile tools such as an in-class visualization tool have been implemented, showing that mobile devices are a viable platform for learning tools [6]. To initiate the development of mobile tools, the interfaces of mobile applications need to be designed to accommodate for the "reduction in real estate" due to their small screen size [4]. One practical approach for creating a mobile Java practice app is to follow a user-centered design process. This process is used commonly in industry (e.g., [12]), leads to better products using fewer resources [26], and makes product development more efficient and effective [1].

Conventional ways to practice coding involve writing out code, which is difficult to do on a mobile phone. Another method of practicing coding, such as answering multiple-choice or fill-in-the-blank questions, must be facilitated instead. Educators might suspect that a student would learn less from not writing out code, but Harrington and Cheng found no statistically significant difference in the achievement gap of students who traced code versus those who wrote it [10]. Utilizing code tracing questions for a practice tool aimed toward novice programmers may also be beneficial, since it is suggested that beginners trace code before writing out code [16]. These findings motivate the use of practice questions on mobile platforms which exclusively require code tracing instead of writing out code.

This study investigates two question formats based around tracing code: multiple-choice (MC) and fill-in-the-blank (FB) questions. Specifically, students had to read snippets of code and complete the code to obtain a desired output. A sample question from the FB prototype is "Write a statement that fills in the blank so that the loop runs 4 times". Similarly, for the MC prototype, the synonymous sample question is "Pick the statement that fills in the blank so that the loop runs 4 times". While both these formats incorporate tracing code, there may exist skepticism between the question formats with regards to their success in helping students practice. Computer science educators may believe that FB questions help students learn more than MC questions, given it is more similar to writing out code. MC, or "closed" question formats, can actually be good for question design because they can promote students to think about all aspects of a question by "remind[ing] respondents of material that they may otherwise not consider" [23]. FB questions, on the other hand, require more cognitive load because students must think of their own answers that are not primed or scaffolded by answer choices. Educational psychologists who study cognitive load theory describe the limitations of working memory and the learning difficulties that can arise from high cognitive load [24]. Cognitive load is especially important to consider in the context of an educational mobile app, since users may be on-the-go, possibly riding the bus or walking from one class to the next, with their attention divided.



(a) Our Process (b) Alternative Process

Figure 1: a) Our process involved think-aloud sessions and focus groups independently influencing A/B testing. (b) An alternative process could involve strictly sequential user studies.

3 USER-CENTERED DESIGN PROCESS

This section describes a user-centered design process and defines techniques for gathering user feedback. This approach involves using a series of user studies to inform the design of a prototype. This paper used think-aloud studies, focus groups, and A/B testing to curate and iterate on the prototype design. For best results, user feedback should be solicited early and often throughout the process.

Note that in our user study, the focus groups and think-aloud studies independently informed the design of the initial prototype (see Figure 1a), but we also recommend a sequential approach in which the focus group findings impact the design of the paper prototype used in the think-aloud studies (see Figure 1b).

3.1 Focus Group Overview

Focus groups are a great way to understand the wants, needs, and opinions of end users. Before exerting resources toward developing a new product, we must determine whether the target users have an interest in it. As the name implies, focus groups involve facilitating a group of participants to focus on and discuss a particular topic or set of topics. A moderator uses a semi-structured interview protocol, which outlines the topics and questions, to direct the conversation. The ideal size of a focus group depends on the topic and the familiarity that participants have with the topic, but Krueger and Casey recommend groups of five to eight participants for noncommercial topics [14]. The moderator should preface the focus group by explaining to all participants that their input is valuable. During the session, the moderator should be impartial and ensure every participant has a chance to express their thoughts, calling on specific individuals to contribute when necessary.

3.2 Think-Aloud Overview

After summarizing user needs, developers can define user requirement specifications and start designing a prototype. Think-aloud studies can be conducted to observe users interacting with a prototype and understand their thought process. A think-aloud study is generally a one-on-one session in which an investigator asks the

participant to complete a series of tasks using the prototype while describing what they are thinking.

Think-aloud studies are valuable and economical ways to obtain usability feedback at different stages of development. During early stages, a think-aloud study using a low-fidelity prototype can inform high-level design decisions such as layout and flow. A low-fidelity prototype, commonly referred to as a wireframe (e.g., Figure 2), is a simple mock-up of an application that allows for feedback on the high-level design idea and application flow by leaving out fine details. During later stages, a think-aloud study using a high-fidelity prototype can inform low-level design decisions such as typography, color scheme, and icons, as well as high-level design decisions. A high-fidelity prototype is functional and detailed, closely resembling the final product (see Figure 4 and Section 4.5 for more detail). Even after deployment, think-aloud studies can be conducted on completed applications to determine areas of improvement.

3.3 A/B Testing Overview

A/B testing can be used to understand the advantages and disadvantages of two versions of an application. Specifically, the versions should be identical except for the one feature or aspect in question. High-fidelity prototypes or deployed applications are most appropriate for A/B testing. There are many different ways to conduct A/B testing and sometimes the goal is not just improved usability, but potentially greater profits [13]. Web developers commonly deploy to a live site and randomly direct traffic between two versions to test new features. Analysis for this type of A/B testing generally involves checking traffic logs or user flow patterns.

For more direct feedback from users, participants in A/B testing should complete surveys on their experiences. The System Usability Scale (SUS) is a validated 10-item survey that is widely used as an industry tool to evaluate the usability of a system or application [3, 19]. Participants score each of the 10 items, such as "I thought the system was easy to use," on a five-point Likert scale from "Strongly Agree" to "Strongly Disagree". The SUS can be modified to replace all instances of "system" with a more appropriate word to describe the technology in question. We used "application" for our studies.

Another important decision for A/B testing is whether to use a within-subjects or between-subjects experimental design. A within-subjects design is when all participants use both versions, in contrast to a between-subjects design in which each participant only uses one of the versions. Our A/B testing employed a within-subjects design so that we could include survey questions that asked users to directly compare the two versions. Although a within-subjects design has the advantage of allowing a direct comparison of versions, it is important to be aware of possible order effects, such as users learning or being biased from one condition to the next. Counterbalancing is a popular method to minimize order effects [18]. In the context of A/B testing, the participants are split into two groups of equal sizes; one group uses Version A first and the other group uses Version B first (see Figure 3).

4 PILOT STUDIES AND FINDINGS

4.1 Laying the Groundwork

All user studies were conducted at a large public university in the Southeastern United States in Fall of 2018 and Spring of 2019. For each user study (think-aloud, focus group, or A/B testing), new participants were recruited such that no student participated in more than one user study. Thirty-five participants were recruited from one of three computer science courses: Computer Programming using Java for non-majors (CS0), Programming Fundamentals 2 (CS2), and Introduction to Computer Organization. Participants ranged in age from 18 to 25. Additional demographic information will be provided in more detail for each user study below. All of the studies reported here were conducted within IRB-approved protocol and students provided informed consent. Students who consented were offered extra credit for their participation, and there was no penalty for students who did not participate.

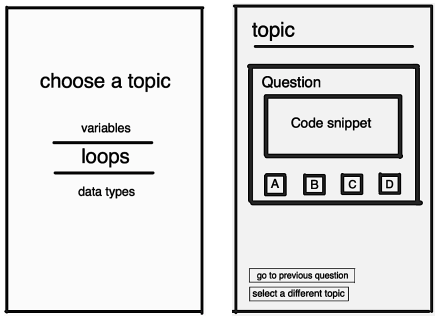


Figure 2: Low-fidelity paper prototype of the multiple-choice version of the mobile application, which was used in the think-aloud sessions.

4.2 Initial Prototype Design Choices

The prototype built is intended for use on a mobile device, designed for a user to practice programming concepts in Java. Each practice question included a snippet of code for the user to trace, and they had to complete one of the lines of code to obtain a desired output. The original wireframe used only a MC question format for uniformity during the think-aloud study (see Figure 2). Participants used the mobile prototype to practice questions regarding the programming concept of loops. This concept was chosen because participants were already introduced to it in their courses. This reflects how we expect participants to use the final product,

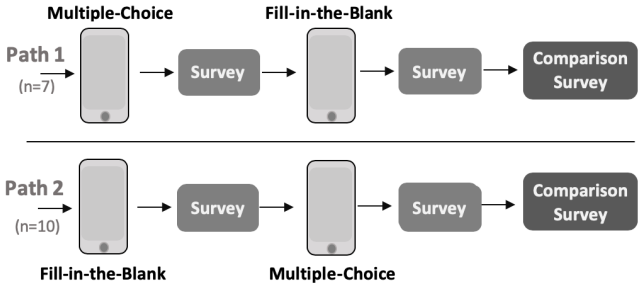
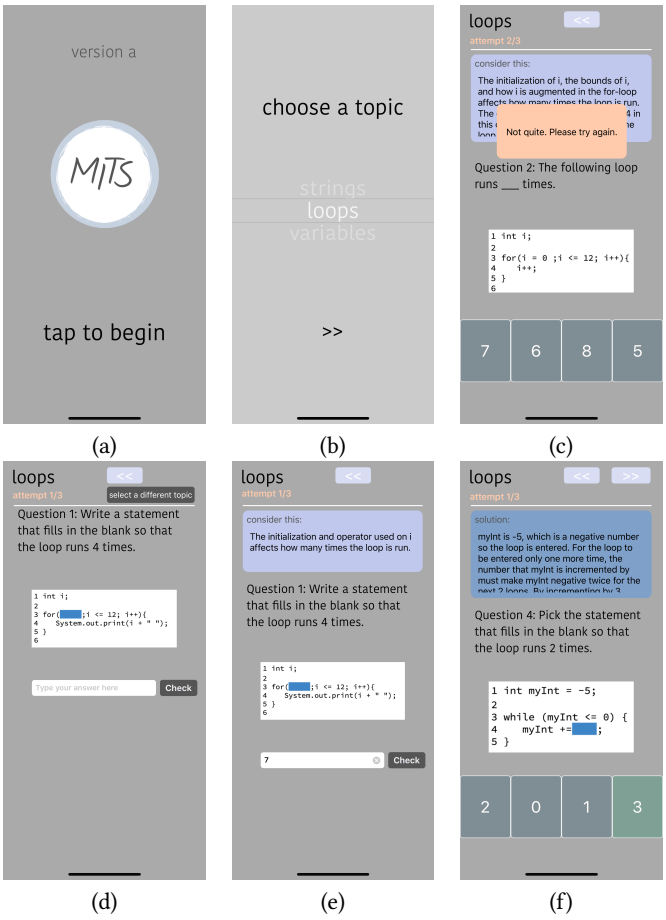


Figure 3: A/B testing within-subjects experimental design. Seven participants followed path 1 and ten followed path 2. Ideally these two groups would be equal, but due to absences, this was not the case for our pilot study.



(a) (b) (c)

(d) (e) (f)

Figure 4: High-fidelity screenshots of the mobile application prototype.

as practice after being introduced to it in class. Participants were also given feedback every time they picked an answer based on the number of times an incorrect answer was chosen. Once the user exhausted three attempts on a question or answered the question correctly, an explanation on how to solve the question was given.

4.3 Think-Aloud Study

4.3.1 Experimental Process. The think-aloud study was conducted to evaluate the perception of our initial mock-up for the mobile app. Four students from the CS0 course participated in a think-aloud session. Three participants were female and one was male, and they ranged in age from 18 to 21 years old. The participants were given a low-fidelity paper prototype and were asked to verbally communicate their first impressions of design features while performing tasks (see Figure 2). The layouts and widgets of the prototype's wireframe were designed using an online tool, InVision[1]. These components were printed and cut out to be presented to the participants. To mimic animations that the application would have, the cutouts of the application were moved and verbalized by the researcher conducting the think-aloud session according to the participant's gestures. These sessions were recorded and later transcribed to document user suggestions and feedback.

In addition to the user interface design, the wireframe included practice coding questions inspired by computer science textbooks such as *Programming in Java* from *zyBooks* [17] and *Introduction to Java Programming* [15] and written by the authors of this paper. This is more detail than usual for a low-fidelity prototype, but we felt it was necessary for participants to be able to contextualize the interface.

4.3.2 Key Takeaways. The original design presented in the think-aloud sessions was a wireframe of the multiple-choice version of the application. It limited the user to three attempts on the practice problems since there were only four answer choices. The prototype also provided unique feedback to users after attempting each question. Restricting the number of attempts a user has to complete the problem received no negative feedback. Additionally, none of the participants mentioned wanting fewer or more attempts for each question, but they were not specifically asked about this. Participants also had positive sentiments toward the answer-choice feedback feature.

4.4 Focus Group Study

4.4.1 Experimental Process. A focus group study was conducted to evaluate user preferences for practicing Java and determine how a mobile prototype could be adapted to fulfill those needs. The focus group study included 14 students in total, with 13 students from CS0 and one student from Introduction to Computer Organization. Five participants were female and nine were male, and their ages ranged from 19 to 25 years old. We aimed to have five to six participants in each focus group, but there were absences and scheduling conflicts when conducting this study. Four focus groups were conducted: two five-participant groups, one three-participant group, and one one-participant interview. Participants were asked questions such as "What types of practice questions would be most helpful for you in learning Java?" and were able to discuss their responses among one another. This study was used to gauge interest and gather user requirements for a mobile Java practice app.

4.4.2 Key Takeaways. In the focus groups, participants were asked about the types of questions they would expect in a mobile application to practice Java. Participants understood the limitations of working on a small screen and therefore mentioned question types such as writing short but complete pieces of code or matching definitions to concepts. To accommodate the concern of interface size while considering student suggestions, we explored the trade-offs between different question types and decided that MC or FB questions would be preferred by users.

During the focus group discussions, students presented concerns with screen size in the case that questions required long blocks of text. Many disliked the implementation of a scroll view for the entire question; they preferred all of the information on a single page view (see Figures 4c-f). Instead of users scrolling through the entire content, widgets were chosen to implement the idea of presenting all components of a question on a single page. If individual sections of the interface required more space for text, the user would be able to scroll through the widgets themselves.

[1]https://www.invisionapp.com/

4.5 High-Fidelity Prototype

The feedback on the paper prototypes, the discussions of ideal application features, and insight from studying practices informed the user interface of the high-fidelity prototype. This prototype is not a full-fledged application but rather a proof of concept.

The prototype was built on iOS using the development environment Xcode. The prototype includes a splash page (Figure 4a), a topic selection page (Figure 4b), and a question page (see Figures 4c-f for examples). The question page has navigation tools at the top right of the screen for progressing through the questions, and a section below the code snippet that contains either buttons for different answer options (MC version), or a text field for users to input their answer (FB version). Once the user inputs the answer, the question area moves down to provide feedback (see Figures 4d-e). A pop-up feature is incorporated to tell the user if the answer chosen was right or wrong (see Figure 4c). To notify the user that attempts have been exhausted or a correct answer has been chosen, the feedback window and either the button with the correct answer (MC version) or the input bar (FB version) changes in color Figure 4f).

Each version of the prototype consisted of an identical set of four coding questions on the concept of loops. As this prototype is merely a proof of concept, a fully developed prototype will have a more complete set of questions. Since data collection was focused on survey responses and not on practice score, researchers decided that having four practice questions was enough for the participant to develop an opinion of the experience with the prototype.

4.6 A/B Testing

4.6.1 Experimental Process. Seventeen participants tested two versions of a mobile prototype outside of class hours. This study process included three students from CS0 and 14 students from CS2. Sixteen participants had ages between 18 to 22; one participant did not disclose their age. Thirteen participants were male and four were female. Each participant went through the study process one at a time with a researcher. Following traditional A/B testing methodology, the participant would individually use both versions of the prototype. The versions were identical except for the question format; Version A consisted of fill-in-the-blank (FB) questions, and Version B consisted of multiple-choice (MC) questions. To minimize order effects resulting from the within-subjects design, roughly half (n=10, 58.82%) of the participants were given Version A (FB) first and the rest (n=7, 41.18%) used Version B (MC) first, as shown in Figure 3.

Identical surveys were given after the participants used each version of the prototype. The survey consisted of three sections: the System Usability Scale (SUS) [3], a section regarding Satisfaction/Desire to Learn, and an Effectiveness section. We created six items each for the latter two sections. An example item from the Satisfaction/Desire to Learn section is "This application meets my expectations for a mobile application for practicing Java." An example item from the Effectiveness section is "I feel more confident in my mastery of Java concepts because of this system." For each section, the participants were given a series of statements to rate using a Likert scale. The SUS and the Satisfaction/Desire to Learn sections were based on a 5-point Likert scale from Strongly Disagree (1) to Strongly Agree (5). The Effectiveness section was based on a 7-point Likert scale from Strongly Disagree (1) to Strongly Agree (7).

To quantitatively analyze the participant responses, the questions in each section of the survey were aggregated for analysis. A paired t-test was used on these aggregate responses by converting the participant reportings to numerical values. For example, if a participant chose "Strongly Disagree," the response would be given a numerical value of 1. To account for responses to the negative statements, the assigned numerical responses were reversed. For example, if a participant chose "Strongly Disagree" for the statement "I would not use this application in the future," the numerical value assigned to this question would be a 5 instead of a 1 for analysis. We performed a two-tailed paired t-test using an alpha level of 0.05 and 16 degrees of freedom.

As SUS is an established usability metric, we calculated and interpreted the SUS score following the traditional procedure [25]. SUS scores of 68 or higher are considered acceptable.

After participants used both versions of the prototype and reported their experiences in the identical surveys, they were asked to complete an additional survey. The survey required them to directly compare the two versions based on four questions. This comparison survey was analyzed using a one sample test of proportions. See Table 1 for the questions.

4.6.2 Key Findings. **System Usability Scale (SUS).** The System Usability Scale was used to measure the usability of the prototype versions on a 100-point scale. Students reported that the MC version was more usable than the FB version. The MC version yielded a mean score of 78.97 (SD=12.69, n=17) and the FB version yielded a mean score of 66.18 (SD=14.39, n=17). This difference is statistically significant (paired-samples t-test, $t(16)$=4.14, p=0.0008).

Satisfaction/Desire to Learn. The Satisfaction/Desire to Learn section was used to gain insight on the motivation users would have to use an application similar to the prototype used in this pilot study. Aggregate scores for this section were out of 30. Students reported higher scores for MC than FB, meaning that students may be more willing to use an application with multiple-choice questions. The MC version yielded a mean score of 21.12 (SD=5.62, n=17) and the FB version yielded a mean score of 18.76 (SD=5.32, n=17). This difference is statistically significant (paired-samples t-test, $t(16)$=2.33, p=0.0334).

Effectiveness. The Effectiveness section was used to understand the user's perceived effectiveness of the app on their Java learning. Aggregate scores for this section were out of 42. The MC version yielded a mean score of 30.59 (SD=6.35, n=17) and the FB version yielded a mean score of 29.00 (SD=6.82, n=17). This difference is not statistically significant (paired-samples t-test, $t(16)$=1.22, p=0.2387).

Comparison of versions. Participants were asked four questions on preference between the two versions of the prototype. Students reported enjoying the MC version more than the FB version (one sample test of proportions, p = 0.0023). They were also more comfortable using the MC version compared to the FB version (one sample test of proportions, p=0.0003). See Table 1 for more details.

5 DISCUSSION

This experience report describes a user-centered design approach for the development of a mobile practice application for Java. The

	Number of Participants (Percent)		P-value
	Fill-in-the-Blank	Multiple-Choice	
Choose the version that was better for practicing Java.	7 (41.18%)	10 (58.82%)	0.6291
Choose the version you enjoyed more.	2 (11.76%)	15 (88.24%)	***0.0023***
Choose the version you would be more likely to use in the future.	5 (29.41%)	12 (70.59%)	0.1435
Choose the application you are more comfortable using.	1 (5.88%)	16 (94.12%)	***0.0003***

Table 1: Results from the one sample test of proportions for the comparison of versions survey. Items that are bold and italicized are statistically significant.

prototype was designed for college students to provide meaningful and easily accessible opportunities for practicing programming concepts. Early in the design process, users expressed concerns about using an educational application on a mobile platform because of the limited screen size. This concern sparked the investigation to determine what question format would be most appropriate for practice problems on the app.

Focus groups and think-aloud sessions were important sources of insight which directly informed the design. For instance, students who participated in the think-aloud study preferred the implementation of a single-page view over a scrolling view. Single-page views require less working memory, which is especially important to minimize in this context, since users will be using much of their cognition on solving the problem. Additionally, participants in the focus groups expressed interest in receiving feedback when approaching a question to help facilitate further understanding of the concepts presented. This also informed the decision to provide feedback even on correct answers so that learners could double check their reasoning.

In efforts to curb issues of limited screen size, we investigated what question format would be most appropriate for a mobile practice application. Ruling out implementation of a question design that requires a user to type code in a free response manner, we narrowed down the choices to a comparison study of multiple-choice (MC) versus fill-in-the-blank (FB) questions.

Results from the A/B testing revealed statistical differences in the student survey responses. The usability scores for the MC version were significantly higher than the scores reported for the FB questions. Applications with SUS scores above 68 are considered acceptable. Since the FB version had an average SUS score of 66.18, it is just under the threshold for acceptability, while the MC version is well over the threshold with a SUS score of 78.97. The comparison survey also indicated significantly greater student satisfaction and desire to learn for the MC version. Users may have found the MC version to be more usable, satisfactory, and desirable due to its convenience and simplicity. Students also reported greater enjoyment and being more comfortable with the MC version. The overall preference for the MC version may be attributed to it requiring less cognitive load from the user. Considering that students using this app will likely be tracing the code in their head (not using scratch paper), the added effort of answering via FB may cause cognitive overload. Renkl and Atkinson suggest minimizing cognitive load during initial cognitive skill acquisition, gradually increasing problem difficulty overtime since cognitive load gradually decreases [22]. For these reasons, MC questions on mobile applications for CS may be most appropriate to motivate repeated practice among novices.

This design process makes a case that meaningful practice opportunities can be provided by a mobile application. Specifically, the prototype supports the integration of MC questions into mobile applications for practicing Java. This experience demonstrates that a user-centered approach can give valuable insight into the effect of application features such as question type on the user experience within a mobile learning tool

Limitations. As the development of this application is in its early stages, its impact on student learning has not yet been investigated. Participants had relatively short interactions with each prototype before completing the surveys on their experiences. Future studies are needed to evaluate the final application's real-world usability and effectiveness.

6 CONCLUSION AND FUTURE WORK

The goal of this work was to gain insight regarding what users would prefer from a mobile application for practicing Java. We used an iterative design process and user-centered approach to curate the design of the prototype. Compared to the fill-in-the-blank version, the multiple-choice version of the prototype (1) had significantly higher usability scores, (2) resulted in greater satisfaction and desire to learn, and (3) was more enjoyable and comfortable to use.

This prototype is still in the early stages of development, with only three iterations of the design. More user studies will be conducted to refine the prototype's usability. Specifically, we will solicit user feedback on different implementations of inter-question navigation: (1) should users be able to navigate and edit their answers to previous questions or (2) should users be required to complete one question before advancing to the next? We will also explore other types of question formats like Parson's problems [7], which involve dragging and dropping code snippets to create a program instead of writing out code.

In the future, we plan to develop the prototype into a fully functioning system and incorporate an intelligent tutoring system that will give users automated feedback and present appropriate practice problems according to their progress and understanding of each concept. Once user studies establish that the design is preferable and has high usability scores, we will test the application's ability to improve learning gains. Insights from investigating the relationship between usage and learning gain will be used to improve the application and create an even better opportunity for meaningful practice.

ACKNOWLEDGMENTS

The authors are grateful to the members of the LearnDialogue Group for their help and encouragement. This material is based upon work supported by the National Science Foundation under grant CNS-1622438. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Chadia Abras, Diane Maloney-Krichmar, and Jenny Preece. 2004. User-Centered Design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications* 37, 4 (2004), 445–456. https://doi.org/10.3233/WOR-2010-1109

[2] Monica Anderson and Jingjing Jiang. 2018. *Teens, Social Media & Technology 2018.* Technical Report May. Pew Research Center.

[3] John Brooke. 1996. SUS: A 'quick and dirty' usability scale. In *Usability evaluation in industry*. 189–194.

[4] Qunicy Brown, Frank J. Lee, Dario D. Salvucci, and Vincent Aleven. 2008. The design of a mobile intelligent tutoring system. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems.* http://www.cs.drexel.edu/{~}salvucci/publications/Brown-ITS08b.pdf

[5] Alan de Oliveira Santana and Eduardo Aranha. 2019. An approach to generate virtual tutors for game programming classes. In *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* 246–252. https://doi.org/10.1145/3287324.3287414

[6] Debzani Deb, M. Muztaba Fuad, and Mallek Kanan. 2017. Creating engaging exercises with mobile response system (MRS). In *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE.* 147–152. https://doi.org/10.1145/3017680.3017793

[7] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. *ICER '08 Proceedings of the Fourth International Workshop on Computing Education Research* (2008), 113–124. https://doi.org/10.1145/1404520.1404532

[8] Margaret Ellis, Clifford A. Shaffer, and Stephen H. Edwards. 2019. Approaches for coordinating eTextbooks, online programming practice, automated grading, and more into one course. In *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* 126–132. https://doi.org/10.1145/3287324.3287487

[9] Richard C Emanuel. 2013. The American college student cell phone survey. *College Student Journal* 47, 1 (2013), 75–81. http://ovidsp.ovid.com/ovidweb.cgi?T=JS{&}PAGE=reference{&}D=psyc10{&}NEWS=N{&}AN=2013-10664-008

[10] Brian Harrington and Nick Cheng. 2018. Tracing vs. writing code: Beyond the learning hierarchy. In *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education.* 423–428. https://doi.org/10.1145/3159450.3159530

[11] Beryl Hoffman, Ralph Morelli, and Jennifer Rosato. 2019. Student engagement is key to broadening participation in CS. In *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* 1123–1129. https://doi.org/10.1145/3287324.3287438

[12] Eeva Kangas and Timo Kinnunen. 2005. Applying user-centered design to mobile application development. *Commun. ACM* 48, 7 (2005), 55–59. https://doi.org/10.1145/1070838.1070866

[13] Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. 2013. Online controlled experiments at large scale. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Vol. Part F1288. 1168–1176. https://doi.org/10.1145/2487575.2488217

[14] Richard A. Krueger and Mary Anne Casey. 2014. Participants in a Focus Group. In *Focus Groups A Practical Guide for Applied Research* (5th ed.). 63–84.

[15] Y. Daniel Liang. 2015. Loops. In *Introduction to Java Programming* (10th ed.). Pearson Education, Inc., 157–202.

[16] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. In *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE.* 161–165. https://doi.org/10.1145/1562877.1562930

[17] Roman Lysecky and Adrian Lizarraga. 2012. *Programming in Java.* zyBooks. https://www.zybooks.com/catalog/programming-in-java/

[18] I. Scott MacKenzie. 2013. Desigining HCI Experiments. In *Human-Computer Interaction: An Empirical Research Perspective* (1st ed.). 157–190.

[19] S Camille Peres, Tri Pham, and Ronald Phillips. 2013. Validation of the system usability scale (SUS): SUS in the wild. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 57. SAGE Publications Sage CA: Los Angeles, CA, 192–196.

[20] Leo Porter, Daniel Zingaro, and Raymond Lister. 2014. Predicting student success using fine grain clicker data. *ICER 2014 - Proceedings of the 10th Annual International Conference on International Computing Education Research* September (2014), 51–58. https://doi.org/10.1145/2632320.2632354

[21] Thomas W. Price, Yihuan Dong, and Dragan Lipovac. 2017. iSnap: Towards Intelligent Tutoring in Novice Programming Environments. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.* 483–488. https://doi.org/10.1145/3159450.3162202

[22] Alexander Renkl and Robert K. Atkinson. 2010. Structuring the Transition From Example Study to Problem Solving in Cognitive Skill Acquisition: A Cognitive Load Perspective. *Educational Psychologist* 38, 1 (2010), 15–22. https://doi.org/10.1207/S15326985EP3801

[23] Fritz Strack and Norbert Schwarz. 2007. Asking questions: Measurement in the social sciences. In *Psychology's territories: Historical and contemporary perspectives from different disciplines.* 225–250.

[24] J Sweller, P Ayres, and S Kalyuga. 2011. *Cognitive load theory.* 89–98 pages. https://doi.org/10.1007/978-1-4419-8126-4_7

[25] U.S. Department of Health & Human Services. 2013. System Usability Scale (SUS). https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[26] U.S. Department of Health & Human Services. 2017. Benefits of User-Centered Design. https://www.usability.gov/what-and-why/benefits-of-ucd.html

[27] Stacey Watson and Heather Richter Lipford. 2019. Motivating students beyond course requirements with a serious game. In *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* 211–217. https://doi.org/10.1145/3287324.3287364

[28] Jun Zheng, Sohee Kang, and Brian Harrington. 2019. Immediate feedback collaborative code tracing. In *Proceedings of the 24th Western Canadian Conference on Computing Education, WCCCE 2019.* 12–13. https://doi.org/10.1145/3314994.3325087

[29] Rui Zhi, Thomas W. Price, Samiha Marwan, Alexandra Milliken, Tiffany Barnes, and Min Chi. 2019. Exploring the impact of worked examples in a novice programming environment. In *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* 98–104. https://doi.org/10.1145/3287324.3287385