Issues in Information Systems
Volume 19, Issue 3, pp. 22-32, 2018

A MOBILE APP DEVELOPMENT PROJECT MODEL EMPHASIZING COMMUNICATION SKILLS, TEAMWORK, AND USER-CENTERED DESIGN

Jamie Pinchot, Robert Morris University, pinchot@rmu.edu

ABSTRACT

The mobile application industry is growing, and there is a clear need for training in mobile application development skills. Mobile app development can be complex and requires a specific set of technical skills for coding, as well as an understanding of interface design for a variety of mobile devices. User-centered design is a necessary component that should be considered a requirement for all software development today, including mobile apps. This paper presents a project model for a mobile app development project in an undergraduate course that incorporates technical components, user-centered design, communication skills, and teamwork. The project model is discussed in detail along with issues and challenges encountered during implementation of the project over four sections of mobile app development courses, two focused on Android and two focused on iOS. Lessons learned are presented and discussed.

Keywords: Mobile App Development, Mobile Development, Soft Skills, Communication Skills, Teamwork, Pair Programming, User-Centered Design

INTRODUCTION

The first iPhone was released in 2007 and the Apple App Store was launched in 2008. The appearance of the App Store marked a transitional shift in software development, allowing for the creation of third-party mobile applications (apps) that could be purchased and consumed on Apple smartphones. Android smartphones and their app marketplace Android Market (now known as Google Play) made their appearance shortly after in 2008. In a surprisingly short period of time, mobile app development has become an essential skill set (Goadrich & Rogers, 2011). Forbes named Application Software Developer the number one in-demand job in 2018, noting that there is a lack of people with needed skills in this area and not enough new graduates in the profession (Stahl, 2018). Business Insider named Mobile Developer the number three best tech job in 2018 (Gillett, 2018).

The mobile application industry is growing, for both revenue and number of downloads. The final quarter in 2017 set records for both app revenue and downloads. The first quarter of 2018 has broken those records already. According to Sydow (2018), "2018 started with the strongest quarter the app economy has ever seen, shattering records set in Q4 2017 for both consumer spend and downloads. Global iOS and Google Play combined downloads grew more than 10% year-over-year to 27.5 billion – the highest of any quarter. Global iOS and Google Play combined consumer spend grew 22% year-over-year to $18.4 billion" (para. 1). These figures are staggering and show the incredible impact of mobile apps on consumers and the need for developers to support the app economy.

There is a clear need for training in mobile application development skills. But, how can universities best prepare students for positions in the rapidly growing field of mobile development? Skills in native platform development, notably Android and iOS, as well as web app development skills will be critical for students seeking positions in application development in coming years. But, in addition to technical skills in these areas, employers are looking for soft skills as well. This paper will address a growing need in programming courses for soft skills such as communication skills and teamwork. It will also discuss the need for user-centered design in mobile application development. The researcher will present a project model that aims to help improve communication skills, teamwork skills, and user-centered design skills while developing a mobile application.

Soft Skills

Soft skills, or "people skills," have demonstrated benefits in the information technology (IT) workplace. Soft skills are a combination of interpersonal skills and personal qualities often associated with employees who are able to work well with others in today's complex and rapidly changing IT landscape. Robles (2012) identified the top ten soft skills perceived to be the most important by business executives as integrity, communication, courtesy, responsibility, social skills, positive attitude, professionalism, flexibility, teamwork, and work ethic.

Emotional intelligence, a concept defined by Goleman (1995) also encompasses many soft skills and has gained a great deal of business attention and popularity in recent years in regard to desired skills in the workplace. Emotional intelligence can be defined as "the ability to carry out accurate reasoning about emotions and the ability to use emotions and emotional knowledge to enhance thought" (Mayer, Roberts, & Barsade, 2008, p. 511). It can also be described as a combination of personality traits and self-perceived abilities related to emotions and interpersonal communications (Joseph, Jin, Newman, & O'Boyle, 2015).

In contrast, the technical skills and knowledge needed for a job are often referred to as hard skills (Robles, 2012). Historically, hard skills were the most important job requirement, but in today's complex workplace that is dealing with rapid technological change, hard skills alone are not sufficient for career success (James & James, 2004; Robles, 2012). The importance of soft skills in the workplace has been studied extensively (Deepa & Seth, 2013; Dixon, Belnap, Albrecht, & Lee, 2010; Klaus, 2010; Kyllonen, 2013; Mishra, 2014; Mitchell, Skinner, & White, 2010; Robles, 2012). Klaus (2010) found that 75% of success is dependent upon soft skills, while only 25% is dependent upon hard skills and technical knowledge.

Software development projects today involve an increasing level of complexity that makes them nearly impossible for individual developers to complete alone. In fact, few IT professionals can expect to work in isolation (ACM & IEEE Computer Society, 2016). Software developers must interact not only with a team of other developers jointly developing code, but also with teams that handle databases, IT infrastructure, and security relating to their software projects. Teamwork within software development is key in today's workforce, and in order to work well in a team, a developer must be capable in a variety of soft skills, such as leadership, coordination, conflict management, and communication (Sancho-Thomas, Fuentes-Fernandez, & Fernandez-Manjon, 2009).

While many universities place a high emphasis on soft skills, these skills are often not taught in conjunction with traditional programming courses, which are considered to be highly technical. Traditional programming courses offered by universities do not seem to succeed in helping students to acquire and cultivate these soft skills (Wilhelm, Logan, Smith, & Szul, 2002). Some posit that soft skills are innate personal qualities that cannot be taught like hard skills (Mishra, 2014), though many researchers believe that soft skills can be effectively learned (Shuman, Besterfield-Sacre, & McGourty, 2005) and call for universities to incorporate more soft skill training into their programming courses (Gonzalez-Morales, Antonio, & Garcia, 2011). In addition, the ACM and IEEE Computer Society updated their 2016 guidelines for Software Engineering curriculum to include a call for communication skills, teamwork skills, and other soft skills in software development courses (ACM & IEEE Computer Society, 2016).

There are examples of capstone courses in information technology programs that emphasize soft skills and teamwork in programming environments (Brown, Lee, & Alejandre, 2009), but this is often a single course that a student experiences in his or her final year of schooling. Gonzalez-Morales, Antonio, & Garcia (2011) note that soft skills are not given sufficient attention in early academic courses for a software engineering degree. This researcher argues that soft skills, and in particular communication skills and teamwork, should be incorporated into lower level programming courses as well as capstone experiences. This will give students varied opportunities to practice and refine their communication skills in relation to programming throughout their time at university. To this end, this paper describes experiences from two mobile app development courses: Mobile App Development with iOS and Mobile App Development with Android. Both of these undergraduate courses teach the basics of mobile app development, including the MVC design pattern, user interface creation, data model creation, and programming and testing techniques for the environment. The researcher teaches both of these courses, and developed a mobile app development project within the course to specifically address and require both communication skills and teamwork, along with technical skills in mobile development. A written report and formal presentation of the mobile app project were required to address communication skills. Pair programming was used to reinforce teamwork skills,

and all projects were completed jointly by pairs of students. In each course, an additional emphasis is also placed on user-centered design.

This paper will describe the goals and structure of the mobile app development project, the use of pair programming teams, and issues encountered with sharing work within teams. It will conclude by presenting lessons learned and opportunities for further research.

PROJECT GOALS AND STRUCTURE

The mobile app development project described in this paper required students to develop a functional, multi-scene mobile application for the indicated platform in teams of two. The project included technical requirements as well as requirements for written documentation and an oral presentation of the app. Pair programming was used to incorporate the goal of teamwork. The written documentation and oral presentation were used to incorporate the goal of enhancing communication skills. Each of these components will be further discussed in later sections. A peer review was required at the conclusion of the project.

The mobile app project was assigned in four sections of mobile app development courses (two sections of Android and two sections of iOS development) taught by the researcher between Fall 2016 and Spring 2018. The project was conducted as a half-semester project. After midterms in each 15-week semester, pair programming teams were selected and students were assigned the project proposal. Students had two weeks to complete the project proposal. For some teams, this may have included a few rounds of iterations based on instructor feedback. Once a proposal was accepted by the instructor, teams could begin to work on the project. Pair programming sessions were facilitated during classroom time each week for four weeks. During these classroom sessions, the instructor was able to observe students working in their pairs and provide guidance and suggestions to teams to improve the process. Students were also encouraged to work on projects outside of class time, during which times the instructor could not observe the behavior of the teams or gauge how well they were following the instructions for pair programming. Once the project development tasks were completed, students were assigned to complete the written user guide and to develop the oral presentation. Students were given a two-week timeframe for these activities. Oral presentations were given during class time on the final day of class.

In addition to incorporation of soft skills into the project as a goal of instructor/researcher, the mobile app project was also aimed to aid the student in accomplishing the objectives laid out for each respective mobile app development course.

The project aimed to address the following course objectives for the Android course:

(1) Develop a mobile user interface using input and display controls
(2) Demonstrate user-centered interface design principles for a mobile app
(3) Create a functional Android app that requests data, stores it, and then modifies that data to produce a result through multiple activities

The project aimed to address the following course objectives for the iOS course:

(1) Develop a mobile user interface using input and display controls
(2) Demonstrate user-centered interface design principles for a mobile app
(3) Develop a multi-scene native mobile app for the iOS platform that demonstrates appropriate use of user interface controls and device or framework functionality
(4) Utilize the Swift programming language to code object-oriented functionality for an iOS app, making appropriate use of Swift syntax

The course objectives listed here are not a full list of the course objectives, but rather those objectives that particularly relate to this mobile app project. Note that objectives one and two for both courses are identical. Objective three differs slightly in wording for each course, based upon the types of project examples and labs that are completed in the two different environments (Android and iOS), thus aiming to give students thorough examples

to pattern for this final project. The main goal of objective three is for the student to develop a multi-scene functional app in the given platform. The iOS course includes one additional goal, objective four, that pertains to the project. This objective relates to the use of the Swift programming language, which was developed by Apple and is used exclusively for iOS/Mac development. Due to the specificity of the language used for iOS development, this course includes a substantive component at the beginning of the course that teaches students the syntax and other rudiments of the Swift language. Objective four attempts to address that additional learning component. Students taking the iOS course are required to have completed a prerequisite of any other introductory programming language; this ensures that Swift is not the student's first exposure to programming, and therefore a "boot camp" style component of the course to teach Swift fundamentals is deemed adequate. The Android course utilizes the Java programming language, and students in that course must have completed a prerequisite course in Java. Therefore, no additional time is spent in the course in discussion of the programming language itself.

Technical Component
The technical component of the mobile app development project included requirements for the actual mobile software development of a functional app. The requirements were broken down as follows (each percentage indicates the percent of the technical component project grade):

- At least Four Interactive Scenes (30%)
- At least Three Interface Controls (20%)
- App Icon, Launch Screen, and Graphics (10%)
- User-Centered Design (20%)
- Successful Build and Smooth Run (20%)

Interactive scenes. In a mobile app, a scene refers to one "screen" or "page view" within the app. In the Android environment, scenes are represented by Activities that are created as XML files, either via code or using a graphic drag-and-drop interface, and controlled programmatically by a Java code file. In the iOS environment, scenes are represented by a combination of a View Controller object on the Storyboard (canvas providing a visual representation of the user interface) and a Swift code file that programmatically controls the View Controller. The Storyboard elements in iOS are also represented behind the scenes by XML, but typically are controlled directly through a drag-and-drop graphical interface, rarely exposing the XML code. In terms of building what the researcher would deem a significant mobile app project, the requirement was set at inclusion of at least four interactive scenes. Interactive scenes were defined for the student as a scene that contains functional user interface controls. Other scenes were encouraged as needed in the app; the four scene requirement was merely a minimum.

Interface controls. Interface controls are constructs used in a mobile app user interface that can allow for input from the user or display output to the user. Some common examples of user interface controls are buttons, which can be clicked to launch specific functionality, text fields, which can accept text-based input from users, labels, which are used to display text-based output, check boxes, which allow selection of one or more options from a set, and radio buttons, which allow for only one selection from a set ("Human interface guidelines", 2018; "User interface", 2018). In addition to these basic types of interface controls, there are many more complex controls available in both mobile development environments. For instance, both the Android and iOS environments contain a variety of spinner and picker controls that provide various types of drop-down menu selection options with varied aesthetics. There are also controls that allow for the display of different types of data. Some examples that exist in both environments include table and list views to display lists of items, and grid or collection views that allow for a variety of grid-based displays for images and other media ("Human interface guidelines", 2018; "User interface", 2018). For this project, students were required to select at least three different interface controls for inclusion in their user interface. Certainly, more interface controls were likely but students had to implement three at a minimum.

App icon, launch screen, and graphics. The aesthetic look and feel of a mobile app can be as important as the app's functionality in terms of appeal to potential users (Morgan, 2017). Though both of the courses are elective courses targeted toward information technology majors, some time and consideration within the course were devoted to a discussion of basic graphic design elements and use of fundamental features of a graphics editing program (Adobe Photoshop Creative Cloud). Students were instructed to make their app look as professional as possible through the use of a custom-designed app icon and launch screen. In addition, students were encouraged to

utilize professional-looking images and graphics throughout the app's content as appropriate. Several copyright-free image repositories and icon-creation tools were reviewed and provided to students as resources.

User-centered design. User-centered design should be incorporated into every software development project. Seyam & McCrickard (2015) note that developers may not give enough attention to user interface and user experience design, especially in small companies. User-centered design has been used to create more compelling, useful, and simple to understand mobile apps (Babich, 2018; Bodine, 2018). User-centered design has its roots in Norman's research (Norman, 1988; Norman & Draper, 1986), and now commonly refers to the study and consideration of how a user will interact with a software system, with the goal of making the user experience as simple and intuitive as possible. User-centered design is often used synonymously with usability, which is defined as a quality attribute of the user interface; concerned with whether a system is easy to learn and efficient to use (Norman & Nielsen, n.d.) and is closely aligned with user experience (UX) design, which emphasizes a full understanding of a user's entire experience while using a system as well as all aspects of the user's interactions with a company and its products (Norman & Nielsen, n.d.).

A portion of the course is devoted to discussing user-centered design concepts, including both aesthetic concepts as well as functional ones. Some examples of aesthetic user-centered design concepts for mobile development are formatting content to fit the screen of a mobile device (often referred to as responsive design), making sure that there is ample contrast between foreground text and background so that text is always easily legible, and using white space effectively so that an interface does not appear too cluttered ("Human interface guidelines", 2018; "UI design", 2018). Though it might be easy to assume that most user-centered design concepts are aesthetic in nature, most of the concepts are actually related to building intuitive and simple functionality.

Some examples of functional concepts of user-centered design include the creation of touch-based controls at a recommended size so that they can be easily and accurately tapped with a finger, organizing interface elements to place controls close to the elements that they modify ("Human interface guidelines", 2018; Kuusinen & Mikkonen, 2014; "UI design", 2018), organizing and labelling menu categories to be user-friendly, allowing users to "go back" easily in one step, prominently displaying a search field, using effective index searching (so that relevant content is returned), providing filter and sort options, and providing clear utility before asking users to register (Gove, 2016). In addition, it is critical to select the proper interface controls to accomplish the task required in the most simple way. For example, when asking a user to input a phone number, the app should provide a numeric keyboard that allows for easy selection of numbers and only numbers (Gove, 2016). A drop-down list can be used to provide easy selection of items rather than allowing free text input, in all cases where this is appropriate. In addition, text-based hints can be used within text fields to provide users with an example of the kind of data (and its format) that is expected in the field. This aids in reducing the time spent in supplying user input because a user will make less mistakes in terms of data formatting. These are just some examples of user-centered design concepts that are discussed in the courses.

Seyam & McCrickard (2016) found that user interface (UI) and user experience (UX) requirements should be made explicit to students for mobile development projects, rather than assuming that students will be able to implement these elements if given a general guideline to pay attention to this area. This provides additional support for inclusion of a specific technical requirement for user-centered design within the technical requirements for the project.

Successful build and smooth run. Mobile app projects are required to build successfully with no syntax errors. In addition, they are required to run in the appropriate simulator software (iOS Simulator or Android Emulator) without runtime errors. Students are required to include error handling for all parts of the application to ensure a smooth run. This portion of the project is more technical in nature and allows for a thorough evaluation of the team's grasp of the programming language used (Java for the Android course and Swift for the iOS course).

Communication Skills Component

ACM and IEEE Computer Society (2016) notes that many problems of teamwork relate to poor communication skills and thus calls for increasing course work at all levels for software engineers that includes formal written reports, formal oral presentations to a group, and opportunities to critique written reports and oral presentations.

These opportunities for practicing and refining communication skills should be present even in early programming courses offered to students.

Communication skills are also emphasized as important by the SWEBOK Guide to the Software Engineering Body of Knowledge that was developed by the IEEE Computer Society (Bourque & Fairley, 2014). It is vital that developers are able to communicate well in both written and oral formats. This type of communication is critical for the successful attainment of software requirements, as accurate requirements depend upon a clear understanding between developers and customers, management, teammates, and vendors (Bourque & Fairley, 2014).

Writing clearly and concisely is extremely important as written emails, status reports, and product documentation will be a primary method of communication within a software development team. It is critical that all written reports and documentation relating to a software product are written in an accessible way so that they are understandable and relevant to their intended audiences. In a professional setting, software developers will often be required to produce project plans, requirements documents, risk analyses, design documents, test plans, user manuals, and technical reports (Bourque & Fairley, 2014).

Developers should be able to present to both large and small groups. Presentation skills are emphasized as a requirement for software engineers (Bourque & Fairley, 2014). In a professional setting, developers will often be required to present prototypes, lead product walk-throughs and review sessions for customers as well as teammates and management, and solicit both feedback and support for products as part of these processes. A software developer's ability to convey concepts effectively in a presentation can influence management and customer support, and therefore significantly impact product acceptance. If a developer is skilled in discussing technical issues with non-technical audiences, it can also influence the ability of customers and other stakeholders to more adequately comprehend and assist in the product design effort (Bourque & Fairley, 2014).

In order to address both written and oral communication skills within the mobile app project, students were required to create a written project proposal, a written user guide, and a presentation of their app which including the development of Powerpoint slides and an oral presentation to the class.

Project proposal. Students were required to write a project proposal for the instructor that concisely outlined the idea for their mobile app project, providing clear plans for overall functionality. The proposal was to include a detailed plan for the implementation of at least four interactive scenes and three different interface controls. The proposal was also to address at least three areas in which user-centered design would be considered and implemented within the project. Project proposals were required to be original, meaning that two teams could not work on projects that were too similar, and a team could not copy something that had already been done in an online tutorial or live app. The subject and functionality of the app was left completely up to the team. Each team was required to submit their project proposal to the instructor for final approval. During review of the proposals, the instructor was able to help students gauge the viability of the project given the current skillset of the class and the timeframe available during the course. The instructor did not turn away any proposal based on topic selection; changes were required or recommended based on the gauged difficulty level and complexity of what the teams were proposing to accomplish. The format of the proposal was to be professional, at least two pages in length, and cover all of the required content. Students were encouraged to write in the active voice and to craft clear and simple descriptions of their plans.

User guide. Teams were required to write a 5-8 page user guide upon conclusion of the development of their mobile app project. The user guide was required to contain a simple description of the app and its functionality, its intended use and purpose, and guidelines for using all features of the app. Guides were to be formatted professionally, and written with end users as the intended audience. Technical jargon was to be avoided, and attention paid to creating a document that was easy to read and relevant to end users. Teams were encouraged to include screenshots and graphical guides to make the instructions more clear to readers.

Presentation. Lastly, teams were required to give an oral presentation about their mobile app project, including a live demonstration of the app's features. Teams were instructed to consider the class as a room full of executives at a company where they work as a mobile app development team. They were to tailor their presentation toward this management-level audience. The presentation was to be primarily explanatory in nature, and teams were instructed

to highlight a technical aspect of the project in a way that would make it easy to understand for an end user unfamiliar with the technology. Lastly, teams were asked to conclude their presentations by providing a set of future development plans for their app, which provided a way for them to practice a small persuasive argument and allowed the instructor to gauge how clearly the team could articulate their development plans.

Teamwork Component

ACM and IEEE Computer Society (2016) emphasize the need for students to have opportunities to work in teams during software development beginning early in the curriculum. This kind of teamwork can be practiced most effectively when employed for a significant project that involves the complex design and implementation of a product, undertaken by a small student team.

In this mobile app development project, teamwork was incorporated through the use of pair programming. Pair programming is an approach where two students work together on the same development task. One student typically writes the code while the other student watches. The coding student is often called the "driver." The student watching plays a "navigator" role, where he or she provides advice and suggestions to the driver, helps to look up syntax and errors as needed, and reviews the "big picture" of the code, looking for errors and scanning for ways to improve. The students exchange roles at regular intervals so that one student is not always coding, and so that each has a chance to experience the benefits of each role (Seyam & McCrickard, 2016).

Pair programming has been shown to be highly effective and to provide additional benefits to students learning programming (Celepkolu & Boyer, 2018). Students in pair programming teams often experience knowledge transfer from one another during the experience, and will also utilize communication skills in addition to technical skills as part of the process (Zieris & Prechelt, 2014). However, there may be an additional challenge in using pair programming for mobile development as the tasks in programming for mobile devices are far more encompassing than merely writing code, especially if you are using actual mobile devices for testing rather than a simulator. Seyam and McCrickard (2016) found that students within pairs working on mobile development spent time searching online resources, reviewing lecture materials, dealing with connectivity issues related to testing their mobile devices (smartphones and smartwatches), and learning how to use the features of their mobile devices. These challenges may not exist in traditional programming courses. Seyam & McCrickard (2016) also found that while pair programming guidelines discourage students from using parallelism, many pairs ended up splitting up the varied project tasks, which may have led to a decrease in the effectiveness of pair programming. Celepkolu and Boyer (2018) confirm that students who create a shared solution show higher learning gains and more satisfaction with pair programming, while students who are responsible for their own code tend to talk less with their teammate and experience fewer benefits.

For this project, students were permitted to self-select into pairs within the classroom environment. In some sections where there was an odd number of students, one group sometimes had to include three students out of necessity. In one section where there was an odd number of students, a student volunteered to complete the project individually. The majority of students in all sections were able to work in teams of two, as suggested.

Seyam & McCrickard (2015) argue that the higher the interaction level among developers using pair programming, the better user experience (UX) design they will achieve, in particular for mobile app development. This provides support for using pair programming in a mobile development project where user-centered design will be emphasized.

ISSUES ENCOUNTERED WITH SHARING WORK WITHIN TEAMS

Two primary issues were encountered amongst project teams during the four sections of the course. These issues were discovered by the researcher via observations of lab time, student comments, and emails sent to the researcher/instructor, and via project scores and peer evaluations.

The first issue related to an imbalance of dedication level within pairs for teams. In a few cases, one student in a pair would carry the workload for the entire team, while the other student slacked off, either due to absences from class during project lab time, or lack of communication and participation with their team member in general. These

issues were noted by the instructor if visible during lab time, and communication with the errant team member was attempted. But, in some cases, the student was simply not responsive and thus the dedicated team member was left with a project experience that was not engaging or satisfactory, and had to do much, if not all, of the work themselves. This problem is a concern, but it is also a common issue with group projects in general and may not be specific to pair programming or to this project.

The second issue related only to the two sections of the course focusing on the iOS platform. Apple requires that all iOS development is done on a Mac. The course was held in a Mac lab, and two other Mac labs were available on campus for students to utilize for work outside of class time during the course. However, students who commuted to campus and did not own their own Mac found it difficult to complete the project work because they were unable to practice or work at home at convenient times, and instead were only able to work on the project when they were able to physically get to the campus. This was not an issue for the Android course sections, as Android apps can be developed on either a PC or a Mac.

LESSONS LEARNED

The researcher feels that this project model for a mobile app development project has been successful for the four course sections reviewed. Student teams were largely able to complete the project and leave the course with a substantial, working mobile app. The project was able to emphasize communication skills via the user guide and oral presentation assignments. Teamwork was emphasized through the use of pair programming. User-centered design was included explicitly as a requirement, which emphasized its importance. However, there were some challenges for the project, and some areas for improvement.

One challenge encountered related to the effectiveness of the pair programming model. While this model worked extremely well for some pairs, others were not as effective. Success in the pair programming model was largely dependent on each team member contributing their half of the work. This was only a problem for a small number of pairs where one member was often absent during lab time or did not respond to their team members communications in a timely manner. In these situations in the future, it may be necessary to regroup and require a non-attending member to complete the project alone, while joining the participating member to an already existing group. Further study is needed to determine the most effective way to handle situations like this when they arise.

In addition, some teams had both members participate fully, but did not equally share the workload for all tasks. For example, rather than jointly working on the interface, the coding, the documentation writing, and the development of the slides for the presentation, some pairs split these tasks up and completed their parts of the project individually. If this was noted during lab time, the instructor encouraged the team to work together. But, work time outside of class could not be monitored and anecdotal evidence from students indicated that some teams took this compartmentalized approach. Future study is warranted to look at ways to further encourage true task sharing within the pair programming model, particularly for mobile development.

In mobile development, the user interface is often complex and created as the first step in an app development project. In this project, development work was treated as one four-week chunk of lab time in which the teams largely organized the tasks for their own development. Some teams spent a great deal of time troubleshooting that could have been avoided had they completed and tested their user interface before beginning the coding for their app. The problems usually occurred because the team was partially finished with the user interface and partially working on the coding. This approach makes it more difficult to pinpoint errors. As a future recommendation, the researcher would restructure the development time to include two weeks dedicated to user interface development followed by two weeks dedicated to project coding. A checkpoint assignment or sign-off after interface development could be very helpful for keeping student teams on track.

Lastly, though the researcher carefully thought through the project goals and intentionally included components to address communication skills, teamwork, and user-centered design, the course objectives for the courses themselves did not reflect any of these items. It was only in writing this paper that the researcher became aware that a number of the skills and components in this project are not adequately represented in the course objectives and need to be

added. This is perhaps indicative of the habit within IT of leaving soft skills out of more technically-based courses. This mindset is something that needs to change.

In conclusion, this mobile app development project is presented as a model to help promote the inclusion of soft skills such as communication skills and teamwork within programming courses in the IT curriculum. It is also intended to promote a strong awareness of the need for user-centered design to be included in mobile development projects.

REFERENCES

ACM & IEEE Computer Society. (2016). Computer engineering curricula 2016: Curriculum guidelines for undergraduate degree programs in computer engineering. Retrieved from https://www.acm.org/binaries/content/assets/education/ce2016-final-report.pdf

Babich, N. (2018). A comprehensive guide to mobile app design. Smashing Magazine. Retrieved from https://www.smashingmagazine.com/2018/02/comprehensive-guide-to-mobile-app-design/

Bodine, K. (2018). The importance of UX and UI in mobile app design. Retrieved from http://colure.co/the-importance-of-ux-and-ui-in-mobile-app-design/

Bourque, P., & Fairley, R. (Eds.). (2014). SWEBOK V3.0: Guide to the software engineering body of knowledge. *IEEE Computer Society*, 1-335.

Brown, Q., Lee, F., & Alejandre, S. (2009). Emphasizing soft skills and team development in an educational digital game design course. *Proceedings of the 4th International Conference on Foundations of Digital Games*, 1-8.

Celepkolu, M., & Boyer, K. (2018). The importance of producing shared code through pair programming. *Proceedings of the Special Interest Group on Computer Science Education (SIGCSE)*, 765-770.

Deepa, S., & Seth, M. (2013). Do soft skills matter? – Implications for educators based on recruiters' perspective. *IUP Journal of Soft Skills, 7*(1), 7-20.

Dixon, J., Belnap, C., Albrecht, C., Lee, K. (2010). The importance of soft skills. *Corporate Finance Review, 14*(6), 35-38.

Gillett, R. (2018). The 20 best tech jobs in America in 2018. Business Insider. Retrieved from http://www.businessinsider.com/best-tech-jobs-in-america-2018-2#3-mobile-developer-18

Goadrich, M., & Rogers, M. (2011). Smart smartphone development: iOS versus Android. *Proceedings of the Special Interest Group on Computer Science Education (SIGCSE)*, 607-612.

Goleman, D. (1995). *Emotional intelligence.* New York, NY: Bantam.

Gonzalez-Morales, D., Antonio, L., Garcia, J. (2011). Teaching "soft" skills in software engineering. *Proceedings of the IEEE Global Engineering Education Conference (EDUCON)*, 630-637.

Gove, J. (2016). Principles of mobile app design: Engage users and drive conversions. *Think with Google*, 1-30. Retrieved from https://www.thinkwithgoogle.com/_qs/documents/23/principles-of-mobile-app-design-engage-users-and-drive-conversions.pdf

Human interface guidelines. (2018). Apple. Retrieved from https://developer.apple.com/ios/human-interface-guidelines/overview/themes/

Joseph, D., Jin, J., Newman, D., & O'Boyle, E. (2015). Why does self-reported emotional intelligence predict job performance? A meta-analytic investigation of mixed EI. *Journal of Applied Psychology, 100*(2), 298-342.

Klaus, P. (2010). Communication breakdown. *California Job Journal, 28*, 1-9.

Kuusinen, K., & Mikkonen, T. (2014). On designing UX for mobile enterprise apps. *Proceedings of the 40th Euromicro Conference on Software Engineering and Advanced Applications, 17*, 221-228.

Kyllonen, P. (2013). Soft skills for the workplace. *The Magazine of Higher Learning, 45*(6), 16-23.

Mishra, K. (2014). Employability skills that recruiters demand. *IUP Journal of Soft Skills, 8*(3), 50-55.

Mitchell, G., Skinner, L., & White, B. (2010). Essential soft skills for success in the twenty-first century workforce as perceived by business educators. *Delta Pi Epsilon Journal, 52*(1), 43-53.

Morgan, P. (2017). Why app design is important to it's functionality. *Socialnomics*. Retrieved from https://socialnomics.net/2017/08/05/why-app-design-is-important-to-its-functionality/

Norman, D. (1988). *The design of everyday things.* Doubleday Press.

Norman, D., & Draper, S. (1986). *User-centered system design: New perspectives on human-computer interaction*. CRC Press.

Norman, D., & Nielsen, J. (n.d.). The definition of user experience (UX). Retrieved from https://www.nngroup.com/articles/definition-user-experience/

Robles, M. (2012). Executive perceptions of the top 10 soft skills needed in today's workplace. *Business Communication Quarterly, 75*(4), 453-465.

Sancho-Thomas, P., Fuentes-Fernandez, R., & Fernandez-Manjon, B. (2009). Learning teamwork skills in university programming courses. *Computers & Education, 53*, 517-531.

Seyam, M., & McCrickard, D. (2015). Collaborating on mobile app design through pair programming: A practice-oriented approach overview and expert review. *2015 International Conference on Collaboration Technologies and Systems*, 124-131.

Seyam, M., & McCrickard, D. (2016). Teaching mobile development with pair programming. *Proceedings of the Special Interest Group on Computer Science Education (SIGCSE)*, 96-101.

Shuman, L., Besterfield-Sacre, M., & McGourty, J. (2005). The ABET "professional skills" – Can they be taught? Can they be assessed? *Journal of Engineering Education, 94*(1), 41-55.

Sidow, L. (2018). Global app store records shattered yet again in Q1 2018. *App Annie*. Retrieved from https://www.appannie.com/en/insights/market-data/q1-2018-apps-record-downloads-spend/

Stahl, A. (2018). Seven top in-demand jobs in 2018. *Forbes*. Retrieved from https://www.forbes.com/sites/ashleystahl/2018/03/26/top-7-jobs-in-demand-in-2018/#5d58055e3628

UI design do's and don'ts. (2018). Apple. Retrieved from https://developer.apple.com/design/tips/

User interface & navigation. (2018). Android. Retrieved from https://developer.android.com/guide/topics/ui/

Wilhelm, W., Logan, J., Smith, S., Szul, L. (2002). Meeting the demand: Teaching "soft" skills. *Delti Pi Epsilon Society*, 1-82.

Zieris, F., & Prechelt, L. (2014). On knowledge transfer skill in pair programming. *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1-10.