https://doi.org/10.48009/4_iis_2020_202-211

Issues in Information Systems
Volume 21, Issue 4, pp.202-211, 2020

USER EXPERIENCE (UX) DESIGN CONCEPTS FOR MOBILE APP DEVELOPMENT COURSES

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

ABSTRACT

Mobile apps have unique characteristics and challenges that differentiate them from traditional software applications. These characteristics and challenges make it critical that mobile apps follow user experience (UX) design principles if they are to be noticed as quality applications within the millions of apps available in the app stores. This paper outlines these unique characteristics and challenges, and discusses the evolution of user experience (UX) design and its applicability to mobile apps. The author asserts that UX design should be taught as a critical component of mobile app development courses, along with coding and other traditional mobile programming topics. A framework of seven UX design principles for mobile app development is described and proposed.

Keywords: User Experience, UX Design, User-Centered Design, Usability, Mobile Application Development, Mobile Programming

INTRODUCTION

Smartphones and other mobile devices have been a worldwide phenomenon, with the global number of mobile phone users growing steadily since 2007. Their use has become so prevalent, that many Americans are foregoing traditional desktop and laptop computers with high-speed home Internet service in favor of mobile devices and cellular service (Anderson, 2019). Within this new mobile computing market, mobile applications (or apps) are projected to continue to increase as well. In 2019, there were more than 2 million Android apps available for download in the Google Play store and 1.83 million iOS apps available in the Apple App Store. While the Google Play store consistently offers more apps and boasts more app downloads, the Apple App Store dominates in terms of consumer spending. In 2019, Apple accounted for 64% of worldwide app store revenue, with Google accounting for the remaining 36%. Mobile apps are projected to generate over 9935 billion U.S. dollars via paid downloads and in-app purchases and advertising by 2023 (Clement, 2019).

It is clear that the future of the web is mobile, and programming mobile apps is a skill that will be necessary for information systems students in the foreseeable future. Quality mobile apps require not only error-free functionality, but also simplicity and ease of use. The mobile nature of the devices running these apps provides a variety of use cases related to on-the-go tasks that do not apply to other types of traditional computing. Innovative use of location data via GPS and accelerometers in mobile devices allow for apps that provide location-aware features. Some of these features include tracking the distance and elevation of a run, providing real-time traffic updates, continually re-routing the optimal path to a driving destination, and displaying a list of restaurants matching cuisine style and customer rating preferences in the local vicinity. The ability to take photos and record audio and video via mobile devices is another example of a set of use cases that are not practical for traditional computing. For instance, a mobile app can allow a user to take a photo of a plant growing on the side of a trail they are hiking and receive instant feedback on the species of plant in the photo. This type of photo identification can be done via traditional computing, but it would not be as useful as it is in a mobile context, where the feedback can be received at the time and place where the unknown plant is encountered. These types of mobile moments, where a smartphone user has an immediate time- or location-bound need to record or receive information, can provide app developers with a plethora of use cases for new and innovative programming features.

However, all mobile apps are not considered to be equal. Apps that are successful are generally considered to be those that are designed well, having both a simple and intuitive interface, and providing clear and focused actionable tasks that can be accomplished when and where the user needs them. Flora et al. (2014) surveyed 130 mobile application developers, experts, and relevant stakeholders and found that 75% of respondents considered “user experience” to be

a crucial and deciding factor for a good mobile app. Respondents noted that "a user should feel comfortable in gadget interaction and feel smart enough to accomplish any task with intuitive use, without any tutorial or additional help" (Flora et al., 2014, p. 24). This notion speaks to the clear importance of user experience design for mobile apps. User experience design has evolved over time from the concept of software system usability.

Evolution of User Experience (UX) Design

User-centered design is often discussed interchangeably with the concept of usability. The term *usability* has been used in many contexts, but in the field of information systems, it has largely been used in regard to software development and testing and generally refers to the ease of use of a particular system. The official ISO definition of usability is "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (Interaction Design Foundation, n.d., para. 2). Nielsen (1993) is widely cited in the field of usability and defines usability as relative, rather than absolute. He notes that a design cannot be said to have good usability, with absolute certainty. Rather, you can say that one design is more usable than another based on a specific measure of effectiveness, efficiency, or satisfaction. Usability, in the area of software and web development in particular, has come to be associated with usability testing, or the process through which users of software systems can be observed working with the system, and ease of use can be improved by noting and fixing issues, including confusing navigation, extraneous clicks (or taps), and other issues.

The trend in the information systems field has moved away from the term usability, and many would argue now focuses on user-centered design. Norman (1988) coined the term *user-centered design* and it later became popularized in his seminal book *The Design of Everyday Things*, which is still largely considered the authority on user experience. This term marked a shift from a focus on software system features and aesthetics to concentrating on the needs and experience of the user. Software developers are often introduced to user-centered design in the context of systems analysis, during the phase of a software project where conceptual design is handled. The concept of a use case, or scenario depicting a specific use of the software system by a user, is often employed to aid developers in understanding software requirements. Further, some analysts and developers create personas to aid in the conceptual design of a system. A persona is a fictional, archetypical user that represents a certain group of expected users for the app (Anvari et al., 2019). A variety of personas may be created. Anvari et al. (2019) further argue that since users are often from many different cultures, personas that represent cultural differences are also necessary for true user-centered design. Lazar et al. (2019) found that personas can also be helpful for addressing accessibility issues for users with disabilities.

Norman (1988) went on to join Apple Computer in the early 1990s as a fellow and then as "User Experience Architect", becoming the first person to have a job title including the "user experience" identifier (Lyonnais, 2017). Norman (2003) expanded his work to consider the emotional impacts of design, further adding to the holistic understanding of the user experience. Not surprisingly, we have seen another trend in information systems toward the term *user experience (UX) design*. UX design is an even broader sense of everything that the user experiences when using an app, from hardware to software to interactions, and encompasses both design characteristics and process. User-centered design clearly improves the user experience and is a facet of UX design (Novoseltseva, 2020). Norman and Nielsen (n.d.) also distinguish between usability and UX by noting that usability is a quality attribute of the user interface concerned with whether the software is easy to learn and efficient to use. UX is a broader concept.

The evolution in terminology from usability to user-centered design to user experience (UX) design is meaningful in that the understanding has shifted from a narrower viewpoint to a broader view. The shift recognizes a more holistic nature of application design, taking into account a variety of concepts including usability, accessibility, aesthetics, information architecture, simplified functionality, intuitiveness, responsiveness to context, user-centeredness, user experience, etc. Bias et al. (2012) describe this by noting "a finished design is a gestalt, where the whole is greater than the sum of its parts" (p. 3). UX design encompasses this holistic view of all user experiences with an application. In this paper, the term user experience (UX) design will be used to refer to this full range of concepts.

The Need for UX Design in Mobile App Development Courses

Logic should dictate that UX design should be included as a necessary component of any software development project. This echoes Bias et al.'s (2012) argument that "any discipline that involves any type of design will (or should) have an essential interest in usability" (p. 276). However, it is not common for UX design to be included in information

systems curricula (Lazar et al., 2019). UX design is often seen as something separate from development and is more often found in universities within graphic design and media arts programs. In fact, studies that have examined the concepts that should be included in mobile programming courses often include user interface and design topics (Dang & Skelton, 2019; El-Tawab et al., 2018; Esakia & McCrickard, 2016; Gordon, 2013; Khmelevsky & Voytenko, 2016; Stuurman et al., 2014; Sung & Samuel, 2014), but do not explicitly include UX design concepts.

Having taught mobile app development for seven years, for both the iOS and Android platforms, I would argue that students in mobile programming courses have a particularly critical need to understand UX design. For mobile apps to stand out from the millions of other apps in the app stores and be successful and profitable, they must be simple and intuitive to use and they must provide functionality that clearly addresses one, or a few, specific use cases. Nicol (2013) notes that mobile apps differ from traditional applications from the PC and web era by requiring simplified information delivery. Successful mobile apps deliver the right information at the right time and place. Nicol (2013) asserts that this can be accomplished by radically simplifying interactions to only expose the essential elements needed on-screen in the moment. This includes delivering the interactions via a simple, intuitive interface. An example of this type of simplified action and interface is using a smartphone to take a photo of a check with your mobile banking app and having this photo automatically allow for a deposit into your checking account. Another example is ride sharing apps such as Uber and Lyft. These apps provide a transformative approach by replacing a user's need to call or hail a taxi. Instead, users can utilize an app to notify drivers that they need a ride. The app knows the user's location automatically. The user is empowered to select a vehicle type and driver from those that are nearby and available. The user gets a time estimate for pickup and can visually see the driver approaching them on a map. Payment is handled through the app so there is no need for cash. These examples illustrate the importance of UX design in developing mobile apps.

The purpose of this paper is both to assert that UX design is a critical topic for mobile app development courses, and also to propose a framework of UX design concepts that should be included in such courses. In order to fully understand the UX design concepts that will be proposed for mobile app development courses, one must first have some understanding of the unique challenges and development environment that surround mobile app development. This environment and its challenges will be discussed in the next section. The remainder of the paper will propose a framework of UX design concepts for mobile app development courses.

UNIQUE CHALLENGES FOR MOBILE APP DEVELOPMENT

According to Joorabchi et al. (2013), mobile apps generally fall into one of three categories: native apps, web apps, and hybrid apps. Native apps are built for one specific platform (namely Android or iOS) while web apps utilize HTML 5, JavaScript, and Cascading Style Sheet (CSS) technologies to build apps that are cross-platform. While cross-platform capability is extremely attractive due to the lower expense of developing and maintaining only one app, web apps are often slower in terms of performance and cannot offer all of the functionality of a native app that takes advantage of a device's hardware features (such as camera, microphone, accelerometer, GPS, etc.). This has led to the development of hybrid apps. As of 2020, there are no set standards for hybrid apps, and thus they are still being defined as new innovative approaches continue to appear in the market. Most hybrid approaches use proprietary software that creates cross-platform apps by building interfaces using web technologies and backend functionality using a "bridge" that takes advantage of native app programming and techniques. Dang and Skelton (2019) identified the leading hybrid app technologies as Appcelerator Titanium (introduced in 2008 by Appcelerator, Inc.), Cordova (formerly PhoneGap, introduced in 2009 by Adobe Systems), Xamarin (introduced in 2011 and now owned by Microsoft), Ionic (introduced in 2015 by Facebook), React Native (introduced in 2015 by Facebook), and Flutter (introduced in 2017 by Google). While some of these technologies have been around longer than others, they all provide excellent development frameworks and can offer a variety of cost savings and advantages to enterprises interested in building apps. However, due to the lack of standards, an enterprise necessarily locks themselves into a proprietary technology in order to truly take advantage of building hybrid apps.

The choice and trade-offs between these three types of mobile apps can be difficult for enterprises and app developers. Each provides both advantages and disadvantages in terms of cost to develop, time to develop, ease of maintenance, ease of delivery to customers, app performance, and overall user experience. Joorabchi et al. (2013) interviewed 12 senior mobile developers from 9 different companies and surveyed 188 mobile developers. Of those surveyed, 82% had native development experience, 11% had tried hybrid app solutions, and 7% had developed mobile web apps.

The majority of the survey respondents were in favor of native apps over the other two types. Their study found that native apps were preferred when better user experience or device features are needed, or when development cost was not an issue. Their participants noted that mobile web doesn't feel or look like any of the platforms; this was seen as a negative for web apps. A number of participants had tried hybrid apps but moved away from that approach in favor of native apps. This echoed Facebook's often cited 2012 announcement that their trial of a hybrid format for the Facebook app was a mistake, noting that the social media company would depend upon native app technology moving forward after suffering slow-downs and multiple other issues with the hybrid version (Hardawar, 2012). However, Facebook has since made advances in the hybrid app arena by introducing the Ionic and React Native platforms (Dang & Skelton, 2019).

In addition, there is a challenge of device and operating system (OS) fragmentation. Android apps can run on a variety of smartphones, tablets, and other devices that physically represent a wide variety of screen sizes, screen resolutions, and device manufacturers. Each of these devices may also include different device hardware features, such as specific camera options, that differ from each other. iOS apps can only run on hardware developed by Apple, but this also includes a variety of iPhones, iPads, iPods, and other devices such as Apple Watch and Apple TV, that vary in screen size, screen resolution, and hardware features. These differences in mobile devices provide a significant challenge for app developers who must create applications that function well for all devices. In addition to hardware differences, app developers must also contend with mobile OS differences. Mobile operating systems tend to be updated frequently, with a new OS launched each year, and many minor updates pushed to users during the year. The many possible combinations of hardware and software for mobile devices can provide a significant challenge for developing and testing mobile apps. The iOS environment tends to have less OS fragmentation than the Android environment. Simulator and emulators can be used to aid in this testing, but are not adequate for all testing needs.

In terms of user interface, both Android and iOS provide design guidelines for app development (Android, n.d.; Apple, n.d.-a; Apple, n.d.-b). Apple names these Human Interface Guidelines (Apple, n.d.-a) while Android calls their guide Material Design Guidelines (Android, n.d.). Apple's Human Interface Guidelines are more extensive and specific and may be one of the reasons that iOS apps are often lauded for simplicity and ease of use. These guidelines are not truly design patterns or coding recommendations, but rather represent a collection of ideas, thoughts, and principles for design best practices for mobile apps. The term *design* might be construed to refer to graphic or visual design, and this is part of what the guidelines discuss. Graphic design issues such as ensuring contrast and legibility of text, and providing images without distortion at proper resolutions are included. But, the guidelines delve further into user experience issues such as using appropriately sized hit targets for tappable areas (for buttons, sliders, etc.) to be convenient for the average human's finger size (Apple, n.d.-b). Information design issues including the use of whitespace, alignment, and proximity to organize content in a way that conveys proper information associations between elements are also covered (Apple, n.d.-b). One challenge for user-centered design applies directly to these interface guidelines. When the same app is developed for two different native platforms (Android and iOS), it can be difficult to create a consistent interface design that adheres to each platform's individual standards (Joorabchi et al., 2013).

Mobile app developers face many issues, including choosing the appropriate type of app and app technology, considering device and OS fragmentation during development and testing, and developing quality user interfaces that are consistent between platforms while still conforming to platform standards. The combination of all of these obstacles presents a major challenge for app developers. In addition, user expectations regarding the user experience of apps is increasing (Holzinger et al., 2012). These challenges further highlight the need for UX design concepts to be included in mobile programming courses.

FRAMEWORK OF UX DESIGN CONCEPTS FOR MOBILE APP DEVELOPMENT COURSES

The concepts of UX design can (and should) be applied to all software and web development projects. Further, there are some concepts that apply specifically and critically to mobile app development. I group these concepts into the following seven categories.

1) Deliver value by leveraging context
2) Simplify information delivery
3) Provide the simplest way to enter data correctly

4) Manage user interactions and navigation
5) Employ information architecture to deliver content logically
6) Use feedback to convey state
7) Leverage aesthetics for clarity of information

The following sections will introduce and discuss each of these concepts in further detail.

Deliver Value by Leveraging Context

One of the unique ways in which mobile apps can provide value to differentiate themselves from other types of software is by leveraging context (Nicol, 2013). Context can be environmental in nature, personal in nature (related to the user), and historical in nature, such as mobile intelligence gathered from enterprise or cloud-based systems. Environmental context can include data gathered from GPS other device sensors such as the accelerometer, camera, and microphone. Data such as location, speed, elevation, position, video, and sound can be captured via these sensors (Nicol, 2013). Environmental context is perhaps the most differentiating feature of mobile devices vs. their traditional software counterparts. Environmental data can be used in many innovative ways, such as using location and time to offer a user open reservations at nearby restaurants, or sensing that a user has just walked into a museum or ballpark and loading a map of the area for them to view.

Personal context can be derived from an understanding of the user and the goals that he or she has for the app. Adjusting the app so that it meets these personalized goals can provide unique value. Some personal context data may be able to be pulled from other mobile apps or the device OS. Other personal context can be derived through the creation of user personas that are representative of specific user groups during the development process. This can be extremely beneficial in personalizing the app, and should take into consideration both cultural and accessibility-related characteristics (Anvari et al., 2019; Lazar et al., 2019). Personal context can also include content that is specific to the current user. For instance, an app could highlight the user's reserved seat within a football arena on a map and provide step-by-step instructions for walking to it. Or, if a user has preferences set in an app, an airline could use those preferences to order a preferred drink or snack while on a flight and have it delivered automatically to the correct seat (Nicol, 2013).

Historical context can be derived from storing user's past actions and preferences and learning from them (Nicol, 2013). It can provide the ability to use previous search and purchase information to enable comparison shopping features and inform future actions. For instance, a movie app might send a notification when tickets go on sale for a new Avengers movie if the user has previously watched other Avengers movies in the theatre on opening weekend. Or, a restaurant app might bring up a history of recently ordered food, allowing easy selection of commonly ordered meals. Historical context can really become powerful when combined with mobile intelligence. Mobile intelligence refers to the ability of a mobile app to connect to database information stored in enterprise or cloud-based systems. While access to this kind of data can be powerful for any software application, when mobile intelligence is combined with environmental context and personal context, it can provide a transformative experience for a user. For example, an application might use business intelligence to search for a flight, searching all airlines for the best price, options, and travel times, and then presenting those options to the user as a comparison. A mobile app using mobile intelligence could do this same thing, but while you are stuck somewhere in an airport after a cancelled flight. In this situation, the app would know your location and final destination, and automatically provide the comparison of best options from your specific location only. Mobile intelligence can thus be used to provide the user with options, but highlighting the next best action (Nicol, 2013).

The combination of the various types of context, environmental, personal, and historical, can provide for unique and innovative ideas for mobile apps. These innovative features have only begun to be discovered.

Simplify Information Delivery

Another hallmark of mobile apps that differentiates them from their traditional software counterparts is the concept of simplicity. Most software applications have a plethora of features and functions. Mobile apps typically have a very simple purpose, focusing on one to three user actions, and handling those actions well. In fact, it is typical for companies to offer a suite of related mobile apps, each with their own distinct purpose (e.g. Walmart online shopping

vs. Grocery pick-up service). This provides for simpler interfaces and functionality, and it also has the added benefit of enabling companies to get mobile apps to market more quickly, because they are smaller and can be developed and tested more quickly (Nicol, 2013). It is important to keep this in mind when creating a mobile app. You should not attempt to replicate a web experience (Babich, 2018) or a software application experience. Rather, know what the purpose of your app is and focus on refining the core experience related to that purpose (Babich, 2018). One best practice in mobile app development is to strive for minimalism by cutting out clutter and prioritizing one primary action per screen (Babich, 2018).

Nicol (2013) also notes that simplified information delivery entails delivering the right information at the right time and place. Interaction should be radically simplified, exposing only the needed interface elements at any given time. A simple, intuitive interface is critical for success. Another facet of simplification relates to how a mobile app works with a company's mobile web. It is common that users will switch back and forth between computers and mobile devices. Take an airline app as an example. A user should be able to search and book a flight on the airline's website on a desktop computer, and then seamlessly transition to the airline's mobile app while on-the-go travelling to and within the airport. The website might have more features and larger graphics/comparisons, while the mobile app should tailor content to the smaller screen and also provide location-specific services such as quickly pulling up a digital boarding pass.

Provide the Simplest Way to Enter Data Correctly

The idea of simplification is one of the overarching themes for mobile app development. The next UX design principle involves providing the simplest way for users to enter data correctly. This involves making user input as easy and as intuitive as possible. First, use context to automate where you can. Apps will often have access to date, time, and location, so make use of these data points to set default values to aid the user. Keep in mind, though, that you should only set defaults; there should always be an option for the user to manually override these values. Consider an app that looks for house rentals. It makes sense to set the app to default to the user's current location, but remember that a user might be searching for a rental in the future (at a different location) and so you should always have an easy option for making changes (Gove, 2016).

Use the appropriate controls to allow for the correct input type and discourage or disallow bad input. This principle involves a solid understanding of the types of input and touch-based controls that are provided by the development environment that you are using. Android and iOS include similar types of controls, but there are some differences. One best practice is to minimize the user's need to type, if possible (Babich, 2018). For example, if you are asking for a person's age, you do not need them to type in a value. Instead, you can provide a drop-down list of appropriate choices. Always make sure that selections are mutually exclusive and that your list of options is exhaustive. Include an "other" choice if you are not certain. Use of an input control with a specified range of values such as a drop-down list, radio buttons, checkboxes, switches, and sliders can virtually ensure that no bad data can enter the app via the input mechanism. This makes the process easy for the user, and also provides benefits to the developer. The developer will not have as many error checking routines to complete on the backend.

If user's do need to type in text or numbers, make sure that you match the appropriate keyboard with the type of text inputs required. For example, if you are expecting all numeric content, choose a keyboard that only allows for numeric data entry. Likewise, for the sake of usability, if you are asking for an email or web address, choose a keyboard that includes helpful buttons for the @ symbol, the . symbol, and .com, etc (Gove, 2016). Again, this type of limitation can also minimize the amount of error checking needed by the developer, so it serves two purposes. You should also provide hints in the form of examples, or simple, on-screen instructions or actions to indicate to the user the requested format for specific data. For instance, if you ask for a phone number, you could code your app to automatically add the dashes at the appropriate spaces in the phone number. This will intuitively answer the user's questions about format as they type. Alternatively, you can provide a hint (often offered in the form of a grayed-out example inside the text input field that automatically disappears when the user begins typing). This can also be an intuitive guide for users. Lastly, if neither option is feasible, provide simple, clear instructions on-screen in the exact location where they will be needed (Gove, 2016). Another nice feature is to provide in-context information that could be helpful (but not necessary) to the data input. For example, when choosing a hotel reservation stay, have the user select check-in and check-out dates via a simple to use date selection control. Then, once dates are selected, automatically show the days of the week just below the selection. For example, "3-night stay from Wednesday to Saturday." This kind of

information is something the user could figure out for themselves, but providing it may help them to quickly notice accidental errors, and makes the entire process more user-friendly.

Manage User Interactions and Navigation

Mobile apps should manage user interactions and navigation so that they are as easy and intuitive to use as possible. Navigation in a mobile app should be self-evident and should feel familiar (Babich, 2018). Most apps use a simple icon with three horizontal lines to indicate a menu. Menu items should be clearly labeled and easy to select. Do not use icons without text labels or descriptions, as this can be confusing to users. There should not be an extensive navigation system if the app provides a simplified purpose with only a few actionable tasks. The navigation should always provide context for the user to show the current location, and provide simple ways to move backward and forward through content. The purpose of navigation is to be functional and allow the user to get to all of the app's functionality; the navigation should not draw focus away from the content (Babich, 2018).

Traditional usability techniques should also be employed to manage and simplify user interactions. Make the registration and login process as simple as possible, providing recall features and forgot password options (Babich, 2018). When using forms, make sure that the on-screen keyboard does not overlap the area for data entry (Babich, 2018), and ensure that all elements can be easily seen and activated. Use finger-friendly hit targets for buttons and other touch controls that are large enough for the average finger to easily tap (Apple, n.d.-b). Ensure that there is enough space between controls so that users do not accidentally tap the wrong control (Babich, 2018).

Employ Information Architecture to Deliver Content Logically

Use principles of information architecture to deliver content in as logical and clear a way as possible. For example, when categorizing content areas of your app, ensure that your categories do not overlap in content, such as offering options for "men's footwear" and "basketball" and "hiking." These categories are confusing because both basketball and hiking may be areas where a user would search for men's footwear. It would be better to organize either by apparel type or by sport in this example (Gove, 2016). Likewise, make sure that you use wording that is clear and will be easily understood by your user. You should avoid jargon (Babich, 2018) and also avoid themed language such as using the terms *roost*, *migrate*, or *fly* instead of *buy*, *rent*, or *sell* (Gove, 2016). You should also place controls and labels in close proximity to the content they modify (Gove, 2016). For example, users should not have to guess whether text belongs with a photo above or below it; the spacing and alignment of elements should clearly show the appropriate information relationships.

Where possible, allow users to customize views and controls. For example, search, filter, and sort capabilities should always be provided and displayed in a prominent location (Gove, 2016). If zoom functionality is needed, for example to zoom in on a photo of a product for sale, always allow the user to control the level of zoom rather than providing screenshots of a distance and close-up option. Users may be interested in a specific area of the photo that they want to see more closely (Gove, 2016).

You should also always show the value of your app up front, and provide clear utility before asking users to register (Gove, 2016). For example, unless your company uses a members-only business model, users should not be required to login to browse products or information, but should only need to login if they wish to purchase or otherwise store selections in their personal account. Similarly, ask for app permissions only if and when they are needed (Gove, 2016). These permissions may include access to the device location, camera, microphone, etc. It makes sense to ask a user for access to the camera before taking a photo within the app. It does not make sense to ask for this permission upon opening of a retail-based app that does not have a clear need for the camera.

Use Feedback to Convey State

Providing feedback is a powerful communication mechanism for a mobile app. Feedback communicates to a user what is happening and can indicate status for actions (Norman, 2013). An example of feedback that communicates form errors in real time is to provide a red outline and text message noting that an email address is invalid just after the user types in an email address into the text box. This kind of feedback is immediate and allows the user the opportunity to correct the error before continuing. A spinning icon or message indicating progress of a task can also

be a feedback indicator that helps a user understand what is happening. A progress bar that shows percentage of a task completed is more user-friendly than a spinning icon. This is because the progress bar ensures the user that something is still happening (vs. wondering if the process is hung and/or how much longer it will take).

Other types of feedback can communicate status of an action (Norman, 2013; Gove, 2016). For example, if you put an item in your shopping cart, you should see a visual indicator of this action. Good examples of this are a message popping up on the screen than an item was added to the cart, a number popping up on the shopping cart icon showing how many items are currently inside, or a sound or vibration activated on the device to indicate that something was accomplished. Another example would be inside a shopping cart view. Here, a user might see a message noting that if they order the product within the next 2 hours, it can be delivered by a specific date. This kind of data provides real-time information to the user that is useful in helping them make a purchase decision.

Leverage Aesthetics for Clarity of Information

Lastly, aesthetics and graphic design principles are also an important part of UX design and should be leveraged to create a pleasing layout and convey clarity of information (Apple, n.d.-a). One best practice for mobile apps is to always create a layout that fits the screen of the device without requiring horizontal scrolling. Apps should also use a text size that is large enough for the average reader (no smaller than 10pt font) (Apple, n.d.-a) and ensure that app is responsive to the user setting their text larger/smaller. This is often controlled in the mobile device OS, but should be respected by the app. Make sure there is ample contrast between text and background so that text is legible, and use whitespace purposefully for ease of viewing/reading. Images should be provided using the appropriate aspect ratio with no distortion, and should be displayed at the correct resolution (Apple, n.d.-a).

SUMMARY

Bias et al. (2012) surveyed 289 graduate students in iSchools who had passed at least one course in usability and user-centered design. The overwhelming majority of respondents, 94%, reported that they used the general principles of usability in their job regularly, though only 20% of respondents stated that usability or user-centered design was part of their job description. This makes a compelling case that user-centered design is appropriate curriculum for students in information systems (IS) fields, and this paper further asserts that UX design principles are appropriate and necessary curriculum for IS students learning mobile app development.

Students learning to develop mobile apps must understand the distinct differences related to user experience (UX) design between mobile apps and traditional software. Mobile apps require awareness of user context, including environmental context (e.g. GPS location data) and data read from other device sensors. These unique data points can and should be utilized to develop apps that leverage user context to provide unique value. Mobile apps should simplify information delivery and focus on one or a few specific user tasks, taking advantage of specific UX design concepts. This paper outlined a framework of UX design concepts for mobile programming courses, including: Delivering value by leveraging context, simplifying information delivery, providing the simplest way to enter data correctly, managing user interactions and navigation, employing information architecture to deliver content logically, using feedback to convey state, and leveraging aesthetics for clarity of information. These concepts are critical to developing quality mobile apps, and this conceptual framework should be considered for inclusion in mobile app development courses.

REFERENCES

Anderson, M. (2019). *Mobile technology and home broadband 2019.* Pew Research Center. https://www.pewresearch.org/internet/2019/06/13/mobile-technology-and-home-broadband-2019/

Anvari, F. Richards, D., Hitchens, M., & Tran, H.M.T. (2019). Teaching user centered conceptual design using cross-cultural personas and peer reviews for a large cohort of students. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET),* 62-73.

Android (n.d.). *Material design guidelines.* Android Developer. https://material.io/design/introduction

Apple. (n.d.-a). *Human interface guidelines.* Apple Developer. https://developer.apple.com/design/human-interface-guidelines/

Apple. (n.d.-b). *UI design dos and don'ts.* Apple Developer. https://developer.apple.com/design/tips/

Babich, N. (2018). *10 do's and don'ts of mobile app design.* Adobe. https://xd.adobe.com/ideas/principles/app-design/10-dos-donts-mobile-app-design/

Bias, R.G., Marty, P.F., & Douglas, I. (2012). Usability/user-centered design in the iSchools: Justifying a teaching philosophy. *Journal of Education for Library and Information Science, 53*(4), 274-289.

Clement, J. (2019). *Mobile app usage – statistics & facts.* Statista. https://www.statista.com/topics/1002/mobile-app-usage/#dossierSummary__chapter2

Dang, D., & Skelton, D. (2019). Teaching mobile app development: Choosing the best development tools in practical labs. *Proceedings of the 10th Annual Conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2019),* 1-6.

El-Tawab, S., Iskandarova, S., Almalag, M., & Ghazizadeh, P. (2018). A methodology of teaching mobile development for undergraduate students in project-based classes. *Society for Information Technology & Teacher Education International Conference, Association for the Advancement of Computing in Education (AACE),* 749-754.

Esakia, A., & McCrickard, D.S. (2016). An adaptable model for teaching mobile app development. *2016 IEEE Frontiers in Education Conference (FIE),* 1-9.

Flora, H.K., Wang, X., & Chande, S.V. (2014). An investigation on the characteristics of mobile applications: A survey study. *International Journal of Information Technology and Computer Science (IJITCS), 6*(11), 21-27.

Gordon, A. (2013). Concepts for mobile programming. *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE),* ACM, 58-63.

Gove, J. (2016). *Principles of mobile app design: Introduction.* Think with Google. https://www.thinkwithgoogle.com/marketing-resources/experience-design/principles-of-mobile-app-design-introduction/

Hardawar, D. (2012). Facebook's Zuckerberg: 'The biggest mistake we've made as a company is betting on HTML5 over native.' https://venturebeat.com/2012/09/11/facebooks-zuckerberg-the-biggest-mistake-weve-made-as-a-company-is-betting-on-html5-over-native/

Holzinger, A., Treitler, P., & Slany, W. (2012). Making apps usable on multiple mobile platforms: On interoperability for business application development on smartphones. In Quirchmayr, G., Basl, J., You, I., Xu, L., & Weippl, E. (Eds.), *Multidisciplinary Research and Practice for Information Systems, 7465*. Springer.

Interaction Design Foundation (n.d.). *Usability.* https://www.interaction-design.org/literature/topics/usability

Joorabchi, M.E., Mesbah, A., & Kruchten, P. (2013). Real challenges in mobile app development. *Proceedings of ESEM'13*, 15-24.

Khmelevsky, Y., & Voytenko, V. (2016). A new paradigm for teaching mobile application development. *Proceedings of the 21st Western Canadian Conference on Computing Education*, 1-6.

Lazar, A., Lazar, J., & Pradhan, A. (2019). Using modules to teach accessibility in a user-centered design course. *21st International ACM SIGACCESS Conference on Computers and Accessibility*, 554-556.

Lyonnais, S. (2017). *Where did the term "user experience" come from?* Adobe Blog. https://theblog.adobe.com/where-did-the-term-user-experience-come-from/

Nicol, D. (2013). *Mobile strategy: How your company can win by embracing mobile technologies.* IBM Press.

Nielsen, J. (1993). *Usability engineering.* Academic Press.

Norman, D. (1988). *The design of everyday things.* Basic Books.

Norman, D. (2003). *Emotional design: Why we love (or hate) everyday things.* Basic Books.

Norman, D. (2013). *The design of everyday things, revised & expanded edition.* Basic Books.

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

Novoseltseva, E. (2020). *User-centered design: An introduction.* UsabilityGeek. https://usabilitygeek.com/user-centered-design-introduction/

Stuurman, S., van Gastel, B.E., & Passier, H.J. (2014). The design of mobile apps: what and how to teach? *Proceedings of the Computer Science Education Research Conference*, 93-100.

Sung, K., & Samuel, A. (2014). Mobile application development classes for the mobile era. *Proceedings of the ACM 2014 Conference on Innovation and Technology in Computer Science Education*, ACM, 141-146.