Issues in Informing Science + Information Technology

An Official Publication of the Informing Science Institute
InformingScience.org

IISIT.org

Volume 14, 2017

HYBRID APP APPROACH: COULD IT MARK THE END OF NATIVE APP DOMINATION?

Minh Huynh*	Southeastern Louisiana University, Hammond, LA USA	Minh.huynh@selu.edu
Prashant Ghimire	Southeastern Louisiana University, Hammond, LA USA	Prashant.ghimire@selu.edu
Donny Truong	George Mason University, Arlington, VA USA	dtruong7@gmu.edu

*Corresponding Author

ABSTRACT

Aim/Purpose	Despite millions of apps on the market, it is still challenging to develop a mobile app that can run across platforms using the same code.
Background	This paper explores a potential solution for developing cross platform apps by presenting the hybrid app approach.
Methodology	The paper first describes a brief evolution of the different mobile app development approaches and then compares them with the hybrid app approach. Next, it focuses on one specific hybrid app development framework called Ionic.
Contribution	The paper presents the hybrid app approach as an emerging trend in mobile app development and concludes with the highlight of its advantages and teaching implications.
Findings	The hybrid app approach reduces the learning curve and offers tools to allow the reuse of code to create apps for different mobile devices.
Recommendations for Practitioners	The experience that the paper describes in using Ionic framework to create a hybrid app can be adopted in a web design or mobile app development course.
Impact on Society	The advance in hybrid framework in general and the growing acceptance of open source framework, such as Ionic in particular, may provide an alternative to the native app domination and may trigger the rapid rise of hybrid apps in the years to come.
Keywords	hybrid apps, mobile app development, cross-platform web app, Ionic framework

Accepting Editor: Eli Cohen | Received: December 21, 2016 | Revised: March 9, March 24, April 20, 2017 | Accepted: April 21, 2017.

Cite as: Huynh, M., Ghimire, P., & Truong, D. (2017). Hybrid app approach: Could it mark the end of native app domination? *Issues in Informing Science and Information Technology Education, 14,* 49-65. Retrieved from http://www.informingscience.org/Publications/3723

INTRODUCTION

According to recent papers published by Huynh and Ghimire (2015; 2017) on the development of mobile apps, there is not yet a perfect approach for creating mobile apps. Despite millions of apps on the market, it is still challenging to design and develop a mobile app that can run across platforms using the same code. Only a few years ago, using native software development kit (SDK or devkit) was the only way to create a mobile app. Each platform, such as Android, iOS, and Windows, requires different SDK and each platform also uses different programming languages for coding. The idea of developing an app without coding with Java or Objective C seemed quite remote. Recently, the landscape in the development of mobile apps has changed significantly in particular with the advent of hybrid mobile app framework (Arora, 2016). In this paper, we are going to explore one emerging approach that potentially offers the solution to this nagging problem in the cross-platform mobile app development.

To understand the significance of the hybrid approach, we first step back and look at the emergence of different approaches over the years. This section provides a glimpse into the evolution of different development approaches ranging from the "separate-sites" web design strategy to the explosion of the native app development. It is then followed by the introduction of the hybrid mobile app approach. In that section, we will describe the background in the hybrid mobile app approach and its advantages in comparison to other approaches. Next, we will focus on Ionic being regarded as one of the top frameworks for hybrid mobile app development. We will also briefly describe how Ionic could be set up and used. The final part of the paper will offer our reflection on the Ionic framework and discuss our perspectives on the potential of the hybrid app approach.

A GLIMPSE INTO THE EVOLUTION OF DIFFERENT APPROACHES FOR DELIVERING WEB CONTENT ON MOBILE DEVICES

Based on a recent study by Pew Research Center on Smartphone (Smith, 2015), nearly two-thirds of Americans are now smartphone owners. Among the smartphone owners, Millennials (ages 18-34) are by far the most technologically advanced user group, with a 95% cell phone ownership. The majority of the cell phones today are smartphones. Smartphones are widely used for navigating numerous important life activities ranging from searching for health information, using educational resources, finding jobs, reading news, sharing contents with others, following the driving direction, performing tasks, to staying connected to the world, and many other usages. For many users, these devices are a key entry point to the online world (Smith, 2015). One of the main driving forces that make people more and more "smartphone-dependent" is the rich resources available from the web.

Initially, as the number of smartphone and tablet users began to grow, web designers often followed the 'separate-sites' web design strategy. The goal was to create a website that allowed users access to the web content from their mobile devices. Normally, this version of mobile website was a separate version from a desktop website. The key advantage of this approach was the ability to deliver the optimal user experience (UX) to both mobile and desktop website users because each website was designed specifically to the respective device type. In other words, when mobile users came to the website, they would be redirected to the mobile web version and the display would be mobile friendly. For instance, there would be fewer images on a mobile web version than on the desktop web version. The text on the mobile web version would be displayed at proper size without the need to zoom in and out. The navigation would be simpler on a mobile web version as shown in Figure 1.

However, this approach required the design, development, and maintenance of two separate websites. Thus, one consequence was the increase in the cost as well as in the amount of effort to develop as well as maintain two websites. Another disadvantage in this approach was the difficulty in detecting a mobile device given plethora of different devices on the market. A mobile website usually relied on the ability of a web server to detect a mobile browser via browser sniffing technology such as embedded JavaScript code. However, this code had to be updated often in order to detect new

devices; otherwise, it would not be able to detect new mobile devices on the market. Because of this disadvantage, the 'separate-sites' web approach has gradually evolved into the present responsive web design (RWD) approach.

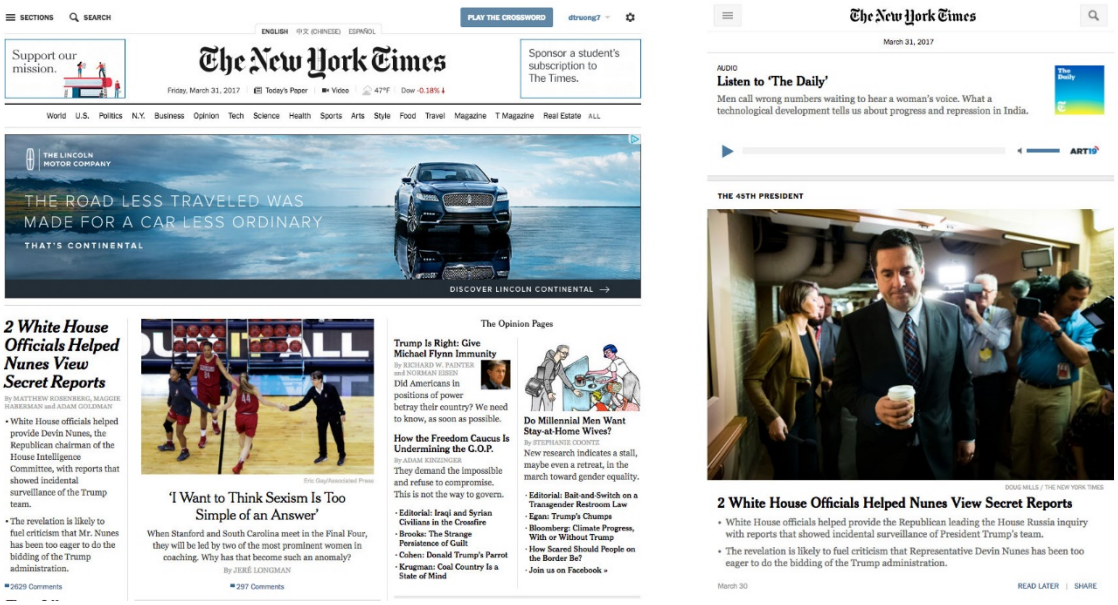


Figure 1. "Separate-sites" web design delivers different versions of the website according the type of devices

(Adopted from source: https://www.nytimes.com/and https://mobile.nytimes.com/)

In the RWD approach, the goal is to optimize the access and display of the web-site content for multiple device types including smartphones, tablets, laptops, and desktops. The key is to enable the website to be responsive to different screen sizes without creating multiple versions of the websites. With the advent of HTML5 and CSS3, responsive web design technique is now less complex because it does not require scripting languages such as JavaScript, PHP, etc. The tags provided by HTML5 and CSS3 are sufficient to deliver the capability to resize, hide, shrink, enlarge, or move the content to make it responsive to any screen size as shown in Figure 2.

The main appeal of the RWD approach is its simplicity in coding as well as in maintaining the website. There is only one version to develop and maintain. Thus, this translates into cost saving in the development and marketing of the website. However, the RWD approach exhibits inherent limitations of a typical website. Although a RWD website is responsive to different devices, it is essentially just a website to display information. Its working environment depends on a web server and a client's browser. Therefore, more advanced UX for different screen sizes and device types are not possible at this point in time. Furthermore, adding more functionalities to the RWD website is often not feasible because of the limited capabilities in its development framework.

As the cost of owning mobile devices decreases and the mobile technology continues to advance, more and more users replace their desktop computers and even laptops with mobile and tablet devices. Instead of relying on desktop computers to access the web, there is a growing demand for accessing web content and doing work with mobile devices. This is where applications designed specifically for mobile devices (commonly known as apps) come into play. Most dominant are the "native apps". Indeed, the growth of native apps is phenomenal. According to one estimate, there were more than two million mobile apps on the market just a few years ago. Recently from the count at Google Play Store and Apple App store alone, users have downloaded all the apps billions of times per year (Malavolta, Ruberto, Soru, & Terragni, 2015). According to the 2014 U.S. Mobile App

Report (Lella & Lipsman, 2014), the mobile apps usage alone makes up a majority of total digital media engagement at 52%. Native app has become the dominant strategy for business to provide its users connection not only to access their web content, to communicate and interact with the business, to conduct work, but also to build its brand and manage its customer relationship. Native apps refer to apps that are written using tools and an application program interface (API) provided by the hardware/device providers such as Google and Apple and are specifically tied to a particular platform such as Google Android, Apple iOS, Microsoft Windows (Lutes, 2012).

Figure 2. Responsive web design delivers different displays based on the type of screen size
(Adopted from https://www.law.gmu.edu/)

Although the native app development approach is currently dominating, it is not perfect for the following reasons. First of all, each native app SDK is tied to a specific platform. For instance, a SDK for iOS is totally different from that of Android. Therefore, it is technically challenging to develop a native app that can run on cross platforms including Android, iOS, and Windows devices. What works on one device does not necessarily work on other devices (Huynh & Ghimire 2017). Another reason is that there are lower-level APIs that allow native code to access device storage, sensors, and data (Charland & Leroux, 2011) and all these APIs are different from one device to another. From the resource perspective, it is economically demanding, especially for smaller shop of developers, to be successful in the native app development. That is, despite so many apps available, only a small percentage of apps are profitable. Furthermore, the native app market has been dominated mostly by large and established companies. Consequently, apps from smaller companies and/or from individual would have a hard time to compete and succeed (Brodsky, 2015). The last reason is the proprietary nature inherent in different hardware platforms. Android, iOS and Windows apps are all designed based on proprietary hardware platforms. Any changes in OS or hardware render the apps inoperable or incompatible (Huynh & Ghimire, 2017).

To provide an alternative to the dominant native app development approach, there are other approaches. One of them is the browser app approach as discussed in the article by Huynh and Ghimire (2017). The proposed browser app approach is a response to the limitations of the 'separate-sites' web design strategy and the RWD technique. In their article, Huynh and Ghimire (2017)

describe the key features in the browser app approach including: deliverable apps with capabilities to run from any mobile browser, to support different devices and not platform dependent, with development based on open-source standards, and with emphasis on code reusability by taking advantage of "write-once-and-run-anywhere" (WORA) concept. The potential of the proposed browser app approach is demonstrated in the simple process of using the existing code to create a series of Store-Retrieve-Display (SRD) apps without a steep learning curve. However, the browser app approach, as pointed out by Huynh and Ghimire (2017), has a few shortcomings. For instance, from the technical perspective, though these browser apps could work on any mobile devices, they are essentially not yet true mobile apps. They are not downloadable because browser apps are accessed from the Web and run mainly from the browser. Hence, they cannot function without a connection to the internet. At the end of their article, Huynh and Ghimire (2017) pointed to a promising strategy called 'hybrid mobile app'. This 'hybrid mobile app' approach will be the focus of what we are going to present in the remaining sections of the paper.

INTRODUCTION OF THE HYBRID MOBILE APP APPROACH

In the previous section, we described a brief evolution of different strategies for delivering web content on mobile devices. As smartphones proliferate, there are more users and their demands for accessing web content are also growing. As technology advances, more capabilities have been added. At the present, the native app strategy is a dominant choice. However, it is far from a perfect solution. There are many challenges to developing a native app to run on cross platforms. In this section, we are going to take a closer look at one of the alternatives called the hybrid mobile app approach.

UNDERSTANDING THE HYBRID MOBILE APP APPROACH

In recent years, the hybrid mobile app approach has grown to be an emerging and promising alternative to the native app development approach. This approach allows developers to create a single mobile app, called a hybrid app, using web standards and to consistently distribute it across multiple mobile platforms with (minimal to) no changes (Malavolta et al., 2015). A hybrid app is essentially a small website written with HTML, CSS, and JavaScript. It is different from normal websites in that it runs only in a browser shell and has access to the native platform layer. To run like a native app, it relies on a native wrapper like Cordova. Therefore, hybrid app is designed and coded like a website, but it is then wrapped with the capabilities to access native platform functionalities. The result is a powerful alternative approach to build mobile apps that are based on web technology but function and look like a native app. Therefore, the hybrid mobile app approach offers a mechanism to turn browser-based apps into mobile apps that could be compiled into binary executable files, could be downloaded from App stores onto users' devices, and could be run both online and offline.

COMPARISON BETWEEN THE HYBRID MOBILE APP APPROACH AND OTHER OPTIONS

As more users use mobile devices, the demand for applications is not only on delivering web content but also on performing more advanced functions. This leads to the emergence of mobile apps. Based on a discussion with industry professionals, Agrawal and Gill (2013) suggested that there are basically three major options available for the development of mobile apps. The first option is to use platform-specific developer tools to develop platform-specific applications. The second option is to create browser apps using HTML/JQuery and CSS (Agrawal & Gill, 2013). The last option is to use RAD tools to create hybrid apps with the characteristics of a web app and a native app.

With the first option, using platform-specific developer tools, the result is the delivery of native apps. Native apps are typically coded in compiled programming languages such as Java, Objective C, or C#. Native apps are often robust because they have access to device specific features (Lutes, 2012). All native apps are obtained via app stores such as Apple App store, Google Play store. A native app

is installed directly onto a device itself, is available locally, and hence can run without a connection to the network.

With the second option using HTML/JQuery and CSS, the result is the delivery of mobile web-browser app. Browser apps function essentially like websites designed with a user interface that is optimized for mobile devices. It is not device-dependent because it is typically coded using standards-based web browser technologies such as HTML, CSS, JavaScript, PHP (Lutes, 2012) rather than compiled program languages. Although it works like a native app, a browser app does not need to be downloaded or installed on a device. The web browser on the mobile device invokes it through a designated URL. Hence, internet connection is required to run it.

The last option is the development of hybrid apps. This kind of app draws upon the characteristics of both a native app and a mobile web app. Like a mobile web app, it is written using HTML, CSS, JavaScript, or PHP, but it is compiled into a native app for distribution via the vendor app stores. From a user's point of view, there is little difference between a native app and a hybrid app. As long as the device has a web browser, a hybrid app can run just like a native app. From a developer's point of view, a hybrid app is truly cross-platform.

Advantages of the Hybrid Mobile App Approach

There are many advantages of using the hybrid mobile app approach. Hybrid mobile apps allow developers to use standardized web technologies such as HTML5, CSS3, and JavaScript. In hybrid mobile apps, the developers also include the capability to deliver all service requests to the Platform API. This part requires the use of a hybrid development framework (e.g. Apache Cordova, ionic framework, etc.) to provide a native wrapper for containing the web-based code, and a generic JavaScript API to bridge all the service requests from the web-based code to the corresponding platform API. It is the native wrapper that enables hybrid mobile apps to be packaged, deployed, and distributed across platform (Irvine & Maddocks, 2013; Malavolta et al., 2015; Wargo, 2012).

Therefore, the hybrid app approach is not practical without the existence of the frameworks. Among the most important frameworks are those that embrace HTML5, CSS, and JavaScript because they allow developers to create cross platform hybrid mobile apps with native look and feel but do not require the use of Java and Objective C, or C#. This is the key factor to make the hybrid mobile app approach the fastest and easiest way to start building cross platform mobile apps using just the common web languages such as HTML5, CSS, and JavaScript.

In addition to lower learning curve, these frameworks also enable the developers to maximize the impact of their coding effort by applying the concept of "Write once, run anywhere" (WORA) (Huynh & Ghimire, 2017). In simple terms, the developers could develop the app as a regular browser app with responsive features provided by the hybrid mobile app frameworks. The access to native hardware components like Camera or notification feature or local storage is accomplished by native JavaScript APIs. The app could then be compiled into platform specific native packages and is ready for use on a mobile device or for distribution on app stores. This is perhaps the strongest reason for the rapid growth in hybrid mobile apps in comparison to that in native apps.

The emergency of solid HTML5 mobile UI/UX frameworks is another driver for such a rapid adoption of the hybrid app approach. Among the top 10 HTML5 mobile UI/UX hybrid mobile app frameworks in the market today are Ionic, Onsen UI, Intel XDK, Sencha Touch, Kendo UI, Framework 7, JQuery Mobile suite, Mobile Angular UI, Famo.us, and Monaca (Arora, 2016). Since reviewing these frameworks is beyond the scope of this paper, we would like to focus our discussion mainly on the Ionic framework. The reason is because the Ionic framework is the most developed framework and we are familiar with it in the development of our hybrid mobile app for demonstration (Arora, 2016; Markov, 2016).

INTRODUCTION TO THE IONIC FRAMEWORK

Only a few years ago, attempting to develop mobile apps using JavaScript was regarding as nothing more than a quirky effort because doing that was like trying to build an iOS and Android app without coding with Java and Objective C (Markov, 2016). Yet these days, what was once viewed as a quirky experiment is increasingly gaining acceptance as an alternative approach to the native app development approach. This significant shift has been driven by the emergence of the hybrid mobile app frameworks. With the advent of these frameworks, it is closer than ever for developers and even non-tech savvy users to create a native-like app without coding with Java and Objective C but with using only web-oriented tools and SDK.

In this section, we will focus our discussion on one of the popular hybrid app development frameworks called Ionic. The description of Ionic will provide a good understanding of what a hybrid app development framework is and what features/capabilities it offers. We will then describe briefly how to obtain Ionic and set it up. More technical details are included in the Appendix.

THE TOP CHOICE AMONG THE HYBRID MOBILE APP FRAMEWORKS: IONIC

As described in Arora's (2016) article, Ionic has established itself as a leader in the hybrid mobile apps development space. It is the most popular and the top pick by many developers. Ionic is a HTML5 mobile app development framework designed for building hybrid apps. Basically, we can divide the hybrid app development into three components: (1) the website built with HTML, (2) the native style mobile UI built provided by framework such as Ionic, and (3) the native wrapper platform like Cordova.

The fundamental premise in the hybrid mobile app framework is that almost everyone was, is, and will be spending his or her computing time in the browser. Browser exists not only on desktop, laptop, but also increasingly on smartphone and tablet. Ionic is what makes an app to run on cross-platforms possible. It focuses on native or hybrid apps. These apps go deeper than the web application running on the browser, because they deal with lower level browser shell (like WebView) and have interface with native platform layer in Cordova. Ionic is open source. The clear benefits of hybrid apps over native apps are the following: supporting cross-platform, writing code once and running them anywhere, and accessing to the 3rd party code.

WHAT IS IONIC?

Ionic is an HTML5 mobile app development framework targeted at building hybrid mobile apps ("Ionic Book," n.d.). Basically, Ionic provides the front-end UI framework that handles all the look and feel and enables UI interactions of a hybrid app. In addition, it supports a wide range of common native mobile components, slick animations, and beautiful designs ("Ionic Book," n.d.).

In a typical HTML5 development framework, there are inherent limitations. For instance, a push notification for a website cannot be implemented on any iPhone or Android device. With Ionic, capabilities of a typical mobile app can be integrated/embedded to overcome the limitations in the existing HTML5 development framework. Thus, Ionic framework can deliver an HTML5 with native-styled mobile UI elements and layouts as well as built-in mobile application capabilities with access to the native device functionality. However, in order for its code to run like a native app, Ionic needs a native wrapper such as Cordova. The main function of Cordova is to allow software programmers to build applications for mobile devices using CSS3, HTML5, and JavaScript instead of relying on platform-specific APIs like those in Android, iOS, or Windows Phone ("Apache Cordova," n.d.). In other words, Cordova acts as a container for running a web application written in HTML, CSS, and JavaScript. Typically, Web applications cannot use the native device functionality like Camera, GPS, Accelerometer, Contacts, etc. With Cordova, developers can access the native device functionality and can package the web application in the device installer format (Virinchy, 2014).

Using Ionic to Create Our First App

As we understand what Ionic is and what capabilities it offers, the next part is for us to implement Ionic and to start creating an Ionic hybrid app. To build a hybrid app with Ionic is a straightforward process. At its core, it is just a web page running a native app shell. As a web page, our app is supported by any kind of HTML, CSS, and Javascript. The key difference is instead of building a website for others to link to, Ionic allows us to build a self-contained application with mobile app-like experience. Beyond the web technology, we can also make use of Cordova plugins or native code and in particular the use of Angular JS. It is the Angular JS that allows developers to deliver many mobile app functionalities in an Ionic app ("Ionic Book," n.d.).

For brevity, we just summarize the whole process of downloading the Ionic framework, installing it, setting up its required environment, and starting to use it to develop the first app in Figure 3. Figure 3 depicts a simple flow diagram of the overall steps that we followed in creating our first Ionic app. For those who are interested in the technical details of this implementation, we include the Appendix to illustrate and explain a step-by-step process in using Ionic to create a simple hybrid app. The Appendix can serve as a useful tutorial or a jumpstart for experimenting with Ionic framework.

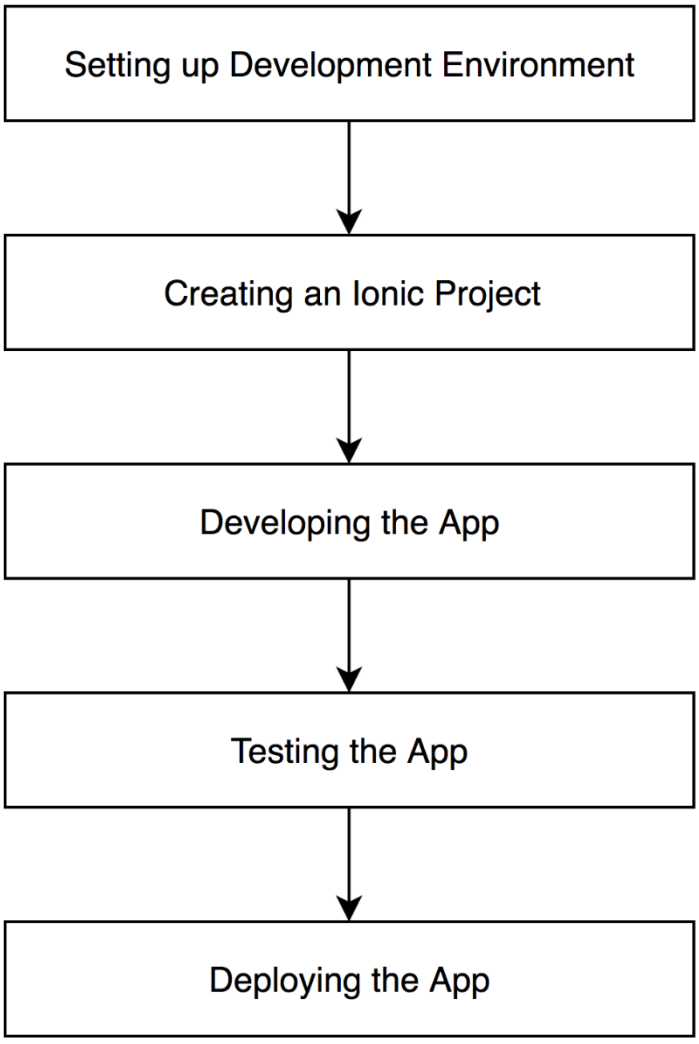


Figure 3. Flow Diagram to illustrate the steps in creating our first Ionic App

CONCLUSION

REFLECTION ON THE IONIC FRAMEWORK

In this paper, we look specifically at Ionic framework as one of the most popular mobile hybrid app development frameworks. As described in Arora's (2016) article, Ionic has established itself as leader in the hybrid mobile apps development space. Ionic is among a handful of open source SDKs that allow an alternative to the native app approach. Its closest competitors sell commercially their framework while Ionic is free to use and is solidly committed to open source technology. It is easy and fast to use Ionic framework to build mobile apps as demonstrated in the Appendix. To get started, it does not take as much effort as it would have taken with the native app approach. Recently, with the advent of Ionic Creator, a simple drag and drop interface building tools was offered to create a simple app with a click of a mouse. To stay ahead of its competitors, the Ionic team continuously keeps the framework updated by adapting to the latest trends. More importantly, its community has expanded so rapidly and the result is the available of valuable and useful development resources for a wide range of projects.

Ionic framework is maintainable and scalable. It uses clean and easy to read markup. It comes packed with highly mobile-optimized library of CSS (Powered by Sass), HTML and JavaScript components. It also features tools and gestures to ensure interactive apps development with ease (Markov, 2016). At the core of the Ionic framework is Angular JS. This is key to the hidden capability of Ionic framework. Like most of the other hybrid app frame-works, Ionic also utilizes Cordova to go native for iOS, Android, windows phone and other platforms.

In sum, Ionic is the most popular framework among various hybrid frameworks on the market and probably the first choice for many developers. Developers can choose to use the CSS portion of the framework to create native looking designs, but to harness the full potential of Ionic, it is best that they pair it with Angular JS. For the tech-savvy developers, a great tool in Ionic is the command-line interface which is full of awesome features including integrated emulators and a Cordova based app packager.

As described in this paper, hybrid app development frameworks can be an alternative to the native approach and this is a remarkable turning point. With the success of hybrid frameworks such as Ionic, it may mark the end of native app domination and may trigger a rapid rise of hybrid apps. Because of its ease of use and its web-based advantages to work on cross platform, hybrid app development frameworks would likely pave its way to become a mainstream approach. More people would use it to create hybrid apps. With a global adoption of hybrid apps, it is likely to see a disruption in the mobile app development with the shifting from native to hybrid approach. Hybrid apps would be increasingly more powerful as web technology continues to advance. With more capabilities, hybrid apps would work just like the native apps and even better in some cases. As mobile devices have now become mobile work, learning and entertainment centers, more apps for cross platform will be in demand. Hybrid mobile app framework allows anyone including non-technical students to learn, acquire, and develop necessary skills to meet the projected demand for the hybrid apps. For those with entrepreneurial spirit, the hybrid mobile app framework such as Ionic makes it viable for even non-technical developers to contribute and participate in the building of more innovative mobile apps.

TEACHING IMPLICATIONS

According to Rajput (2016), the mobile app market was soaring with 58% growth in 2015 in app usage, and the mobile users spent 17 percent more time with their devices. At any time, a new mobile app is being launched with new features. It is important for developers to deliver apps that work on the maximum number of devices. However, as mentioned earlier, it is not economical to make a different app for each platform used by various mobile devices. Based on a recent research, the finding

shows that 60 percent of the enterprises are already using cross platform app technology for their business. It is estimated that cross platform apps would increase with a compound annual growth rate of about 38 percent thereby reaching $4.8 billion by 2017 (Rajput, 2016). This shows a tremendous potential in the hybrid mobile app development framework and a bright future for cross platform hybrid apps.

Over the past couple of years, there has been much progress in the app development platform and mobile app tools market. A few years ago, it would take a team of highly-skilled mobile developers and designers to create an app and quite often the app would be built from scratch in a sophisticated integrated development environment. However, at the present time, non-technical developers can access numerous of types of tools that enable them to create and launch apps faster than ever. Ionic is the case in point. Most importantly, the hybrid mobile app development framework can now deliver cross-platform apps that will enable developers to reach more users of any web-based mobile devices.

While the advent of the hybrid mobile app framework opens a great opportunity for developers to design and build cross-platform apps, it also offers access for IT instructors to explore and teach advanced web design and hybrid mobile application development to non-technical students. In the past, the native app approach requires steep learning curves, demands high-level of technical skills in coding and highly technical knowledge in SDKs. As a result, most courses involved mobile app development have been offered in the computer science and engineering departments. However, the hybrid app approach reduces the learning curve and offers tools to bypass the need for coding and the use of SDKs. Therefore, it is possible to teach mobile app development using the hybrid app approach to non-technical students. The advance in hybrid framework in general and the growing acceptance of open source framework such as Ionic in particular helps open more doors for mobile app development to non-technical students.

REFERENCES

Agrawal, M., & Gill, G. (2013). Mobile application development strategy at Vology. *Journal of Information Technology Education: Discussion Cases*, 2, Case Number 13. Retrieved February 19, 2015 from https://www.informingscience.org/Publications/1918

Apache Cordova. (n.d.). In *Wikipedia*. Retrieved March 3, 2017 from https://en.wikipedia.org/wiki/Apache_Cordova

Arora, S. (2016, Mar 23). *10 best hybrid mobile app UI frameworks: HTML5, CSS and JS*. Retrieved November 4, 2016 from http://noeticforce.com/best-hybrid-mobile-app-ui-frameworks-html5-js-css

Brodsky, I. (2015, December 21). Deathmatch: The mobile Web vs. mobile apps. *Computerworld*. Retrieved September 18, 2016 from http://www.computerworld.com/article/3016736/mobile-wireless/the-mobile-web-vs-mobile-app-death-match.html

Build amazing mobile apps, faster. (n.d.). Retrieved December 12, 2016 from http://ionic.io/products/creator

Charland, A., & Leroux, B. (2011). Mobile application development: Web vs. native. *Communications of the ACM*, *54*(5), 49-53.

Get started with Ionic Framework. (n.d.). Retrieved December 12, 2016 from http://ionicframework.com/getting-started/

Huynh, M. & Ghimire, P. (2015). Learning by doing: How to develop a cross-platform web app. *Journal of Information Technology Education: Innovations in Practice*, *14*, 145-169. Retrieved February 19, 2015 from https://www.informingscience.org/Publications/2252

Huynh, M., & Ghimire. P. (2017). Browser app approach: Can it be an answer to the challenges in cross-platform app development? *Journal of Information Technology Education: Innovations in Practice*, *16*, 47-68. Retrieved from http://www.informingscience.org/Publications/3667

Ionic Book. (2017). Retrieved March 3, 2017 from http://ionicframework.com/docs/guide/

Irvine, M., & Maddocks, J. (2013). *Enabling mobile apps with IBM worklight application center*. IBM Redbooks.

Lella, A., & Lipsman, A. (2014). The U.S. mobile app report, 2014. *comsCore* [white paper]. Retrieved November 18, 2015 from https://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report

Lutes, K. (2012). Cross-platform mobile app software development in the curriculum. *Issues in Informing Science and Information Technology*, 9, 115-124. Retrieved February 19, 2015 from http://iisit.org/Vol9/IISITv9p115-124Lutes120.pdf

Malavolta, I., Ruberto, S., Soru, T., & Terragni, V. (2015). End users' perception of hybrid mobile apps in the Google Play Store. *Proceedings of the 4th International Conference on Mobile Services : IEEE*. Retrieved November 18, 2015 from http://www.ivanomalavolta.com/files/papers/MS_2015.pdf

Markov, D. (2016, Oct 13). Comparing the top frameworks for building hybrid mobile apps. *Tutorialzine*. Retrieved November 4, 2016 from http://tutorialzine.com/2015/10/comparing-the-top-frameworks-for-building-hybrid-mobile-apps/

Rajput, M. (2016, May 11). *Device agnostic: Top 5 cross platform app development tools in 2016*. Retrieved December 12, 2016 from http://www.business.com/mobile-apps-and-tools/top-5-cross-platform-app-development-tools-in-2016/

Smith, A. (2015, April 1). *U.S. smartphone use in 2015. PewResearch Center Report on Internet, Science & Tech*. Retrieved October 1, 2015 from http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/

Virinchy P. (2014, July 27). *What is Cordova and how does it work?* Retrieved March 4, 2017 from https://blogs.sap.com/2014/07/27/what-is-cordova-and-how-does-it-work/

Wargo, J. (2012) *PhoneGap essentials: Building cross-platform mobile apps*. Addison-Wesley.

APPENDIX

In this Appendix, we illustrate a step-by-step process in using Ionic to create a hybrid app. The details include the installation of Ionic, the description of required components, the process of running the Ionic project as described from "Get started with Ionic Framework" (n.d.).

INSTALLATION

NodeJS

First off, we need to make sure that NodeJS is installed. Ionic app requires ionic command line interface (CLI) to create and run ionic projects. Ionic CLI comes as a NodeJS package. Therefore, we need to install NodeJS prior to installing anything else mentioned below. Latest version of NodeJS for Windows, macOS or Linux can be downloaded from its official website https://nodejs.org

Ionic and Cordova

Once NodeJS is installed, we are ready to install ionic. Since Ionic is based on Cordova, Cordova also needs to be installed along with ionic. In fact, Cordova utilities provide the key support to build, deploy and test the apps.

Cordova can be installed with a simple command
$ npm install –g cordova

Ionic can be installed using the following command:

$ npm install –g ionic@1.x

Since the new release of Ionic 2.0, just executing without any version specification $ npm install -g ionic will install ionic CLI version 2 by default. Using Ionic CLI 2 will create Ionic 2 projects instead. To install the earlier version such as Ionic 1 CLI, we would need to specify the version 1. For instance, the command would look like the following: $ npm install –g ionic@1.x.
This would install the latest 1.x version of the ionic CLI. The –g flag represents that the NodeJS packages, Ionic and Cordova in our case, are installed globally and can be accessed from anywhere in the computer. This allows us to create, execute and run an ionic project in any location in our development machine.

To make sure that ionic and cordova are installed, we would run $ ionic –v and $ cordova –v on the command line. This should show the version of ionic and cordova if they were installed correctly.

Starting a Project

Once we are done with installing NodeJS followed by Ionic and Cordova, we are ready to start building apps with Ionic.

Ionic app can be created using Ionic CLI. The following command will create a new Ionic project with tabs template:

$ ionic start mySimpleApp tabs

Ionic comes loaded with starter templates to speed up the development process. They include template tabs, sidemenu, google maps, etc. For the purpose of our demo, we will choose to install a tabs template.

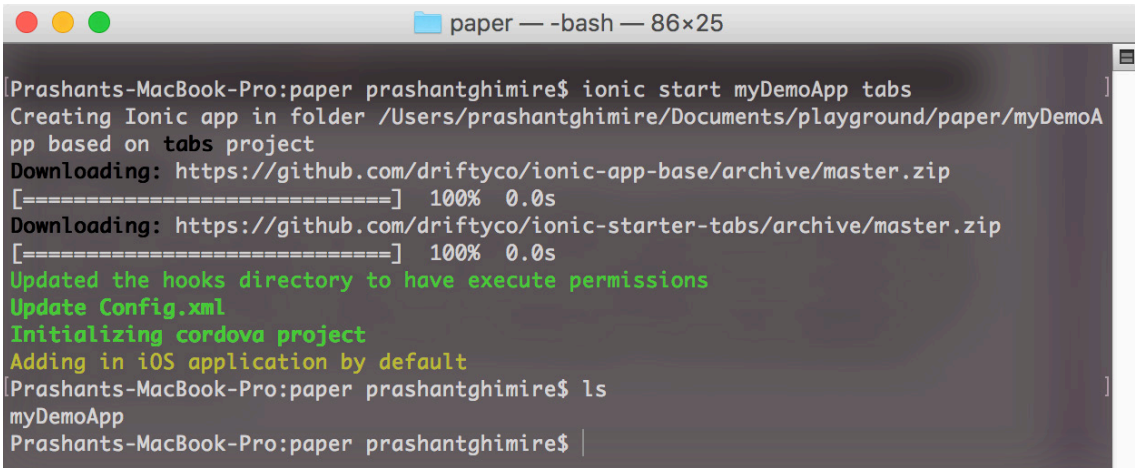


Figure A1: Creating a tabs Ionic Project

Executing the above command would create a folder with the provided project name, i.e. myDemoApp. Upon completion, we could explore the directory and we would see the files and folders as below:

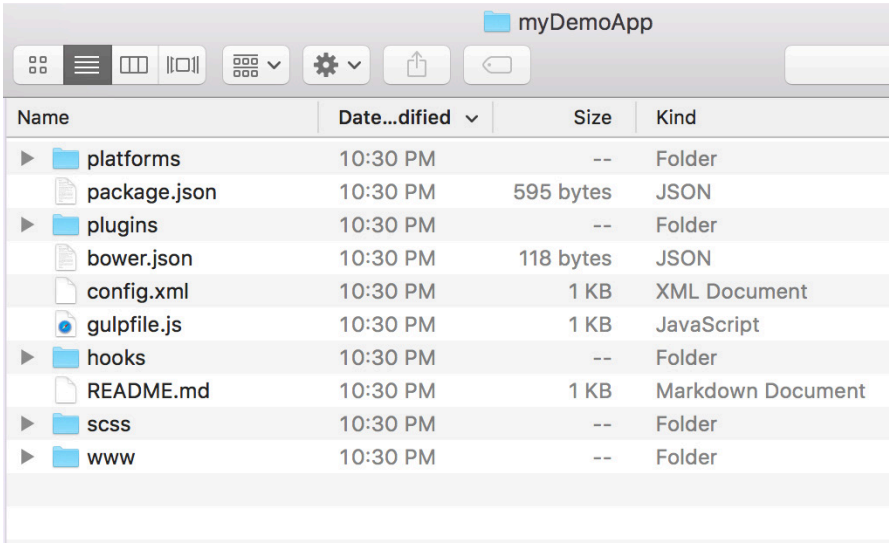


Figure A2: Ionic project files and folders

The project directory contains many directories and configuration files. They are there to define the project dependencies to external CSS/JavaScript libraries and to build tools.

The platform directory contains projects for each platform we choose to build, like Android or iOS. These projects are valid Native SDK, for instance a valid iOS project for iOS platform. They can be opened using XCode and run on a simulator.

The plugins directory contains Cordova plugins. Cordova plugins are modules that provide JavaScript interface for accessing the native device APIs. This is one of the advantages for using Cordova. Native API access could include access of Camera, File Storage, Geo Location, Bluetooth connectivity, etc. There are plenty of Cordova plugins available online for free by active Cordova community.

The main content of the application resides under www directory. This is the directory in which most of our custom code is going to reside. In fact, this is the directory that gets wrapped in the WebView by Cordova when the app is packaged for a specific platform. Within the www directory, index.html is the file that gets loaded on the app start up.

The contents in index.html could be modified and used according to our need. Basically, sub views are injected onto the index.html file by AngularJS. An ionic project is basically a valid AngularJS project.

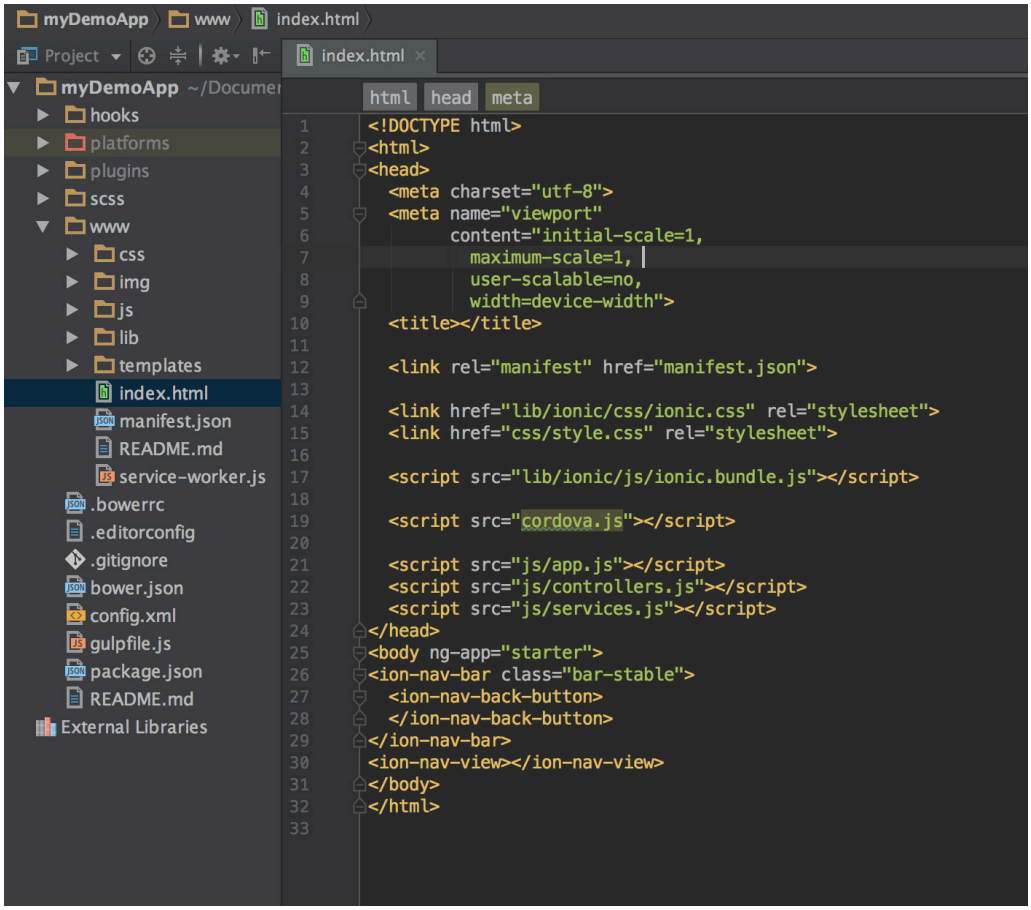


Figure A3: Ionic project index.html file

RUNNING THE PROJECT

On the browser

At this point, we have created a sample ionic project and we are ready to run the app. There are many ways to run the app. In our case, we want to run the app on the browser. This is one of very useful options in Ionic because we could instantly see how the app is going to look like on the phone without having to install it on the device. This can be achieved by issuing the following command:

$ ionic serve

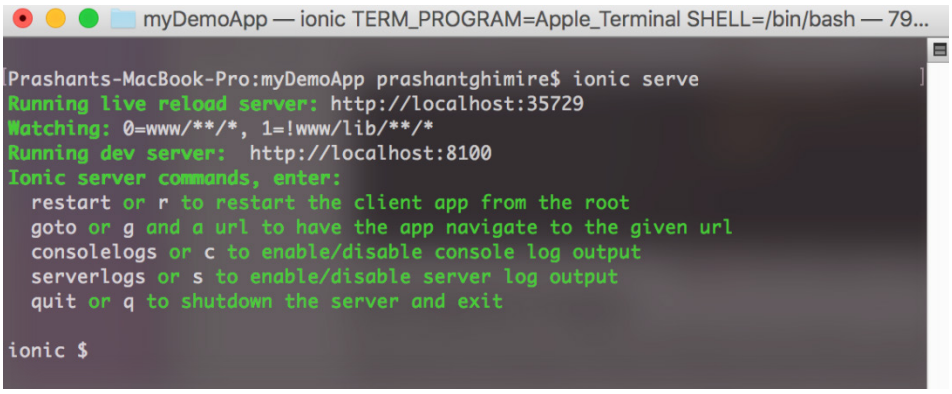


Figure A4: Running Ionic app on the browser

The above command will open the app on the browser.

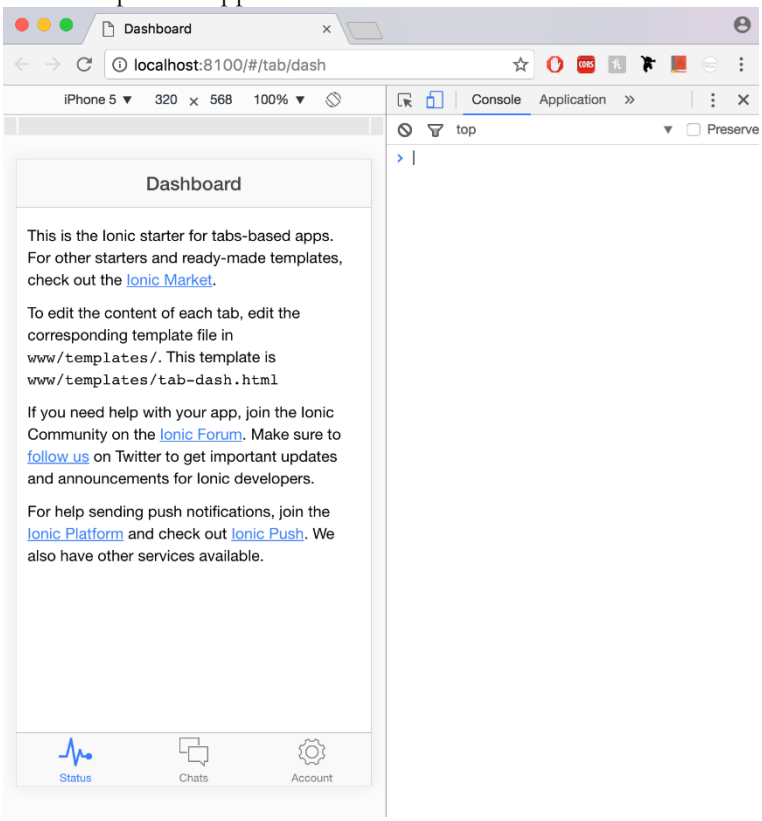


Figure A5: Ionic app running live on the browser

Ionic is integrated with Gulp out of the box. Gulp is a built-in tool that automates build processes such as JS and Sass compilation. Ionic could reload the app to reflect the changes on HTML, CSS and JS project files. Any errors logged could be seen on the browser. This saves development time drastically and makes the development process simpler.

On the simulator

This is another way to run the app. Sometime, we might want to run the app on the device during the development phase. For instance, one of the reasons for running the app on the device is to be able to test Cordova plugin. Since Cordova plugins access native APIs which browser could not pro-

vide, they could only be tested using the actual hardware or simulator. Using simulator saves development time and requirement of actual physical hardware. Ionic app could be run on the simulator using the following command in the format of ionic emulate platform, where platform is the targeted platform to run on. In our case, we want to deploy on iOS so we would issue the following command:

$ ionic emulate ios

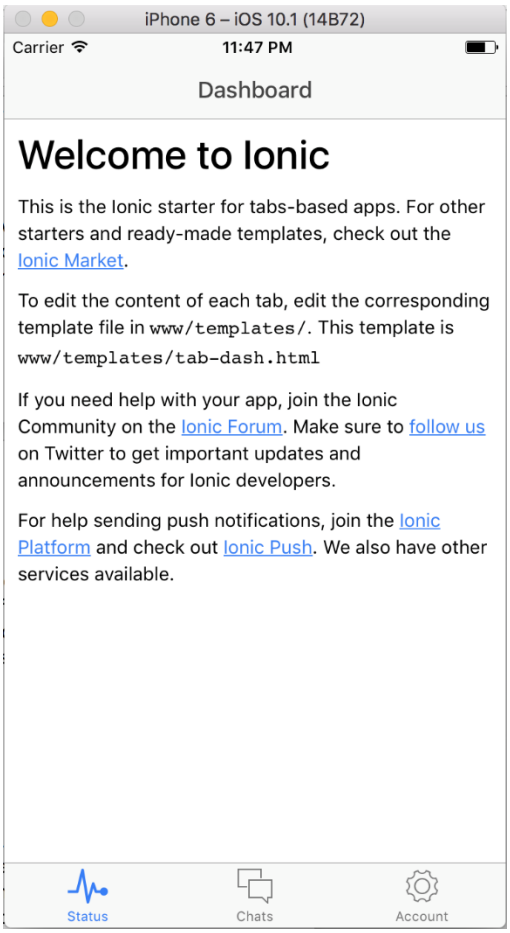


Figure A6: Ionic app run on iOS simulator

In order to run the app on the device itself, we will issue command: $ ionic run [platform]. Similar to running on simulator, platform argument for the command would be ios, android, etc.

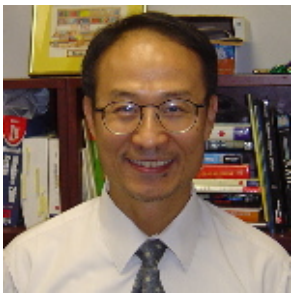
As demonstrated, we could create an ionic application using simple steps above. If encountering any problem during any phase of the development process, we would turn to the forums. Ionic has active community on forums such as Stack Overflow to provide help. Ionic Blog, http://blog.ionic.io/, is also a great way to keep update with the latest Ionic news.

Ionic Creator

Recently, Ionic offers an amazingly simple tool to create an app from A to Z by just drag and drop. The tool is called Ionic Creator. With Ionic Create, it is possible to create an app with just the drag of a mouse. It makes the application development process even more faster and simpler. It comes

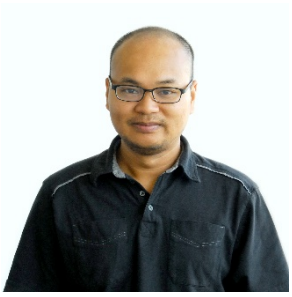
with all the Ionic UI components including tabs, side menus, slide boxes, modals and forms. All we need to do is drag and drop the components into pages. It also has readymade templates for building login and signup pages, tabbed interfaces, Google Maps, etc. Furthermore, Ionic Creator allows us to export the ready to go project. This feature is offered by Ionic as free developer plan. Higher level premium plan with more services and features is also offered ("Build amazing mobile apps, faster," n.d.)

BIOGRAPHIES



Minh Q. Huynh is an Associate Professor at Southeastern Louisiana University. He received his Ph.D. from State University of New York at Binghamton, and his B.S. in Computer Sciences from University of Maryland University College. His teaching interests are in the areas of Digital Marketing, Web design, Introduction to MIS. Research interests include software development using open source, web database implementation, web apps, technology supported distance education, IT for small businesses. His publications appear in journals such as the Journal of Information Technology Education: Innovations in Practice (JITE:IIP), the Communications of ACM, Journal of AIS, Communications of AIS, European Journal of IS, Journal of Electronic Commerce in Organizations, Journal of Information Systems Education, Business Studies Journal.

Prashant Ghimire completed his Bachelor degree in Computer Science from Southeastern Louisiana University. He is passionate about modern JavaScript technologies, mobile application development and UI/UX design. He has worked with many popular web development technologies such as AngularJS, Ionic, Loopback, jQuery, Bootstrap, ASP.NET, Laravel, CakePHP, MySQL, MongoDB, etc. Prashant has also been involved in research project on web application development and cloud computing. His work on "Linking QR code to service learning" was featured at the IARSLCE-Research poster session. His research on QR Code/Barcode processor was published on Business Studies Journal. He also presented this research at the 6th Annual General Business Conference at Sam Houston State University. He is currently working as a Software Engineer at Leviton Security & Home Automations.



Donny Truong is Director of Design & Web Services at George Mason University Antonin Scalia Law School. He received his M.A. in Graphic Design from George Mason University School of Art and his B.A. in Digital Arts & Multimedia Design from La Salle University. He published his thesis on *Vietnamese Typography* and taught courses in web design, accessibility, and usability.