2. Literature Review

2.1 Introduction

The client reasons that a mobile application will enable coachees stay properly engaged with their programmes resulting in improved completion rates, which will lead to higher incomes by Coach in a Box – completion rates of coachees enrolled in a programme is proportional to the money paid by the coachees’ employer, who are the clients of Coach in a Box.

The basis of this literature review is to:

* review the effects of mobile applications on learning (mobile learning) i.e. do mobile users engage with mobile learning apps than with e-learning apps?
* review current mobile learning applications
* review native and hybrid applications

2.2 Mobile Learning

2.2.1 What is Mobile Learning

In it’s simplest form, mobile learning can be defined as a type of learning delivered using mobile technologies. A comprehensive definition given by Gikas and Grant (2013, p. 19) describes mobile learning as:

* learning that’s more than that delivered and supported by mobile devices: Learners are able to access information and knowledge ubiquitously on devices that are portable and especially personal -- owned by the user
* learning that’s both formal and informal: Formal learning is structured and designed, i.e. learning delivered in universities. Whilst informal learning is often defined as learning that results from daily work-related or leisurely activities: informal learning is unstructured but intentional.
* learning that is context aware and authentic for the learner: Learners can personalise their interactions with the content. Mobile learning allows for learning to take place in environments that are conducive to the learner i.e. your learning is not restricted to a particular environment e.g. lecture halls and you can learn at your own pace and time.

2.2.2 Existing studies that support the effectiveness of mobile learning

(i) Alexander (2004, p. 61) contends that students prefer to work, and work differently, with mobile learning devices than with desktop computers. He states that public computers will still be seen as impersonal regardless of the emotional investment of the student. Also, according to Alexander (2004) personal desktops retain an external, semi-public face -- their screens readable by passers-by or, worse yet, room-mates. But he notes that mobile devices are personally intimate; they are held close to the body -- in a purse, on the lap, in a pocket, and on the floor next to the user. Their screens are easily hidden from prying eyes (Alexander, 2004). An example is given of Michele Forman, the 2001 National Teacher of the Year in the United States, who notes that her high school students became very attached to their wireless laptops. Michele concluded that the students increased their personal writing and composition after the introduction of wireless laptops (Alexander, 2004).

(ii) Smartphones are increasingly popular in the practice of nursing and education of nurses (Garrett & Klein, 2008, cited by Pimmer, Brysiewicz, Linxen, Walters, Chipps & Gröhbiel, 2014, p. 1398). Smartphones support a broad range of practices in formal educational settings as well as in clinical environments (George, Davidson, Serapiglia, Barla & Thotakura, 2010, cited by Pimmer et al., 2014, p. 1398). For example, Taiwanese nursing students used smartphones as clinical examination tools in simulations and this resulted in higher learning outcomes compared to a control group who used pen and paper to record and evaluate patients’ symptoms (Wu, Hwang, Tsai, Chen & Huang, 2011, cited by Pimmer et al., 2014, p. 1398).

(iii) In a British study, midwives used multimedia podcasts on iPods at bedsides to support the learning of the Newborn Infant Physical Examination (Clay, 2011, cited by Pimmer et al., 2014). The analysis of the small-scale pilot scheme revealed that the tool was well received and the participants specifically appreciated the “just-in-time” learning facilitated by these devices.

(iv) Similarly, studies from the USA (United States of America) and Canada came to the conclusion that nurses and nursing students view mobile devices as effective means to support their learning in the workplace by enabling access to several sources of expertise in decision-making processes. Reference tools such as drug and diagnostic/laboratory applications were found to be particularly popular and valued (Garrett & Klein, 2008; George et al., 2010; Kenny, Park, Van Neste-Kenny, Burton & Meiers, 2009, cited by Pimmer et al., 2014).

(v) In addition to the provision of information and communication features, a mobile e-portfolio allowed Canadian students to document their clinical experiences using different modes including text, audio and image. The students greatly valued the reference functions, but also appreciated the opportunity to capture clinical events in the form of photographs (Garrett & Jackson, 2006, cited by Pimmer et al., 2014, p. 1398).

(vi) Similarly, in a UK (United Kingdom) study reported by Morley (2013), cited by Primmer et al. (2014, p. 1399), a range of Web 2.0 tools was piloted to support peer-to-peer and learner–to-tutor interactions of nursing students working in isolated placements settings using digital technology to support learning in clinical placements and to address professional isolation.

(vii) Whilst these studies mentioned so far were conducted predominantly in high-income countries, the investigation carried by Pimmer et al. (2014, p. 1403) showed that even nursing students in low-income countries also used mobile technology as educational tools. These learning practices involved socio-cognitive processes, which are, learning in the form of joint problem-solving and reflection, and showed the effectiveness of mobile learning in remote and resource-poor areas in informal learning contexts in low and middle-income countries (Pimmer et al., 2014).

Due to the pervasiveness of mobile devices and ability to ubiquitously access resources and information on the web via the Internet, mobile learning knows no boundary, is effective, keeps users engaged and results in higher learning outcomes.

Disadvantages of mobile learning are usually associated with the size of mobile devices used. For instance the size of the mobile device used may be an issue if the device has a small screen. But recent smartphones from iPhone 6, iPhone 6 Plus, Nexus 5 and Android Phablets have reasonable large screen sizes. Alternatively tablets and iPads can be used instead of smartphones for a better learning experience.

Distractions from other applications like Facebook, twitter, is another disadvantage. Lots of distractions exist when using smartphones and tablets but the same can also be said of e-learning and yet e-learning has been found to be very effective – it all boils down to the focus and determination of the user not to be distracted.

2.3 Existing leadership development and management mobile applications

Several leadership development and management applications exist in Android’s Google Play Store and iPhone’s App Store, the two leading mobile operating systems. Whilst most of these leadership development and management applications are standalone applications, the “My Page” application compliments an existing website.

A Google search of “business coaching UK” listed many companies offering business or executive coaching as a service but I will be looking at a couple, namely: Notion’s BusinessCoaching, Shirlaws, Vistage and CoachDirectors.

1. Notion Coaching (UK): There’s no application by the name of Norton Business Coaching in Google’s Play Store or iPhone’s App Store.
2. Shirlaws (Australia): This is an interactive conference tool application. You can obtain the conference’s agenda and biography of the speaker. It’s has only one review implying it’s not used by lots of people.
3. Vistage International Events (UK): This application enables you to find and chat with other attendees through the in-app event messaging; be up-to-date during events; obtain the conference’s agenda and biography of the speaker. It has three reviews. Whilst better than having one review, it still implies it’s not used by lots of people.
4. My Vistage (UK) – With My Visatage, you can add meetings to your calendar, locate venue of meeting, mark meeting attendance and update your profile. It has two reviews with one review stating: “Used to work, (sic) now just hangs up …”. This review indicates the importance of the performance and maintenance of the app.
5. CoachDirectors (UK): There’s no application by the name of CoachDirectors in Google’s Play Store or iPhone’s App Store.

These companies are direct competitors of Coach in a Box and operate in similar domains (leadership development, business coaching, executive coaching etc.), albeit with a slight nuance – Coach in a Box caters to the many -- executives and other employees, rather than the few (only executives). Strangely, especially for companies that should know better, there isn’t a major investment in mobile learning by companies offering business and executive coaching services. Some companies like CoachDirectors don’t have a mobile application and even companies with mobile applications have few or poor reviews.

2.4 Native or Cross-platform application

2.4.1 Native Applications

Native applications refer to applications that are developed to target a specific mobile platform (Jobe, 2013 p. 28). Kohan (2015) defines native application development as the use of native programming languages of the platform to build an application. The two main major mobile platforms are iOS and Android. Vendors of these platforms support specific programming languages and tools to develop native applications. For example to develop iOS applications, you can either use Objective-C or Swift programming language with the Xcode IDE, whilst Java programming language and Android Studio IDE is used to develop Android native applications. Native applications have direct access to hardware devices and support all, simple and complex, user interfaces and interactions available in their respective platforms (Jobe, 2013 p. 28).

Advantages

1. Better performance: Native applications provide a better performance compared to most cross-platform applications. Native applications are responsive and provide an all-round fluid experience -- a user can navigate and interact with the application without noticing any loading delays.
2. User friendly: Additionally, native application development environments (IDEs) provide design and user interaction widgets that come standard with each platform. So, the users will have a better experience because they are familiar with the standard native icons, actions, transitions or navigation (Kohan, 2015).
3. No limitations: Native applications are developed using native application IDEs (Xcode and Android Studio), so implementing many native features and interactions are easily adoptable and built into the IDE (Kohan, 2015).
4. Support and Resources: There is a lot of support and resources to develop native apps; there is more support and available resources to build native applications than to build cross-platform applications (Kohan, 2015).
5. Dedicated IDE: Native application development occurs in dedicated and specific IDEs that provide better debugging features than cross-platform development. Some cross-platforms don’t even have a dedicated IDE and developers have to improvise.
6. Tools and Debugging: Native development provides better development tools so it is much easier and less time-consuming to find and fix bugs; test app; build and deploy app using native development IDEs (Kohan, 2015).
7. Popularity: Native applications are more popular with large companies because they provide a better and fluid user experience; and are less likely to have feature limitations (Kohan, 2015).

Disadvantages

1. The expertise in native programming languages are less than those in most cross-platform programming languages.
2. Programmers in this field are usually more expensive to hire because native applications are more difficult to develop and require experienced and technically skilled developers (Xanthopoulos & Xinogalos, 2013, p. 213).
3. Development time and maintenance costs: Code reuse is not supported in native app development. Source codes used to develop an iOS app can’t be used to develop the same app in Android: the same app has to be re-developed using Java from scratch. This increases the time spent in developing the same app on Android and iOS platforms. It also increases the cost of maintaining two distinct source codes for the same app (Xanthopoulos & Xinogalos, 2013, p. 214).
4. Native app development favours large organisations with huge resources and considerable manpower because of the reasons stated above.

It is apparent that native application development is a better approach but this suits large companies with several skilled and experienced developers in the relevant programming languages e.g. Java, Swift or Objective-C. Coach in a Box is not a large organisation and the current development team is not skilled in native app development. Therefore the team will find it difficult to maintain the app. Alternatively Coach in a Box may have to hire a skilled native developer but this increases the cost to maintain the app. Considering I have no experience in building native applications, the demand on my time from other units, impending project deadline (April, 2016) and also the desire by Coach in a Box to target Android and iOS platforms, the app will not be native.

2.4.2 Cross-platform applications

The aim of cross-platform mobile development is to target all mobile platforms with the same source code or to reuse some bits of the source code and achieve near-native performance that’s indistinguishable from native performance. There are a lot of options when it comes to cross-platform applications. The factors to be considered are:

1. price: Does the vendor charge any subscription to use the tools and/or programming language? If so how expensive is it?
2. learning curve: Am I familiar with the programming language? If not how much effort is required to learn the programming language? Can the development team at Coach in a Box easily grasp the language or framework?
3. performance: Is the performance at par with a native app? Do apps developed using the programming language lag? If they do lag, can the app be optimised to deliver a near-native performance?
4. stability: How stable is the programming language? If it’s new, will support for the programming language continue in the foreseeable future? Do I think the creators are in it for the long haul? If issues exist do I think they will be resolved quickly?
5. online resource and support: Is there enough online support of the language? Is the documentation of the programming language comprehensive and clear? Are examples given? Does the language have a presence in Stack Overflow: does Stack Overflow support the language, are questions asked and answered on Stack Overflow?

The main types of mobile applications produced are hybrid, interpreted and generated applications (Xanthopoulos & Xinogalos, 2013).

(i) Hybrid mobile applications

Hybrid applications aim to leverage the advantages of web and native applications. Hybrid applications are primarily built using JavaScript frameworks. These frameworks utilise web technologies like HTML5 and JavaScript; and a detailed knowledge of the target platform is not required.

Similarly to web apps, the source code of hybrid apps is executed by a browser, which is packaged with the final app. Unlike web apps, whose source code is downloaded from the web, hybrid apps are installed on the device and can access the underlying hardware devices through specialised APIs (Application Programming Interfaces). Examples of containers used to create hybrid mobile applications are PhoneGap and Cordova (Xanthopoulos & Xinogalos, 2013, p. 215).

The performance of a hybrid app depends on the WebView: a minimal browser that delivers web content (Looper, 2015) i.e. UI (User Interface) rendering and running of JavaScript codes using its embedded JavaScript engine (Khanna, 2015).

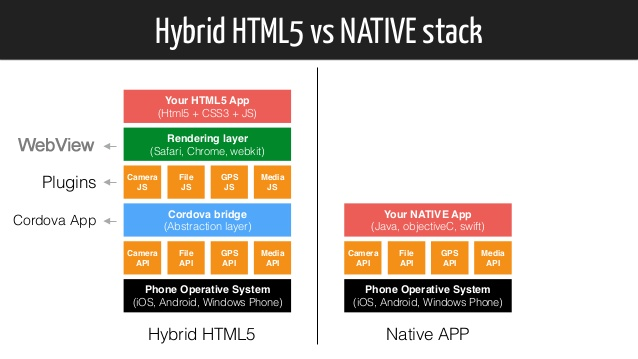


Fig 1: Hybrid HTML5 vs Native Stack (Rodrigues, 2015)

The WebView used in earlier versions of iOS and Android were not the latest rendering or JavaScript engines used by their web counterparts e.g. Safari and Chrome for iOS and Android respectively. This made the performance of hybrid apps lag in comparison to its web counterparts (Khanna, 2015).

In 2012, the lag in performance of hybrid apps led Facebook’s CEO Mark Zuckerberg to admit that developing Facebook as a mobile app using HTML5 rather than as a native app was a big mistake. Facebook later converted its mobile app to native apps (Xanthopoulos & Xinogalos, 2014).

But recently Apple has updated its UIWebView to WKWebView API in iOS 8. This WebView leverages the Nitro JavaScript JIT compiler used by Safari. Nitro compiles JavaScript codes into native codes, making it faster than the older UIWebView. This update will greatly increase the performance of JavaScript execution in hybrid apps on iOS platform (Khanna, 2015).

The situation is more fragmented on Android devices due to the many devices running Android and different manufacturers of Android devices e.g. Samsung, HTC and Sony, to mention a few. Whereas earlier Android versions relied on the WebKit engine to power its WebView, as of Android 4.4 various versions of Chromium are implemented. This obviously causes backward compatibility issues for developers who must support earlier Android versions (Looper, 2015).

To address this fragmentation, Google, in its Android 5.0 Lollipop release, has updated the WebView so that users can get updates for WebView directly from Google’s Play Store. This will ensure that users receive timely performance enhancements and security upgrade patches (Khanna, 2015).

Examples of some JavaScript frameworks used in hybrid app development are Mobile Angular UI, Ionic, jQuery Mobile, Sencha Touch and React Native.

(ii) Interpreted mobile applications

Interpreted apps make use platform-specific native components to implement the user interface, whilst the application logic is implemented independently of the platform using programming languages or frameworks like Ruby (Rhodes) and JavaScript (Appcelerator) (Behrens, 2010).

This approach yields improved efficiency because of the native user components, but a major drawback is that users are tied to the features provided by the framework. For instance, new platform-specific features (e.g. iPhone’s Retina Display) can be available to apps only when the framework supports them (Xanthopoulos & Xinogalos, 2013, p. 215).

(iii) Generated mobile applications

Frameworks used to develop these apps are written in one of several programming languages and compiled to the executable native code of the targeted platform. This approach produces truly native apps for each platform from a single source code (Behrens, 2010). Examples include Xamarin (C#), RubyMotion (Ruby) and B4X (Visual Basic).

The performance of a generated app is indistinguishable from that of a native app because of the compiled source code. In theory, it is possible to alter the resultant compiled code to meet specific needs e.g. a suggestion to correct a deprecated call. In practice however, the use of generated compiled codes are difficult because of their automated structures. An example is Applause, which is open source and uses a domain-specific language based on the XText framework (Xanthopoulos & Xinogalos, 2013, p. 216).

2.4.3 Comparison of some cross-platform mobile app frameworks

Some frameworks used in cross-platform mobile app development will be compared based on the following criteria: price, learning curve, performance, platform, stability and online resource and support. Appendix C shows this comparison.

2.5 Conclusion

From the comparison, it is obvious that Xamarin, RubyMotion and B4X apps yield native or near-native speeds. This is because their source codes are compiled just like native programming languages i.e. Java (Android), Swift or Objective-C (iOS). But they require developers to be skilled in C# (Xamarin), be knowledgeable of native APIs (RubyMotion) and Visual Basic (B4X). These are skill-sets the Coach in a Box’s development team and I don’t have. Also one needs to pay a subscription fee or license to access the full features of some of these frameworks, which isn’t ideal for a university project. These frameworks are ruled out for these reasons.

Rhodes Mobile looks ideal as it’s a Ruby framework, a language the Coach in a Box’s website is based on. But the cheapest paid package is $299 a month – it’s very expensive. Appcelerator Titanium isn’t an alternative because I’m not familiar with the Titanium framework and it’s not free. Therefore I’m left with the hybrid (HTML5 and/or JavaScript) frameworks.

After an in-depth analysis of the hybrid frameworks, I have decided to go with the Ionic framework for these reasons:

* it’s free;
* though it’s fairly new, there are lots of online tutorials and support. The official documentation is comprehensive for Android and iOS platforms;
* the design and UI components look sleek. You can achieve a native look and feel with the framework;
* it has a large ecosystem that provides functions and features from rapid prototyping (Ionic Creator) to a robust development environment (Ionic CLI) to the ability to preview apps in iOS and Android views while developing (Ionic Serve) to the ability to share apps without installing on the targeted devices (Ionic View).
* Ionic is based on Angular.js and the development team is a bit familiar with it. I’m confident that the effort required to learn Ionic won’t be so much for them, so maintaining the app will be feasible.
* it has a vibrant community – The upcoming Ionic 2.0 promises to improve the performance of Ionic apps and backward compatibility of Ionic 1.X apps has been assured.
* though there are some performance issues with page transitions, animations, scrolling and two-way binding, there are also solutions to avoid, fix or mitigate some of these issues.
* because it’s an open source framework, there are lots of third-party plugins that try to address some of the shortcomings in the framework

Admittedly the Ionic framework isn’t comparable with native frameworks and will struggle to handle complex UIs. But considering the duration of the project; competing demand by other course-works; nature of the project (UI is not complex); and size and skill-set of the development team at Coach in a Box, I believe Ionic is an apt choice.

.