*Impact Factor (SJIF): 3.632*

**International Journal of Advance Research in Engineering, Science & Technology**

*e-ISSN: 2393-9877, p-ISSN: 2394-2444*

***(Special Issue for ITECE 2016)***

# Enterprise Mobility : Analogy among Popular Cross Platform Application Development Tools

**Darshan Thoria[1], Drashti Hirani[2], Vishal Kansagara[3]**

*[1]CSE, SLTIET*
*[2]M.E.C.S.E.*
*[3]CSE, SLTIET*

***Abstract — Mobile computing in the enterprise is still relatively immature, primarily due to the complexity of the applications that support mobility. Up to now, the development of these applications has traditionally spanned operating systems, devices, and any number of development frameworks. The vendors in each of these categories have had distinct approaches to mobile application development that reflect their core competencies. Yet, as is often the case with technologies that are rapidly maturing, some of these approaches are on the verge of becoming obsolete. Due to the rise of mobile application development platforms (MADPs) in the enterprise, it is becoming much simpler to develop mobile applications that can run on almost any platform. In this paper, we present an elaborate comparison of several popular cross-platform mobile application development tools, namely, Xamarin, PhoneGap/ApacheCordova, Appcelerator Titanium, Kony and Corona. We provide a pragmatic comparison from different perspectives like ease and cost of development including programming language and tool support, end-product capability and performance and community support. This comparison aims to support mobile application vendors from a large scale, from freelancers to enterprise development teams, when making a choice with respect to their specific requirements and constraints.***

***Keywords-enterprise mobility, cross-platform development tools, native API, operating system, Gartner analysis***

## I. INTRODUCTION

The market for enterprise application development is expected to reach $61B by 2018, according to Strategy Analytics. Gartner predicts that by the end of 2015, 50 percent of business processes that involve human tasks will require near-real-time responses that are mobile-enabled. Mobility has quickly become an expected norm as opposed to a new trend. The fact remains, however, that the market for Mobile Application Development Platforms (MADP) is crowded and confused. It is highly fragmented and is bracketed by small ideal players that are providing lightweight and focused solutions on the low end and global firms which provides “do everything” platforms, from design and development through test and deploy. Gartner lists 22 platforms alone in their Magic Quadrant, and those companies needed to exceed Gartner’s stringent financial guidelines to be considered.

Mobile is complicated by nature. Companies are increasingly turning to mobile as a way to engage customers, in addition to increasing use as an internal productivity tool. Those who develop applications for consumers have already understood the need to support at least the two main platforms: Apple and Google. Thanks to BYOD (Bring Your Own Device), enterprises are finding they too must support at least the two main platforms.

The challenge has become how to develop mobile applications for at least two platforms, without needing to develop in Objective-C (Apple), Java (Google), and .NET (Windows Phone/RT). In most cases, consumer and enterprise mobile developers have only one, if any, of these development capabilities already. Thus, organizations are faced with the reality that they must literally build almost the exact same application two or more different times on completely different technology stacks and languages. In the case of Apple, you can’t even build iOS apps without having a Mac. As a result, mobile computing has emerged as one of the highest enterprise IT priorities and finding a way to build these applications not only quickly but also as cost-efficiently as possible, and so cross-platform mobile development tools are gaining popularity in the world due to their characteristic to compile the application source code for multiple supported OS’s.

## II. MOBILE APPLICATION DEVELOPMENT OFFERINGS

The mobile application development market today consists of three dominant types of offerings as follow:


***Fig: 1 Approaches to MobileApp Development [18]***

### *A. Native App(Platform-based)Developemrnt*

Native apps are the ones specifically designed and developed for a given mobile platform using the development tools and languages that the respective platform natively supports [11]. For example, Xcode IDE (Integrated Development Environment) and Objective-C language (or Apple's new Swift language) are used for iOS, Eclipse IDE and Java language are used for Android, and finally Visual Studio IDE and C# language are used for Windows Phone environment. Native apps can offer the best look-and-feel and user experience to users with smooth and fluid interface. They can also provide rich user experience with a rich set of device-specific features like Camera, Microphone, Accelerometer, GPS, Bluetooth, NFC, Fingerprint Scanner, and so on. Moreover, native apps can be published in app stores and can be discovered by users.

Despite its superior features, on the other hand, software vendors targeting multiple platforms with their apps have to have separate code bases and separate development teams for each platform they target for the same apps if they are to follow the native app development approach. This approach can become very costly depending on the complexity of the apps developed. As a result, several cross-platform solutions have emerged over the years, each with unique features, and most mobile software vendors have been considering these cost-effective alternatives instead of sticking to native app development unless the developed apps strictly require platform-/device-specific capabilities.

### *B. Hybrid App Development*

In hybrid development approaches, generally, a web app is wrapped within a native container app that can be installed and run just like any other native app of the respective platform [12]. The development approach and the end products are very similar to web pages/web apps. The web app runs inside a native user interface layer known as Web View [8]. Therefore, hybrid apps usually struggle to establish the smooth and slick usability of native apps. There can also be platform-specific problems like page transitions that do not work smoothly on Android due to lacking CSS/CSS3 implementation. In addition, browsers on different platforms do not uniformly support all the latest HTML features and APIs, which can make development and testing a challenge [11]. The only advantage of using a hybrid app over a mobile web app is the ability to access to most commonly used device features such as Camera or Contacts, which is almost impossible via a mobile web app. However, access to advanced device features or interaction with other apps requires native development, which eliminates the advantage of hybrid platforms.

### *C. Web App Development*

In this type of mobile app development, Responsive Web Design (RWD) pattern is usually followed. RWD is an approach to web site design aimed at creating web sites to provide an optimal viewing and interaction experience like easy reading and navigation with a minimum of resizing, panning, and scrolling, across a wide range of devices from desktop computer monitors to tablets and mobile phones [7]. Either a new mobile version of an existing traditional web site is created keeping the existing one for desktop access, or the web site is created totally with this approach from scratch. Standard web technologies such as HTML5, JavaScript, and CSS are used in the development of mobile responsive web apps. Although it looks advantageous at first to create mobile apps this way, as with any other website, these type of apps are restricted to the features of the browser used to view the app. In addition, they cannot provide rich user experience that native apps can do by using several device-related APIs (Application Programming Interface) such as Camera, Accelerometer, GPS, and so on [8].

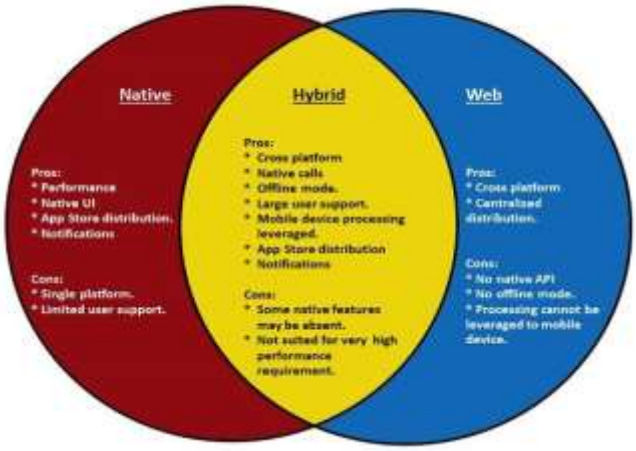


***Fig: 2 Pros and Cons of each Approach [18]***

## III. CROSS PLATFORM APP DEVELOPMENT TOOLS

Cross-platform app development is a totally different approach than the hybrid one. In this approach, a true native app is generated from the same single codebase for each target platform supported. UI controls are truly native and not visually emulated through CSS. Therefore, cross-platform native apps usually provide better performance and user experience than hybrid apps [4]. While development tools for this type of apps are usually "write once, run anywhere", some tools allow platform-specific tuning to handle different UI requirements with a "write once, adapt everywhere" philosophy. In general, the user interface code and the code that deals with the device-specific features tend to be written for each platform, while the code for other tasks can potentially be reused such as service client logic, client-side validation, data caching, and client-side data storage, saving a significant amount of time and effort [9].

Most cross-platform tools are maturing quickly -- many of the features you need in enterprise mobile apps (barcode scanning, image recognition, forms, data connection and text-voice) have long been included in the core of the cross-platform tools. Having the ability to create apps that can be delivered very quickly to many platforms is a huge win for cross-platform app development tools. This is very important for small to midsize businesses (SMBs) that do not have budget to support mobile developers for Android, iOS and Windows. Indeed, even large companies can supplement smaller project development with cross-platform tools.

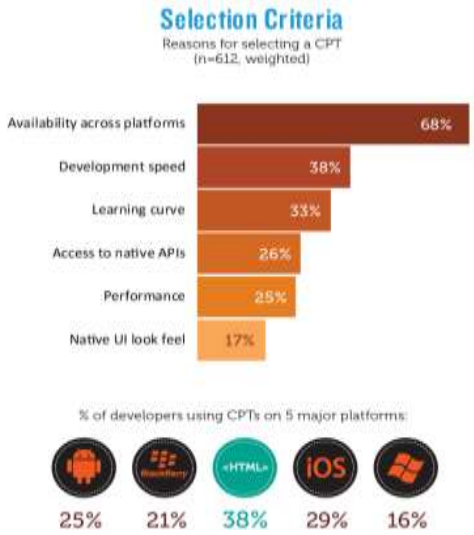


***Fig: 3 Selection Criteria for CPT [17]***


***Fig: 4 Five Stages in Life of a Cross Platform App [17]***

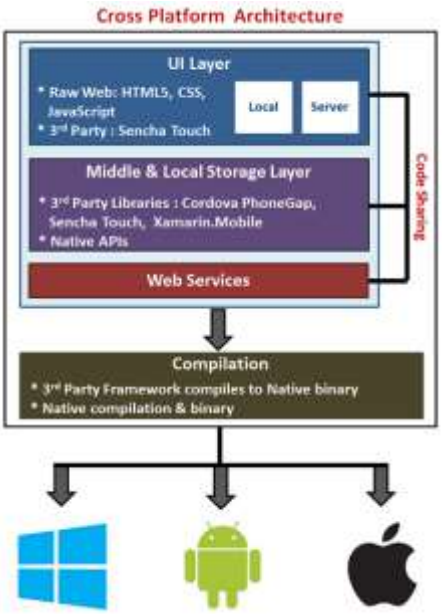


***Fig: 5 Cross Platform Architecture [18]***

In this study we have mainly considered six popular cross-platform mobile app development frameworks which are Xamarin, Appcelerator, PhoneGap/Apache Codova, Kony, and Corona.

### *A. Xamarin*

Xamarin is a development platform that allows developers to code native, crossplatform iOS, Android, and Windows Phone apps in C# language, which is the flagship and the most popular programming language for the .NET platform of Microsoft [6]. The Xamarin platform has been descended from the open source Mono Project that has brought .NET to Linux, and it is now a port of .NET to iOS and Android operating systems with support for Windows Phone. Xamarin uses C# bindings, called Xamarin.Android and Xamarin.iOS, to the native Android and iOS APIs for development on mobile and tablet devices. Xamarin provides development environments and designers to build mobile apps on Windows or Mac operating systems. The two common choices for Xamarin development environments are Xamarin Studio on Mac or Windows, or Visual Studio on Windows with the Xamarin for Windows plug-in. A Mac is always required for compiling iOS apps, even if Visual Studio is used as the development environment.

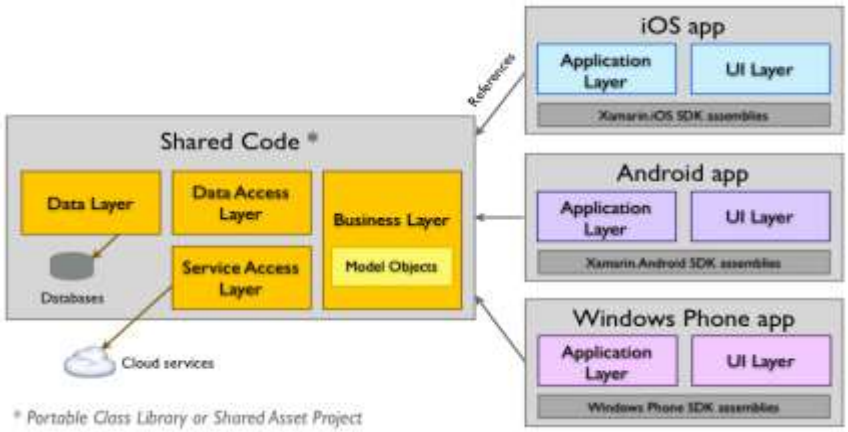


***Fig: 6 Xamarin Architecture [18]***

### *B. Appcelerator*

Appcelerator Titanium is a platform for building cross-platform native mobile apps using well-known web technologies, such as JavaScript, CSS, and HTML. Titanium is an open source project developed by Appcelerator Inc and licensed under the Apache Public License [4]. Titanium has the architectural goal of providing a cross-platform JavaScript runtime and API for mobile development, which differs from the hybrid frameworks. Besides Android and iOS, Titanium can produce native output for Windows Phone 8. Titanium is not a "write once, run anywhere" application framework, but more of a "write once, modify for each platform" one, especially for the user interface of the apps developed. Appcelerator reports 60-90% code reuse across device platforms in general [3].

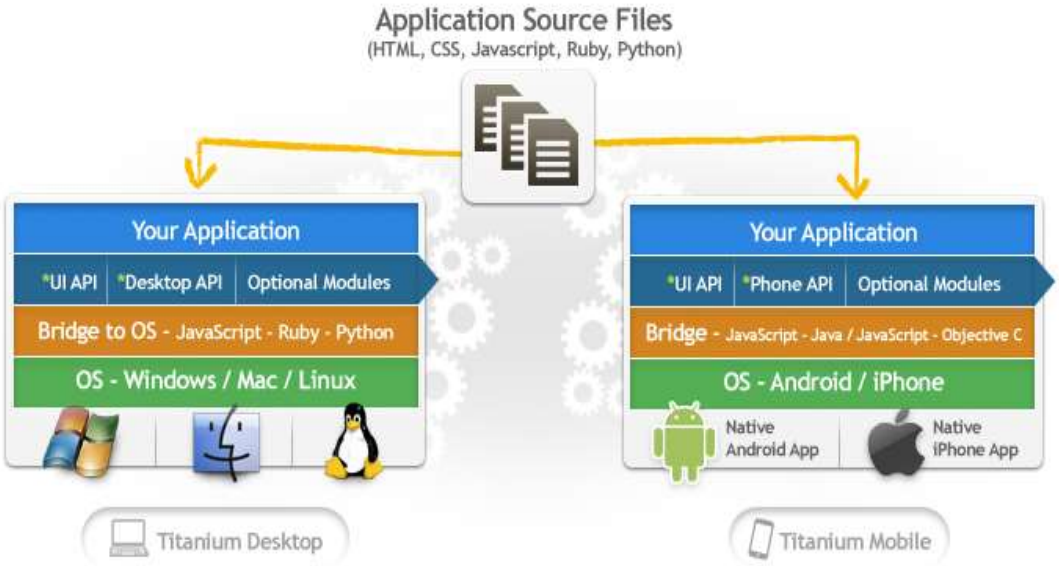


***Fig: 7 Appcelerator Architecture [18]***

### *C. PhoneGap/Apache Cordova*

PhoneGap, also known as Apache Cordova, is an open source hybrid mobile app development framework. It was originally developed by Nitobi and the company was acquired by Adobe in 2011. After Nitobi was acquired, Adobe donated the PhoneGap code base to the Apache Software Foundation (ASF) under the project name Cordova [8].PhoneGap uses standard and well-known web technologies such as HTML, CSS, and JavaScript to create cross-platform mobile apps without using native development languages. It packages app code into an executable program that can run on an array of mobile devices. Developers can write code once and deploy their app across multiple mobile operating systems such as iOS, Android, Windows Phone 8, BlackBerry, and Amazon FireOS. PhoneGap provides a JavaScript programming interface that allows developers to access platform-specific features with plain JavaScript [5]. A PhoneGap application should not just be a wrapped static web site that does not take advantage of its platform capabilities via APIs. In fact, for example, Apple is known to refuse apps to the App Store if it feels the app is solely a bundled web site that provides no entertainment value.

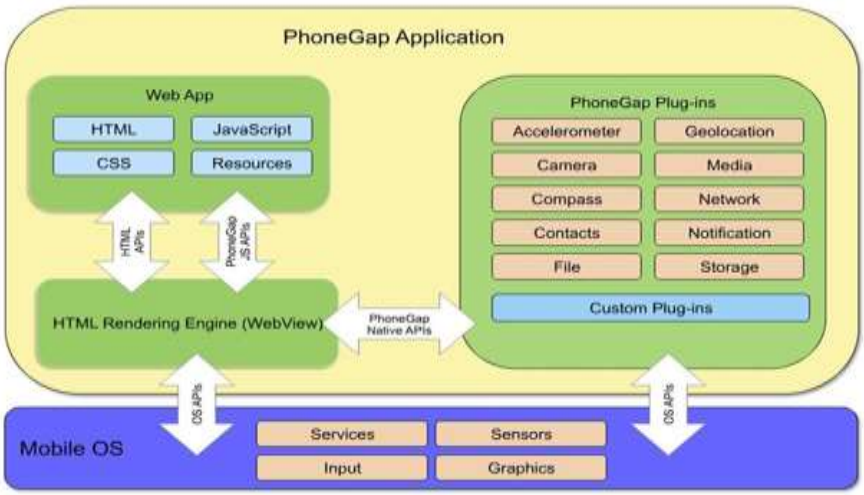


***Fig: 8 Phonegap Architecture[18]***

### *D. Kony*

Kony is a mobile application development platform for building mobile, tablet, desktop, and kiosk applications. Kony is one of several platform supports that developers can use to build mobile applications for consumers and enterprises.

Kony has been integrated with native platform SDKs to execute natively across devices. The run-time consists of a Mobile App Server, and a virtual machine that implements the scripting language (JavaScript) used by Kony, coupled with the library for each device platform – BlackBerry, Windows Mobile, Java, Symbian, and browser clients.[15]

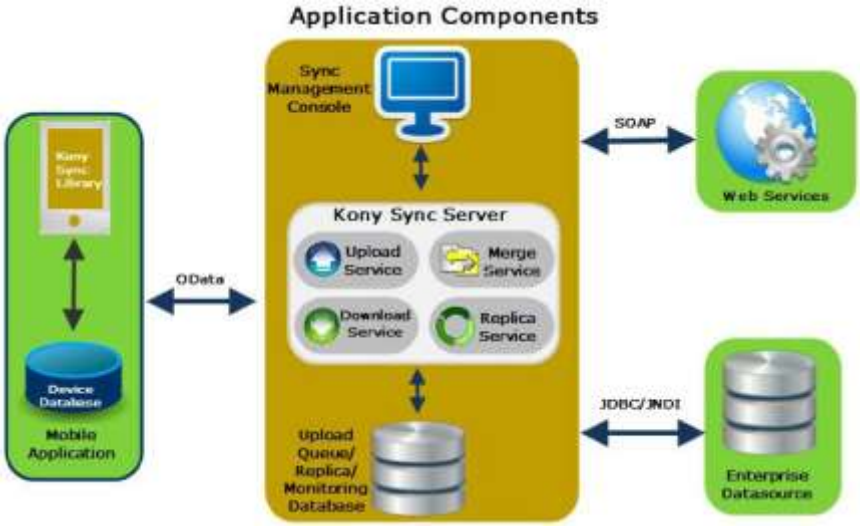


*Fig: 9 Kony Architecture*

### *E. Corona*

Corona's SDK comes with the promise that you can start coding your new app in as little as five minutes after the download. It's another cross platform mobile development tool that's optimized for 2D gaming graphics and helps you make games 10 times faster than it would take to code everything from scratch.Corona's programming language is Lua, which is written in C, making it a cross platform language. Corona chose Lua because they found it to be really robust with a small footprint for mobile apps.[16]


*Fig: 10 Corona Architecture [16]*

## IV. COMPARISION FACTORS AMONG DEVELOPMENT TOOLS

### *A. Type*

| XAMARIN | APPCELERATOR | PHONEGAP | KONY | CORONA |
|---|---|---|---|---|
| Natively Compiled | Interpreted on the top of the natively compiled | HTML5 | Interpreted on the top of the natively compiled | Natively Compiled |

### *B. Language Used*

| XAMARIN | APPCELERATOR | PHONEGAP | KONY | CORONA |
|---|---|---|---|---|
| .Net | JavaScript | HTML5,CSS,Javascript | JavaScript | .Net |

### C. Tool Set

| | Development Platform | Testing | Profiling | Analytics |
|---|---|---|---|---|
| **XAMARIN** | ✓ | ✓ | ✓ | ✓ |
| **APPCELERATOR** | ✓ | ✓ | ✓ | ✓ |
| **PHONEGAP** | ✓ | | | |
| **KONY** | ✓ | ✓ | ✓ | ✓ |
| **CORONA** | ✓ | | | |

### D. Platforms

| | iOS | Android | Windows/RT | Blackberry |
|---|---|---|---|---|
| **XAMARIN** | ✓ | ✓ | ✓ | |
| **APPCELERATOR** | ✓ | ✓ | ✓ | ✓ |
| **PHONEGAP** | ✓ | ✓ | ✓ | ✓ |
| **KONY** | ✓ | ✓ | ✓ | ✓ |
| **CORONA** | ✓ | ✓ | ✓ | |

### E. Strengths and Weaknesses

| | Strengths | Weaknesses |
|---|---|---|
| **XAMARIN** | The native app with cross-platform benefits, stability | Learning curve |
| **APPCELERATOR** | UI abstraction, popularity, relatively easy language | Log time with OS release complex native code integration, UI abstraction can lead to non-standard UI |
| **PHONEGAP** | Low technology requirements, easy to learn, common language | UI limitations, functional limits, complex native code integration |
| **KONY** | Fast with complete toolset, integration tools | UI limitations, complex native integration |
| **CORONA** | Market leading in graphics/games support | Limited naïve UI support |

### F. Comments

| XAMARIN | APPCELERATOR | PHONEGAP | KONY | CORONA |
|---|---|---|---|---|
| Compiles into a true native APP or APK ,use native UI tools | One of the largest and most well-known platform | Considered one of the fastest frameworks to learn | One of the most complete players | Considered one of the top platforms for mobile game development |

## V. CONCLUSION

The following are guidelines you can use when selecting tools for your enterprise:

Medium-sized enterprises will face the challenge of having to build apps for both iOS and Android due to a mix of staff and consumer needs. Cross-platform technologies such as Xamarin or Appcelerator work well in this type of environment. The use of C# in Xamarin will make it easier for .NET-based teams to migrate to mobile.

Large Fortune 1,000 enterprises: The demands for larger companies are obviously more complex than SMBs because large, back office systems and the need for sophisticated security demand more complex apps. Due to this, it is highly recommended to have a multi-tier approach to app development. For small teams, choose a cross-platform tool; mission critical apps should be developed with native tools; and consumer-facing public apps should also be developed with native app development tools to leverage the latest security features in the mobile OS.

When security is a high concern in your apps, you will want to build native apps and potentially combine the deployment of the apps with a mobile application management product.

When speed of delivery is the highest concern, you will want to leverage a combination of COTS products with template apps your team has written that can be repurposed. Kony, PhoneGap, and Xamarin all have tools that fall into this category.

When cost is the highest concern, open source technologies are a great option. Apache Cordova with HTML5 development will be a good choice.

When support for three or more mobile platforms is critical, you will want to focus on cross-platform app development tools. The leader in this category is Apache Cordova. A few of the products listed above are built on Apache Cordova.

When support for Graphics/Games is the highest concern Corona will be a good choice.

It is not an easy task to choose a mobile app development tool because there are a lot of alternatives with a lot of different capabilities and offerings. Therefore, software vendors need to evaluate each one carefully, and we believe that it is best to try the ones that seem reasonable on small pilot projects to see if they are satisfactory in terms of the criteria we described.

REFERENCES

[ 1 ] Francese, R.; Risi, M.; Tortora, G.; Scanniello, G., "Supporting the development of multi-platform mobile applications", Web Systems Evolution (WSE), 2013 15th IEEE International Symposium on, vol., no., pp.87,90, 27-27 Sept. 2013
[ 2 ] Alcocer, R. (2015). Build Native Cross-Platform Apps with Appcelerator: A Beginner's Guide for Web Developers: J.B. Orion.
[ 3 ] Appcelerator. (2015). Mobile App Development Platform – Appcelerator , from http://www.appcelerator.com/
[ 4 ] Bahrenburg, B. (2013). Appcelerator Titanium Business Application Development Cookbook. Birmingham, UK: Packt Publishing.
[ 5 ] Fernandez, W., & Alber, S. (2015). Beginning App Development with Parse and PhoneGap New York, NY: Apress.
[ 6 ] Hermes, D. (2015). Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. New York, NY: Apress.
[ 7 ] LePage, P. (2014). Responsive Web Design Basics – Web Fundamentals, from https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/
[ 8 ] Ramanujam, P., & Natili, G. (2015). PhoneGap Beginner's Guide (3rd ed.). Birmingham, UK: Packt Publishing.
[ 9 ] Reynolds, M. (2014). Xamarin Mobile Application Development for Android. Birmingham, UK: Packt Publishing.
[ 10 ] Barkan Saeed "Comparing cross platform app development tools", a publication of mobilenext, 2015.
[ 11 ] Smartface. (2013). Native vs. Hybrid Retrieved August, 18, 2015, from http://www.smartface.io/nativevs-hybrid/
[ 12 ] Smartface. (2014). Native Smartface vs. PhoneGap/Sencha/Cordova Hybrids Retrieved August, 18, 2015, from http://www.smartface.io/smartface-app-studio-vs-phonegapcordova-cross-platformnative-vs-hybrid-environments/
[ 13 ] Smartface. (2015a). Smartface vs. Xamarin (Cross-Platform Native Frameworks) Retrieved August, 18, 2015, from http://www.smartface.io/smartface-app-studio-vs-xamarin-c-sharp-javascript-basedcross-platform-native-frameworks/
[ 14 ] Smartface. (2015b). Smartface: Develop Cross-platform Native iOS & Android Apps, from http://www.smartface.io/
[ 15 ] "KonyOne Platform with Mobile App Manager", a publication by www.kony.com
[ 16 ] Accessed https://docs.coronalabs.com
[ 17 ] Accessed http://www.developereconomics.com
[ 18 ] Accessed http://bluetubeinc.com, http://social.technet.microsoft.com/, https://developer.xamarin.com, http://mammoth.digital/, http://ltslab.blogspot.in/