Global Scientific JOURNALS

GSJ: Volume 8, Issue 5, May 2020, Online: ISSN 2320-9186
www.globalscientificjournal.com

HYBRID MOBILE APPLICATION DEVELOPMENT: A BETTER ALTERNATIVE TO NATIVE

ENIHE RAPHAEL OZIGHOR, JOSHUA JIMMY
Veritas University, Abuja, Nigeria

Keywords
Native Mobile Application Development, Hybrid Mobile Application Development, Comparison between Native and Hybrid Mobile Development

ABSTRACT

Developing applications targeting mobile devices is a complex task involving numerous options, technologies and trade-offs, much so due to the proliferation and fragmentation of devices and platforms. With the rising demands of smart phones and tablets, mobile apps are becoming ubiquitous, hence developing applications for mobiles are getting quite challenging in terms of cost, effort and marketing. There are varieties of operating systems in the market that are unalike, which are an obstacle to developers when it comes to developing a single application for different operating system.

The results of this work indicate that even though cross platform tools are not fully matured they show great potential and reduce the cost associated in developing native mobile applications. Hybrid mobile development is equally suitable for rapid development of high-fidelity prototypes of the mobile application as well as fairly complex, resource intensive mobile applications on its own right. As the upcoming future trends and the evolution of HTML5 continues to redefine the web, allowing its growth as a software platform, there remains great opportunities for cross-platform mobile development and hence provides an attractive alternative for the native mobile development.

I. INTRODUCTION

Applications for mobile platforms have over the last decade been the driving force for the smart phone revolution. The success has spread from smart phones to a variety of devices such as tablets, wearables and sensors, all recognized today as part of the mobile device's platform. Despite the huge success and substantial progress in relation to software platforms, hardware specifications, development methods and use there is still a long way to go to be at a standardized level (Behrens, H.:, 2010).
The number of different operating systems for smart phones are plentiful for example android, iOS, windows phone, blackberry and other that are not mentioned here. This makes it increasingly harder for developers and especially hobby-developers to create applications across operating system platforms (Anmol K, Rashmi G, B. Sindhya, 2015).
Generally, every system uses its own programming language, which makes it hard for a hobby-developer to learn all different languages and port their application to them all. When the first Hybrid app platforms were introduced, it essentially made it easier to create mobile-applications for all operating systems (Paulo R, Adriano B, 2015). An application can easily be created using HTML5, CSS and JavaScript and then simply port it using a Hybrid platform to any operating system you want (Jeff Whatcott, 2011).

This introduction triggered many developers to switch to Hybrid app development. Development companies concluded that they already had extensive knowledge about HTML5, CSS and JavaScript, but less experience in Android, Objective-C and other smart phone languages. The step into this new world of mobile development seemed substantially easier when switching to hybrid development (Jeff Whatcott, 2011).

OS	**Programming Language**	**Development Environment**	**Application Store**
Google's Android	JAVA, Kotlin	Android Studio, Android SDK	Play Store
Apple's iOS	Objective-C/Swift	XCode	Appel-iTunes
Microsoft Windows phone	Visual C#, C++	Visual Studio	Window Phone Market
RIM BlackBerry OS	JAVA	BlackBerry Plug-in for Eclipse	BlackBerry Apps World

Differences between mobiles platforms from a development point of view

The proposed solution to overcome the above issues is the native mobile application's development with the hybrid development platforms. This solution allows the code to be written once and deployed in several platforms.
Choosing from the available option becomes a major challenge to developers, first a developer is presented with two options; native and hybrid and then after that the developer is then presented with many different options to pick single or multiple out of; if native, common options are presented in the table above and if cross platform, common options are React, Ionic, Flutter. This decision point is now a huge challenge to individual developers and firms and has a downside of having less resources to give an extensive and a comprehensive information about the trend.

To assist in the possible decision making and to contribute to the existing research, native applications development and hybrid development is analysed taking new latest upgrade and key features into consideration. Also, comparing the two from existing data, analysis, research and new findings.

This research paper aims to compare native and hybrid application development on a feature level to provide scientific evidence for researchers and companies choosing application development approach as well as providing vital information about both native and hybrid applications.

The aim of this research is to explore the hybrid mobile development as an alternative to native mobile development; how can they be achieved, how can they tackle the aforementioned challenges in mobile development, and what benefits can they bring. Hence this research has been formulated for which the research work tries to find plausible answer. To put it succinctly, the work hinge on the following objectives to achieve the aim:

i. Investigating the mobile development approach that leads to cross-platform (hybrid) mobile solutions which can alleviate those mentioned challenges and problems.
ii. To compare the two development methods.
iii. To analyse existing research and to draw conclusion on why hybrid is a good alternative to native mobile application development

2. LITERATURE REVIEW

Much related work can usually be identified for an article that compares various technologies. However, if it deals with cutting-edge technology, the number of similar papers shrinks drastically. General papers on the technologies dealt with in this paper are cited in the appropriate sections. Thus, this section assesses existing work on native and cross-platform mobile app development.

The snowballing approach (ClaesWohlin, 2016) was used to find relevant and useful papers.

2.1 Review of Related Works

Even though previous studies in this subject area exists most of them focus on performance and have not taken features and functionality such as HCI and UX and Framework Comparison into account and also dated out the recent update and upgrade and also platforms. We believe that while performance is an important topic, features and functionality are almost equally important when developing.

The fundamental papers on Hybrid and native such as Heitkötter et al. (Henning Heitkötter, Sebastian Hanschke, and Tim A Majchrzak, 2012) and Corral et al. (Luis Corral, Andrea Janes, and TadasRemencius, 2012) tend to be framework comparison studies, mapping approaches, requirements and important factors. Also, newer research such as Majchrzak et al. (Tim Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli, 2017) focus on framework-level differences, but do tend to draw more from technical assessments, thus help validate the need for this very research.

We have also identified a newfound research interest of analysing data from the app stores. Such studies help form the foundation of technical baselines. For instance, Ali and Mesbah (Mohamed Ali and Ali Mesbah, 2018) answer questions such as the prevalence of Hybrid apps in the App Stores by traversing the code of 1.1 million apps, finding the Phone Gap framework to be highly popular – thus its inclusion in this research.

Mercado et al. (2016) analyse the language of more than 780,000 app reviews. Their contribution is of immense value and helps to better understand users' perception of cross-platform apps on a massive scale, thus help validate the need for this very research.

2.2 Native Applications

Native applications are built by using of the native programming language of the device which it needs to be created for. If an application is built for iOS, it must be written in Objective-C or the new language, Swift. Applications for Android uses its native language Java. Native applications provide a development environment with tools and widgets for creating desired interfaces with native user interaction experience, which are yet not there in the case of hybrid application development tools (William Jobe, 2014) (Paulo R, Adriano B, 2015).

2.2.1 Feature of Native Application Development

1. Best overall experience. Some of the typical process that native application would process are multi-touch, faster graphic APIs, fluid animation, built-in components and ease of use (Paulo R, Adriano B, 2015).
2. The native application multi-touch features makes it possible for the user to interact with the device with complex UI (User Interface) gestures. For ex. users could double tap to zoom. Pinch-spread and other advanced gestures (Bernard K and Joseph M,2015.).
3. Depending on the different device characteristic, native applications provides fast graphic API. Animation which is an essential when providing gaming experience on the device. It is also needed for highly interactive reporting and compound computational algorithms (Paulo R, Adriano B, 2015).

2.2.2 Limitations in Native Application Development

1. While native applications offer benefits in graphics, app store distribution, and device integration, their lack of portability poses significant problems for businesses.
2. Besides facing the risks of an unstable mobile-platform landscape and limited app control, native applications require large investments in terms of time and money. While native app development costs vary according to complexity, it is definitely the most expensive and time-consuming approach. For example, Forrester Research estimates that most native apps require at least six months of full-time work, and cost between $20,000 and $150,000, depending on their complexity.
3. When placed in the app store, a native application is controlled by the app store’s owner (like Apple or Google). Thus, the app-store model places companies at the mercy of a third-party vendor.
4. There could be increased maintenance costs because native apps work in a silos-based model. Since each operating system is different, updates will need to be repeated for every application to ensure its compatibility with the device.
5. Lastly, since each platform has a specified process by which applications are approved, organizations will have to go through multiple processes to ensure successful deployment of their application on each device, making the process labour intensive.

2.3 The Rise of Hybrid Applications

App-development has gone through a lot of changes since the first smart phones were released. One interesting thing that not too many people think about or even remember is that the first iPhone which was announced on January 9, 2007 did not even contain an App Store. At that time, there were no any applications, the only app-like thing you could have was a bookmarked website which gave a shortcut from the home-screen. It was not until fourteen months later, in March of 2008, that Apple introduced their app SDK and the iTunes App

Store which was a huge success. Apple's original vision for applications drew a bright line between web applications and native applications. Apple believed that native coded applications was going to yield the best end user experiences (Jeff W, 2011).
Over time a variety of platforms for smart phones were introduced, e.g. Android, Blackberry, Windows. By now it started to be complex and expensive to develop native coded applications. For every platform companies and communities had to have one team for iOS, one for Android and one team for every other platform they wanted to publish applications for. Compared to Web development this was a huge difference. In Web development one doesn't need e.g. one team for Internet Explorer, one for Chrome, one for Firefox; one simply have one (1) code to "rule them all".
During the early phases when mobile web applications came to life and became increasingly common, they were basically websites designed to run on smart phones. The design and functionality was adopted to work on smart phones, not created specifically for them. This was a new and cheaper way to develop applications for smart phones. The drawback with these applications was that the performance was not especially good to begin with. Another disadvantage with the mobile web applications were that they required a network connection.

In 2009 at iPhone DevCamp event in San Francisco, Apache Cordova was developed and went to win the People's Choice Award at O'Reilly Media's 2009 Web 2.0 Conference. Apache Cordova was originally created by Nitobi, but was in 2011 bought and rebranded as PhoneGap by Adobe Systems. PhoneGap is today one of the most popular hybrid application platforms.
A hybrid platform is basically a platform that allows you to write one code to "rule them all". The language differs depending on what platform you choose, but most hybrid platforms use JavaScript, HTML5 and CSS3. The code is written like the code for a website. Then it is simply thrown into the hybrid platform and out, you got applications to match your desires (iOS, Android, Windows, etc.). One of the big differences that made hybrid applications so popular compared to mobile web applications, were the ability to run offline, since they were no longer a "website", they were now an actual application. Hybrid applications are like a combination of mobile web applications and native applications; the main advantages are:

1. Code once, deploy on all
2. Ability to make native calls to hardware using the "Native Shell" though JavaScript
3. Offline mode, ability to run the application without internet
4. Allows a large number of users to access the application due to the multiplatform support
5. Distribution through official stores

2.3.1 Hybrid Application Development Technology

Hybrid mobile development can involve either developing the original app on a native platform (which could be iOS, Android, Windows Mobile, BlackBerry/RIM, etc.), or developing the original app in a singular environment that will then allow the app to be sent to different native platform(s). Hybrid evangelists firmly support their method, as it empowers applications to be modelled in an abstracted form and provides better user experience across multiple devices.
There are various technologies and two approaches based on development with one environment and deployment in many platforms (Cross-Platform). The design of these approaches can give a beneficial result in time and cost minimizing, as it allows developers to write with one of the languages only and use a single framework that would be translatable to many platforms. These two approaches are as follows:

2.3.1.1. Web Approach

The web approach is based on web browsers for mobile devices. Applications based on the web approach are implemented with the use of HTML, CSS and JavaScript; and rely on the browser as its runtime environment and benefit from the browser support of mobile platforms. Within this approach, the application is implemented as a single optimized website for mobile. This optimization has to take into consideration the different screen sizes of the devices and their usage philosophy.

The advantage that comes of web mobile applications is that they exist in a similar fashion across mobile web browsers on all platforms. Thus, no mobile application updates are required. The drawback of the web approach is the fact that access to the device's native functionalities (such notifications system, GPS, contact list, etc) is limited. A second disadvantage is that the time it takes to render the web pages by loading them from the network is longer than that of the native mobile user interface. Moreover, web applications are only accessible via a URL and cannot be made readily available on mobile app stores. This would have a diminishing impact on the approach's attractiveness.

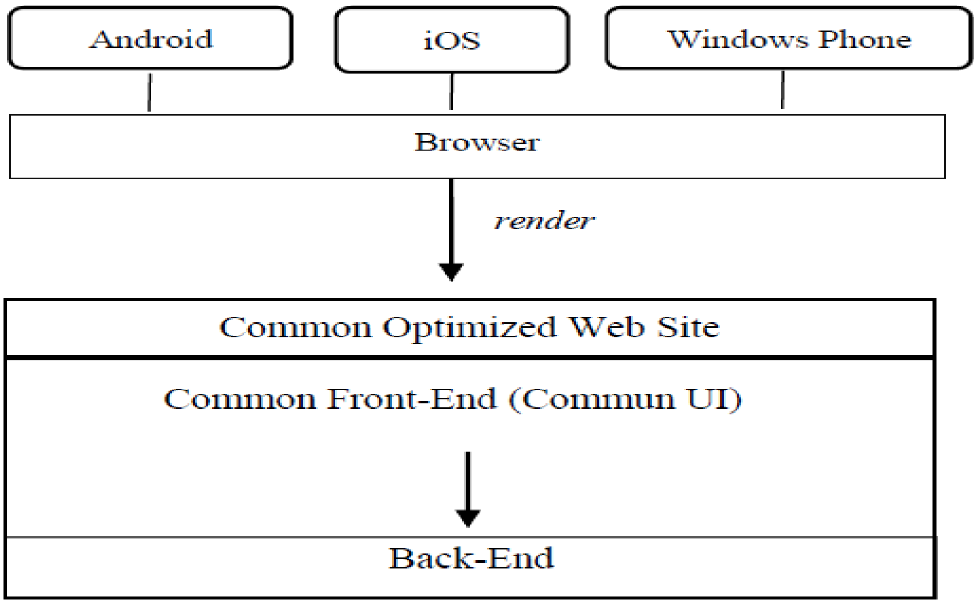


Fig.2: web approach

2.3.1.2. Hybrid Approach

The Hybrid approach used to be a combination of the advantages of web technologies and those of native functionalities. This approach uses the browser engine in the device and embeds the HTML content in the native web container (WebView in android, UI WebView in iOS). The native functionalities are accessible through the use of an abstract JavaScript bridge Fig.3.

As opposed to web applications, Hybrid applications are distributable through application stores and the native features are available through the abstract layer.

The modern hybrid application development is beyond the use of the WebView or UI WebView of android or iOS, the recent platforms such as flutter using the dart language has its own compiler that compiles the source to a binary equivalent of the native system.

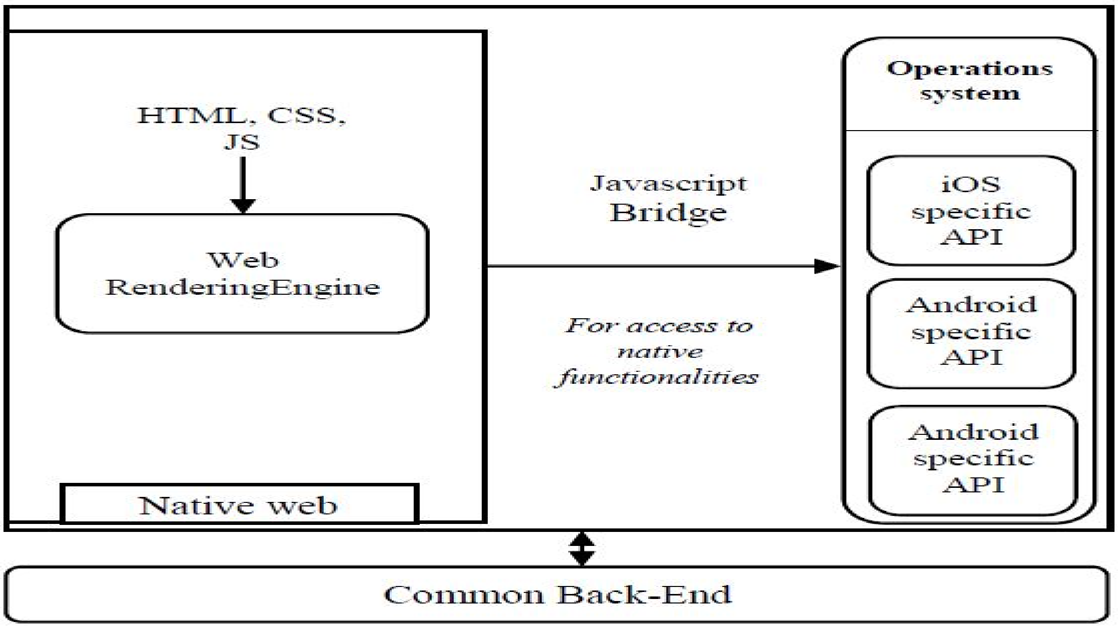


Fig.3. Hybrid Approach

i. Interpreted Hybrid Approach

Interpreted approach use common language (like JavaScript or others) to write the code of user interface and generate the equivalent for native component for each platform. The native features are provided by an abstract layer that interprets the code on runtime across different platforms to access the native APIs Fig.4.

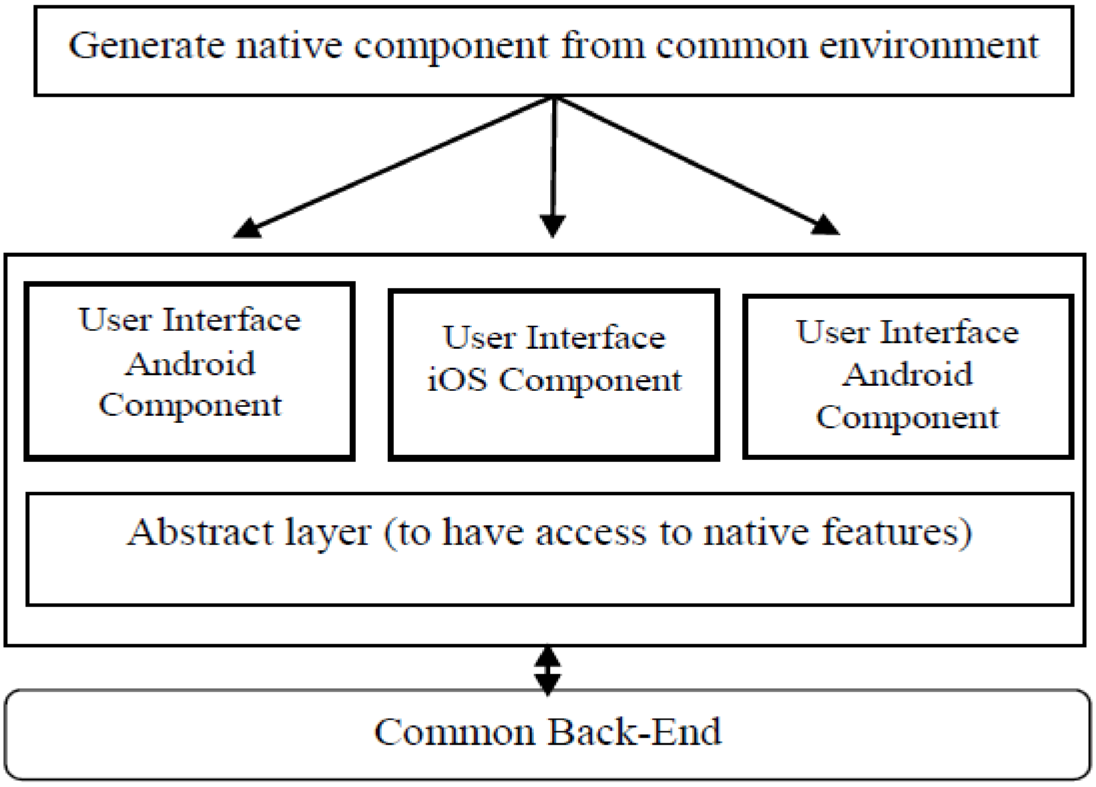

Generate native component from common environment
User Interface Android Component
User Interface iOS Component
User Interface Android Component
Abstract layer (to have access to native features)
Common Back-End

Fig.4. Interpreted Approach

The advantage of this approach is that it allows for native user interfaces. However, the downside is the dependence on the development environment. To be more exact, new platform-specific features such as new user interface features would not be made available to

applications unless they are supported by the development environment. There is also an application performance degradation that is caused by calling the abstract layer on runtime.
Appcelerator Titanium, React, ionic and Smartface App Studio are the most poplar's interpreted environment using JavaScript to write user interface code.

ii. Cross-Compiled Hybrid Approach

In the cross-compiled approach (or generated approach), developers write codes with the use of any common programming language. These codes are transformed by cross compilers to a specific native code. This method is not fully reliable and is still in the phase of development.
Even though mobile web applications support multi-platform it is in a restricted manner, since internet access is required algorithms (Paulo R, Adriano B, 2015).

2.3.2 Limitations in Hybrid Application Development

1. While cost saving can be one of the advantages of cross-platforms, the real meaning behind the term should be fully understood. A typical mobile application development process has cost overheads related to requirements gathering, analysis, and high-level design. It also has platform-specific design considerations, such as form factor, capabilities of operating systems, and hardware besides other similar concerns.
2. Pertaining to single code base, if a particular issue is found and fixed, or a new capability/feature on one platform is added, the entire suite of target applications should be retested fully. This is a serious concern. The same is applicable even if the change is required for only one platform. The fact that the code is used for all platforms introduces a mandatory overhead – i.e. to test on each and every platform every time a change is ready to be submitted. Any change for a particular platform may have unforeseen effects on an unrelated platform.
3. Application size also takes impact from the overhead of having to download the contents of the application (mainly consisting of the graphic and audio/visual components packaged in the app, as well as the code for the app). There is also the "runtime" component of the cross-platform solution, and the potential overhead introduced into the compiled code if the tool does not adjust the generated code optimally. This may, in some cases, double the size of the app to impact down-time, potential data cost and finally, end-user experience.
4. A limited user interface: hybrid apps have a design that doesn't have a native feel. The user interface thus isn't as seamless. Possibilities (3D for example) are also restricted due to the fact that Web View is used and that this doesn't allow the exploitation of the devices' full potential.
5. Many cross-platform solutions allow non-developers to generate an application using alternate skills, but issues that crop up require specialized knowledge of the underlying platform or language.
6. There is likely to be further fragmentation of mobile devices and technologies, all of which will play a huge role in escalating costs and time frames. At the same time, they will be adding to the complexity of the development process. There will also be more issues related to security, integration, and upgradation. There could also be new distribution channels, wherein developers can market their apps directly to consumers, instead of going through app stores. Social media and its power will continue to increase, and its effect will be palpable in the future mobile space. Therefore, organizations should adopt a flexible approach, for which support, scalability, and integration become factors to consider. When that happens, decisions about the right platform or approach will fall into place.

i. The supplier understands that one cannot create a perfect system for all requirements, and therefore allows the supplier to 'plug-in' specific pieces of native code to resolve certain issues the supplier knows exists.
ii. Identifying people with the right skill set to create the plug-in code.

7. Performance is a concern, and an experienced programmer will always be able to get better performance out of a specific platform when programming an app natively. Generic cross-compilers or run-time interpreters simply cannot make the same assumptions about what the app is trying to achieve.
8. Most hybrid platforms provide for a capability mechanism. By using a native plug-in, it is possible to provide access to some functionality on a particular platform which cannot be encapsulated in the system.

2.4 Comparison between Native and Hybrid

There are many different frameworks for building hybrid applications. A conscious choice of framework must be made in order to develop a hybrid application with considerations to the overall performance, and a smooth and appealing interface.
Designing the mobile interface could be achieved both using native and hybrid application. Using hybrid approach to design the mobile interface is however more flexible since the developer uses languages such as HTML5 and CSS3. Using hybrid approach to build complex and compacted application, the developer faces design problems such as slower response, and the consumption of time.
Thus, it is easier to design such applications using native approach due to the tools and design widgets provided by the native platform technologies.

1. Native applications provide the user with a better performance and smoother experience without delay compared to a hybrid application.
2. User may need to click a specific button more than once to get a response using hybrid application, which could lead to unsatisfied end-users.
3. Building applications with large animation can also create problems in the case of hybrid applications while native application gives a greater fluidity.
4. Native application is more time consuming since the same application has to be developed for different operative systems and requires knowledge in different programming languages.
   (William J, 2014,), (Felix M Kho’i and Jawed J, 2019).

Consideration	Native	Hybrid
Effort of supporting platforms and versions	High	Medium
Device Capabilities access	Full	Medium
User Experience	Full	Full
Performance	Very High	Full
Upgrade in the Client	Needed	Needed
Ease of publication/distribution	High	Medium

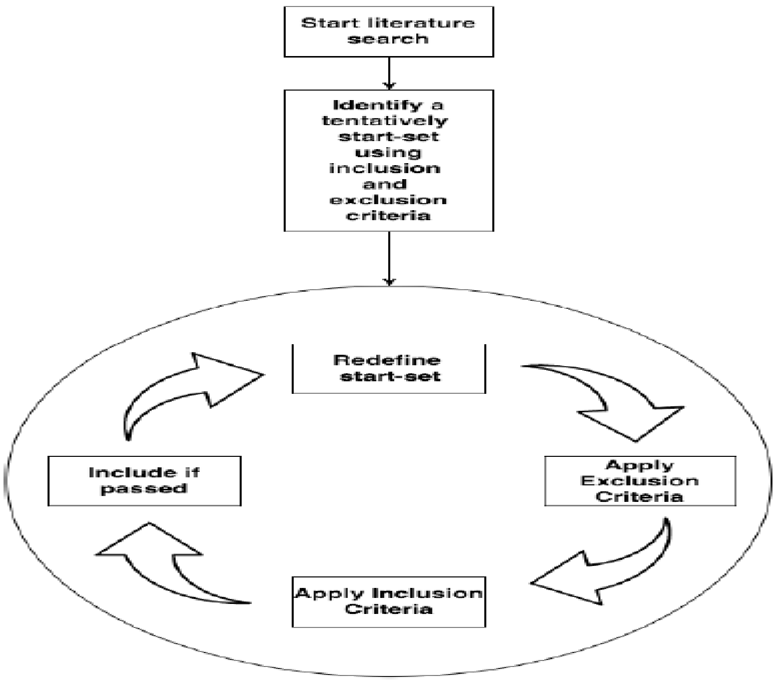
Approval cycle	Mandatory	In some cases
Monetization in app store	Available	Available

Developers with native applications skills are not only expensive and harder to come across, they are also extremely specialized in native applications. It´s rarely feasible to take a group of iOS developers right away and redeploy them on an Android project because of the highly specific platform skills; workflow and pacing creates differences that doesn't compare to web development (Jeff W, 2011).
Moreover, hybrid applications are being slower compared to native applications due to the fact that hybrid applications must be run via native container. However, some of the worlds' largest companies in the business, like Facebook, LinkedIn and Netflix have changed to hybrid approach building applications. Of course, there will be cases where native application is more suited than hybrid applications but for a very wide different content-centric applications, hybrid applications are performing well. However, building hybrid application means that the developer must rely on mobile app development frameworks and tools and its capability, provided feature to build the application. This means that if the chosen feature is not up-to-date, hybrid developer may face disadvantages and would not be able to implement the features that are not provided by the chosen framework (B. Siegfried, 2017).

3. RESEARCH MEHTODOLOGY

3.1 Data Gathering Techniques

The snowballing approach was utilized to find relevant papers and articles for this study,our start set contains eight different papers/articles. We defined both inclusion and exclusion criterions to make sure we got relevant and useful information. We customized my iterations to match our way of writing, below is a figure describing our customized iterations.



Customized Snowballing Iterations

3.2 Sources of Data Collection

Primary and secondary sources were also used to gather information for this research project. The data collection instruments issued for this study were Internet, books, related literature review, seminar documentations, online developers forums and highly rated organisation that provides statistics of usage, performance, users of various technologies (statista.com, github.io, iee.org)

3.4 Eligibility Criteria

Every research study has guidelines for who can or cannot participate and what can or cannot be included in the study.
Enrolling inclusion and exclusion criteria ensure that the results will be due to what is under study and not other factors. In this way, eligibility criteria help researchers achieve accurate and meaningful results.

Inclusion Criteria

Language: English
Timeframe: 2014-2020
Title: Is the title relevant to the study?

i. Yes, review Abstract, Questions, Keywords and Conclusion. Does it answer any of our questions, or contain relevant information.
ii. No, exclude the paper

Exclusion Criteria

Non-peer reviewed
We began the literature study with a starting set of six different papers. These papers where defined as relevant and useful using the Inclusion and Exclusion criteria.
In the article "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering", ClaesWohlin described what the characteristics for a good start set is. In this study we chose to interpret these characteristics as:

i. If the relevant papers may come from different communities, then it is important to have these covered in the start set. The reason is that the papers may be in independent clusters, i.e. in clusters of papers not referring to each other.
ii. It is important to get the right amount of papers in the start set. The number differs depending on the area of study, in a smaller, more focused area the amount requires fewer papers than a bigger area of study.
iii. If the search result of papers is too large, for example due to having to general search terms, then an alternative is to identify a number of relevant and highly cited papers.
iv. The start set should include different publishers, years and authors, i.e. diversity.
v. The start set ought to be formulated from keywords in the research questions, preferably also take synonyms into account.

4. ANALYSIS OF RESULTS AND FINDINGS

This section contains the research perspectives and analysis of result which include the comparison of the two development approaches and more details based on the newest technologies and updates in these two approaches. The section discuses the native application development and hybrid application development based on the latest updates and upgrades which former publications covers less or non, giving detailed explanation, features, and limitation of these two development platforms and then gives comparison of the two platforms and finally providing findings from the research work.

4.1 Native Application Development

'Native' is a term used for software development in which the developer uses the main language, tools, and a framework for the platform being targeted, while using an Integrated Development Environment (IDE).
Native apps are typically built using development tools and languages (XCode and Objective-C for iOS apps, Eclipse, Android Studio; Java for Android, Visual Studio; and C# for Windows) that the respective platforms support, and they run only on those platforms. Since native apps are written for specific platforms, they can interact with and take advantage of operating system features and the other software programs installed on the platforms.

4.1.2 Features of Native Application Development

i. Multi-touch – double taps, pinch-spread, and other compound User Interface (UI) gestures
ii. Fast graphics API – extremely speedy graphics
iii. Fluid animation – crucial in gaming, highly interactive reporting, or intensely computational algorithms for transforming photos and sounds
iv. Interaction – interacts with other apps and provides for widgets on the homepage, also, can respond to hard keys, i.e. the Android's search button and volume control
v. Documentation – there are nearly 3,000 books on iOS and Android development, along with several online articles, blog posts, and technical threads

As Reviewed from existing works in chapter two the limitation of native application development did not change even with the inclusion of the latest feature as compared to the hybrid approach. Despite regular updates, native applications development for various devices has not changed from previous approach but rather makes the existing functionality do better and adaptable to new technologies and approaches, this makes the system more reliable and maintains its existing advantages and, in some cases, creates more and the downfall, its limitations almost the same.

4.2 Hybrid Application Development

Hybrid application development empowers the developer to create an application using a single language or tool set, and instantly deploy it across a variety of platforms. In general, any program that can run on more than one device with different systems is a cross-platform program and hybrid applications is one approach of development cross-platform application, leaving web approach as the other.

4.2.1 Hybrid Application Development Technology

As reviewed from previous works in chapter two hybrid application development technology is still evolving and one of the types of hybrid approach which previous work state as "still in development phase" is now fully functional and has change the face of hybrid application development. This approach is explained as follows;

4.2.1.1. Cross-Compiled Hybrid Approach

In the cross-compiled approach (or generated approach), developers write codes with the use of any common programming language. These codes are transformed by cross compilers to a specific native code Fig.5.

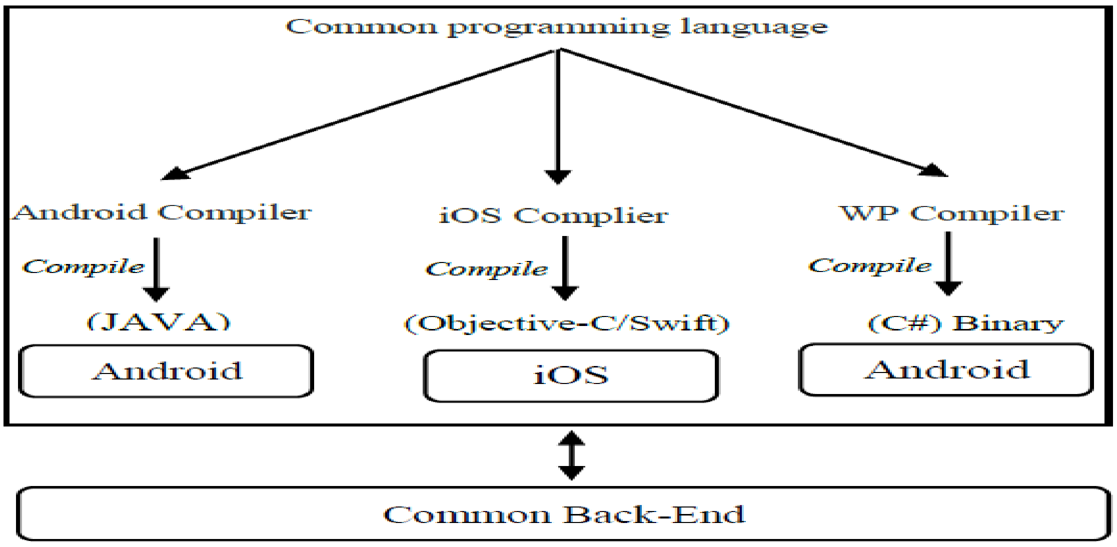


Fig.5: Cross-compiled Approach

The main benefit of this approach is that the applications are able to attain native performance and deliver all the features of native applications along with its native interface components.

There exist two powerful platforms based on this approach; the first one is Flutter which uses Dart shared codebase, the second is Xamarin, which uses C# shared codebase and lastly React-Native. Developers can use Xamarin tools to write native Android, iOS, and Windows apps with a shared code across a number of platforms and native user interfaces. However, Xamarin requires writing a specific code in order to benefit from the platform specific features; contrary to the Flutter and React that also uses Code-Name. Code-Name presents more advantages by having a free and open source version, which is more generous than the free Xamarin version (Nfaoui, Es-Sbai and Abdellah, 2019).

4.2.2 Features Hybrid Application Development

i. Familiar languages/relatively simpler language – apps can be created with HTML, CSS, and JavaScript. Simpler language C#, Dart.
ii. Integration – integrates with existing JSP and .NET infrastructure
iii. Single code for building mobile apps for iPhone and Android platforms
iv. Advanced capabilities – leverages features such as GPS, camera, etc.
v. Flexible – applications adapt to different resolutions, screen sizes, aspect ratios, and orientations
vi. Multiple devices – can be used to build for desktop, tablet, and mobile web devices
vii. Single-page architecture – generates self-contained web apps that execute locally on the device.
viii. Hybrid application simplifies marketing by enabling the use of multiple media with generalized messages targeting potential customers.

4.2.3 Limitations Hybrid Application Development

i. While cost saving can be one of the advantages of cross-platforms, the real meaning behind the term should be fully understood. A typical mobile application development process has cost overheads related to requirements gathering, analysis, and high-level design. It also has platform-specific design considerations, such as form factor, capabilities of operating systems, and hardware besides other similar concerns.

ii. Pertaining to single code base, if a particular issue is found and fixed, or a new capability/feature on one platform is added, the entire suite of target applications should be retested fully. This is a serious concern. The same is applicable even if the change is required for only one platform. The fact that the code is used for all platforms introduces a mandatory overhead – i.e. to test on each and every platform every time a change is ready to be submitted. Any change for a particular platform may have unforeseen effects on an unrelated platform.

4.3 Comparisons

From the collection and analysis of existing data, the two key elements of our research are compared with each other, giving both the pros and cons of each element and discussing their major features in detail.

Consideration	**Native**	**Cross-Platform**
Multiple OS Support	No	Yes
User Interface Quality	High	Medium to High
Performance	High	High
Cost of Ownership	High	Medium
Application Updates	Native Market	Native Market
Application Maintenance	High	Medium
Development Languages	Java, C, C++, Objective C, Objective C++	Dart, Java,HTML,CSS, JavaScript,

Comparisons Between Native and Mobile Platforms

4.4 Results

After reviewing relevant publications, articles and research works and also performing an investigative research on the latest development in the areas that covers native and hybrid application development but the once that are published and those that are in use but too recent to have any research publication and also, comparing the two platforms by the publications and from the publications the following results were realised:

i. Performance: with the introduction of the new base languages that are supported by some devices and also the introduction of the cross-compiler, the performance of hybrid application has skyrocketed to be as much as that of native.
ii. UI: hybrid application now does not necessarily needs the web view in androids or the UI web view for iOS. It compiles to the native UI equivalent and render it on the device.
iii. Access Control: Hybrid applications has now covered the problem of not having access to most functionality of device such as camera, GPS, Hardware Accelerator, OS. That is now covered, hybrid application development platforms such as flutter, React Native, Xamarin can now have access to functionality that native does.
iv. Design: Hybrid application development platforms offers more flexibility and design properties providing for more dynamic and beautiful designs.
v. Offline: hybrid applications now provide almost all functionality that native applications provides.

4.4 Findings
As time goes the disadvantages of hybrid application is becoming it advantages and hybrid application development is evolving rapidly and will continue to evolve providing more feature and functionality that covers both what native can do and even more.
Missing features of hybrid application such as; hardware access, Gps optimizations, OS interactive access, Encryption, Security, Utility access and optimization are now covered and are been done by hybrid platforms efficiently.
Hybrid application platforms are becoming closer and closer to native and with the introduction of the dart language by google and the flutter platform, hybrid applications produced are in all ways equivalent to native application and flutter having its own language and doesn't run on the web view of it target device and also having its own compilers means a big break through for the hybrid community.
All a developer need is to have a test of hybrid development and falling in love with it is an unforeseen event that will come to pass. More and more communities and forums are being created and they keep getting bigger and more interesting and from developers' point of view, the moment flutter and react native breaks out the love for mobile application development in general has increased drastically.
Even though hybrid application development comes with all it ever increasing feature from all angles of this research and previous once, hybrid application development will not replace native application development but rather might even work hand in hand to provide more quality and advance applications for mobile devices.

5. SUMMARY, RECOMMENDATOIN AND CONCLUSION

5.1 Summary of Result
As apps continue to play an important role in the business world, developers and organizations struggle to find the best development approach, but most of them realize that both approaches have their advantages and disadvantages.
There is likely to be further fragmentation of mobile devices and technologies, all of which will play a huge role in escalating costs and time frames. At the same time, they will be adding to the complexity of the development process. There will also be more issues related to security, integration, and upgradation. There could also be new distribution channels, wherein developers can market their apps directly to consumers, instead of going through app stores. Social media and its power will continue to increase, and its effect will be palpable in the future mobile space. Therefore, organizations should adopt a flexible approach, for which support, scalability, and integration become factors to consider. When that happens, decisions about the right platform or approach will fall into place.

5.2 Recommendation
As this study shows that Hybrid application is not just another way of developing mobile application but also provides efficient and quality features that native approach canot provide and even more, we recommend the following;

i. To companies, developers and students that want to develop or learn how to develop not just for hobby but also has a target and a customer -base, adapting hybrid application development will increase quality, productivity and reduce cost.
ii. Keeping up with latest technology and new upgrades is a key feature that every developer and organisation should have and this feature should be a key element for every hybrid application development.
iii. Having knowledge of and using it, are two different things that can comfortably be separated in this instance. Due to fragmentation there are many approach to solving a

mobile application development problem and having a good knowledge of them all is very important but using them all is not necessary and so, even if native has been around and is not doing bad at all for a while and hybrid is now doing better individual developers should try to learn more than one approach and not necessarily use all and organisations should be able to handle all approaches.

5.3 Conclusion

Although native apps benefit from an optimal integration into the respective mobile operating system and good developer support, the analysis showed that cross-platform approaches are a viable alternative. As soon as mobile apps must be developed for multiple platforms under tight budgets, with small developer teams, and in a short time frame, a cross-platform approach is necessary. However, these approaches are more than a second-best alternative. Developers might prefer using a cross-platform solution even in the absence of these constraints.

Cross-Platform apps constitute an ideal starting point for new developers and a focus for native developers, because they do not require advanced knowledge and enable developers to start implementing the app right away. Cross-Platform apps are a simple approach benefiting from very good support by devices on all platforms. Furthermore, they can be easily ported to other cross-platform approaches.

However, the results of our evaluation are only general guidelines that can be adapted and interpreted for each project individually. The Result can be used to support decisions, for example in semi-formal multi-criteria decision methods. Basic decision support can be obtained by weighing the criteria according to the requirements of a given project and calculating a weighted grade, carefully interpreted and analysed for sensitivity. The result might yield first insights on which solution best matches the requirements at hand.

REFERENCE

Anmol K, Rashmi G, B. Sindhya, May 2015. “An Introduction to Hybrid Platform MobileApplication Development” Accessed 2019-12-08 http://research.ijcaonline.org/volume118/number15/pxc3903463.pdf

B. Siegfried, "Enhanced Student Technology Support with Cross-Platform Mobile Apps," in 39th annual ACM Special Interest Group on University and College Computing Services, San Diego, California, 2017.

Behrens, H.: Cross-Platform App Development for iPhone, Android & Co. (2010), http://heikobehrens.net/2010/10/11/cross-platform-app-development-for-iphone-android-co-%E2%80%94-a-comparison-i-presentedat-mobiletechcon-2010/

Bernard Kohan and Joseph Montanez,January 2015. “Native vs Hybrid / PhoneGap App Development Comparison” Accessed 2019-11-03 http://www.comentum.com/phonegap-vs-native-app-development.html

ClaesWohlin, "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering” Accessed 2019-11-18 http://www.wohlin.eu/ease14.pdf

El Habib Nfaoui, Najia Es-Sbai and Sidi Mohamed Ben Abdellah Conference Paper, March 2019, Cross platform approach for mobile application development: A survey https://www.researchgate.net/publication/304371091

Felix MohammadiKho’i and Jawed Jahid, April 2019. “Comparing Native and Hybrid Applications with focus on Features” Accessed 2019-12-08

Henning Heitkötter, Sebastian Hanschke, and Tim A Majchrzak. “Comparing crossplatform development approaches for mobile applications.” In Proceedings 8th WEBIST, pages 299–311. SciTePress, April 2052.

Iván Tactuk Mercado, NuthanMunaiah, and Andrew Meneely. The impact of cross-platform development approaches for mobile applications from the user’s perspective. In Proceedings of the International Workshop on App Market Analytics, WAMA 2016, pages 43–49, New York, NY, USA, 14 November 2016. ACM.

Jeff Whatcott, 30 Nov 2011, “HTML5 and the Rise of Hybrid Applications”. Accessed 2019-12-18 https://blog.brightcove.com/en/2011/11/html5-and-rise-hybrid-applications

JQuery Mobile Accessed 2019-12-18 https://jquerymobile.com/

L’ubošstaráˇcek and Valentino vrani´c. 2017 “MDA Based Multiplatform Mobile Application Modeling with Platform Compliant User Interfaces”.

Luis Corral, Andrea Janes, and TadasRemencius. Potential advantages and disadvantages of multiplatform development Frameworks–A vision on mobile environments. In Procedia Computer Science, volume 10, pages 1202–1207. SciVerse ScienceDirect, 9 August 2012.

Mohamed Ali and Ali Mesbah. Mining and characterizing hybrid apps. In Proceedings of the International Workshop on App Market Analytics, WAMA 2016, pages 50–56, New York, NY, USA, 2016. ACM.

Native vs Hybrid / PhoneGap App Development Comparison, January 2015. Accessed 2016-04-03 http://www.comentum.com/phonegap-vs-native-app-development.html

Paulo R, Adriano B, February 2015. “Cross Platform App A Comparative Study” Accessed 2019-12-08 http://uv3sv3ds3g.search.serialssolutions.com/?ctx_ver=Z39.882004&ctx_enc=info%3Aofi%2Fenc%3AUTF8&rfr_id=info:sid/summon.serialssolutions.com&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.atitle=CROSS+PLATFORM+APP+A+COMPARATIVE+STUDY&rft.jtitle=International+Journal+of+Computer+Science+%26+Information+Technology&rft.au=Paulo+R.+M.+de+Andrade&rft.au=Adriano+B.+Albuquerque&rft.date=20150201&rft.issn=09753826&rft.eissn=09753826&rft.volume=7&rft.issue=1&rft.spage=33&rft.epage=40&rft_id=info:doi/10.5121%2Fijcsit.2015.7104&rft.externalDBID=DA&rft.externalDocID=oai_doaj_org_article_0a54a17ec193467d9c4b66c4d98f1d9&paramdict=en-US

Tim Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli. Comprehensive analysis of innovative Cross-Platform app development frameworks. In Proceedings of the 50th Hawaii International Conference on System Sciences, pages 6162–6171. scholarspace.manoa.hawaii.edu, 2017.

William Jobe, Nov 12, 2014, “Native Applications vs. Mobile Web Applications”. Accessed 2019-11-08. https://www.researchgate.net/profile/William_Jobe/publication/268153001_Native_Applications_Vs._Mobile_Web_Applications/links/546346b30cf2cb7e9da765c3.pdf