American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

http://asrjetsjournal.org/

Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development

Oluwatofunmi Adetunji[a]*, Chigozirim Ajaegbu[b], Nzechukwu Otuneme[c], Olawale J. Omotosho[d]

[a]*Department of Software Engineering, Babcock University, Ilishan-Remo, Ogun State, Nigeria*

[b,d]*Department of Computer Science, Babcock University, Ilishan-Remo, Ogun State, Nigeria*

[c]*Department of Computer Science, Wesley University, Ondo, Ondo State, Nigeria*

[a]*Email: adetunjio@babcock.edu.ng*

[b]*Email: ajaegbuc@babcock.edu.ng*

[c]*Email: nzechukwu.otuneme@wesleyuni.edu.ng*

[d]*Email: ojomotosho@gmail.com*

Abstract

Over the years, there has been a constant increase in the demand for mobile software due to the constant increase in the number of smart phones. Mobile developers have the liberty to adopt different development architectures or strategies which includes the native app, mobile web app, hybrid app and the new Progressive Web App (PWA). PWA which combines the features of the native and web development strategies emerged as a better alternative to other development approaches due to additional benefits such as offline capability, background synchronization and so on despite several concerns that have been raised towards the efficiency of PWAs. Hence, this research work aims at performing a comparative study on the existing mobile development architectures using the Systematic Literature Review (SLR) technique, performing feature comparison on the native, hybrid and PWA architecture and finally argues for the PWA development architecture based on the comparisons. The comparison will aid researchers and development firm in understanding the concept of PWA thereby motivating them to adopt this strategy for further development.

Keywords: Progressive Web Apps; Mobile Application Development; Native Apps; Hybrid Apps.

---

* Corresponding author.

1. Introduction

In recent years there has been a constant increase in the number of mobile devices and its users. As at mid-2019, the world's population has reached 7.7 billion (United Nations, 2019) which has in turn affected the number of mobile users. Mobile technology has evolved rapidly over the last decade which has made more than five (5) billion people possess a mobile device in which 57 percent of mobile devices are smart phones [1,2]. These statistics shows the constant increase in the affinity of people towards mobile devices especially smart phones. Therefore, it is the imperative to satisfy the needs of the increasing number of smart phone users by constantly developing applications (apps) that spans through different sectors of life ranging from education to health to entertainment and so on. This has given mobile applications a different nomenclature such as mEducation, mHealth, mGoverntment, mEntertainment and so on in order to completely differentiate it from other form of applications. Different smart phone vendors adopt a particular mobile platform such as android, windows, iOS, blackberry, Symbian and so on [3] upon which mobile applications are built. Broadly, mobile application architecture can be divided into the native app which is completely dependent on a mobile device platform, mobile web app which makes use of web technologies such as Hypertext Markup Language (HTML), Cascading Style Sheet (CSS) and JavaScript providing more flexibility for mobile development across platforms and the Hybrid architecture which harnesses the pros of both the native and mobile architecture [4,5]. The highlighted architectures have one form of limitation or the other which will be discussed in detail in section 3. These identified limitations brought about the PWA architecture which was developed by google[a]. PWA is an emerging technology that has been embraced by some mobile developers in the industry, however, due to existing applications that has been developed over the years using the native, mobile web and hybrid architecture, doubts have been raised about the need, success and acceptability of PWA. This research work aims at performing a comparative study on the existing mobile development architectures using the Systematic Literature Review (SLR) technique. Feature analysis and comparison on the native, hybrid and PWA architecture will be carried out and finally an architecture will be recommended based on the comparison. Section 2 provides supporting texts from existing literatures, section 3 provides a broad discussion on the traditional mobile development architectures as well as the emerging PWA. Section 4 provides a comparison and analysis of features of the mobile development strategies as well as a recommendation based on comparison while section 5 concludes this research work.

2. Literature Review

PWA is an emerging technology that is gradually gaining academic involvement in terms of research. This is evident from the handful of research articles as regards PWA across various academic search engines which will be duly reviewed. Mobile software development team or organizations can adopt one or more of the existing development strategies ranging from native apps to mobile web apps, hybrid apps and now the PWA. Native app development strategy happened to be the first that existed, it consists if binary executable files that are directly downloaded and stored in to a user's mobile device [6]. Apps developed using this architecture is platform dependent and is solely distributed via a dedicated app store (Google Play Store, Apple App Store,

[a] https://developers.google.com/web/progressive-web-apps

BlackBerry App World) depending on the platform adapted by the mobile device vendors. [7] identified high development time, high testing and maintenance cost as a major challenge of the native app, [6] called this a challenge of mobile fragmentation which implies that a code written for one mobile platform (for example, java codes for android app) cannot be used for another platform such as Apple iOS app which is written in Objective-C. in an attempt to overcome the challenges of the native app where each platform has its own Software Development Kit (SDK) with different development capabilities, several cross platform architectures were developed which allows deployment of mobile solutions using a single SDK. A survey of several cross-platform approaches was carried out by [8] while [3] discussed the taxonomy of these cross-platform approaches. These approaches identified are the web approach which are used in developing mobile applications using web technologies (HTML, CSS and JavaScript) hosted on a remote server thereby making it platform independent because the mobile-optimized website/app are accessed via a browser app such as Chrome, Firefox or Safari which must be pre-installed on user's mobile devices [7,9]. A major challenge of this approach is that apps are only accessed via a Uniform Resource Locator (URL) using a reliable and constant internet connections which implies that apps cannot be downloaded via various app stores. The hybrid approach according to [3,8] tried harnessing the advantages of the native and web architecture. In the hybrid approach, mobile solutions are developed using the web technologies but rendered inside the native apps and are distributed via various app stores. Other approaches discussed were the interpreted approach which uses a common programing language such as JavaScript to write a code which in turn generates the equivalence for the native component for each platform, the cross-compile approach which enables developers write codes using any common programming language which are then transformed by cross compilers to a specific native code. To overcome the challenges posed by the various mobile development approaches (architectures) as identified by the above researchers, another development approach known as Progressive Web App (PWA) as coined by [10] was developed. Reference [11] provided a general introduction to the concept and technologies behind PWA by showcasing some major features and providing technical comparison alongside existing mobile development architectures. Biørn-Hansen and his team performed a measurement-comparison of the size of installation, launch time and time from app-icon tap tool bar render among the hybrid, interpreted and PWA mobile development approach. The result showed that PWA had the least size of installation as well as the smallest launch time but has the highest time from app-icon tap tool bar render. To further elaborate the general concept and technology of PWA, [12] discussed the architectural pattern on which the PWA is based that is responsible for the improved loading time of mobile apps. An assessment of PWA was carried out using the Analytical Hierarchy Process (AHP) by [5] to decide the architecture type that suits the development of a mobile app. The assessment was done on the major features (application size, multi-platform supports, offline accessibility) across four types of mobile development architecture, the result showed that PWA has more weighted score over others. The background operation of the service workers in PWA might make mobile app developers and users think it has an adverse effect(s) on the battery life (energy) which is one of the scarcest resources of a mobile device. To nullify such assumption, Reference [13] assessed the impact of service workers on the energy efficiency of PWAs by carrying out an empirical experiment on seven (7) existing PWAs using two (2) devices (low and high-end devices) over a 2G and Wi-Fi network. The result showed that the service workers have no significant impact over the energy consumption on both devices irrespective of the network conditions. However, the load times if PWAs as regards to its counterparts was not evaluated. Also, the assumption that the caching of

contents by the service workers might reduce the performance of PWAs was nullified by [14] in the analysis of the cache component in the service workers in comparison to other mobile development pattern, the google lighthouse (beta) was used to prove that the performance of a PWA is better than its counterpart – native app (android) due to the caching process embedded in it. However, there was no result showing the performance of the iOS counterpart. A PWA monitoring system for smart farming was developed by [15]. This application was tested using oil palm farm at Indonesia which allowed field employee of the palm plantation to send reports about the farm to the supervisor who resides in the office (a different location) irrespective of the network condition. The developed application was subjected to a black box testing using the google light house. This application however was not bench marked against any counterpart (android, iOS), the application also made use of an existing Application Programming Interface (API) from a previous research which brought about conformity issues between the existing API and the specified user interface. Based on the reviews, it is evident that different mobile architecture can be adopted for the development of mobile solutions with each having their pros and cons. An attempt to leverage on the advantages of these architectures brought about the PWA of which analysis has been carried out on some of its major components.

3. Mobile Application Development Architecture

Mobile applications commonly referred to as an app are software programs developed and optimize for mobile devices such as smart phones and tablets [16]. Mobile apps are like the traditional software application but have some distinct features that distinguishes it from regular apps. [17] identified requirements that clearly distinguishes a mobile apps from traditional apps some of which are:

1. **Potential Interactions with Other Applications:** This mean that mobile devices might have numerous apps from different sources which likely interacts with other applications residing in the device.
2. **Sensor Handling:** Mobile applications can access several sensors local to a mobile device such as accelerometer, GPS, microphone, cameras and so on.
3. **Families of Hardware and Software Platforms:** There are different mobile platforms which might require developers to build several apps for different platforms.
4. **Security:** Mobile platforms are vulnerable to attacks because they are ‘open’ which can allow the installation of new malware applications that can affect the overall operation of the device.
5. **User Interface:** Mobile apps cannot be designed in a singular manner due to the fact that mobile devices come in different sizes and shapes.
6. **Complexity of Testing:** This is a difficult task as a simple application need to be tested on several devices as well as under different network conditions this is so because the development platform is not the same as where the application will be used.
7. **Power Consumption:** Software must be optimized to maximize battery life.

Table 1 shows a detailed difference between mobile development platforms. Despite the differences across platforms, the uniqueness of each platform – specific API, tools and technologies enable developers to create apps with good user experience and increased performance [7,9].

Table 1: Difference between four mobile platforms

Platforms	Virtual Machine (VM)	Programming Language	Integrated Development Environment (IDE)	User Interface	Devices	Application Store
Android (Google)	Dalvik VM	Java	Eclipse, Android Studio, Android SDK	XML files	Heterogenous	Google Paly Store
IOS (Apple)	No	Objective-C or Swift	XCode	Cocoa Touch	Homogenous	Apple iTunes Store
Windows (Microsoft)	Common Language Runtime (CLR)	C# or C++	Visual Studio	XAML files	Homogenous	Windows Phone Market
Blackberry OS (Research in Motion – Rim)	BlackBerry Enterprise Server VM	Java	BlackBerry Plug-in for Eclipse	XML files	Homogenous	BlackBerry Apps World

There are four (4) ways in which mobile app can be developed leading to four different types of apps which are native app, mobile web app, hybrid app and the emerging PWA. A comparative study on the various development approaches will be carried out based on a SLR.

3.1. Native Applications

These are apps developed using tools and programming languages dedicated for a certain mobile platform [3]. Native applications are platform dependent hence programmers must conform to the specific languages and tools needed to successfully develop the app. A major disadvantage to this development approach is mobile platform fragmentation as identified by [13] – meaning that for a development firm to reach more audience across varying platforms, there must be the 're-development' of the same app across different technologies and tools specific to each desired platform. This leads to an increase in development time, development cost, effort, maintenance cost and low portability. Figure 1 diagrammatically shows the approach of native mobile development.

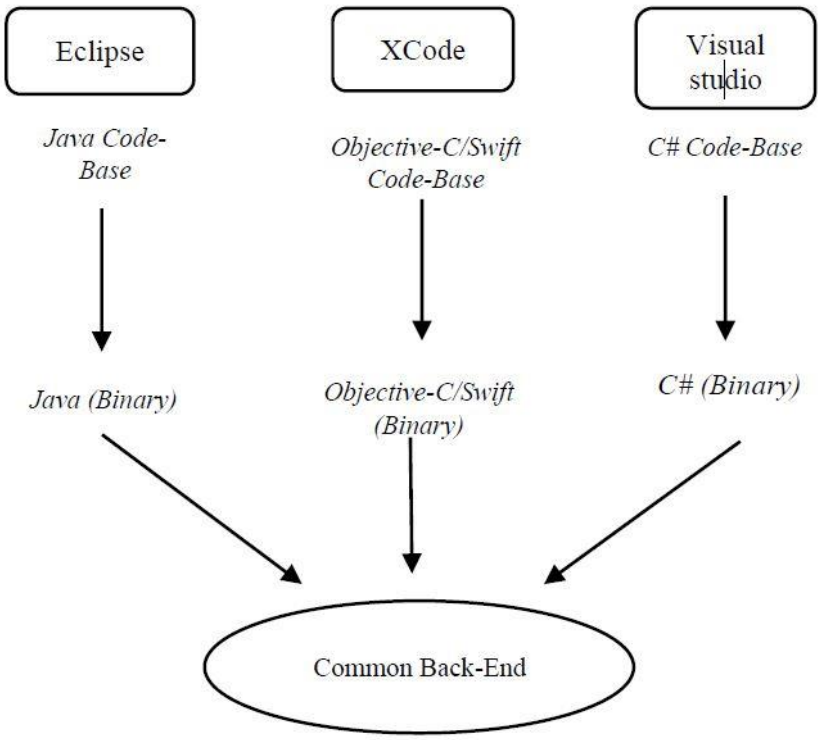

Figure 1: Mobile Native Development Approach [8].

Highlighted below are strengths and weaknesses of the Native apps.

Strengths

1. Native apps have full access to mobile device features and sensors.
2. There is a native look and feel of the user interface.
3. They are easily accessed via a dedicated app store.
4. They have higher performance than web apps [3].

Weaknesses

1. Development languages and tools are platform specific – the same app needs to be developed for each platform.
2. There is high development time.
3. Application testing is done across mobile devices which leads to high testing and maintenance cost.
4. They are difficult to develop which means prospective developers must have high level of experience [18].

3.2. Mobile Web Applications

These are mobile optimized web apps developed based on web technologies such as HTML, CSS and JavaScript. They are hosted on remote servers and are accessed using specific URL via web browsers installed

on a user mobile device [3,7,9]. This makes the mobile web platform independent because the web browser serves as its runtime environment. This approach enforces optimization of web application such as taking into consideration the screen sizes of various devices as well as their usage philosophy. Figure 2 diagrammatically shows the web approach of mobile development as amended from [8]. Mobile web apps adopt the client-server model where a service requester (client) makes certain calls or request to a service provider (server) which in turn respond to the request of the client. The back and forth **`1**communication is handled by an application level protocol (HTTP).

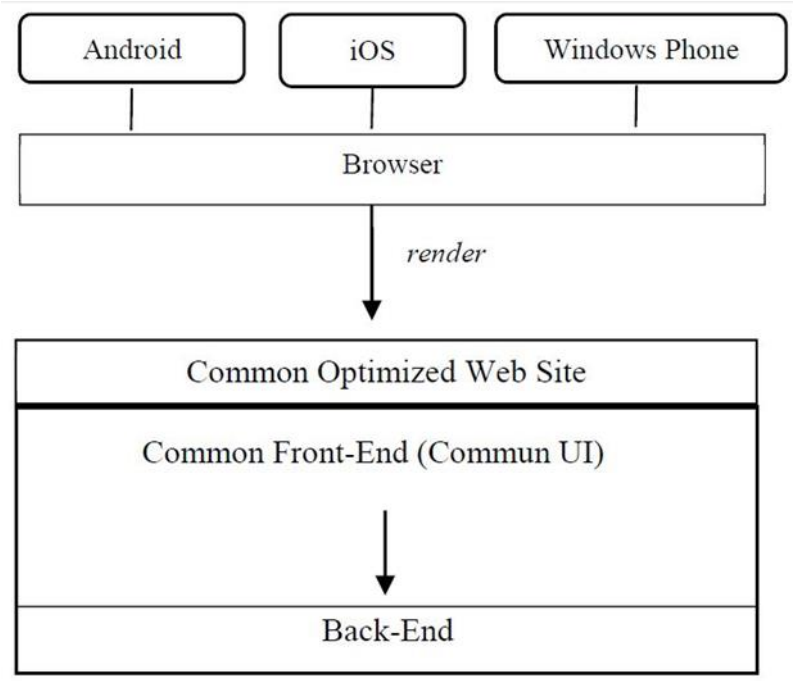

Figure 2: Mobile Web Development Approach [8]

The strengths and weaknesses of the mobile web development approach are stated below.

Strengths

1. Web app provides uniform experience to users across all platforms.
2. No mobile application update is required [8].
3. Easy to learn and develop using web technologies [3].
4. App development is done once and can run on any platform
5. No form of processing is done on the user's device – processing takes place on the server.
6. It has a fast development time compared to the native approach.
7. It is more portable than native apps.

Weaknesses

1. They have limited access to the device native or low-level features and functionalities [7].

2. Rendering of user interfaces is dependent on the available internet connection
3. Web apps are only accessible via URLs and not a dedicated app store.
4. It has a lesser performance compared to native apps due to the HTML and JavaScript that are parsed and implemented through web browsers [19].

3.3. Hybrid Approach

The approach tries to herness the benefits of both the native and web approach thereby overcoming some limitations posed by both approaches. Applications developed using the hybrid approach uses the browser engine in the mobile device and embeds the HTML content in the native web container (for example, WebView for android, UIWebView for iOS) [8]. The provision of certain mobile hybrid development frameworks such as Cordova, Ionic, PhoneGap, MoSync provides a native wrapper that contains the web-based codes and also a generic JavaScript API that serves as a bridge of the service request from the web-based code to corresponding platform's API [7]. Figure 3 shows a diagrammatic view of the Hybrid approach.

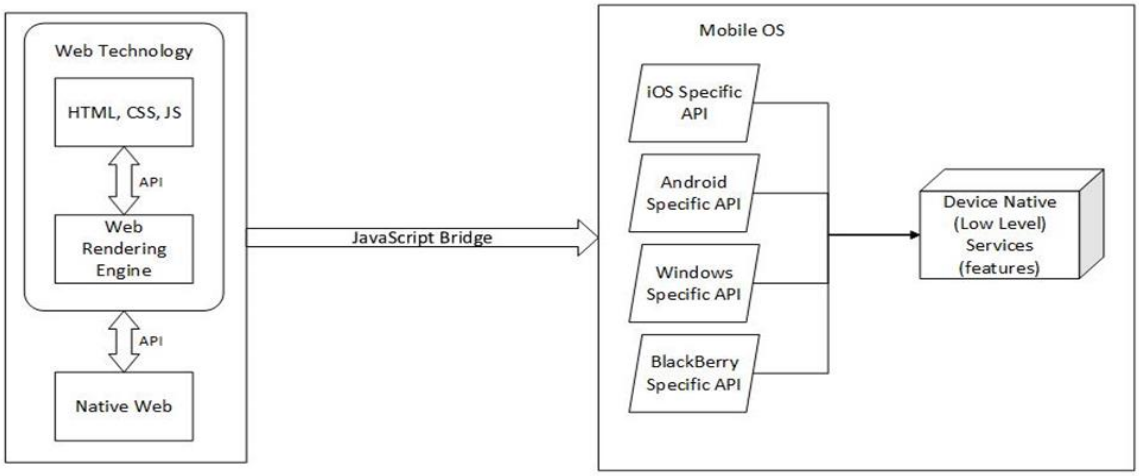

Figure 3: Diagrammatic view of the Hybrid Mobile Development Approach [8]

The strengths and weaknesses of the approach are discussed below:

Strengths

1. Hybrid apps are distributable through dedicated Appstore as opposed to web apps.
2. Hybrid apps can be packaged and distributed to any supported platform [20].
3. Development process is simplified because a single code base maintained for all platforms
4. Hybrid apps can be adopted for both server backend and standalone apps.
5. Hybrid apps can access device native features of mobile devices.

Weaknesses

1. Hybrid UI are inferior in performance when compared to its native counterparts due to the fact that execution happens in the browser engine [8].
2. The existence of JavaScript bridge imposes an additional overhead in performance when accessing the device specific platform API [7].
3. User experience provided by hybrid app is the same across all platforms which might not fit or integrate into various mobile device structure style as some devices have a physical back button while the back button of some phones is managed on the screen.
4. Hybrid apps are most time dependent on internet connections.
5. The hybrid app is limited to what the JavaScript bridge is capable of translating [9].

As discussed, the native, web and hybrid application development have different strengths and weaknesses of which several researchers have argued for or against a particular approach. [21] pointed out a trade-off in terms of performance and user experience between web app development (which are seen as a cheaper alternative) as compared to native development. The hybrid leverage on the advantages of both the web app and native app. The fourth and emerging approach to developing mobile app is the PWA. This will be discussed in section 3.4 as it is the main focus of this research work.

3.4. Progressive Web Application (PWA)

PWA is a mobile development approach that seek to overcome the challenges or weaknesses of earlier approaches. Adopting this approach produces special kind of web apps which requires no installation before using and is served from a remote server via a secured Hypertext Transfer Protocol (HTTPS) unlike regular mobile web apps which might be served using the HTTP [7,12,22]. User of PWA are provided with a native app like experience by promoting the PWA to a top-level mobile app with a full screen support (no browser) after deciding to install the PWA on the user's device [7]. The PWA is based on the concepts of a single application for all platforms [5] just like the hybrid approach. However, it possesses distinct capabilities such as instant loading, push notification even in the offline state. Figure 5 diagrammatically shows the PWA development approach.

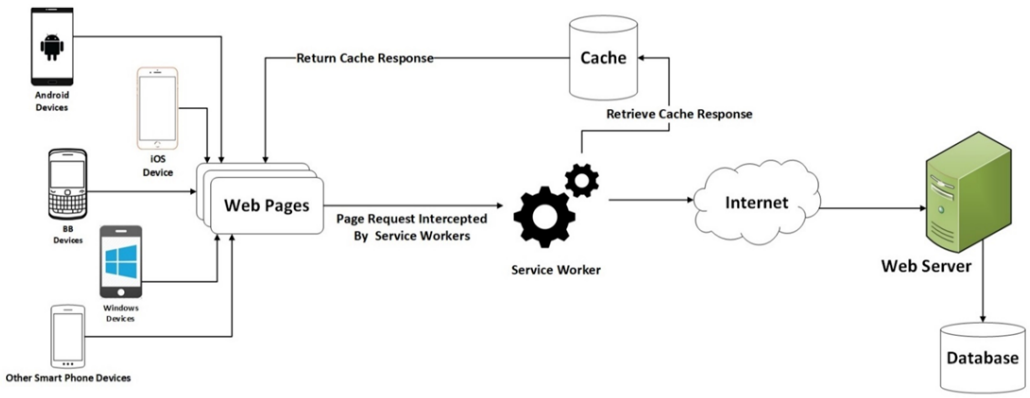

Figure 5: PWA Development Approach Architecture (Researcher's Diagram)

Features of PWA

The PWA were advocated by google[b] and has compiled a list of considered features that are the baseline requirements for a PWA as identified below.

1. **Offline Capabilities:** PWAs have the ability to work to a great extent even if the device is offline (airplane mode or out of network coverage).
2. **Push Notification:** PWAs have the ability to display re-engaging notifications as defined in the push API.
3. **Add to Home Screen:** Ability to install the web app to the user's device at will.
4. **Background Synchronization:** Ability to synchronize data in the background.
5. **Storage Estimation:** ability to estimate the available storage that an application uses and also to know the amount of storage left.
6. **Web Share:** Ability to make use of the native sharing widget belonging to the Operating System (OS) as specified by the web share API.
7. **Cross-Browser Usage:** ability to work on major browsers.
8. **Page Unique Identity:** Every page has a unique URL which makes it linkable with other pages.
9. **Payment Request:** Ability to use the web payment request API to act as an intermediary among merchants and users.

The identified features have made PWA to be a special kind of mobile web app. [7] highlighted four areas in which PWA is aimed at improving the general web experience as listed below:

1. **Conversion:** PWAs are based on progressive enhancement strategy in which the lower level functionalities are cached initially after which the advanced functionalities (depending on the browser) are progressively enacted.
2. **Reliability:** With the help of Service Workers, PWAs can be loaded instantly with low or without network connection – dependencies on networks are eliminated.
3. **Performance:** There is a constant background process of the service workers so as to ensure instant and reliable experience for users.
4. **Engagement:** Engaging users have been made easy with PWA as it supports push notification in the cloud.

Components of PWA

There are three major components of PWA which are Service Workers and App Shell.

1. **App Shell:** This is used to store static contents of an application such as the navigation bar, home page and other resources which remains the same across the app (HTML, CSS-Minimal and JavaScript). This is done to provide a skeleton of the application when an offline request is made. This feature help

[b] https://developers.google.com/web/progressive-web-apps

to reduce the loading time of applications which further reduces as the user revisits the web application as evident in a load time test performed by [12].

2. **Service Workers:** This offers technical ground work such as background synchronization and push notifications [14]. This is efficiently done because the service worker runs a separate browser thread alongside other APIs to provide the native like application features [12]. Service worker is a script that runs in the background to receive messages even if the application is not active. As indicated in a research carried out by [13] service workers does not adversely affect the energy stored in a mobile device.
3. **Web Application Manifest:** This is a file that exposes certain modifiable setting to the app developer such as the logo image path, app name and so on. It is used to modify the behavior and style of PWA [11].

The strengths and weaknesses of PWA are considered below.

Strengths

1. It is easy to learn and develop using existing web technologies.
2. Installation of app on user's device before usage in not mandatory.
3. App can be accessible by users while offline.
4. It promotes user engagement.
5. PWAs run only on the HTTPS protocol making it highly secured.
6. The single app is developed and can run on any platform using mobile web browsers.
7. Saves development and maintenance cost as there is no need for development firm to hire different developers for different architectures.

Weaknesses

1. PWAs do not have full access to all low-level features of mobile devices.
2. Users cannot decide to update the app as the app automatically updates once it is visited.
3. Not too many browsers support this technology as of today.

4. Introduction (use bold for main headings like this one. do not use italic)

The goal of the feature comparison is to objectively recommend the best approach to be adopted in mobile development. To achieve the specified goal some development features will be compared across three (3) mobile development approaches (Native, Hybrid, PWA). Table 2 shows a feature comparison among the Native, Hybrid and PWA Mobile Development Approach

Analysis of Feature Comparison

1. **Installable:** This is the ability of mobile applications to be installed on the user's mobile device. This feature is possible in the three mobile development approach compared above.

2. **Offline Capability:** This is the ability for a mobile app to work without an internet connection (in airplane mode or out of network coverage). This feature is limited in both the Native and Hybrid development approach due to the fact that some apps are designed to work without the internet once installed on the mobile device such as an offline dictionary, offline game apps and so on. Apps such as Facebook, Instagram and so on that requires an internet connection to function cannot run on the Native and Hybrid architecture once the network is cut off. On the other hand, every app developed using the PWA approach has the ability to run to a great extent without an internet connection due to the presence of service workers.

Table 2: Feature comparison among the Native, Hybrid and PWA Mobile Development Approach

FEATURES	NATIVE	HYBRID	PWA
Installable	Yes	Yes	Yes
Offline Capability	Limited	Limited	Yes
Testable Before Installation	No	Yes	Yes
App Market Place Availability	Yes	Yes	Yes
Push Notification	Yes	Yes	Yes
Cross Platform Availability	No	Yes	Yes
Hardware and Platform Access	Yes	Yes	Limited
Background Synchronization	Yes	Yes	Yes
Security Layer	No	No	Yes
Link-Ability	No	No	Yes
Bookmark-Ability	No	No	Yes
Constantly Updated	No	No	Yes
Friction of Distribution	High	High	Low
Desktop Capability	No	No	Yes

3. **Testable Before Installation:** This implies that an app can be tried to see how it functions or operates before installation on the user's device. This feature is negative for the Native development approach and positive for the Hybrid and PWA approach.
4. **App Market Place Availability:** This explains the distribution of mobile applications via dedicated app stores. Apps developed using the Native and Hybrid approach are distributed via the Google Play Store, Apple iTunes Store, Windows Phone Market, BlackBerry App World depending the development platforms. PWA apps are only accessed via a dedicated and unique URL, however from Google Chrome version 72 (android platform), the Trusted Web Activity (TWA) feature has been embedded which allows PWAs to be distributed via the Google Play Store.
5. **Push Notification:** This is the ability to display re-engaging information to users. This feature is available for the three development approach being compared.
6. **Cross Platform Availability:** This is the ability for a mobile app to be distributed or made available on all mobile platform such as the android, iOS, Windows, BlackBerry and so on. This feature is not possible for the Native development approach except the app is re-developed using the specialized

SDK for the mobile platform. For the Hybrid development approach, this is possible with the help of the JavaScript Bridge but on the long run imposes performance overhead. The PWA development approach is the only approach that makes a mobile application available to all mobile platforms without re-development for each platform and also with no overhead incurred.

7. **Hardware and Platform Access:** The Native and Hybrid development model have full access to the hardware features and sensors of the host mobile device irrespective of the platform residing in the mobile device. However, the amount of hardware features and sensors that can be accessed by PWA depends on the type of smartphone use. PWAs have greater possibilities of accessing more device features on Android smart phones compared to the iOS. This can easily be confirmed by visiting *What Web Can Do Today*[c] on the smart phone that interest the developer.
8. **Background Synchronization:** All the mobile development approach as compared in table 2 have the ability to synchronize data with the server in the background.
9. **Security Layer:** Mobile applications developed using the Native and the Hybrid development approach are not deployed on a secured layer which can lead to a compromise in the integrity of the application. On the flip side, PWAs can **only** be accessed via a secured layer – Hypertext Transfer Protocol Secured (HTTPS) which provides a high level of security for the app.
10. **Link-ability:** Only the PWA development approach is equipped with the link-ability feature. This means individual page in a PWA has a URL through which it can be connected with other pages or through which other pages can link up to it.
11. **Bookmark-ability:** This feature allows desired pages of a mobile application to be bookmarked using the browser. This feature is only available in the PWA development approach.
12. **Constantly Updated:** Applications developed using the Native and the Hybrid development approach are usually downloaded to the user's mobile devices and can only be updated whenever an update is triggered and accepted by the owner of the mobile device. This is not the case for apps developed using the PWA approach due to the fact that the apps are loaded from the web server, once an update is made by the developer, the apps are automatically updated and integrated on all mobile devices where the app resides which also facilitates the same view for all users.
13. **Friction of Distribution:** The friction of distribution is high in both the Native and the Hybrid approach because apps developed in this approach can only be distributed via a dedicated app store. Whereas, the friction of distribution in the PWA approach is low due to the fact that the apps can be accessed by visiting a specified URL across any smart phone.
14. **Desktop Compatibility:** Applications developed using the PWA development approach are desktop compatible, that is, they can be viewed and used on laptops and desktop computers without any distortion or hindrance. However, this is not the case for the Native and Hybrid approach where apps can only be accessed on mobile devices with the required and specific platform.

5. Conclusion and Recommendation

The promises offered by PWAs can neither be underestimated nor compared to existing (traditional) mobile

[c] https://whatwebcando.today

development strategies. Development firms strive to reduce development time, testing time and cost as well as general maintenance cost – which is relatively impossible while adopting the native and hybrid development architecture. The mobile web development approach has completely eradicated the challenge of mobile fragmentation which implies that a mobile app can now run on any mobile platform with the help of a browser and does not need to be re-developed. PWA has completely brought in a new dimension with the help of the service worker, app shell and other components which has facilitated the offline loading, background synchronization, push notification of mobile applications thereby making web apps look, feel and act similar to native and hybrid apps. This research makes a recommendation of the PWA to mobile app developers based on feature comparison and analysis. However, further experiments on the mobile development approach can be carried out in terms of memory management and efficiency on smartphones to further validate the claims of this work.

References

[1]. Mayuran Sivakumaran and P. Iacopino, "The Mobile Economy 2018," GSMA Intelligence, pp. 5–11, 2019.

[2]. B. Y. K. Taylor and L. Silver, "Smartphone ownership is growing rapidly around the world, but not always equally," Pew Res. Cent., no. February, 2019.

[3]. W. S. El-Kassas, B. A. Abdullah, A. H. Yousef, and A. M. Wahba, "Taxonomy of Cross-Platform Mobile Applications Development Approaches," Ain Shams Eng. J., vol. 8, no. 2, pp. 163–190, 2017.

[4]. N. Pande, A. Somani, S. Prasad Samal, and V. Kakkirala, "Enhanced Web Application and Browsing Performance through Service-Worker Infusion Framework," in Proceedings - 2018 IEEE International Conference on Web Services, ICWS 2018 - Part of the 2018 IEEE World Congress on Services, 2018, pp. 195–202.

[5]. A. I. Khan, A. Al-Badi, and M. Al-Kindi, "Progressive Web Application Assessment Using AHP," Procedia Comput. Sci., vol. 155, pp. 289–294, 2019.

[6]. IBM, "HTML5 , Hybrid or Native Mobile App Development," White Paper, IBM Corporation, p. Document Number: WSW14182USEN, 2012.

[7]. I. Malavolta, "Beyond Native Apps: Web Technologies to the Rescue! (Keynote)," in Mobile! 2016 - Proceedings of the 1st International Workshop on Mobile Development, co-located with SPLASH 2016, 2016, pp. 1–2.

[8]. M. Latif, Y. Lakhrissi, E. H. Nfaoui, and N. Es-Sbai, "Cross platform approach for mobile application development: A survey," 2016 Int. Conf. Inf. Technol. Organ. Dev. IT4OD 2016, pp. 1–5, 2016.

[9]. F. Johannsen, "Progressive Web Applications and Code Complexity-An analysis of the added complexity of making a web application progressive," Linköping University, 2018.

[10]. A. Russell, "Progressive Web Apps: Escaping Tabs Without Losing Our Soul," Infrequently Noted, 2015. [Online]. Available: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/. [Accessed: 05-Feb-2020].

[11]. A. Biørn-Hansen, T. A. Majchrzak, and T. M. Grønli, "Progressive web apps: The possibleweb-native unifier for mobile development," in WEBIST 2017 - Proceedings of the 13th International Conference on Web Information Systems and Technologies, 2017, no. Webist, pp. 344–351.

[12]. K. Behl and G. Raj, "Architectural Pattern of Progressive Web and Background Synchronization," Proc. 2018 Int. Conf. Adv. Comput. Commun. Eng. ICACCE 2018, no. June, pp. 366–371, 2018.

[13]. I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirovic, "Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps," in Proceedings - 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2017, 2017, pp. 35–45.

[14]. A. Gambhir and G. Raj, "Analysis of Cache in Service Worker and Performance Scoring of Progressive Web Application," Proc. 2018 Int. Conf. Adv. Comput. Commun. Eng. ICACCE 2018, no. June, pp. 294–299, 2018.

[15]. L. E. Nugroho, A. G. H. Pratama, I. W. Mustika, and R. Ferdiana, "Development of monitoring system for smart farming using Progressive Web App," 2017 9th Int. Conf. Inf. Technol. Electr. Eng. ICITEE 2017, vol. 2018-Janua, pp. 1–5, 2018.

[16]. [V. Sharma, R. Verma, V. Pathak, M. Paliwal, and P. Jain, "Progressive Web App (PWA) - One Stop Solution for All Application Development Across All Platforms," Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., vol. 5, no. 2, pp. 1120–1122, 2019.

[17]. A. I. Wasserman, "Software engineering issues for mobile application development," in Proceedings of the FSE/SDP Workshop on the Future of Software Engineering Research, FoSER 2010, 2010, pp. 397–400.

[18]. S. Xanthopoulos and S. Xinogalos, "A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications," in In Proceedings of the 6th Balkan Conference in Informatics, 2013, pp. 213–220.

[19]. Y. Chang and S. Oh, "A study on the development of one source multi use cross-platform based on zero coding," Multimed. Tools Appl., vol. 74, no. 7, 2014.

[20]. R. C. . Rahul and S. B. Tolety, "A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach," in In 2012 Annual IEEE India Conference (INDICON), 2012, pp. 625–629.

[21]. A. Charland and B. Leroux, "Online advertising, Behavioral targeting, and Privacy," Commun. ACM, vol. 54, no. 5, pp. 0–5, 2011.

[22]. T. Steiner, "What is in a Web View: An Analysis of Progressive Web App Features When the Means of Web Access is not a Web Browser," in In Companion Proceedings of the The Web Conference 2018, 2018, pp. 789–796.