Progressive Web Apps for An Effective Cross-Platform Mobile App Development

Oforji, Jerome Chikwado
Department of Computer Science, Abia State Polytechnic, Aba
ofordelima@yahoo.com

Agbakwuru, Alphonsus O.
Department of Computer Science, Imo State University, Owerri

Amanze, Bethran C.
Department of Computer Science, Imo State University, Owerri
amanzebethran@yahoo.com

DOI: 10.56201/ijcsmt.v10.no2.2024.pg62.72

Abstract

The advent of cross-platform application development frame-works have made it much easier to create applications for multiple platforms for mobile devices. In spite of reduced learning effort, usually lower costs, and a faster time-to-market cross-platform methods always do not prevail in most cases. Although there are normal exclusions – like graphic-intensive games, which should to be programmed with the native software development kits (SDKS), choice between native apps, cross-platform generated ones, and Web apps can remain delicate. Whereas many diverse efforts have been made with respect to how cross-platform development frameworks ought to work, no technology is deemed unequivocally superior than the others. But a cross-platform mobile app have got an edge over native app development. This paper however looks into an effective cross-platform mobile app development using progressive web apps. It also recommends that developers adopt this technology of mobile app development due to its huge gains.

Keywords: *PWAs, Web, App, Android, iOS, Windows, Cross-platform, Native app*

1.0 Introduction

The driving energy for the smart phone revolution over the years have been the applications for the various smart phone platforms. As a result, mobile applications or apps have become essential for many businesses across numerous industries. It is however, vital that such applications run seamlessly on various platforms such as Android, iOS and Windows. Taking advantage of the cross-platform mobile development method, one can create such applications quickly and cost-efficiently. Developing applications for mobile devices is a multifaceted task which involves various options, trade-offs and technologies, also as a result of the rapid increase and disintegration of devices and platforms. With the increasing demands of smart phones and tablets, mobile apps are becoming wall-to-wall, consequently developing applications for mobiles phones are getting rather challenging with respect to cost, effort and marketing. There are wide ranges of operating systems in the market that are different, and these are an obstacle to developers when it comes to developing a single application for diverse operating system (Enihe and Joshua, 2020).

Whether a mobile application is created for the app stores or for internal use, there is always the challenge of needing to develop it for several platforms. This typically means aiming at both Android and iOS devices. Nevertheless, while developing in each platform's native framework does give one the benefit of being closer to the platform, but it comes with a cost. There is the definite cost of engaging native developers from a smaller group of talent, but, more significantly, none of the code base one is building with is sharable. The codes the iOS team develops, cannot be reused by the Android team and likewise that developed for Android cannot be reused for iOS. There now exist two separate code bases, which must be independently maintained. As a result, if there is an alteration to the business logic or design that runs within the application, business must update and test the different both code bases for the alteration to business logic or design. Apart from that, there is the extra task of keeping the user experience aligned across each platform. While you should follow the platform-specific end user experience designs, developers still want to the capability to offer custom touches to their application. Over again, more work must be repeated (Griffith, 2022). Having different code bases for different platforms makes it difficult for application maintenance, high cost and creates a situation for code reusability issues.

Currently, several opportunities exist to cut costs on mobile application development. Unlike a decade ago, where all a developer could do was build a native app. The disadvantage of this was that the developer had to spend twice the development costs and time to make apps for iOS and Android platforms. Nowadays, with the emergence of broad cross-platform development options, developing a single app for multi-platform become increasingly more popular (Vyshnova, 2021). Cross-platform development is simply the opposite of native app development. A native mobile app is an application that meets the requirements of a particular operating system or device platform by using its Software Development Kit (SDK) and primary technology stack, as well as hardware memory, camera, sensors, and other programs installed on a device, but a cross-platform application is a mobile app that is compatible with several operating systems or device platforms such as iOS, Android and Windows and can, consequently, run on any smartphone, tablet, PC, smartwatch, and connected TV. Cross-platform mobile app development is the process of creating an application that would be compatible and able to run on multiple platforms of smartphones. This means, same code could be used for the applications that run on all the platforms (Yewale, 2021).

Though development implementations for mobile applications have grown, the cross-platform development is still an important subject matter. Applications should generally, always support both iOS and Android devices. These are the two main mobile platforms. They should to run efficiently on several hardware, and also possess compatibility with a group platform versions. Furthermore, other categories of devices other than smartphone and tablets have emerged, which makes cross-platform support even more delicate. However, creating an app once and serving the host of possible targets remains a concern in spite of having cross-platform frameworks that are recognized by practice and research. The technology unifier is still to be discovered, but Progressive Web Apps (PWA) is a huge step towards it (Majchrzak, et al, 2018).

However, several frameworks for developing a cross-platform mobile application development exist such as React Native, Kotlin, Xamarin, PhoneGap, Native Script, Flutter but progressive web app (PWA) is becoming more of a unifier between Android and iOS devices. As it is gaining ground in mobile app development so is native app development losing ground. Besides, the PWA mobile app can equally run on windows using any browser. The PWAs are mobile apps experiences delivered through web browsers. In order words, it can run on any devices.

2.0 Developing Cross-Platform Applications

Software that can run in several platforms is called a cross-platform application. Cross-platform apps are often hybrid apps implemented using web technologies, but there are also several frameworks that enable cross-platform native development, such as React Native or Xamarin. React Native is a framework that enables native mobile app development with JavaScript and React. As react, a JavaScript framework developed by Facebook, turned out very successful, Facebook created React Native for streamlining the application development process for multiple platforms. Traditionally native applications for Android are made with Java, while applications for iOS have been made with Objective-C, or lately, Swift. However, in order to publish an application in multiple platforms, learning several programming languages takes time, and maintaining multiple codebases consumes a lot of resources. React Native pursued to make this easier allowing developer to use a single framework for cross-platform development (Boduch, 2017). Even though it is theoretically enough to know just JavaScript for developing a React Native application, there can often be platform specific issues where knowledge of native application development is beneficial; according to Discord, development is significantly easier for native developers, than pure web developers. Additionally, there are also some performance issues when developing with React Native. Java, Objective-C and Swift are considerably faster than JavaScript, especially when heavy calculations are included (Boduch, 2017).

Xamarin is another cross-platform tool that enables native development for android, iOS, OS X and UWP (Universal Windows Platform) with C# using Visual Studio. Xamarin applications can share codebase that includes most of the app logic and user interface, counting about 75 % of the code in general. In addition to the shared code base, the applications usually need some platform specific code. Developing with Xamarin makes it easier to develop for multiple platforms but some platform specific programming is needed, nevertheless (Xamarin, 2018).

2.1 Progressive Web Apps: Concept and Overview

Progressive enhancement is a web development strategy that has an emphasis on creating great user experiences regardless of the device capabilities or web connection speeds. Progressively enhanced applications get progressively better when connections and device capabilities improve. In case of connectivity, everything loads instantly if the connections are fast but on slower speeds the most important parts will be prioritized, and less relevant items are loaded later (Domes, 2017). Progressive Web Apps are web apps with a progressively enhanced, native-like user experience. PWAs can be installed on a user’s device, they have access to some native functionality like push notifications and they are able work offline as well. Once pinned to home screen, user experience should not be very different from that of a native app. They also integrate in the device settings and the app launcher. PWAs still run in the browser but they have their own app window, and the browser UI elements are not visible making the app appear as a native application (Domes, 2017). Google describes Progressive Web Apps as being Reliable, Fast, Integrated, and Engaging.

For an application to be considered a Progressive Web App, it needs to meet certain technical requirements:

1. Have a manifest file declared in the HTML head. Web manifest should be a standard JSON file placed in the app directory that includes information about the application and its behavior. Mandatory fields for PWAs are background color, display, icons, start_url and either name or short name (Domes, 2017).
2. Include an appropriate icon for home screen. There should be several differently sized icons for optimal display in all different devices and device sizes (Domes, 2017).
3. Register a service worker. Service workers are web workers that reside between the app and the network, handing network requests and caching (Domes, 2017).
4. Serve the application over HTTPS. HTTPS (Hypertext Transfer Protocol Secure) is an internet protocol. HTTPS prevents intruders from exploiting resources and tampering with data because the communication is encrypted and so it protects the integrity of the website. Service workers only work on applications where HTTPS is enabled. During application development, testing and developing is usually done on local host, which is an exception for this rule. It is always a good practice to secure applications in the web, regardless of the nature of the data that is handled inside the application (Basques, 2018). Services like Let us Encrypt by Internet Security Research Group (ISRG) allow anyone owning a domain to get the needed certificates for free.

Essentially almost any web app can be made a Progressive Web App as long as they meet the requirements. React, Angular, Vuejs and SPAs in general are very popular frameworks for PWA development, but it’s possible to make a CMS website, like Drupal or Word Press, a PWA too.

The PWA is based on the concepts of a single application for all platforms just like the hybrid approach (Khan, et. al. 2019). However, it possesses distinct capabilities such as instant loading, push notification even in the offline state. Figure 2.1 diagrammatically shows the PWA development approach.

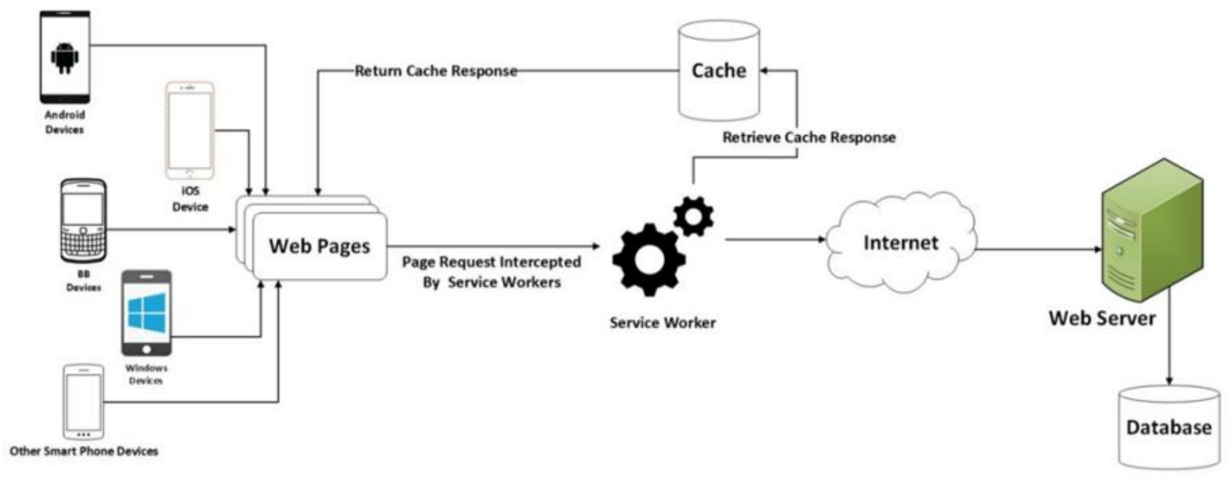

Figure 2.1: PWA Development Approach Architecture (Oluwatofunmi, et al., 2020)

The PWA was advocated by Google 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. Malavolta, (2016) 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.

2.1.1 Components of PWA

There are three main components of PWA which are Service Workers, App Shell and Web Application Manifest.

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 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 (Behl and Raj, 2018).
2. **Service Workers:** This offers technical ground work such as background synchronization and push notifications (Gambhir and Raj, 2018). This is efficiently done because the service worker runs a separate browser thread alongside other APIs to provide the native like application features (Behl and Raj, 2018). 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 (Malavolta et al., 2017), 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 (Biørn-Hansen, 2017).

The strengths and weaknesses of PWA are given 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.

2.1.2 Service Workers

One of the core building blocks that enable to build PWAs is a Service Worker. Without Service Workers, developing a PWA with offline functionality, Push Notification and Background Synchronization is impossible (Gaunt, 2018). It is a script that runs in the background. Web app will be used to install Service Worker into the browser. Like a Web Worker, Service Worker runs in a separate thread and does not access the Document Object Model (DOM) directly. It is a programmable network proxy, which acts as a liaison for every user request and manages those requests in a web application (Gaunt, 2018). Figure 2.2 below depicts the difference between JavaScript and Service Worker.

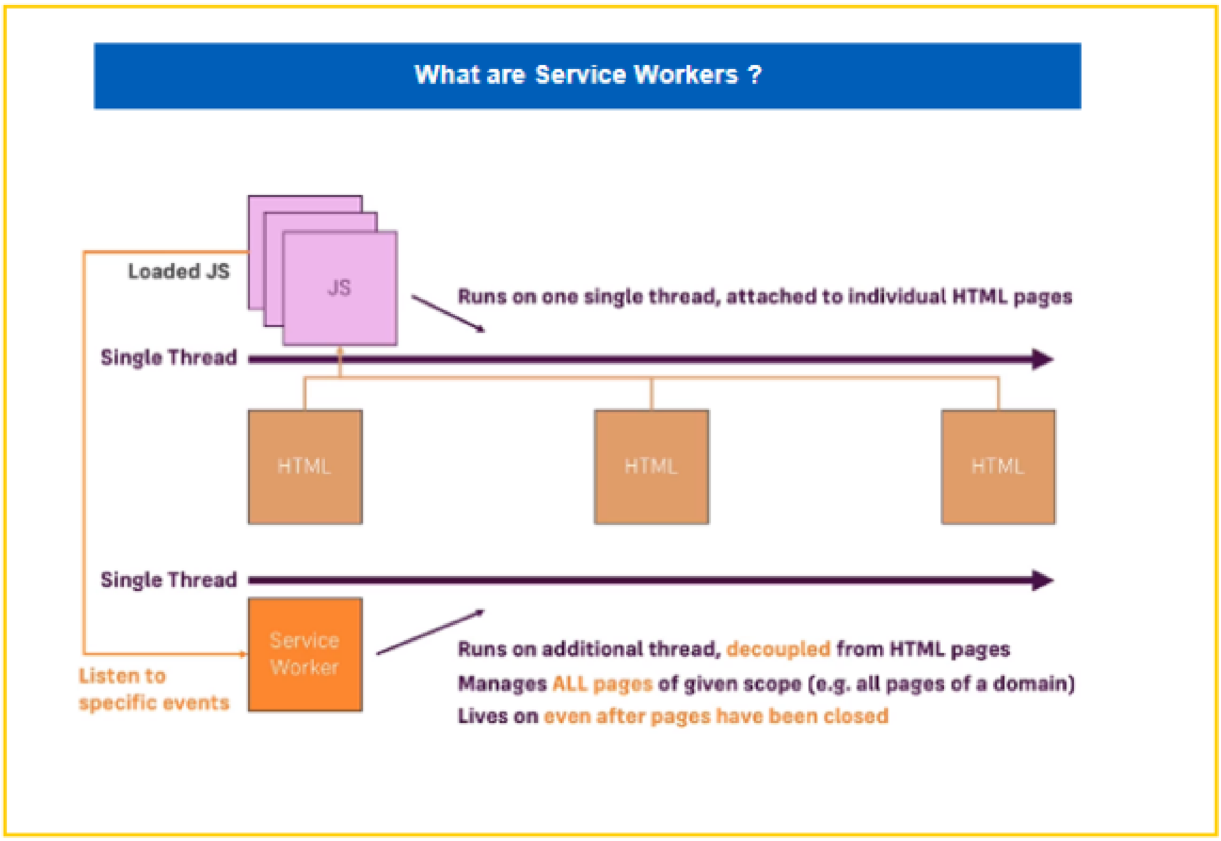
What are Service Workers ?
Loaded JS
JS
Runs on one single thread, attached to individual HTML pages
Single Thread
HTML
HTML
HTML
Single Thread
Service Worker
Listen to specific events
Runs on additional thread, decoupled from HTML pages
Manages ALL pages of given scope (e.g. all pages of a domain)
Lives on even after pages have been closed

Figure 2.2: Difference between JavaScript and Service Worker (Gaunt, 2018)

To install a Service Worker, the application developer initiates a request by writing a special script, which will be written with JavaScript (Gaunt, 2018). The script will initially help to download those static assets, which are required to be available when the user is offline. During installation, failure in any of these static assets' results termination of the previous installation, and a new installation will be retired when the consumer uses the app another time. Hence, the Service Worker will be triggered when the client uses the app following the first installation (Gaunt, 2018). Regarding the network connection, the Service Worker requires a secure connection to reduce the impact of interfering network requests. Moreover, using Hypertext Transfer Protocol Secure (HTTPS) helps to minimize possible attacks like hijacking a connection and fabricating a response with a man-in-the-middle. However, during development, it is possible to use local host with a Service Worker (Gaunt, 2018). Regarding events, Service Workers have a bunch of listenable events. Some of the most known listenable events that are available in Service Workers are: fetch event, push notification event, notification interaction event, background sync event, and service life cycle event. The Fetch event is an HTTP request event, which is initiated by the client or page-related JavaScript. Furthermore, Service Worker receives Web Push Notifications from the server and the user interacts with displayed notification. In addition, Service Worker receives Background Sync event for instance when an internet connection is restored. Finally, Service worker has an event, which changes the life cycle of the Service Worker. The function of Service Worker is to progressively enhance the application, which improves the contents as well as the functionality of the application. It can be added to the new or existing fully functional applications. However, in order to get the best out of this technology, the user needs to use the application with Service Worker compatible browsers (Gaunt, 2018).

2.2 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 (Sharma, 2019). Mobile apps are like the traditional software application but have some distinct features that distinguish it from regular apps. Wasserman (2010) identified requirements that clearly distinguish mobile apps from traditional apps some of which are:

1. **Potential Interactions with Other Applications**: This means that mobile devices might have numerous apps from different sources which likely interact 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 simple applications 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 2.1 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 (Malavolta, 2016).

Table 2.1: **Difference between four mobile platforms** (Malavolta, 2016)

Platforms	**Virtual Machine (VM)**	**Programming Language**	**Integrated Development Environment (IDE)**	**User Interface**
Android (Google)	Dalvik VM	Java	Eclipse, Android Studio, Android SDK	XML files
IOS (Apple)	No	Objective-C or Swif	XCode	Cocoa Touch
Windows (Microsoft)	Common Language Runtime (CLR)	C# or C++	Visual Studio	XAML files

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.

2.3 Advantages of Using PWAs for Cross-Platform Mobile App development

The advantages of using PWAs include the following:

i. It greatly reduces mobile app development cost as only one development team is needed to development mobile app for various platforms.
ii. It also reduces app development effort as only one code base is needed for multi-platform of devices.
iii. It enhances code reusability.
iv. It enhances and eases app maintenance.
v. It improves mobile app compatibility across several mobile platforms and even windows operating system.

3.0 Conclusion and Recommendation

Mobile app development has been a challenging area because of the multiplicity of platforms that are available in mobile phone industries. Deploying applications on mobile phones have depended so much on versions of the mobile phone and this leads to developing native apps for each version of the phone differently. This has a lot of cost and time consumption implications. The Cross-Platform Mobile App Development using progressive web apps will be beneficial to application users, management and application developers. The application users will have a better reach than mobile apps as many users can have access to the web app. Also, the progressive web apps are more engaging and native-like than regular web apps as it will have tools that will interact with the users. It will also consume less data and use less local storage, because it can work offline which requires no data. It works across different platforms, hence reduces the cost of developing different applications for different platforms.

It is however recommended that Progressive Web Apps be adopted by mobile App developers as it will beneficial to both users and developers in several aspects such app size which becomes smaller, renders speed, cost effective and easy of update and maintenance.

References

Enihe, R. and Joshua, J. (2020). Hybrid Mobile Application Development: A Better Alternative to Native. 10.11216/gsj.2020.05.39825.

Majchrzak, T. A. Biørn-Hansen, A., Grønli, T. (2018). Progressive Web Apps: the Definite Approach to Cross-Platform Development? Proceedings of the 51st Hawaii International Conference on System Sciences | 2018. URI: http://hdl.handle.net/10125/50607, ISBN: 978-0-9981331-1-9.

Griffith, C. (2022). What is Cross Platform Mobile Development? Iconic. Retrieved on 1st June, 2022 from https://ionic.io/resources/articles/what-is-cross-platform-app-development.

Vyshnova, J.(2021). Flutter vs PWA: What Is the Future of Cross-Platform Mobile App Development? Dinarys. Retrieved 1st June, 2022 from https://dinarys.com/blog/flutter-vs-pwa-what-is-the-future-of-cross-platform-mobile-app-development.

Yewale, K. (2021). Cross Platform Application Development: Benefits and Technology. ClarioTech. Retrieved 2nd June, 2022 from https://www.clariontech.com/blog/cross-platform-application-development-benefits-and-technology.

Boduch, A. (2017) [E-book]. React and React Native. Packt Publishing Limited.

Malavolta, I. (2016) 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.

Malavolta, I., Procaccianti, G., Noorland, P. and Vukmirovic, P. (2017) "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.

Gaunt, M., (2018) Service workers: an introduction. URL: https://developers. Google. com/web/fundamentals/primers/service-workers/#cache_and_return_requests. Accessed, 20.10. 2018 , p.2018.

Domes, S. (2017) [E-book]. Progressive Web Apps with React. Packt Publishing Limited.

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

Basques, K. (2018). Why HTTPS Matters. Google Developers. https://developers.google.com/web/fundamentals/security/encrypt-in-transit/why. Updated 21.9.2018. Accessed 25.10.

Biørn-Hansen, A., Majchrzak, T. A. and Grønli, T. M. (2017) "Progressive web apps: The possible web-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.

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

Gambhir, A. and Raj, G. (2018) 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.

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

Xamarin Documentation (2018) [Online]. Microsoft. https://docs.microsoft.com/enus/xamarin/. Accessed 1.11.2018.

Oluwatofunmi, A., Chigozirim, A., Nzechukwu, O., Olawale, J. O. (2020). Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development. American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS) ISSN (Print) 2313-4410, ISSN (Online) 2313-4402.

Sharma, V., Verma, R., Pathak, V., Paliwal, M. and Jain, P. (2019). 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.

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