Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2021, 8(4):60-68

Research Article ISSN: 2394 - 658X

Progressive Web Apps: Merging Web Accessibility with Native Performance

Mounika Kothapalli

Senior Application Developer at ADP Consulting Services
Email: moni.kothapalli@gmail.com

ABSTRACT
Progressive Web Apps combine the best of the web and native applications in the new technology. It allows functionality similar to an application to be presented directly in a web browser. Such an application introduces features like offline capability, push notifications, and the possibility of installation of the application right on the home screen. This paper outlines exactly what Progressive Web Apps are and describes their core features, together with benefits for developers and users. It addresses technical approaches to building PWAs, including issues in developing them. A deep explanation of successful deployments of PWAs which may impact improvements in user engagement and conversion metrics is placed in a nutshell. Finally, the paper concludes with future directions and forecasts further developments in the sphere of Progressive Web Apps.

Key words: PWAs, web application, native application, offline functionality, push notifications, responsive design, performance optimization.

INTRODUCTION

Progressive web applications are one of the critical technologies that enable making web application accessibility mesh with strong abilities of native applications. With the rapid proliferation of mobile devices and ever-larger demands for seamless, cross-platform user experience, PWAs represent the revolutionary answer to the strength of web and native ecosystems. This paper aims to explain the architecture, benefits, implementation techniques, and the empirical effects of progressive web applications.

A. Research Background

Over the last ten years, this landscape for the web and mobile application development landscape has been dramatically changed. Native applications dominated, for advanced functionality, better performance, and direct access to device-level features. However, developmental and maintenance issues and costs are significant, considering the number of platforms involved. On the other extreme, traditional web applications deliver universal access through web browsers, but often lack performance and the quality of user experience that native applications afford.

One of the answers to these two opposing challenges was Progressive Web Apps. They leverage the latest in web technologies like Service Workers, Web App Manifests, and secure protocols like HTTPS. PWAs aim to offer improved capabilities, the ability to work offline, and a performance boost. Back in 2015, Google introduced PWAs to the world and immediately became popular among developers and companies because of its compelling benefits.

B. Objectives

This article outlines several central goals:

- Provide a thorough analysis of Progressive Web Apps, explaining their central features and core concepts.
- Evaluate the advantages for developers and users, focusing on the enhancing user experience, offline availability, and platform independence.
- Review the technical fundamentals of creating PWAs, such as deploying Service Workers, using Web App Manifests, and performance optimization.
- Demonstrate empirical examples and case studies in which PWAs show positive effects on user engagement and metrics of conversion.
- Outline the problems and limitations of PWAs and discuss potential solutions as well as predictions for future trends.
- Thus, this article achieves the above tasks as it contributes to the scholarly knowledge on Progressive Web Apps and provides actionable advice for developers, researchers, and companies considering the adoption of this technology.
-

C. Paper Structure

The paper has the following structure: Section II provides a general view on the concept of Progressive Web Apps, while Section III analyzes the benefits of the technology in terms of PWA development and the end-users' perspective. Section IV follows a technical view on the context of PWA development, with tools for development, Service Worker implementation, caching mechanisms, and performance optimization. Section V will cover some of the most common challenges and limitations in terms of the use of PWAs and the ways these problems can be alleviated. Section VI will cover some of the future directions and trends in the PWA domain. Finally, Section VII concludes the paper with the summary of the main findings, their implications on stakeholders, and research opportunities.

LITERATURE REVIEW

PWAs are a new modus operandi for developing a web app that manages to capture all the best features of a web and native app. This section gives the introductory or initial understanding of PWAs—the definition of the same and its characteristics and differentiate the same from traditional web apps and native mobile applications.

A. Definition and Characteristics

A Progressive Web App uses state-of-the-art available in web technologies and design principles to offer a similar end-user experience to native applications within a web browser [1]. In other words, the defining characteristics of the PWAs include:

- Progressiveness: PWAs are universally functional across all browsers, making their functionality progressive depending on the device and browser technology available [2].
- Responsiveness: PWAs are optimally designed for viewing on every device, and they adapt seamlessly to different screen sizes and orientations [3].
- Connectivity Independence: They work offline or under a poor network condition, thereby making them accessible everywhere [4].
- App-like Interface: PWAs simulate the fully immersive, full-screen experience of native apps [5].
- Freshness: The usage of Service Workers allows PWAs to update their content in the background without requiring user interaction [6].
- Security: That's delivered through HTTPS, which makes their data transmission secure and thus ensures enhanced user data security [7].
- Discoverability and Installability: They are easily searchable and are installed directly on devices without the mediation of an app store [8].
- Re-engagement: PWAs have the ability to use push notifications and thus can engage their users even when they are not actively in use [9].

B. Comparison with Native Apps and Traditional Web Apps

PWAs uniquely put together the attributes that set them apart from both native mobile applications and traditional web applications. Unlike Native apps, PWAs are reachable from web browser too [10]. Whereas, the distinction between classical web applications and PWAs is that PWAs have more of a native application look and feel, the load times are higher, and they can even be installed on a user's device [11], in addition to having offline functionality and providing native functionality such as: push notifications and offline sync using device memory.

C. Key Components of PWAs

Progressive Web Apps employ several core components and technologies as shown in Fig. 1, to gain their advanced functionality:

1. Web App Manifest
   This is a JSON file that holds metadata for the PWA. It includes the app's name, icons, theme colors, and display preferences [12]. It allows installation of a home-screen icon and sets up appearance and behavior of the app.
2. Service Workers
   Service Workers are scripts that handle the user interaction for the PWA and the network [13]. They provide offline access, caching of resources, background data synchronization, and push notifications. It runs in its own thread, separate from the main browser thread; a Service Worker handles the network requests and caching of resources to achieve easy delivery of content without an internet connection.

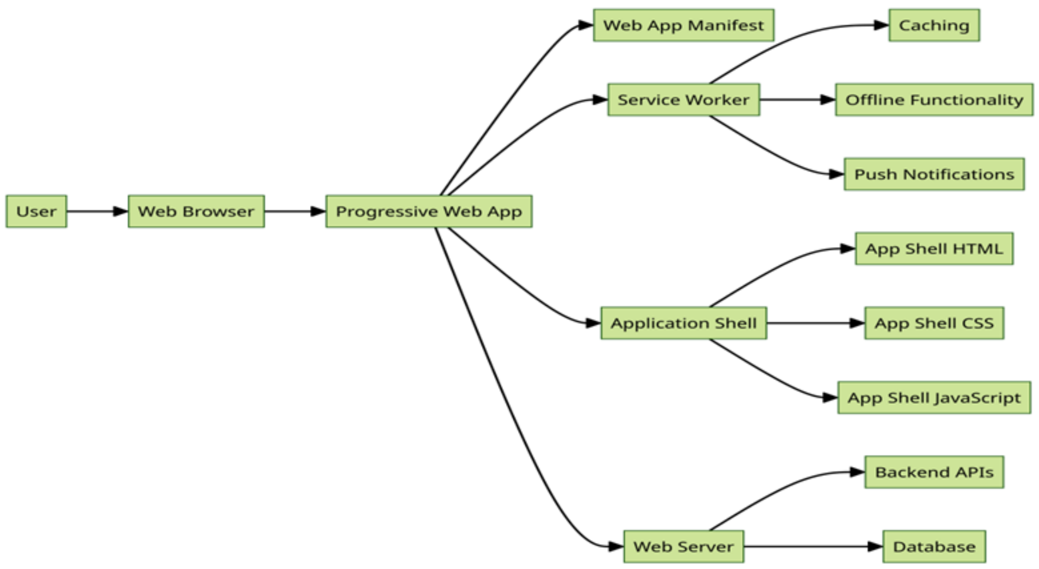

Figure 1: Key components of PWA

Padlock: PWAs must be served over HTTPS for the connections between the user's device and web server to be encrypted [15]. HTTPS ensures the integrity and confidentiality of data from being intercepted and modified.

3. Responsive Design
   PWAs use responsive design principles that guarantee a good experience in most devices and across different-sized screens [16]. They dynamically adjust the layout and content of the user's viewport and offer a level of usability and aesthetic coherence that is the same for all devices and screen sizes.
4. App Shell Architecture
   This architecture model separates the lightweight application skeleton, known as the shell, from dynamic content [17]. The shell is cached and is immediately loaded and easily interactive, while dynamic content updates content freshness and supports offline performance.
   These components make up the structural composition of a Progressive Web App that makes it possible to deliver a strong, app-like experience within a web browser.

BENEFITS OF PROGRESSIVE WEB APPS

Progressive Web Apps offer multiple benefits that make them an attractive option for developers and users. The following discussion is on the key benefits that come with the PWAs, including better user experience, offline access, support for push notifications, being installed to home screens, better performance, cross platform compatibiltiy, and cost-effectiveness.

A. Better User Experience

PWAs offer an intuitive user experience that is similar to native mobile apps. They offer a responsive and app-like user interface that gives a smooth and intuitive experience across devices [18] and different screen sizes.

B. Working Offline

Fig. 2 the use of Service Workers. Through the use of Service Workers, the PWAs can work offline, or in poor network conditions since they cache the required resources and data that work even when they are offline [19]. Overall, it enhances the user experience when poor or intermittent network connectivity is experienced [20].

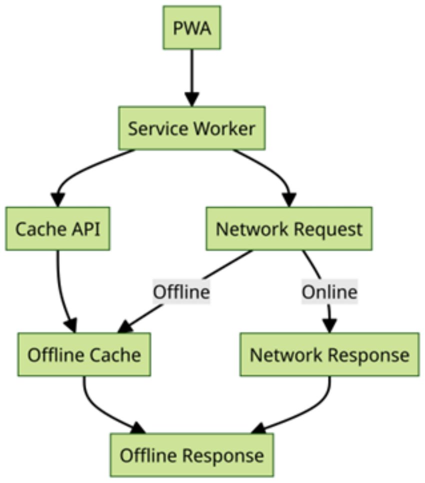

Figure 2: Offline functionality illustration

C. Push Notifications

Progressive Web Apps can also engage users with push notifications, just like native apps. Another way of keeping the user engaged and loyal is by sending timely, relevant information, or personalized content for them [21]. Push notifications keep the user attached to the application, increasing interaction and retention.

D. Home Screen Installation

PWAs can be easily installed directly onto a device's home screen, much like native applications are placed. This will make it easier to launch PWAs directly without going through a web browser, thus making the platform easily accessible to ensure user adoption. Installation would be done outside app stores, removing all barriers to user engagement.

E. Performance

PWAs are built for speed and performance, ensuring smooth operations on low-end devices and on devices with slow network conditions. Techniques such as caching, lazy loading, and code splitting reduce initial load times and provide a good user experience. Service Workers enhance performance by managing content loading and reducing dependency on network connections [22].

F. Cost-Effectiveness and Ease of Development

Developing a PWA, therefore, usually costs less than developing a native application for each of the platforms. In turn, based on the common-used web technologies, PWAs benefit from a large developer community and standard practices [23]. Development is more efficient since troubles about the app stores and lengthy procedures for installation are minimized. PWAs allow updating instantly for all users, and update procedures become simple to manage [24].

G. Cross-Platform Support

Native web technologies make PWAs inherently cross-platform. This means no need for separate development for native platforms like iOS and Android. Any device with a modern web browser is, therefore, accessible, as seen in Fig. 4. A single codebase allows for a lot of reduction in development efforts and enables the platform to be marketed to a much larger market [10].

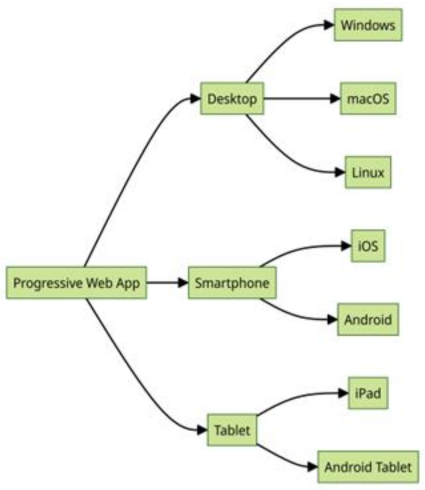

Figure 3: Cross platform compatibility

BUILDING PROGRESSIVE WEB APPS

Building a Progressive Web App requires a lot of tools, frameworks, and methodologies in order to construct effective and seamless user interfaces. This section will explain the building blocks of PWA construction: adoption of development tools and frameworks, implementation of Service Workers, methods of caching of resources, use of push notifications, performance optimization, and testing and debugging methods.

A. Development Tools and Frameworks

A range of development tools and frameworks can augment development of PWAs. Some popular ones include Angular, React, Vue.js, and Ionic一toolbox because of their complete ecosystems, large libraries, and powerful tooling that will help in efficient building of PWAs. Other tools include Google's Lighthouse and PWABuilder, which can audit and optimize PWAs to earn optimum performance and best practices compliance [25].

B. Implementation of Service Workers

Service Workers hold great importance for PWAs because they make possible the ability to work offline, resource caching, and the delivery of push notifications. Development of Service Workers involves writing scripts in JavaScript to intercept network requests and control resource caching and self-define what the user's experience is when offline. There are libraries, such as Workbox, which facilitate the implementation and maintenance of the Service Workers [26].

C. Caching Strategies

Caching holds a lot of importance in PWAs. Proper caching by the developer of the PWA ensures reliability and fast delivery of the contents even in conditions of unavailability of the network. There are various caching approaches available. The approach usually depends on the needs of the application and balances between the need for real-time data and the user's tolerance of stale contents. There is a set of approaches from cache-first to network-first to stale-while-revalidate [27].

D. Push Notifications

The push notification is the key element that a PWA needs. Push notifications remain a very effective way of keeping the user engaged and notifying them of time-critical information. Push notifications usually are designed with the Push API and Notification API, which is usually combined with services such as Firebase Cloud Messaging. Proper handling of the user subscriptions, the contents of the notifications, and the permissions is essential to provide a good user experience with the notifications [28].

E. Techniques to Optimize Performance

Performance optimization ensures that the PWA is fast and responsive. Code splitting, lazy loading, and small bundle sizes of JavaScript increase load times. Other techniques like asset compression, efficient caching, and optimization of the critical rendering path also help in performance improvement [29].

F. Testing and Debugging

Testing and debugging are utilized in developing quality built and reliable PWA. Tools such as Lighthouse, Chrome DevTools, and test tools integrated within the PWA frameworks help in detecting and fixing performance issues, debugging issues, and ensuring PWA best practices are followed [25]. Thorough testing of the application across various devices, network conditions, and user interactions is required to ensure the durability and reliability of the PWA.

PROBLEMS AND LIMITATIONS

Though Progressive Web Apps have lots of benefits, they also face certain problems and limitations which developers and enterprises should watch out for. The paper shall discuss the critical problems: browser compatibility, app store acceptance, user awareness and acceptance, and security issues.

Browser compatibility remains a very important issue for Progressive Web Apps, despite improvements over the years. Different browsers, and their respective versions, indeed still reveal inconsistencies in supporting the core PWA features, such as push notifications and background sync [11][30]. Also, Progressive Web Apps bypass traditional approval processes and fees of app stores by not being hosted on platforms such as Apple's App Store or Google Play Store, which, while convenient, also significantly lowers their visibility and discoverability in comparison to native apps. And, some app stores have their policies and limitations that further work against the functionality and acceptance of PWAs [31].

On the user's side, there is a huge challenge to increase awareness and acceptance of PWAs. Most users are accustomed to downloading apps from app stores themselves; hence, strong emphasis needs to be given to users about the benefits and capabilities of PWAs in order to foster acceptance [5]. However, the main point at which users resist Progressive Web Apps is the issues related to trust, security, and perceived inferiority to native apps. The security of PWAs is especially critical due to their access to sensitive information of the users and features of a device. In order to protect these, the developers have to enforce stringent security practices: secure communication via HTTPS, strict input validation, secure data storage, and protections against cross-site scripting attacks [11].

A. Browser Compatibility

The first problem with PWAs is browser compatibility. Though the situation has been improved over recent years, still the problem exists on browser compatibility among different browsers and versions [11]. Not all browsers support the PWA features fully, such as push notifications and background sync, which may hinder the functionality and the user experience of the PWA [30].

B. App Store Acceptance

The absence of application stores like the Apple's App Store or Google Play Store allows the PWA to bypass the difficult app store approval processes and fees, but it reduces its visibility and discovery compared to native apps. Besides, some app stores can have some policy or limitation that can degrade the functionality and possible acceptance of the PWAs [31]

C. User Awareness and Acceptance

Increasing user awareness and acceptance of PWAs would be a challenge because many users are used to downloading apps from app stores. It's important to inform users of the benefits and capabilities of PWAs to make them accept it [5]. On the other hand, users' resistance to the installation of PWAs is largely due to concerns about trust, security, and the relatively lower value compared to native apps.

D. Security Issues

Security is indeed a critical aspect of PWAs, given the access to sensitive users' information and device capabilities. The apps should show secure communication by using HTTPS. However, developers must implement strong security measures. This includes the validation of input, safe data storage, and protections from cross-site scripting attacks [11].

FUTURE TRENDS AND OPPORTUNITY

Some of the future trends and opportunities for PWAs include: Advancement in web technologies, integration with novel technologies, prospects for enterprise adoption, and its role as a complement to native apps.

A. Advancement in Web Technologies

As web technologies continue to improve, PWAs will benefit from new capabilities and improvements. New achievements in Web Assembly and Web GPU are likely to enable PWAs to benefit from high-performance computing and rendering graphics, thus giving a lot more opportunities for complex applications and rich user experiences [30]. Further enhancements in web performance optimizing and 5G networks shall come in to provide fast and response speed to PWAs.

B. Integration with Emerging Technologies

PWAs have the potential to integrate with emerging technologies such as Augmented Reality and Virtual Reality. With the emergence of Web AR and Web VR technologies, there is a prospect for PWAs to offer immersive and interactive experiences directly in the web browser. This capability opens new frontiers in e-commerce, education, and entertainment.

C. Prospects for Enterprise Adoption

For enterprise use, they would be very attractive because of the benefits of cross-platform compatibility, cost efficiency, and ease of deployment. PWAs can be used in developing both internal tools and employee-facing applications and customer solutions by enterprises [31]. More importantly, PWAs can facilitate B2B interaction. They provide a seamless platform in which businesses can interact with partners and customers.

D. PWAs as a Complement to Native Apps

Although PWAs can take the place of native apps in many cases, they can also augment them. Organizations might use PWAs to offer a simple, cross-platform version of their app, along with a native app for those users who want advanced features or who insist on a native experience [24]. PWAs can provide a steppingstone or entry-level platform for users to test the value of the app before deciding to make a full installation.

CONCLUSION

Progressive Web Apps (PWAs) have evolved as a revolutionary technology, essentially bridging the gap between web and native mobile applications. The present paper has described the architecture of PWAs, elaborating on their key characteristics, benefits, and the issues and opportunities they present.

A. Summary of Key Points

PWAs combine the best aspects of both web and native apps by offering one consistent and seamless user interface across different devices. They use the latest web technologies such as Service Workers, Web App Manifests, and HTTPS to offer offline access, push notifications, and installation to the home screen. PWAs provide some great benefits such as enhanced user experience, better performance, cross-platform support, and cost savings [30].

On the downside, PWAs face a few challenges: browser compatibility, limited acceptance by app stores, and low user awareness and adoption [32].
In the future, PWAs seem to be promising as advancements in web technologies may occur; also, integration of PWAs with future technologies, like AR and VR, and high-end business opportunities [24]. PWAs can also act as a lightweight, accessible alternative to native apps.

B. Implications for the Development Community and Businesses

The findings of the research are immense for the development community and businesses. For the development community, PWAs give them a full-scale framework for development where they use web technologies to develop cross-platform applications [10]. Developers will need to stay up-to-date on changes in web standards, best practices, and tools to develop high-quality PWAs that can meet the expectations of users.
PWAs give businesses an opportunity to gain more reach, better user engagement, and cut down on the cost of development [31]. Businesses have to develop a strategy to introduce PWAs into their mobile strategy in situations where one needs cross-platform support, offline support, and easy distribution.

C. Future Research Directions

This paper, however, has given a panoramic view of the Progressive Web Apps. Several areas still warrant detailed investigations. For example, future studies can look at long-term user adoption and retention compared to native applications. Insights into how to optimize PWAs can be derived by comparing its performance and user experience across various devices and network conditions.
There is also room for researching the security dynamics of PWAs and shedding light on vulnerabilities and best practices for secure development. Integration of PWAs with the latest technologies, such as augmented reality, virtual reality, and IoT, can also result in novel applications and use cases.
The impact of the PWAs on current app store ecosystems and alternative distribution models is also deserving of study. This may give deeper insight into what the future of mobile app development and distribution may look like.
In summary, Progressive Web Apps represent a paradigm shift in the realm of mobile application development. As the world of web technologies continues to evolve and the need for seamless interfaces continues to grow in demand, PWAs are poised to play an important role in the future of mobile app development.

REFERENCES

[1]. A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive Web Apps: The Possible Web-Native Unifier for Mobile Development," in Proc. 13th Int. Conf. Web Inf. Syst. Technol. (WEBIST), 2017, pp. 344–351, doi: 10.5220/0006353703440351.
[2]. A. Russell, "Progressive Web Apps: Escaping Tabs Without Losing Our Soul," Infrequently Noted, Jun. 15, 2015. [Online]. Available: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/
[3]. G. Heaton, "Why Progressive Web Apps Are The Future of Mobile Web," Sitepoint, Mar. 13, 2019. [Online]. Available: https://www.sitepoint.com/progressive-web-apps-future-mobile-web/
[4]. M. Gaunt, "Service Workers: An Introduction," Google Developers, Dec. 1, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/primers/service-workers
[5]. I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, "Hybrid Mobile Apps in the Google Play Store: An Exploratory Investigation," in Proc. 2nd ACM Int. Conf. Mobile Softw. Eng. Syst. (MOBILESoft), 2015, pp. 56–59, doi: 10.1109/MobileSoft.2015.15.
[6]. A. Osmani, "The App Shell Model," Google Developers, May 29, 2020. [Online]. Available: https://developers.google.com/web/fundamentals/architecture/app-shell
[7]. M. Gaunt, "Web App Manifest," Google Developers, Nov. 5, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/web-app-manifest/
[8]. A. Osmani, "Installable Web Apps," Google Developers, May 29, 2020. [Online]. Available: https://developers.google.com/web/fundamentals/app-install-banners/
[9]. M. Gaunt, "Push Notifications: Timely, Relevant, and Precise," Google Developers, Nov. 8, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/push-notifications/

[10]. T. A. Majchrzak, A. Biørn-Hansen, and T.-M. Grønli, "Progressive Web Apps: the Definite Approach to Cross-Platform Development?" in Proc. 51st Hawaii Int. Conf. Syst. Sci. (HICSS), 2018, pp. 5735–5744, doi: 10.24251/HICSS.2018.718.

[11]. A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive Web Apps for the Unified Development of Mobile Applications," in Web Information Systems and Technologies, Springer, 2018, pp. 64–86, doi: 10.1007/978-3-319-93527-0_4.

[12]. Google Developers, "Web App Manifest," Web Fundamentals, Nov. 5, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/web-app-manifest/

[13]. J. Archibald, "The Service Worker Lifecycle," Google Developers, Dec. 1, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/primers/service-workers/lifecycle

[14]. N. Alim, "The Offline Cookbook," Google Developers, Dec. 12, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/

[15]. E. Enge, "Progressive Web Apps (PWAs) - The Complete Guide," Search Engine Journal, Oct. 2, 2019. [Online]. Available: https://www.searchenginejournal.com/progressive-web-apps-guide/272885/

[16]. P. Walton, "Responsive Web Design Basics," Google Developers, Feb. 12, 2020. [Online]. Available: https://developers.google.com/web/fundamentals/design-and-ux/responsive/

[17]. A. Osmani, "The App Shell Model," Google Developers, May 29, 2020. [Online]. Available: https://developers.google.com/web/fundamentals/architecture/app-shell

[18]. S. Valtolina, D. Cascini, and B. R. Barricelli, "Progressive Web Apps: An Analysis of Features and Performance," in Proc. 28th Int. Conf. Inf. Syst. Dev. (ISD2019), 2019, pp. 1–12.

[19]. M. Gaunt, "Service Workers: An Introduction," Google Developers, Dec. 1, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/primers/service-workers

[20]. A. Osmani, "The Offline Cookbook," Google Developers, Dec. 12, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/

[21]. A. Osmani, "Push Notifications: Timely, Relevant, and Precise," Google Developers, Nov. 8, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/push-notifications/

[22]. M. Gaunt, "Service Workers: An Introduction," Google Developers, Dec. 1, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/primers/service-workers

[23]. S. S. Gill, I. Chana, and R. Buyya, "IoT-Based Agriculture as a Cloud and Big Data Service: The Beginning of Digital India," J. Organ. End User Comput., vol. 29, no. 4, pp. 1–23, Oct. 2017, doi: 10.4018/JOEUC.2017100101.

[24]. A. S. Simoens, "Towards Progressive Web Apps as Default," in Proc. 15th Int. Conf. Web Inf. Syst. Technol., 2019, pp. 488–495, doi: 10.5220/0007729004880495.

[25]. Google Developers, "Tools for Web Developers," Web Fundamentals, Nov. 5, 2019. [Online]. Available: https://developers.google.com/web/tools/

[26]. Google Developers, "Workbox," Nov. 5, 2019. [Online]. Available: https://developers.google.com/web/tools/workbox

[27]. A. Osmani, "The Cache API: A quick guide," Google Developers, Nov. 5, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/cache-api

[28]. Google Developers, "Push Notifications: Web Push Protocol," Nov. 8, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/push-notifications/web-push-protocol

[29]. Google Developers, "Performance," Web Fundamentals, Nov. 5, 2019. [Online]. Available: https://developers.google.com/web/fundamentals/performance

[30]. S. Valverde and A. Hernández, "Progressive Web Apps: A New Way to Deliver Web Content," in Proc. 15th Iberian Conf. Inf. Syst. Technol. (CISTI), 2020, pp. 1–6, doi: 10.23919/CISTI49556.2020.9140864.

[31]. M. Lochrie, P. Garbett, and P. Coulton, "PWA vs Native: Which Should You Choose for Your Enterprise App?" The Manifest, Jul. 10, 2019. [Online]. Available: https://themanifest.com/app-development/pwa-vs-native-enterprise-app

[32]. I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirović, "Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps," in Proc. 4th Int. Conf. Mobile Softw. Eng. Syst. (MOBILESoft), 2017, pp. 35–45, doi: 10.1109/MOBILESoft.2017.7.