IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Special Issue 2, April 2023

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | SJIF: 8.423|| Volume 12, Special Issue 2, April 2023 ||

8th National Conference on Advanced Computing Technologies (NCACT'23)

Organized by

Department of CSE, Velammal Engineering College, Chennnai, India

Progressive Web Apps for the Unified Development of Mobile Applications

Kamireddy Vinish Reddy[1], Bharathi Sri[2]

U.G. Student, Department of Computer Science and Engineering, Velammal Engineering College, Chennai, Tamil Nadu, India [1]

Assistant professor, Department of Computer Science and Engineering, Velammal Engineering College, Chennai, Tamil Nadu, India [2]

ABSTRACT: This paper mentions Progressive Web Applications (PWAs) and how they make it possible to design web applications first and foremost offline. PWAs can be downloaded, installed, and utilised offline on a variety of platforms, including mobile devices and personal computers, in contrast to standard web apps. The paragraph gives a summary of the most recent developments in PWA research and practise, including the underlying theories and methodologies. On both technical and general levels, it contrasts PWAs with cross-platform app development strategies and makes recommendations for further study in this area. The paragraph finishes by urging academics to stay up to date with the most recent developments in mobile and web development, particularly PWAs, which might offer intriguing research topics with practical applications.

KEYWORDS: Progressive Web Apps, Service workers, Cross-platform development, Mobile web.

I. INTRODUCTION

The creation of web- and app-enabled electronics like smartphones, smartwatches, TVs, and laptops has raised the need for multi-platform technical developers. Codebases used to only have one or a few target platforms. This is referred to as the Native development technique in mobile development, where apps are created using a platform-specific language, such as Swift or Objective-C for iOS and Java or Kotlin for Android.

Yet, due to their lack of platform knowledge, developers are constrained by the profusion of devices that can run apps and display web content. Cross-platform development frameworks and tools have grown in popularity as a result of this. These frameworks and technologies are designed to speed up product development and maximise code reuse across platforms. In recent years, various cross-platform development frameworks have been the focus of academic study. These frameworks are typically grouped into development methodologies like hybrid, interpreted, cross-compiled, model-driven (MDD), and component-driven (CDD).

The Progressive Web App (PWA) strategy is one that has drawn attention. PWAs enable webpages to behave less like a website and more like an app, allowing users to download and install them on their phones in a manner akin to how normal applications are downloaded and installed via app stores. PWAs can also function well offline, which makes them very helpful in locations with limited or expensive internet connectivity.

There hasn't been a lot of new research on Progressive Web Applications since 2017. The other sections of the paper cover the study methodologies used by the authors, related work on cross-platform development and Progressive Web Applications, underlying technologies and concepts, conclusions on feature and performance comparisons, and recommendations for further research.

II. RELATED WORK

Although there is a well-established body of knowledge with an industry focus, it is clear from the literature study that scholarly involvement and research on Progressive Web Applications are lacking. The tutorials and blog posts are produced and published by the Google Web Fundamentals group, which should act as the foundation for future study until a strong academic knowledge base is established. In addition to the content created by Google, there are two early-access books and a huge number of industry pieces that cover a wide range of subjects, such as fundamental ideas, difficulties, and the effects of web innovation.

|| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | SJIF: 8.423|| Volume 12, Special Issue 2, April 2023 ||

8th National Conference on Advanced Computing Technologies (NCACT'23)

Organized by

Department of CSE, Velammal Engineering College, Chennnai, India

Some app development strategies mentioned in the literature, such as cross-platform app development and underlying techniques, however, have a more substantial academic background. There have been a number of articles on these themes that have been discovered in the past; these papers provide the theoretical underpinning and early research from which later papers draw. The papers discuss classifications, performance measurements, and technical framework and method comparisons. Recent studies have been found on issues including end-user perception of cross-platform apps, energy usage comparisons, development method evaluation frameworks, and requirements for cross-platform tooling, among others. These contributions ought to be regarded as significant foundations for progressive web apps research in the future.

The industry's considerable interest should serve as a drive for additional study and broader academic involvement given the paucity of academic contributions regarding Progressive Web Applications. In Section 7, research ideas are offered to pique the interest of pertinent research communities.

III. METHODOLOGY

PWAs encompass multiple technologies and concepts that work together to enhance user experience. This section provides an overview of the different aspects of PWAs.

The use of techniques to provide a progressive user experience defines a Progressive Web App (PWA), which is essentially a web app. While all PWAs are progressive web apps, not all web apps that are progressive meet the criteria to be considered PWAs. To avoid confusion, this paper uses the upper-case variant and acronym PWA to refer to the collection of technologies and concepts that qualify an app to be a PWA. The lower-case variant is used when addressing concepts more generally. As the number of PWAs increases, a clear denomination will be necessary. A taxonomy of app-enabled devices has been proposed as a reference. Technologies

Service Workers:

Most of the features and concepts (see Sect. 4.3) that separates a PWA from a regular web app are available through Service Workers. A Service Worker is a JavaScript script which performs background operations separate from the rest of the website as it cannot interact with the DOM. It acts as an application-level network proxy, allowing developers to fully intercept network requests and control such as caching of data and assets for offline availability, background synchronization (e.g., fetching data while app is in the background) and registering for push notifications. As per July 2017, Apple's Safari browser does not support the Service Workers API, rendering their platform's users unable to use PWAs to their full extent.

Application Shell (AppShell):

A well-designed and optimized AppShell is of utmost importance for an optimal user experience. This is the first-to-render user interface of the PWA, ensuring that the user experiences immediate rendering of some content to avoid the feeling of a slow app. The AppShell is made up of static assets and interface components, meaning that it should not depend on any external dynamic data to render. Examples of such static assets include toolbars, navigation bars, splash screens and the like developed in HTML, CSS and JavaScript. In contrast, dynamic data requires an asynchronous action to be fetched before render (e.g., network call or local database query). Web App Manifest:

The Manifest is a JSON file web developers can use to configure their PWAs. Within the file, configurations such as app name, icon paths, splash screen image, background colours and display types (e.g., full screen, with or without browser artefacts) can be specified. Such configurations are important for enhancing the app-like feeling of a PWA, a concept further discussed in Sect.

HTTPS:

Progressive Web Apps must be served via HTTPS. This ought to " prevent snooping and to ensure content hasn"t been tampered with.", according to Google's LePage.

Concepts:

These 10 concepts are the foundation of Progressive Web Apps Progressive, Responsive, Installable, Connectivity Independent, App-Like, Fresh, Safe, Discoverable, Re-engageable, Linkable.

The concept of Progressive Enhancement means that a product, such as a website, can progressively become better and more advanced based on the user's browser and device. Responsive design is at the heart of mobile-compliant web design and allows for different form factors to engage with the content of a website optimally.

Progressive Web Apps are installable, which means that they can be downloaded and installed directly from a user's internet browser. They can also manage offline contexts, unlike regular web apps or websites, by leveraging Service Workers, as well as local data and asset caches.

|| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | SJIF: 8.423|| Volume 12, Special Issue 2, April 2023 ||

8th National Conference on Advanced Computing Technologies (NCACT'23)

Organized by

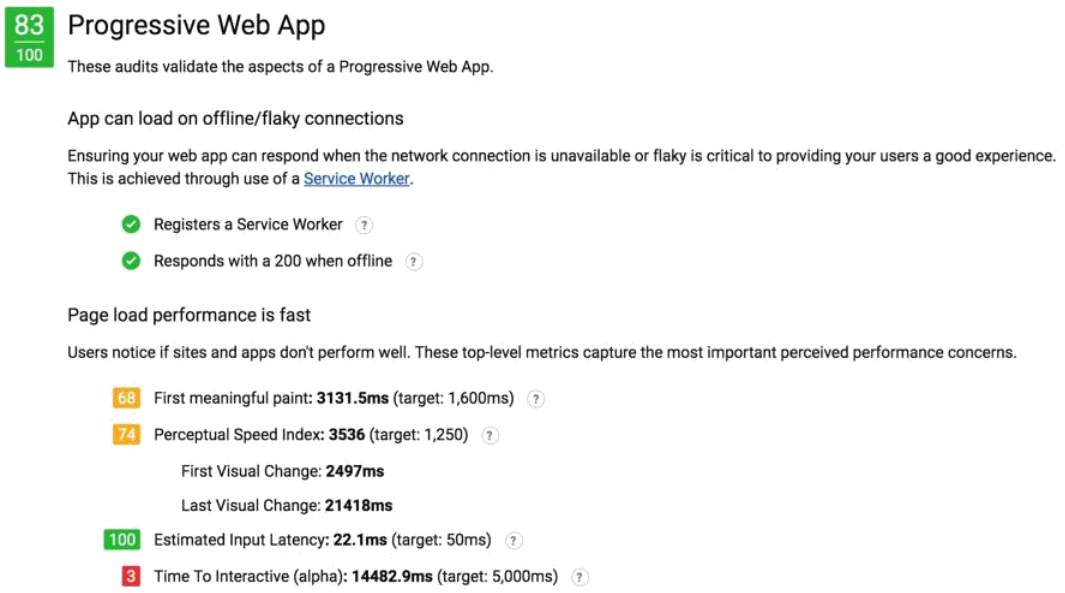
Department of CSE, Velammal Engineering College, Chennnai, India

A well-designed PWA can act, look, and feel like a regular native or cross-platform app. They can fetch new content from an associated web service whenever the app requires it, ensuring that the app is both available offline and online with fresh content.

PWAs must be served via HTTPS for optimal end-user security. Search engines can discover and index the content of Progressive Web Apps, which are linkable and adhere to the same standard as any reachable website. Lastly, PWAs can deliver the same re-engaging experience as regular apps by leveraging push notifications to draw users back into the app.

Lighthouse: PWA Testing Tool:

While not being a technology or concept, the Lighthouse Chrome extension is an official Google product targeting PWA developers. The Lighthouse extension will aid in benchmarking websites against certain measurements and aims Google assess as important, especially for PWAs. While the extension works on non-PWA websites too, it is designed to help optimize websites for better rendering speed, time to first possible user interaction and general compliance with Google's aim for PWAs and the future of (mobile) web experience. Lighthouse will generate a verbose report on the state of the tested website, provide resources on how to optimize code and assets, and give an overall score, as seen in Fig below.



Example of a Lighthouse report, generated using the reddit Art PWA

IV. EXPERIMENTAL RESULTS

In this section we present both technical and more overarching results from our study. The data presented should aspire academia to continue researching Progressive Web Apps and cross-platform development in general. Feature Comparison

The purpose of this section is to compare and highlight the differences between interpreted apps, Progressive Web Apps, hybrid apps, and native apps, while also discussing technical frameworks and unification of user experience. Table 1, presented below, provides a list of features available in PWAs as of July 2017, along with their compatibility. The table is not exhaustive, and further remarks are provided in the subsequent sections.

|| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | SJIF: 8.423|| Volume 12, Special Issue 2, April 2023 ||

8th National Conference on Advanced Computing Technologies (NCACT'23)

Organized by

Department of CSE, Velammal Engineering College, Chennnai, India

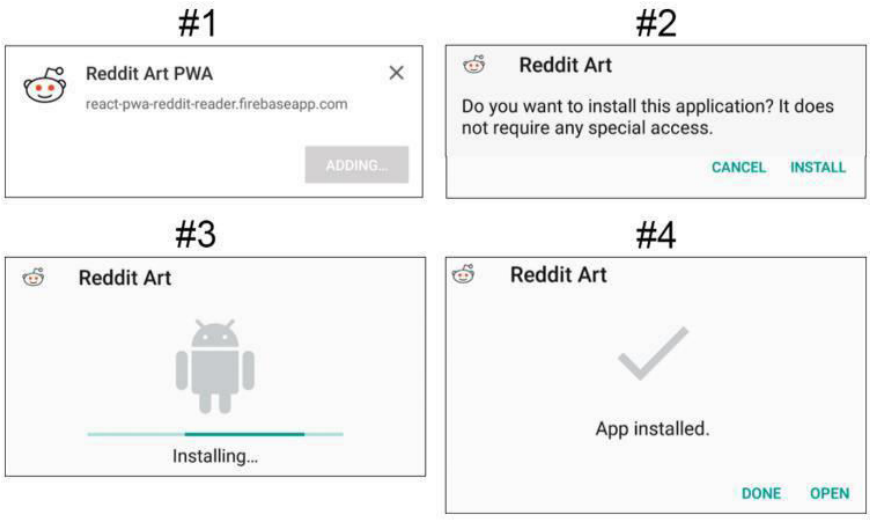
Feature	Interpreted	PWA	Hybrid	Cross-compiled	Native
Installable	Yes	Yes[a]	Yes	Yes	Yes
Offline capable	Yes	Yes	Yes	Yes	Yes
Testable before installation	No	Yes	No	No	No
App marketplace availability	Yes	Yes[b]	Yes	Yes	Yes
Push notifications	Yes	Yes[c]	Yes	Yes	Yes
Cross-platform availability	Yes	Limited[d]	Yes	Yes	No
Hardware and Platform API access	Yes	Limited[e]	Yes[f]	Yes	Yes
Background synchronisation	Yes	Yes	Yes	Yes	Yes

Table 1. Feature-comparison of approaches.

Unification of Mobile App and Web Experiences

A significant difference between web apps and regular mobile apps is their accessibility. While a regular app requires search and installation via an app marketplace, Progressive Web Apps offer the best of both worlds. End-users can easily explore an application through their web browser and then choose to install it via an "Add to Home screen" banner prompted to the user if certain criteria are met or can be displayed programmatically as controlled by the website's developer. The installation process for the PWA developed for this study is shown in Figure below. However, this experience is only achievable by enabling the Improved PWA Installation flag in the Chrome Canary for Android settings as of July 2017. Without this flag enabled, the experience is similar to bookmarking a site to the home screen. The installation prompt, together with the full-screen experience of PWAs, can be seen as an advancement in unification of end-user experience and mobile web perception. Rather than forcing users to download an app from a marketplace, they can first experience the product in their web browser as a regular website before installing it via the banner. However, we suggest that research on potential security concerns should be of interest to all parties. Unification of Desktop App and Web Experiences

Using JavaScript frameworks like Electron.js for cross-platform desktop application development has become increasingly popular in recent years. However, in this section, the authors suggest that Progressive Web Apps (PWAs)



PWA installation flow in Chrome Canary.

|| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | SJIF: 8.423|| Volume 12, Special Issue 2, April 2023 ||

8th National Conference on Advanced Computing Technologies (NCACT'23)

Organized by

Department of CSE, Velammal Engineering College, Chennnai, India

may be the future approach for developing such applications, without the need for Electron.js or similar frameworks. PWAs are not limited to mobile devices and can be deployed on Chrome OS, Windows, and MacOS, with some browsers such as Google Chrome offering desktop PWA support. As of July 2017, a PWA can be installed on a user's desktop via an installation banner and downloaded to the Chrome Apps folder, allowing offline use. However, there is a difference in user experience between desktop and mobile PWAs, with desktop PWAs opening as a new tab in the Chrome browser, unlike mobile PWAs which execute in their own browser-less shell. The authors propose further research on using PWAs as a means of cross-platform desktop development, which should be of interest to both industry and academia. Performance Comparison Between Native, Hybrid, Interpreted, Cross-Compiled and PWA

Table 2 presents a comparison of four different mobile app approaches based on their installation size, launch time, and toolbar render time. The Progressive Web App (PWA) has the smallest installation size, at 104 KB, compared to the other three approaches. Specifically, it is 242 times smaller than the Xamarin-based cross-compiled app (25.19 MB), 157 times smaller than the React Native-based interpreted app (16.39 MB), and 42 times smaller than the native Android app (4.37 MB). However, it is still larger than the Ionic Framework-based hybrid app, which has an installation size of 4.53 MB, making it 43 times smaller than the PWA.

Measure	Native	Hybrid	Interpreted	Cross-compiled	PWA
Size of installation	4.37 MB	4.53 MB	16.39 MB	25.19 MB	104 KB
Android activity launch time	1408 ms	467.5 ms	246 ms	4190 ms	230 ms
Time from app-icon tap to toolbar render	1688 ms	3999 ms	1716 ms	4590 ms	(a) 3152 ms (b) 1319 ms

In terms of render-speeds, the PWA's toolbar render time depends on whether Chrome Canary is running in the background or not. If it is not running in the background, the PWA's toolbar render time is 367 ms, which is faster than the React Native-based app (513 ms) and the Xamarin-based app (671 ms). However, if Chrome Canary is running in the background, the PWA's toolbar render time increases to 1115 ms, which is slower than the React Native-based app (918 ms) but still faster than the Xamarin-based app (2165 ms).

V. CONCLUSION

The full potential of the Progressive Web App approach is dependent on Apple implementing Service Workers into their iOS Safari browser. However, PWAs offer fast user interface rendering times, compliant platform and device API integrations through HTML5 and JavaScript, and require minimal space on devices, making them a promising contender in the cross-platform space. While PWAs lack native user interfaces, they offer unique testing capabilities since they can be tested in any web browser before installation. PWAs also have the ability to perform tasks such as background synchronization and push notification registration, previously only available in native apps and other cross-platform approaches. Additionally, PWAs have the potential to be used for developing desktop apps, with Microsoft and Google already making plans to incorporate them in their platforms. However, there is a lack of academic literature on the subject, presenting opportunities for further research.

REFERENCES

[1] Biørn-Hansen, A., Majchrzak, T.A., Grønli, T.M. (2017). Progressive web apps: the possible web -native unifier for mobile development. In Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST).

[2] Google Developers. (2022). Progressive Web Apps. Retrieved from https://developers.google.com/web/progressive-web-apps

[3] Malavolta, I., Ruberto, S., Soru, T., &Terragni, V. (2015). Hybrid mobile apps in the Google play store: an exploratory investigation. In Proceedings 2nd ACM International Conference on Mobile Software Engineering and Systems, MOBILESoft 2015.

[4] Osmani, A., & Cohen, M. (2017). Offline storage for progressive web apps. Retrieved from https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/offline-for-pwa

[5] Rieger, C., &Majchrzak, T.A. (2016). Weighted evaluation framework for cross-platform app development approaches. In SIGSAND/PLAIS 2016.

|| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | SJIF: 8.423|| Volume 12, Special Issue 2, April 2023 ||

8th National Conference on Advanced Computing Technologies (NCACT'23)

Organized by

Department of CSE, Velammal Engineering College, Chennnai, India

[6] Russel, A., &Berriman, F. (2015). Progressive web apps: escaping tabs without losing our soul. Retrieved from https://goo.gl/e6pZHF
[7] Sharma, R., Chauhan, R., & Sharma, N. (2019). A survey on progressive web applications: Opportunities, challenges and solutions. In Proceedings of the 2019 5th International Conference on Computing Sciences (ICCS).
[8] World Wide Web Consortium (W3C). (2018). Progressive web apps. Retrieved from https://www.w3.org/TR/appmanifest/
[9] A Comprehensive Guide to Progressive Web Apps: https://developers.google.com/web/progressive-web-apps/guide
[10] Why Progressive Web Apps Are The Future Of Web Development: https://www.forbes.com/sites/forbestechcouncil/2021/06/23/why-progressive-web-apps-are-the-future-of-web-development/?sh=54cd2f484bbd
[11] The Benefits of Progressive Web Apps: https://www.smashingmagazine.com/2018/11/benefits-progressive-web-apps/
[12] Building Progressive Web Apps: https://www.udacity.com/course/building-progressive-web-apps--ud811
Best Practices for Building Progressive Web Apps: https://www.digitalocean.com/community/tutorials/best-practices-for-building-progressive-web-apps
[13] "The State of Progressive Web Apps in 2021" by SimiCart: This report provides insights into the current state of progressive web apps and how they are being adopted by businesses. It also includes case studies and examples of successful implementations.
[14] "The State of Progressive Web Apps" by Smashing Magazine: This report provides an overview of the benefits and challenges of building progressive web apps, as well as best practices and real-world examples.
[15] "Progressive Web App Development: Pros and Cons" by Altar.io: This article provides a balanced view of the advantages and disadvantages of building progressive web apps, with a focus on the development process and user experience.
[16] "Progressive Web Apps: A Business Perspective" by Google: This white paper provides insights into the business case for building progressive web apps, including improved engagement, reduced development costs, and increased customer satisfaction.
[17] "The Impact of Progressive Web Apps on SEO and UX" by Moz: This article explores how progressive web apps can improve search engine optimization and user experience, and provides tips for optimizing them.
[18] "Progressive Web Apps: The Future of Mobile Web" by Econsultancy: This report provides an overview of the benefits and challenges of building progressive web apps, as well as case studies and best practices for implementing them.
[19] "Progressive Web Apps: The Definitive Guide" by Scotch.io: This comprehensive guide covers everything you need to know about building progressive web apps, including the development process, user experience, and best practices.

INNO SPACE
SJIF Scientific Journal Impact Factor

SJIF: 8.423

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INTERNATIONAL CENTRE

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details