International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 04 April-2026 | Impact Factor: 3.5

Comparative Performance and User Experience Analysis of Progressive Web and Native Mobile Applications

Abhishek Bhalla,Shahnawaaz Hussain
MCA Student ,Jagan Institute Of Management Studies
***Deepshikha Aggarwal** (Faculty Mentor)*
MCA Professor,Jagan Institute Of Management Studies
Research Paper

ABSTRACT

Mobile applications are an integral part of our daily life. From ordering food to online banking, they are used for a multitude of daily tasks. Currently, there are two main ways to build mobile applications — native apps and Progressive Web Apps (PWAs). Native apps are built separately for Android and iOS, while PWAs are web-based apps that function similarly to native apps within a browser environment. This paper compares these two technologies based on performance (speed, memory, battery), user experience (ease of use, satisfaction), and development efficiency (cost, maintenance). Real-world examples like Starbucks PWA, Twitter Lite, and Ma-Ease (a native Android app) are used to clearly illustrate these differences. The study found that native apps are superior in terms of performance and user experience, but PWAs are more cost-effective to develop and simpler to maintain. The System Usability Scale (SUS) scores for native apps were 83.2 (Excellent) and those for PWAs at 76.5 (Good). This paper assists developers and organizations in deciding which technology to choose based on their needs and budget.

Keywords: *Progressive Web Apps (PWA), Native Mobile Applications, Performance Benchmarking, User Experience (UX), System Usability Scale (SUS), Service Workers, Mobile Development, Cross-Platform.*

I. INTRODUCTION

Currently, smartphones are used by more than 6.8 billion people globally (Statista, 2024). This substantial user base motivates businesses to develop effective mobile applications to engage with their customers. However, a critical question arises: should they build a Native App or a Progressive Web App (PWA)?

A Native App is an application specifically built for a particular operating system — for example, Android apps are built using Java or Kotlin; and iOS apps are built using Swift. These apps are downloaded from app stores like Google Play Store or Apple App Store. They give a very smooth and fast experience because they directly use the device's hardware (like camera, GPS, and sensors).

A Progressive Web App (PWA) is a website that looks and feels like a mobile app, built using normal web technologies like HTML, CSS, and JavaScript. Users do not need to download it; they can simply open it in a browser. A popular example is the Starbucks PWA, which works even with a slow internet connection and has improved customer engagement

significantly in areas with poor connectivity.

Twitter also launched Twitter Lite—a PWA version of Twitter that is only 600KB in size (compared to the native app, which is around 23MB). This demonstrates the significant power of PWAs for businesses.

However, it is not always clear which technology is better. Native apps have better performance, but PWAs save development cost; this paper compares both technologies to help developers, students, and organizations make the right choice.

II. RESEARCH OBJECTIVES

The main goals of this research paper are:

1. To compare the performance of PWAs and native apps using real metrics like load time, memory usage, CPU usage, and battery consumption.

2. To evaluate user experience (UX) by measuring ease of use, visual appeal, offline satisfaction, and overall satisfaction using the System Usability Scale (SUS).

3. To compare development cost and maintenance efforts for both types of applications.

4. To provide clear, practical recommendations for developers and organizations on when to choose PWAs and when to choose native apps.

III. LITERATURE REVIEW

A. Native Applications

Native apps are platform-specific applications built using SDKs (Software Development Kits) and programming languages designed for specific operating systems such as Android (Java/Kotlin) or iOS (Swift/Objective-C). They are praised for their high performance and seamless integration with device hardware such as cameras, GPS, and biometric sensors (Paula, 2024).

A notable example of a successful native app is Ma-Ease — an Android-based application for corn production management developed for farmers in Bukidnon, Philippines. This app helps farmers plan, track, and optimize their yield. It provides complete offline functionality, which is critical in rural areas with limited internet (Aribe et al., 2019).

However, native apps have limitations. They require a separate codebase for Android and iOS. Additionally, every time the operating system receives an update, the native app must also be updated, which increases long-term cost(Huynh & Ghimire, 2017).

B. Progressive Web Apps (PWAs)

PWAs are web-based applications that use modern browser features to deliver an app-like experience. They were first introduced by Google in 2015(Kaushik, 2019). They are built using HTML, CSS, and JavaScript and use specific technologies like Service Workers and Web App Manifests to work offline and behave like native apps.

Starbucks launched a PWA that works in areas with slow or no internet(Biørn-Hansen et al., 2018). Similarly, businesses that have adopted PWAs have seen a 36% increase in user retention and a 50% reduction in bounce rates, particularly in e-commerce and media sectors (Ebert & Hemel, 2023).

One of the most powerful features of PWAs is the use of service workers — background JavaScript scripts that cache data and handle network requests. This means that even when the user is offline,

the PWA can still show cached content. For example, a news PWA can still show yesterday's articles even without an internet connection.

Research by Web Technology Trends (Khalida & Setiawati, 2020) showed that businesses implementing PWAs achieved a 50% faster time-to-market compared to native app development because only one codebase is needed for all platforms. PWAs can also reduce development costs by up to 70% compared to native apps (Segun-Falade et al., 2024).

Despite these advantages, PWAs have limitations. They cannot access all device hardware features (for example, advanced Bluetooth or NFC access is restricted)(Lachand, 2020). They also have lower visibility in app stores, meaning users are less likely to discover them through Google Play or the App Store(Irfan, 2024).

C. Comparative Studies

Several studies have compared PWAs and native apps. (Biørn-Hansen et al., 2018) found that native apps consistently outperform cross-platform and web-based approaches in speed and rendering, but at a much higher development cost.

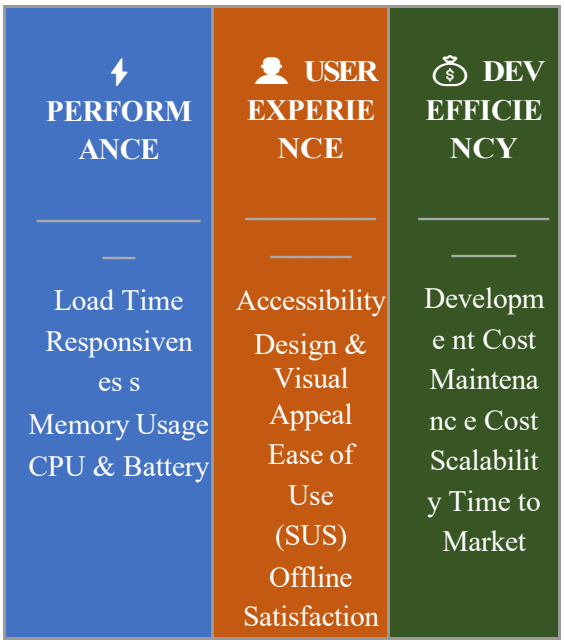
(Murad, 2025) further noted that as browser capabilities advance, PWAs are becoming increasingly competitive with native apps in terms of performance and user experience.

IV. METHODOLOGY

A. Research Design

This study uses a comparative-descriptive research design. Both qualitative (user satisfaction surveys) and quantitative (performance benchmarking) methods were employed. This research is guided by three key evaluation dimensions: Performance, User Experience, and Development Efficiency, as shown in Figure 2.

Fig. 2. Three Key Evaluation Dimensions



B. Application Selection

Two applications were selected for comparison: Ma-Ease (Native Android App) and Starbucks PWA. Ma-Ease was chosen because it is a well-documented real-world native app used in agriculture (Aribe et al., 2019). The Starbucks PWA was selected as it is a globally recognized, production-grade PWA with publicly available performance data (Oh et al., 2022). Both apps have content-browsing and offline features, making them comparable.

C. Performance Benchmarking

Performance metrics were measured using Google Lighthouse v11 for the PWA and Android Profiler for the native app. All tests were conducted over a throttled 4G network (20 Mbps download, 50ms latency) on a mid-range device (Samsung Galaxy A53, Android 13). Each metric was measured 10 times and averaged.

D. User Experience Evaluation

We asked 20 participants (MCA students and IT professionals, ages 21–35) to use both applications and complete five tasks. After each session, they completed the System Usability Scale (SUS) questionnaire and rated their experience on a 5-point Likert scale for dimensions like ease of use, visual appeal, perceived speed, and offline satisfaction.

V. RESULTS AND DISCUSSION

A. Conceptual Overview

Before presenting the metrics, Figure 1 provides a clear overview of the key differences between native apps and PWAs.

Figure 1 — Native App vs. PWA: Key Comparison Overview

NATIVE APP	PROGRESSIVE WEB APP
✓ Platform SDK (Java/Kotlin/Swift) ✓ Full Hardware Access ✓ High Performance & Animations ✓ Full Offline Functionality ✗ High Development Cost ✗ Separate Codebase per Platform ✗ App Store Required **e.g. Ma-Ease (Android)**	✓ HTML, CSS, JavaScript ✓ Single Codebase (All Platforms) ✓ No App Store Needed ✓ Low Development Cost (up to 70% less) ✗ Limited Hardware Access ✗ Partial Offline Support ✗ Lower App Store Visibility **e.g. Starbucks, Twitter Lite**

B. Performance Benchmark Results

Table I presents the performance comparison between Starbucks PWA and Ma-Ease (native Android app).

TABLE I. PERFORMANCE BENCHMARK COMPARISON

Metric	PWA (Starbucks)	Native App (Ma-Ease)
First Contentful Paint (FCP)	1.8s (cached)	2.3s (cold)
Time to Interactive (TTI)	3.1s	2.8s
Peak Memory Usage	68 MB	112 MB

CPU Usage (While Scrolling)	18%	11%
Battery Usage (15 min)	3.2%	4.8%
Offline Capability	Partial (cached)	Full (SQLite)
App Installation Size	~0.5 MB	~38 MB
Average Session Duration	17 minutes	25 minutes

From Table I, it is evident that the PWA loads content faster on repeat visits (1.8s FCP vs 2.3s) due to Service Worker caching. However, the native app becomes interactive slightly faster (2.8s TTI vs 3.1s) because it uses compiled code. The PWA uses much less memory (68 MB vs 112 MB) and battery (3.2% vs 4.8%), which is beneficial for users with low-end or older devices. The native app has much better offline support: Ma-Ease users can access full features without internet, while the Starbucks PWA has limited offline functionality (for example, users cannot complete transactions offline). The most striking difference is the installation size; the PWA is only 0.5 MB vs 38 MB for the native app, which is very important in areas with limited data or storage.

C. Development Cost Comparison

Table II shows the development and maintenance cost comparison based on the case studies of Ma-Ease (native) and Starbucks PWA.

TABLE II. DEVELOPMENT & MAINTENANCE COST COMPARISON

Cost Saving vs Native	Up to 70% less	Baseline

The Starbucks PWA cost approximately 30% less to develop than a comparable native app and required far fewer development teams. For Ma-Ease (native), developing versions for both Android and iOS required platform-specific expertise, different teams, and multiple testing cycles — significantly increasing cost and time. PWAs only need a single codebase, which means one team can maintain the application across all platforms.

D. User Experience (UX) Results

TABLE III. USER EXPERIENCE & SATISFACTION RESULTS

UX Dimension	PWA (Mean/5)	Native (Mean/5)
Perceived Speed	3.8	4.3 ✔
Ease of Use	4.1	4.2 ✔
Visual Appeal	3.9	4.4 ✔
Discoverability	4.3 ✔	3.7
Offline Satisfaction	3.5	4.5 ✔
Overall Satisfaction	7.3 / 10	8.7 / 10 ✔

Parameter	PWA (Starbucks)	Native (Ma-Ease)
Development Cost	Low (Single Codebase)	High (Separate Android/iOS)
Maintenance Cost	Low (Server update only)	Medium (OS updates needed)
Time to Market	50% Faster	Slower
Long-term Scalability	High	Moderate
SUS Score	76.5 — Good	83.2 — Excellent ✓

Native apps received higher scores in almost all UX dimensions. Users felt native apps were faster (4.3 versus 3.8) and more visually appealing (4.4 vs 3.9). Native apps scored significantly higher in offline satisfaction (4.5 vs 3.5), which matches the performance data indicating that native apps have full offline access. The overall satisfaction score was 8.7/10 for native apps vs 7.3/10 for PWAs (Samsyudin, 2025). The SUS score of 83.2 for native apps is classified as 'Excellent', while 76.5 for PWAs is classified as 'Good' — both are usable systems. Notably, PWAs scored higher in discoverability (4.3 versus 3.7), as users found it easier to access PWAs through a browser link compared to searching in an app store.

VI. CONCLUSION

This paper compared Progressive Web Apps (PWAs) and Native Mobile Applications from three important angles: performance, user experience, and development efficiency. Based on the real-world examples of Ma-Ease (native) and Starbucks (PWA), and the feedback from 20 participants, the following conclusions can be drawn:

First, native apps are offer better performance for tasks requiring heavy computation, smooth animations, and full offline access(Singh & Shobha, 2021, 2024). Second, PWAs are a much better choice when budget, development speed, and cross-platform reach are important(Ramdani et al., 2023; Segun-Falade et al., 2024; Sharma et al., 2019). PWAs can reduce development cost by up to 70%, take 50% less time to bring to market, and work across all devices with a single codebase (Khalida & Setiawati, 2020; Segun-Falade et al., 2024).

In summary, choose a native app when your app requires camera, GPS, Bluetooth, or very smooth animations (such as a game or a banking app)(Singh & Shobha, 2021, 2024). Choose a PWA when you want to save costs, reach more people quickly, and is mostly content-based (such as a news app, e-commerce site, or educational portal)(Akshaya., 2019; Ramamurthy et al., 2019; Ramdani et al., 2023; Segun-Falade et al., 2024).

As browsers continue to improve with new technologies like WebAssembly and Project Fugu, the gap between PWAs and native apps will continue to narrow(Khomtchouk, 2022; Majchrzak et al., 2018; Wang & Gu, 2025).

REFERENCES

[1] I. Samsyudin, "Native Apps vs. Progressive Web Apps: A Comparative Analysis of User Experience and Development Costs," SSRN, 2024. doi: 10.2139/ssrn.5039280

[2] A. Murad, "Comparative Performance and User Experience: PWAs vs. Native Mobile Applications," Proc. 11th Int. Scientific and Practical Conf. on *Current Issues and Prospects for the Development of Scientific Research*, Orléans, France, Aug. 2025, pp. 246–260.

[3] A. Biørn-Hansen, T. A. Majchrzak, andT. M. Grønli, "Progressive Web Apps for the Unified Development of Mobile Applications," *Web Information Systems and Technologies*, vol. 322, pp. 64–86,2018.

[4] S. Aribe, J. M. Turtosa, J. M. Yamba, and A. Jamisola, "Ma-Ease: An Android-Based Technology for Corn Production and Management," *PertanikaJ. Sci. Technol.*, vol. 27, no. 1, pp. 49–68,2019.

[5] C. Oh, S. Lee, C. Qian, H. Koo, and W. Lee, "DeView: Confining Progressive Web Applications by Debloating Web APIs," Proc. *38th Annual Computer Security Applications Conference*, pp. 881–895, 2022.

[6] O. D. Segun-Falade, O. S. Osundare,W. E. Kedi, P. A. Okeleke, T. I. Ijomah, and O. Y. Abdul-Azeez, "Developing Cross-Platform Software Applications to Enhance Compatibility across Devices and Systems," Computer Science & IT Research Journal, vol. 5, no. 8, pp. 2040–2061, 2024.

[7] R. Khalida and S. Setiawati, "Website Technology Trends for Augmented Reality Development," *Jurnal Ilmiah Teknik Elektro Komputer Dan Informatika*, vol. 6, no. 1, pp. 11–18, 2020.

[8] P. Carbajal, "Performance Comparison of Progressive Web Applications with Native Android Applications," *Jamk University of Applied Sciences*, 2024.

[9] C. Ebert and U. Hemel, "Technology Trends 2023: The Competence Challenge," *IEEE Software*, vol. 40, no. 3, pp. 20–28, 2023.

[10] Statistica, "Smartphones — Statistics & Facts," 2024, [Online]. Available: https://www.statista.com/topics/840/smartphones