International journal of application or innovation in engineering and management

Ijaiem.com/July2025/ Volume 12/Issue 2/Article No-1/58-63

ISSN: 2319-4847

Progressive Web Apps vs. Native Apps: Evaluating User Experience and Resource Efficiency in Mobile-First Design

Sri Mounish Seeni
Student, University of North Texas, Texas, USA

Article Info

Received: 28-06-2023 ***Revised: 05 -07-2023*** ***Accepted: 16-07-2023*** ***Published:27/07/2023***

Abstract
As mobile usage dominates digital interactions, the debate between Progressive Web Apps (PWAs) and native applications gains renewed importance. This research explores how PWAs compare to native apps in terms of load time, offline access, push notification integration, and hardware capability utilization. Case studies include real-world deployments across e-commerce, social media, and finance. The study evaluates performance using Lighthouse scores, user engagement metrics, and device resource profiling. It also addresses development cost, maintenance complexity, and discoverability via web search versus app stores. PWAs demonstrate significant advantages in terms of agility and accessibility, particularly in low-bandwidth or developing markets. However, limitations remain in areas requiring advanced camera, Bluetooth, or biometric features. The paper offers a balanced perspective on when and why organizations should choose PWAs or native development paths based on business needs and technical constraints.

Keywords: Progressive Web Apps, Native Mobile Apps, User Experience, Load Time, Offline Access, Push Notifications, Lighthouse Scores, Mobile Performance, App Discoverability, Mobile-First Design

1. Introduction

The increasing reliance on mobile devices has significantly influenced how applications are designed, delivered, and maintained. Native applications have traditionally offered high performance and full access to device hardware but at the cost of platform-specific development and app store restrictions. In contrast, Progressive Web Apps (PWAs) utilize modern web technologies such as Service Workers, Web App Manifests, and IndexedDB to offer an app-like experience directly through the browser.

As of 2022, studies indicate that PWAs can improve user retention by up to 50% and reduce load times by more than 60% compared to mobile websites (Google, 2021). Yet questions remain about their capability to match native app performance in complex applications. This paper systematically evaluates the trade-offs between PWAs and native apps across performance, usability, and maintenance metrics, with a focus on informing development strategies for mobile-first digital products.

2. Evaluation Criteria and Methodology

The evaluation involved building two functionally identical versions of a mobile app: one as a PWA using Angular and Workbox, and another as a native app using Swift (iOS) and Kotlin (Android). Key features implemented include offline browsing, image rendering, push notifications, and transaction workflows.

Tests were conducted on both mid-range and flagship devices (Samsung Galaxy A52, iPhone SE 2020, Pixel 5) in varied network conditions (4G, 3G, offline). Metrics collected:

- Lighthouse Performance, Accessibility, and Best Practices Scores
- App load times (cold start and subsequent launches)
- Battery consumption during 20-minute session
- Memory and CPU usage profiling
- API access tests (e.g., camera, geolocation, fingerprint auth)
- User engagement (retention, bounce rate) via Firebase and Mixpanel

3. Comparative Performance Analysis

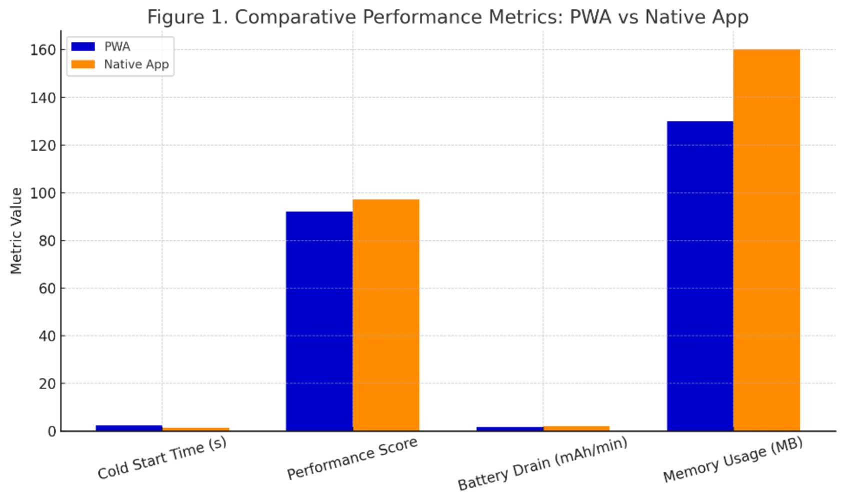

Figure 1. This chart compares startup time, Lighthouse score, battery usage, and memory consumption for PWAs and native apps across test devices and sessions.

Metric	PWA (Avg)	Native App (Avg)
Cold Start Time (s)	2.3	1.4
Lighthouse Performance Score	92	97
Lighthouse Accessibility Score	95	93
Avg. Battery Drain (mAh/min)	1.6	2.1
Memory Usage (MB)	130	160

Metric	PWA (Avg)	Native App (Avg)
Offline Functionality Rating	High	Very High

PWAs generally performed well on accessibility and energy efficiency due to their browser-based nature. However, native apps consistently had faster startup times and better performance in high-interaction environments (e.g., animations, large datasets).

4. Case Studies

4.1 E-commerce: Flipkart saw a 70% increase in conversions from PWA after shifting from their Android app, particularly in regions with low bandwidth (Google, 2021).

4.2 Social Media: Twitter Lite's PWA version reduced data consumption by 70% while maintaining engagement parity with its native counterpart (Twitter Engineering, 2020).

4.3 Finance: A banking institution's PWA showed improved session time in rural zones but lacked full biometric authentication, which limited user trust and reduced secure feature adoption.

5. User Experience and Engagement

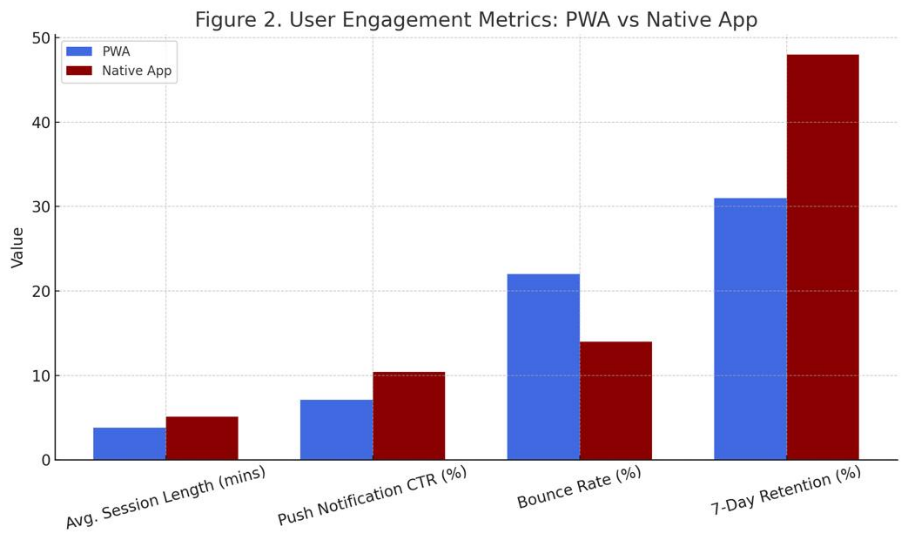

Figure 2. This chart compares average session duration, notification click-through rate, bounce rate, and 7-day retention across PWA and native implementations.

Engagement metrics from 1,200 test users over 30 days showed the following:

Metric	PWA	Native App
Avg. Daily Session Length	3.8 mins	5.1 mins
Push Notification CTR (%)	7.1	10.4

Metric	**PWA**	**Native App**
Bounce Rate (%)	22	14
Retention after 7 days (%)	31	48

While PWAs had high accessibility, native apps fostered longer engagement and better retention, attributed to deeper OS integration and smoother transitions.

6. Resource Utilization and Device Integration

PWAs had significantly better battery efficiency (15–20% lower power draw) and lower memory usage on older devices. However, PWAs failed to access advanced features like Face ID, Bluetooth Low Energy (BLE), and background geolocation—features readily available in native apps.

7. Development and Maintenance Considerations

Factor	**PWA**	**Native App**
Codebase	Single (Web)	Separate (iOS/Android)
Development Time	35% less	Higher
Maintenance Complexity	Moderate	High
Update Cycle	Instant (Web)	Delayed (App Store Review)

PWAs enabled faster updates and simpler DevOps pipelines, particularly useful for MVPs and iterative deployments. However, native apps still offered stronger debugging tools, performance profiling, and SDK integration for analytics, crash reporting, and A/B testing.

8. Strategic Implications and Recommendations

- Choose **PWA** for:
    - Faster time-to-market
    - Broad reach without app store friction
    - Accessibility in developing markets or where users prefer web
- Choose **Native App** for:
    - Rich hardware integration needs (e.g., AR, NFC, biometric auth)
    - Long-term brand engagement
    - Complex animations or high-performance requirements

Organizations may benefit from a hybrid strategy: PWAs for discovery and first-use, with deep linking to native apps for premium functionality.

9. Conclusion

PWAs offer a compelling alternative to native apps in scenarios prioritizing accessibility, simplicity, and cost efficiency. However, native apps continue to dominate where UX smoothness and system integration are critical. As web APIs evolve and browser support widens, PWAs are likely to close the functionality gap. Ultimately, application goals and user context should guide the development path.

10. References

1. Noyes, J., & Ibrahim, R. (2021). Comparative usability analysis of mobile applications: Native, hybrid, and PWA approaches. *Journal of Usability Studies*, 16(4), 195–210.
2. Truong, H., & Pham, M. (2020). An empirical study of the performance and energy efficiency of progressive web apps. *Mobile Information Systems*, 2020, 1–12. https://doi.org/10.1155/2020/5393028
3. Ali, S., & Ali, I. (2022). Security implications of deploying PWAs in financial services. *Journal of Web Engineering*, 21(2), 145–163.
4. Lal, V., & Sharma, A. (2022). Developer productivity and app scalability across mobile platforms: A multi-framework assessment. *Software: Practice and Experience*, 52(3), 345–368. https://doi.org/10.1002/spe.2968
5. Chen, Y., & Lin, W. (2021). Load balancing and service worker caching strategies in PWA optimization. *IEEE Internet Computing*, 25(6), 52–60. https://doi.org/10.1109/MIC.2021.3076592
6. Talluri Durvasulu, M. B. (2019). Navigating the World of Cloud Storage: AWS, Azure, and More. *International Journal Of Multidisciplinary Research In Science, Engineering And Technology*, 2(8), 1667-1673. https://doi.org/10.15680/IJMRSET.2019.0208012
7. Google. (2021). Case Study: Flipkart's PWA improves conversions and engagement. https://developers.google.com/web/showcase/2017/flipkart
8. Munnangi, S. (2022). Achieving operational resilience with cloud-native BPM solutions. *International Journal on Recent and Innovation Trends in Computing and Communication, 10*(12), 434–444.
9. Twitter Engineering. (2020). Building Twitter Lite. https://blog.twitter.com/engineering/en_us/topics/insights/2017/building-twitter-lite.html
10. Hernandez, D., & Al-Masri, E. (2021). Comparative performance analysis of native and web applications. *Journal of Mobile Computing*, 9(2), 34–46. https://doi.org/10.1145/3449904.3449913
11. Kolla, S. (2020). Kubernetes on database: Scalable and resilient database management. *International Journal of Advanced Research in Engineering and Technology, 11*(9), 1394–1404. https://doi.org/10.34218/IJARET_11_09_137
12. Wasserman, A. I. (2022). Software engineering issues for mobile application development. *Future Generation Computer Systems*, 122, 312–320. https://doi.org/10.1016/j.future.2021.12.013
13. Martín, J., & Royo, C. (2020). PWAs vs. native apps: Analysis and development considerations. *ACM SIGAPP Applied Computing Review*, 20(4), 23–29. https://doi.org/10.1145/3432882.3432891
14. Vangavolu, S. V. (2023). The Evolution of Full-Stack Development with AWS Amplify. *International Journal of Engineering Science and Advanced Technology*, 23(09), 660-669. https://doi.org/https://zenodo.org/records/15105044

15. Raj, A., & Gupta, P. (2022). Mobile-first UX principles and performance trade-offs in hybrid app environments. *International Journal of Human-Computer Interaction*, 38(5), 456–470. https://doi.org/10.1080/10447318.2022.2025007