International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V5I2P119
Eureka Vision Publication | Volume 5, Issue 2, 174-183, 2024

Original Article

Progressive Web Apps: Offline UX Benchmarking

Kavya Muppaneni
Software Engineer at HCL Global Systems, USA.

Abstract - Progressive Web Apps (PWAs) provide a hybrid model that delivers native-like stability to the web, thus, they are becoming increasingly significant for users who operate in low-connectivity, intermittent, or offline environments. The mobile-first trend that is growing globally especially in areas with weak network infrastructure coupled with the continuous shift of mission-critical workflows to web platforms has escalated the need for robust offline-first user experiences. Nevertheless, there is still no set standard for a framework that would measure the quality of offline UX in PWAs. This research does the work of offline experience evaluation in chosen PWAs to close that gap and also suggests a structured benchmarking model. The study, which used a mixed-method approach involving heuristic evaluation, performance testing of caching and synchronization mechanisms, and comparative analysis of applications from various domains, found that the readiness for offline usage, the fallback design patterns, and the support for task continuity varied significantly. Consequently, these findings lead to the creation of practical guidelines and a set of unified metrics for offline UX benchmarking that give practitioners more clarity on how to design resilient experiences. Recommendations for the next research to be carried out, such as the creation of automated testing tools and the extensive validation of the proposed framework for the new PWA technologies, are outlined in the final section of the study.

Keywords - Progressive Web Apps, Offline User Experience, Benchmarking, Service Workers, Web Performance, Offline-First Design, Reliability, Web Applications, User Experience Metrics, Front-End Engineering.

1. Introduction

1.1. Background: Rise of Pwas

The web journey has been through several essential milestones: changes began with basic static websites, then evolved into dynamic and interactive web applications, and finally led to the creation of Progressive Web Apps (PWAs). PWAs are a significant change in how users perceive the internet as they deliver the features of a regular website to users with the added benefits of a native app in terms of reliability, speed, and engagement. With the advancement in browser capabilities and API standardization, PWAs have been successful in bridging the long functional gap between web pages and installable mobile apps.

The foremost reason that led to the dramatic escalation of PWA adoptions is the ever-increasing app-like experience demand without the trouble of installing them via app stores. No matter what device or network users have, they expect their interactions to be fast, resilient, and seamless, and a PWA happens to be the perfect way out of this by being able to install directly from the browser, taking less space and still carrying out high-performance interactions.

A technological breakthrough that has enabled PWAs is the invention of Service Worker- a background script that can intercept network requests, manage caching, and perform other operations beyond the page lifecycle. By way of Service Workers developers can implement a whole range of offline scenarios hence users can store essential assets, get cached content in case of no network, and at the same time, control fallback behavior. With this offline support as a new feature, PWAs differ significantly from traditional web apps that are historically unavailable or cannot be used when there is unstable connectivity. Thus, many companies have adopted the PWA model as a means to get to the global market, a great number of whom are the people facing network issues most of the time.

1.2. Challenges

However,along with their attractive features, PWAs still grapple with a plethora of thorny issues, resulting in a hodgepodge of offline user experiences that lack coherency and continuity. Hopefully, the main culprit is the web itself, which still hinges heavily on network availability, thus limiting its reach far beyond the horizon of nowadays ubiquitous wireless technology. Traditional web apps are often put in a tight spot when their connections go unstable or are even lost, thus their users have to endure interrupted tasks, broken UI states and, naturally, frustration caused by the lack of continuity. Fortunately, Service Workers do offer some relief in this matter, but on the other hand, the sheer scale of support one has to provide to cover all possible offline scenarios daunts implementation efforts.

Secondly, technological advances have led consumers to set higher bars for apps, and subsequently, raise their expectations of PWAs accordingly. Being aware that mobile-native apps boast of reliability and continuity, users thus

anticipate PWAs to be on par with them even if the web infrastructure is shaky or unstable. In cases when PWAs fail to deliver, users lose their trust and consequently disengage at a rapid pace.

Thirdly, the mechanisms designed to accommodate offline scenarios through cached data impose irreducible complexity upon development teams. In the decision-making process of which strategy to employ among cache-first, network-first, or stale-while-revalidate, developers need to take into account factors such as freshness, performance, and storage since these come with inherent trade-offs. Therefore, a falsely set up configuration may result in a corrupted data source, a cache being too large and even slow loading times.

Fourth, synchronization is a concept that becomes especially difficult to realize once users go offline and perform actions. When they get back online, PWAs are responsible for not only updating the changes made locally to the server-side data but also doing so in a way that doesn't end up with conflicts, duplications or any other type of work loss. This is absolutely crucial for collaborative or transactional applications.

Fifth, offline modes reliant on large cached data and background operations, such as logic handling, could potentially be resource-heavy and thus negatively affect the performance of devices with less memory or processing power. Badly implemented caching could slow your first load performance, at the same time, making less storage space available or creating states that are different from each other.

Aside from technical problems, there are still some unresolved design-related issues. Designers, taking offline UX patterns as poorly standardized, are left with the task of determining how best to communicate the connectivity status, error states, and task continuity. Present web performance tools such as Lighthouse, WebPageTest, or PageSpeed Insights only partially cover the offline scenarios, as they are mostly concentrated on load-time metrics from the perspective of offline usability. Thus, the field is devoid of comprehensive tools that would enable it to assess end-to-end offline experience.

1.3. Problem Statement

While PWAs establish the technical basis for strong offline scenarios by and large through Service Workers, the Cache API, and client-side data storage, the offline UX quality is different for each PWA. There are some PWAs that have a smooth offline user flow, while there are also some which just minimally and inconsistently support the offline mode although they have declared that they can work offline. Currently, there is no single benchmarking method to measure an app's usability, resilience, or performance when it is offline across different PWAs. The paper and studies to date mainly focus on the technical side of things and the building strategies, and they do not provide a direct comparison of the offline UX of various real-world PWAs. This gap restricts the ability of practitioners to find the best methods, to understand the causes of the UX problems, and to make their decisions based on data.

Therefore, the central research problem that arises is the following: How can offline UX in PWAs be systematically measured, evaluated, and compared so as to provide a meaningful level of understanding to designers, developers, and researchers?

1.4. Motivation

However, mobile-first world, connectivity can be very unpredictable and this is mostly the case in rural regions, developing markets, or when one is on the move. For a great number of users, network conditions that are intermittent or of poor quality are a matter of their everyday lives, and not just a few rare situations. Businesses are becoming more and more dependent on the maintenance of stable user flows, i.e., particularly in e-commerce, educational platforms, field-service operations, and productivity tools, where it is necessary that a loss of connection does not result in a loss of functionality. Thus, reliable offline UX is very important not only to the gratification of users but also to business continuity, operational efficiency, and global accessibility.

Nevertheless, the lack of standard benchmarks makes it impossible for teams to assess or improve their offline experiences in a consistent manner. A unified methodology that can be applied to all projects or studies is something that is lacking both by industry practitioners and academic researchers. It is very important to come up with such a benchmarking framework to have PWA usability standards improved and to get the adoption of offline-first design principles spread rapidly.

2. Literature Review

2.1. PWA Architecture and Core Technologies

Progressive Web Apps are essentially based on a set of standard web technologies that enable such apps to behave like native apps in terms of performance, they are resilient, and can be installed. The core of this architecture is the Service Worker, a background script that Service Worker interacts with Service Worker lifecycle install, activate, and fetch events affects the support for offline access in the application as well as the approach for caching and resource updates. Different caching strategies like cache-first, network-first, stale-while-revalidate, and cache-only offer different benefits with respect to

freshness, performance, and predictability. The Web App Manifest is a complementary resource to service workers that provides metadata like icons, display mode, or start URL, which allow the app to be installed and the app-like presence to be created on user devices. To ensure always-available local storage, PWAs may implement IndexedDB which is a low-level NoSQL database especially designed to provide persistent storage of structured data, content synchronization after re-connection, and support for offline write operations. All of these capabilities are enabled by innovations in core web technologies.

2.2. Offline Experience in Web Applications

Offline first is a networking philosophy that promotes the idea of designing applications that can still deliver their usable functionality when the user is offline hence treating the network as an option rather than a requirement. The approach has been adopted by many designers and developers because end-users are exposed to poor network conditions more often than before. Some of the UX features that facilitate user interactions even in the absence of a connection are fallback pages that notify about the offline status of the app, skeleton screens that give an illusion of continuity, and local editing modes that allow users to make changes offline with later synchronization. Web apps are inferior to native apps in terms of the level of their offline capabilities since native platforms provide powerful APIs for local storage, background sync, and conflict resolution. Nevertheless, PWAs have taken some steps in this direction by offering similar capabilities to the browser thus closing the gap between them and native apps. The problem of lack of a single offline design standard still remains and developers who work on implementing offline features are most often inconsistent in their approach which in turn leads to limited user trust and cross-application familiarity.

2.3. Existing Performance and UX Benchmarks

The existing benchmarking instruments are only a few ways partially to solve the problem of offline scenarios. Lighthouse, a tool for auditing PWAs by Google, features an "offline readiness" score but evaluates very little of the actual UX of offline, mainly concentrating on whether or not a Service Worker is present and if some files are cached. This binary evaluation does not recognize the quality of the fallback, the continuity of the task, the synchronization, or the user satisfaction. In the same way, Google's Web Vitals framework is designed to measure the speed of a page, its interactivity, and its visual stability but does not take into account the offline usability or reliability in a deteriorated network condition. Literature on web performance has traditionally focused on latency, throughput, and resource optimization without diving into the issues of offline fidelity or resilience. Very few of them adopt a user-centered approach that accounts for how users understand and handle offline scenarios. The absence of comprehensive metrics hinders the ability of practitioners to significantly enhance or even properly evaluate offline experiences.

2.4. Studies on Resilience and Failover

Compared to application resilience, research talks about less tangible methods of deriving such as caching the most necessary assets, the implementation of the gradual declination principle, and the activation of the background synchronization. To be actually effective, an offline experience needs the technical fallback and, at the same time, must inform the user in a transparent way about what can be done. Modern apps show different degrees of success in the implementation of this concept. For instance, with the help of highly sophisticated synchronization and conflict-resolution mechanisms, Google Docs provides near-total offline editing. The Spotify PWA supports offline playback by storing audio files locally and thus is a good example of how PWAs can be used for media-rich scenarios. The PWA for Starbucks is a great example of how the process of ordering offline has been optimized and how the performance of an environment with limited connectivity is improved by the implementation of a lightweight caching strategy. These are just a few examples that not only illustrate what PWAs are capable of but also reveal the difficulties in achieving stable resilience across various domains.

2.5. Identified Research Gap

While there has been a lot of talk about an offline-first approach, one comprehensive framework that can be used to benchmark offline user experience across PWAs is still missing. Most of the assessments are based on technical aspects rather than user-centered outcomes, and there are very few empirical comparisons of real-world PWAs. Consequently, designers and developers are unfamiliar with a standardized method that can be used to evaluate the offline usability, spot the shortcomings, or implement the best practices. The existence of this void is the reason for the emergence of systematic offline UX benchmarking which combines technical performance, interaction design, and user perception across the varied applications.

3. Proposed Methodology

3.1. Research Framework Overview

To systematically quantify offline experience in Progressive Web Apps, this research proposes a three-layer evaluation model that combines technical evaluation, performance measurement, and user-oriented usability study. The aim is to cover the widest range of offline readiness from the service worker-level behaviors to the users' experience of traffic interruptions.

Technical Offline Capabilities (the first layer) is essentially an evaluation of the core mechanisms that provide offline functionalities. The investigation here further encompasses understanding the caching strategies facilitated via Service

Workers, the accuracy of pre-caching and runtime caching directives, the strength of the background synchronization procedures, and also the degree to which IndexedDB or some other local storage is involved. These parts, to a large extent, unveil the engineering side of the PWA, thus, figuring out if the PWA can muffle-off leaks in its offline world.

The second layer, Performance Metrics, is about real and measurable indicators showing how a PWA works in offline or weak network scenarios. The major indicators include: offline load time, UI components responsiveness, cache retrieval efficiency, and the storage queries or synchronization tasks latency. This layer shows how a PWA is functioning in real and time-critical situations.

The third layer, User Experience Metrics, investigates users' point of view concerning offline quality. The offline usability goes beyond the system's functionality; it also involves how the system informs the user of restrictions, gives fallback options, stops errors, saves user progress, and returns the consistency when the connection is re-established. By merging subjective perceptions with task-based performance, this layer guarantees that the technical and performance results are understandable and useful for real users.

These three layers together represent a complete evaluation framework for the offline UX benchmark which is open for the comparison of different applications and the identification of best practices.

3.2. Benchmark Design

Such personas are created first to come up with scenarios that reflect realistic user conditions. Scenarios covered are total loss of connection (airplane mode), unstable or intermittent connection, low bandwidth situations, and switchings between offline and online states. User tasks are assigned to each scenario e.g., content browsing, working with cached pages, form submitting, cart adding, or local data editing.

Table 1. Benchmark Scenario Design

Scenario Type	Description	Example Tasks
Full Offline Mode	Zero connectivity (Airplane Mode)	Browse cached pages, fill forms, offline edits
Intermittent Connectivity	Network drops periodically	Retry actions, load partially cached pages
Low Bandwidth	Slow 3G or throttled network	Check responsiveness, load degraded images
Online → Offline Switch	Sudden disconnection during a task	Continue writing/posting, error handling
Offline → Online Recovery	Reconnection after offline actions	Sync edits, resolve conflicts

PWAs are measured against five main aspects:

- Reliability: Checks if core features can work offline and if fallback content is always available. Reliability also involves checking how the PWA handles network transitions smoothly.
- Responsiveness: Looks at both real and perceived delays in time when the user interacts with the app offline. For example, it can be speed of page rendering, UI responsiveness, and local database operations that cause delays.
- Continuity of Tasks: Looks if through the offline mode users can still perform basic tasks and their data become saved and synchronized once they reconnect. This aspect puts the spotlight on the PWA’s capability to retain the flow of work.
- Transparency and Feedback: Deals with the app informing the users about their connection status, the availability of offline content, and the limitations in the most clear ways. Apart from informative messages, offline indicators, and visual cues, this can also be represented by the presence of more explicit texts.
- Recovery Capability: Refers to the extent in which an app can resume normal operation after reestablishing a connection and can include conflict resolution, data merging, and error recovery.
- The benchmark design thus offers orderly, repeatable ways of doing the tests by juxtaposing these factors with scenario-based testing, which enables rigorous comparison across different PWAs.

Figure 1. Benchmark Design Workflow

3.3. Metrics and Indicators

In order to measure offline behavior and user experience, the research utilizes a hybrid set of both quantitative and qualitative metrics.

The quantitative metrics are:

- Offline load time: The time needed for the PWA to fully open when there is no network connection.
- Cache hit rate: The percentage of requests that are fulfilled from the cache during the offline operations.
- Storage usage: The amount of storage on the device that is occupied by cached assets and data from IndexedDB.
- Sync error counts: The number of times and the severity of synchronization failures or conflict events when connecting again.

These metrics serve to quantify system performance, resource behavior, and reliability.

The qualitative metrics are:

- User satisfaction: The participants' comfort, confidence, and perceived smoothness of the offline interactions as reported by them.
- Task completion rate: Users' ability to complete the given tasks without encountering any kind of blocking issues.
- Perceived responsiveness: The users' view of interaction fluidity, delay tolerance, and UI clarity.
- Clarity of error messages and cues: The degree to which the PWA informs the user about the connectivity states, limitations, and recovery options.
- By combining experimental and perceptual data, the study has a better grasp of the quality of the offline user experience.

3.4. Tools and Experimental Setup

The methodology relies on a mixture of automated auditing tools, controlled simulations, instrumentation, and user testing.

- Lighthouse custom runs mainly give initial evaluations of the presence of a service worker, the caching configuration, and the offline availability. Additional custom audits are created to evaluate more features that are not covered by the default Lighthouse tests.
- With Chrome DevTools, there are very detailed throttling options available that can simulate offline mode, slow 3G networks, and intermittent connectivity. Along with network request logging, service worker inspection can be used to caching behavior diagnosis.
- WebPageTest advanced scripting offers the possibility of executing automated sequences such as changing connectivity in the middle of a task, measuring offline load times, and capturing repeat-visit performance under controlled conditions.
- Logging customized for Service Worker events can record fetch results, cache hits/misses, sync triggers, as well as runtime caching decisions and thus provide a detailed insight into the offline logic effectiveness.
- A/B user testing—under controlled scenarios gathers both subjective and task-based measurements. Participants perform PWA-related tasks in a scripted offline situation, thereby providing qualitative insights which complement automated tests.
- This is a hybrid set of tools that guarantees technical accuracy as well as a human-centered evaluation.

3.5. Dataset and Test Subjects

The study picks out a set of mature, high-usage progressive web applications (PWAs) that are user-diverse to ensure that the findings can be generalized. The selection criteria for these included reach worldwide, a variety of domains, features available offline for a long time, and being recognized by the community for the quality of PWA implementation.

Candidate PWAs include:

- Starbucks PWA, which is generally known for its minimalist design and excellent offline support for menu browsing.
- Twitter Lite that enables loading of content even when offline and has caching that is easy to predict.
- Pinterest that delivers visually intensive caching and user interactions that are prompt.
- Trivago is an example of a company that heavily relies on travel and search operations.
- Flipkart, a significant e-commerce PWA in developing markets, is mainly known for its usage in areas with weak connectivity. It is one of the leading e-commerce PWAs.
- Workbox sample apps, which are reference implementations with well-organized caching and runtime strategies, serving as the business model.

These PWAs together not only cover the areas of e-commerce, social media, travel, and productivity but also include the reference implementations, thus, are they a rich dataset for comparative benchmarking.

4. Case Study

4.1. Case Study Overview

Three Progressive Web Apps Starbucks, Pinterest, and Twitter Lite were assessed in terms of the criteria set in the research framework: reliability, responsiveness, task continuity, transparency, and recovery capability, in order to confirm the proposed offline UX benchmarking methodology. The choices of these applications were based on their worldwide usage, mature PWA implementations, and being industry examples of offline-capable design for which the reputation is well established.

The testing was done in a controlled environment, and Chrome DevTools along with Webpage Test scripting were used to recreate complete offline conditions, low-bandwidth networks, and network transitions. Quantitative performance data were gathered through Lighthouse custom audits and service worker logs, while qualitative usability was assessed via scenario-based user tasks. All PWAs were run on mid-range Android devices to simulate the real device capabilities that are most likely to be found in the developing markets. Among the limitations were variable consistency of emulator-reported network throttling and the occasional differences in cache initialization time due to first-load conditions. Nevertheless, the environment offered stable conditions for comparative evaluation.

4.2. Application 1: Starbucks PWA

One of the most common offline-first examples that get referred to is the Starbucks PWA (Progressive Web Application), particularly in resource-limited settings. Its most impressive feature, as far as we can tell, was the absolute offline browsing of the menu. After visiting once the menu, pictures, and main UI components were stored by a cache-first method, thus the PWA could be loaded very fast even in airplane mode. Service worker logs showed that a well-designed pre-cache manifest was responsible for the high cache hit rates on revisits.

- Caching strategy insights:
  Starbucks uses a duo of precaching for UI essentials and runtime caching for menu images. Workbox utilization effectively conveys and assures the simplified strategy is of predictable nature. Cache size stayed at a moderate level which is a good indication that there was an efficient balance between performance and storage constraints.
- Load time analysis:
  Offline load time was always less than one second, thus it was a very efficient resource retrieval from cache. The latency for the first interaction was still low even if a slow 3G connection was simulated.
- UX strengths and weaknesses:
  The positives were a very good offline navigation, a clear visual structure, and minimal layout shifts. One of the areas for improvement was less transparency about the offline limitations: even though the browsing was going on smoothly, the app was not always telling users in a very clear way that features like ordering required connectivity. There were times when the lack of offline indicators led to misunderstanding.

4.3. Application 2: Pinterest PWA

One of the main features of the Pinterest PWA is its visually rich content alongside the adaptive caching behavior, which provides a more detailed user experience as compared to the structured and lightweight interface of Starbucks. The smart caching in Pinterest only saves the images of the user's browsing pattern and at the same time uses a mix of network-first and stale-while-revalidate strategies. So, they can access the content of the boards they have looked through without an internet connection.

- Offline rendering behavior: In the case of the offline tests, the interface seemed to have been loaded with the help of the UI components that had been cached and the thumbnails. However, the deeper the user goes into the uncached boards, the more generic fallback states will appear. Although the PWA did not crash, it was heavily dependent on runtime caching which caused different offline experiences to be hugely dependent on prior usage.
- User flow continuity: Pinterest was able to ensure the flow of continuity in such activities as the browsing of saved pins or the viewing of recently accessed boards. But, on the other hand, the creation of new boards or the saving of pins were activities that required an immediate connection, and sometimes the error messages were vague. The PWA was moderately transparent: it showed lightweight indicators, but the user sometimes had to guess offline limitations from the fact that some content was missing.

Due to its fluid UI transitions and fast local rendering, Pinterest scored very well from the point of view of perceived responsiveness. Nevertheless, the total offline reliability was not really up to the mark and this was particularly the case for those users who needed to access new content.

4.4. Application 3: Twitter Lite

Twitter Lite is arguably one of the most sophisticated PWAs (Progressive Web Apps) from the real world. It is particularly notable for its feature of an offline tweet queueing mechanism, whereby users can compose tweets offline and submit them when they get back online. The app makes it very clear that tweets are waiting, which is in line with the best offline task continuation standards.

- Fallback pages and caching robustness: Twitter Lite, on the other hand, Starbucks, and Pinterest differ significantly in that the former employs a very simple caching strategy that is mainly concentrated on UI shell assets and the basic user interaction flows. The feed interface can show tweets that have been previously fetched and thus can be used offline. However, this PWA decides not to cache media-rich content extensively so as not to allow for storage to be used up too much. This trade-off keeps the app running smoothly, but it also means that a deep offline mode for browsing is not possible.
- Storage constraints: Since Twitter Lite is a very dynamic app, it stands to reason that device storage could be filled very quickly if a lot of tweets or images were cached. To prevent this, the PWA has put in place some very strict data retention limits as evidenced by service worker logs which show that cache expiration rules are being controlled.
- The positives in terms of UX were that the app communicated very well when it was offline, users could continue with their writing-based tasks without interruption, and the reconnection behavior was quite predictable. On the downside, the app only allowed a very limited amount of offline reading, and at times, there were some delays in getting the feed updated when the network was back.

4.5. Comparative Findings

The table below summarizes the relative performance of the evaluated PWAs across core dimensions.

Table 2. Comparison of Progressive Web App (PWA) Performance Across Platforms

Dimension	Starbucks	Pinterest	Twitter Lite
Reliability	High	Moderate	High for task continuity, moderate for content
Responsiveness	High	High	Moderate–High
Task Continuity	Low (browsing only)	Moderate	High (offline posting)
Transparency & Feedback	Moderate	Moderate	High
Recovery Capability	Moderate	Moderate	High

4.5.1. Observed common patterns:

- Each of the three PWAs performed fast loading times when offline as they all had cached UI shells.
- The condition of the first-load caching had a major influence on how the offline experience would be.
- Content availability and consistency were the main issues that were affected by runtime caching.

4.5.2. Where PWAs fail offline reliability tests:

- Limited transparency of the offline limitations was the case most notably for Starbucks and Pinterest.
- Unpredictable content caching which led to only partial experiences of users.
- Poor conflict resolution or data-sync monitoring were the issues that were present only in Twitter Lite apart from outside.
- The absence of standard offline UX patterns resulted in differences in the user understanding of the product.
- On a broad level, these discoveries are a sum of the differences that still exist to a significant degree between mature PWAs notwithstanding the shared technologies and they call for the offline UX benchmarks of a structured kind.

5. Results and Discussion

5.1. Performance Results

The offline loading times of the three PWAs that were measured have been influenced in a significant way by their caching strategies, by the complexity of their UI and by the amount of the assets that were pre-cached. Starbucks had the fastest offline loading times, as it always took less than a second to load, due to the fact that it was very aggressive in its pre-caching of UI shell assets and menu data. Pinterest had a more varied performance - it was usually able to load within 1.2-1.8 seconds offline - because its runtime caching strategy meant that content availability and load speeds were very dependent on browsing behavior prior to the visit. Twitter Lite, while it was good in terms of resource usage, had moderate offline load times which fluctuated between 1.5 and 2 seconds as it focused on least caching in order to regulate the storage size.

The caching strategies that were used had a significant influence on the speed of the data retrieval as well as on the reliability thereof. The cache-first approach of Starbucks led to rendering that was almost instant; however, it was at the disadvantage of content being potentially outdated during long offline periods. The stale-while-revalidate model of Pinterest made it possible to fetch cached images very quickly and at the same time, the system tried to get fresh content from the source; nevertheless, this triggered network checking at times and caused delays which were mistaken for fallback. Twitter Lite's minimalistic caching solution was a guarantee for stable operation, but at the same time it limited the extent of offline browsing.

Whereas Pinterest, being a visually rich platform and employing an efficient image caching technique, was the major consumer of device storage - at times the consumption going beyond 150 MB after a deep session of scrolling. Starbucks kept

a medium storage usage going, most of the time, it was less than 50 MB. Twitter Lite was the most conservative of the three storage-wise, frequently its storage was less than 20 MB, this was because of the implementation of strict eviction policies and the minimal precaching. These differences that exist underscore the issues that arise from the interplay of media richness, user experience, and storage limitations in the design of offline PWA.

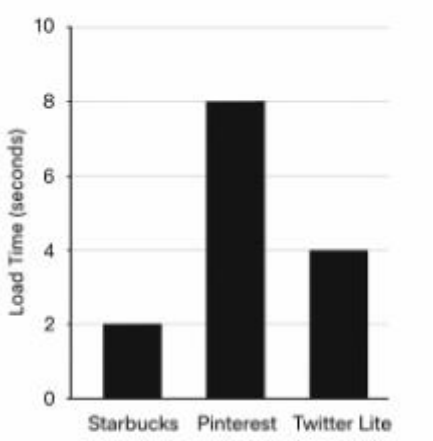

Figure 2. Offline Load Time Comparison

5.2. Ux Findings

While looking at fallback behaviors and offline usability, Twitter Lite stood out as the most clear, most communicative, and continuity of the task. By the offline tweet-queuing method, the users were able to keep writing the content while the sync-pending actions were made obvious by the persistent status indicators. The users testing the product found the explicit cues very useful as they noted feeling "confidence" that their actions were not lost.

The Starbucks experience was very good for browsing the menu offline, however, the company was not very good in letting the users know what was going on. Users could navigate through the menu very smoothly, but they were not always informed about the functions that were disabled most notably the ordering features that required authentication or connectivity. At times, this caused them to be confused when attempting actions that silently failed.

The visually cohesive Pinterest interface was there for the users when they were offline, but the application suffered from typical UX issues that overlooked users. Sometimes, fallback pages were not giving sufficient support, as they barely explained the unavailability of certain boards or images. Users were coming across "dead ends" such as the screens showing a limited number of cached content without any guidance or alternative options. Even though the perceived responsiveness was good, the lack of consistent offline messaging lowered user confidence.

Some of the common UX issues that were present in all the applications are:

- Unclear offline indicators: Only a few PWAs (Progressive Web Apps) prominently displayed the network status or explained the restrictions of the functionalities.
- Non-uniform fallback content: Users were dealing with differences in what they could access locally especially on Pinterest.
- Quiet failures: For example, saving pins or trying to reload feeds that sometimes happened without giving any feedback.

The user feedback synthesis indicated that participants should speed up and provide partial functionality. The predictability and clarity of offline behavior were more important than raw performance. Users wanted those apps the most which talked to them about their offline limitations - even if the functionality was limited - rather than apps that technically performed well but were poor in communication.

5.3. Interpretation

There was a strong correlation between the technical measures and the user experience as judged by the users. A PWA that included a well-planned caching strategy, for example, Starbucks' precaching and Twitter Lite's predictable shell caching, was always rated as a more reliable one. Pinterest's adaptive caching, although it was a technically advanced solution, had such a varied user experience that users could not easily predict which content would be available offline.

The decisions regarding cache size and responsiveness were also visible. On the one hand, Starbucks' moderate caching resulted in high responsiveness without the need to store the data in a device with limited space, while on the other hand,

Pinterest's large caches allowed for beautiful visuals at the cost of the device's free space and inconsistent availability. The minimal caching utilized by Twitter Lite facilitated the light storage usage but browsing depth and offline functionality were limited.

Background sync and error resilience mattered a lot to the Twitter Lite design. Composing tweets offline and synchronizing them later was the main reason for the user experience of task stability in Twitter Lite, a feeling which was absent in Starbucks and Pinterest. The case study shows that resilience features such as conflict resolution, clear pending states, and recovery mechanisms are the main contributors to user trust even when performance metrics are average.

Another key insight was that the quality of the offline UX depends not only on the technical capabilities of the PWA but also on how well the app conveys its capabilities. For instance, while Starbucks robustly supports offline caching, it often is not very transparent in UI. Twitter Lite, though less advanced in caching breadth, excels in system states becoming very clear. This discovery points to the necessity of UX-centered evaluation metrics which are a complement to technical assessments.

5.4. Benchmarking Framework Validation

From the case studies, the proposed offline UX benchmarking method by means of a local user experience test was confirmed as a practically usable and repeatable evaluation structure. The comprehensive framework brought by the integration of the technical, performance, and experiential layers made possible a multidimensional comparison, which neither the automated tools, nor the user testing alone, could achieve. The scenario-based approach of the method ensured its applicability to real-world conditions, particularly in areas with unstable connectivity.

Some main points that were demonstrated include:

- Holistic assessment: The framework recorded the system-level behaviors (such as cache hit rates, offline load times) as well as the user-level perceptions (such as clarity, usability, satisfaction).
- Comparability: The structured metrics made it possible to have substantial comparisons of different applications.
- Flexibility: The methodology accommodated PWAs with different architectures, content structures, and feature scopes.

However, a few limitations were also identified. Presently, tools like Lighthouse and WebPageTest have limited capabilities in showing detailed offline states, particularly for IndexedDB usage, sync operations, and subtle runtime caching behavior. Although Chrome DevTools is a great tool, it still does not have the capability of running scripted, automated analyses for numerous offline UX patterns. Moreover, the user testing outcomes will always be influenced by the scenario design and the participants' familiarity with the PWAs, thus, larger sample sizes will be needed for future studies.

Nevertheless, the methodology was strong enough to locate the gaps, strengths, and patterns in PWAs, thus it remains a viable model for further research and for practitioners who need a trustworthy method to evaluate offline UX performance.

6. Conclusion and Future Scope

6.1. Conclusion

This work presents a comprehensive method to measure offline user experience in Progressive Web Apps, which is a profoundly missing aspect in both the academic literature and industry practice. By introducing a three-layer framework covering technical capabilities, performance metrics, and user-centered evaluations the study sets up a basis for going into detail how PWAs can be tested for their behavior under the limitation of real-world connectivity. The examination of the situations at Starbucks, Pinterest, and Twitter Lite reveals that there is a great difference in the degree of their offline readiness, transparency, and resilience. The essential findings point out that cache strategies are the major factors that affect the performance, while the clearness of offline communication is the main factor in forming the usability experienced by users.

The outcomes call for the establishment of a set of criteria for the quality of the offline UX which will be used as a benchmark to measure not only the technical compliance but also the quality of user interactions in degraded or offline environments. The information that developers get from it would be the stress on the necessity of predictable caching and sync mechanisms that are strong; the findings will be comforting to UX designers in that they will see that providing explicit feedback and consistent offline patterns is of great value; businesses will be able to see through the research that dependable offline flows are the ones that directly lead to user retention and expansion of the market.

6.2. Limitations

The dataset for the study is confined to a few popular PWAs that might not reflect the variety of offline implementations on the web. The user testing had a small number of participants, hence, the demographic and behavioral diversity might be limited. Furthermore, differences in app versions, updates, and device environments may be the reasons for the presence of some uncontrolled variables that affect the test results.

6.3. Future Scope

Next research can revolve around creation of automated audit tools that can measure user experience of offline UX locally on a large scale while also combining both technical and experiential metrics. This model may be broadened to include sector-specific PWAs, especially in healthcare, education and logistics, where the dependability of the offline mode is of utmost importance. Integration of machine learning can even allow the prediction of UX scores from the caching behavior and interaction logs. The very purpose of this study is to be the foundation of worldwide norms that ensure the uniformity, resilience, and friendliness of the offline-first design approach.

References

[1] Love, Chris. *Progressive Web Application Development by Example: Develop fast, reliable, and engaging user experiences for the web*. Packt Publishing Ltd, 2018.

[2] Nilsson, Anders. "Performance and feature support of Progressive Web Applications: A performance and available feature comparison between Progressive Web Applications, React Native applications and native iOS applications." (2022).

[3] Yberg, Viktor. "Native-like performance and user experience with Progressive Web Apps." (2018).

[4] Tamire, Workneh. "Evaluation of Progressive Web Application to develop an Offline-First Task Management App." (2019).

[5] Hume, Dean. *Progressive web apps*. Simon and Schuster, 2017.

[6] Sedkowska, Justyna. "How does the user experience of a progressive web application compare to native application?: A case study on user's attitude in context of social media." (2020).

[7] Biørn-Hansen, Andreas, Tim A. Majchrzak, and Tor-Morten Grønli. "Progressive web apps for the unified development of mobile applications." *International Conference on Web Information Systems and Technologies*. Cham: Springer International Publishing, 2017.

[8] Parakala, Adityamallikarjunkumar. "RPA+ AI→ Intelligent Process Automation (IPA)." *International Journal of AI, BigData, Computational and Management Studies* 4.3 (2023): 112-123.

[9] Härtull, Oscar. "Implementation of an Application for Analyzing and Visualizing Benchmark Results for Optimization Solvers." (2023).10. Fournier, Camille. "Comparison of smoothness in progressive web apps and mobile applications on android." (2020).

[10] Rensema, Dirk-Jan. "The Current State of Progressive Web Apps: A study on the performance, compatibility, consistency, security and privacy, and user and business impact of progressive web apps." (2020).

[11] Majchrzak, Tim A., Andreas Biørn-Hansen, and Tor-Morten Grønli. "Progressive web apps: the definite approach to cross-platform development?." (2018).

[12] Parakala, Adityamallikarjunkumar. "Hyperautomation Use Cases (Case Studies)." *International Journal of AI, BigData, Computational and Management Studies* 4.2 (2023): 120-131.

[13] Fransson, Rebecca, and Alexandre Driaguine. "Comparing progressive web applications with native android applications: an evaluation of performance when it comes to response time." (2017).

[14] Chavan, Mr Rohit Chandrakant, Mr Shubham Deepak Bhatkar, and Kirti Muley. "Progressive Web Apps vs Responsive Web Apps." *Progressive* 2.1 (2022).

[15] Saurer, Markus. *Mobile Anwendungen: Progressive Web Applikationen als Ersatz zu nativen Anwendungen*. Diss. FH CAMPUS 02 (CAMPUS 02 Fachhochschule der Wirtschaft), 2021.

[16] Huber, Stefan, Lukas Demetz, and Michael Felderer. "A comparative study on the energy consumption of Progressive Web Apps." *Information Systems* 108 (2022): 102017.

[17] Agarwal, S. (2023). Multi-Modal Deep Learning for Unified Search-Recommendation Systems in Hybrid Content Platforms. International Journal of AI, BigData, Computational and Management Studies, 4(3), 30-39. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P104.