Progressive Web Apps (PWAs)

Aaron Ismael Sales Flores

I. Introduction

Progressive web apps (PWA) are web apps that combine the best features of web apps and native apps. These apps are designed to offer an enhanced user experience, similar to that of a native app, but can be accessed directly through a web browser. They represent a significant evolution in the web and mobile development landscape, offering a hybrid solution that combines the best of web and native applications. These applications have gained popularity due to their ability to provide users with fast, reliable interactive experiences while removing some of the traditional barriers associated with mobile applications.

II. What is a PWA?

Progressive Web Application (PWA) is a type of web app that can operate both as a web page and mobile app on any device. Using standard technologies, PWA is aimed at delivering native-like user experience, with speedier conversion and cleaner browsing even with a poor Internet connection. PWAs are written in JavaScript, CSS, and HTML. They look and behave just like regular web pages. However, they also deliver functionalities identical to those provided by mobile apps: they are fast, can work offline, send push notifications, and use the features of user devices.

III. PWA features

- They work offline:Unlike standard web applications, PWAs are network independent, which allows them to work even when users are offline or have an unreliable network connection. On a very simplistic level, this is accomplished by using Service Workers and APIs to revisit and cache page requests and responses, thereby allowing users to browse content they previously viewed.
- They're Search Engine Optimized: PWAs are intended to be more discoverable and compatible with search engines. To support that goal, PWAs adhere to certain global standards and formats that make it easier to catalog, rank, and surface content in search engines. In addition to providing SEO advantages, these modern technical requirements make the application more secure.
- They're installable: Unlike traditional web apps, PWAs have the ability to be installed on a device. This empowers users to access that app via an app icon and ultimately creates a more seamless and integrated user experience.
- They're linkable: Unlike native mobile applications, traditional web applications are accessible via a direct URL without requiring setup or installation. That URL structure makes it easy to encourage users to engage with specific content by linking directly to a page you'd like them to view and even anchoring to specific text. Responsive design is table stakes for delivering on these heightened expectations. PWAs leverage modern technologies to ensure a web application UI is flexible and responsive.
- They provide built-in security benefits: PWAs are built using HTTPS, which encrypts data shared between the app and the server. This protocol makes it inherently more challenging for hackers to access sensitive data. Compared with native applications, PWAs have more limited permissions, which typically reduces exposure to security threats.
- They provide built-in security benefits: PWAs are built using HTTPS, which encrypts data shared between the app and the server. This protocol makes it inherently more challenging for hackers to access sensitive data. Compared with native applications, PWAs have more limited permissions, which typically reduces exposure to security threats.
- They're more cost-effective to develop than native mobile apps: PWAs can be built using web technologies such as HTML, CSS, and JavaScript, which makes them a more cost-effective alternative to developing a native mobile application for each operating system (iOS, Android, etc.).
- They outperform standard web apps: Because PWAs are designed to be lightweight from a data consumption standpoint, they have better load times, impeccable responsiveness, and more seamless animations than traditional web apps. This all equates to a more delightful, scaleable, and flexible user experience across various devices.

IV. Differences between web application, PWA and SPA

- Security: SPAs are less secure than PWAs as they are vulnerable to cross-site scripting (XSS). PWAs on the other hand have fewer security risks since all requests go through HTTPS. This help protects the privacy and integrity of the data being transmitted.
- SEO: PWAs are designed are websites and as such, they are discoverable and crawlable by search engines. SPAs, on the other hand, are web applications that are designed to be fast and responsive. However, since SPAs only load a single HTML page and dynamically update the page with new content as the user navigates, they are more difficult for search engines to index and rank.
- Development cost: Developing a PWA tends to be more expensive than developing a SPA because it involves building both a server-side application and a client-side application. On the other hand, an SPA is a client-side

application that is loaded entirely from the client and does not require a back-end server. As a result, the development cost for a SPA may be lower than that of a PWA.
- Performance: PWAs are generally faster and more responsive than traditional web apps because they are designed to be loaded from the server and then cache content for offline use. SPAs are loaded entirely from the client and do not require a full-page refresh to update content. This can make the app feel more interactive and responsive. However, SPAs may be slower than PWAs when it comes to initial load time, as they must download all of the necessary resources before the app can be rendered.
- Maintenance and ease of update: PWAs are often easier to maintain and update since they are hosted on a server and updates can be rolled out centrally. This means that any updates or bug fixes can be made on the server and will be automatically reflected in the app for all users. SPAs, on the other hand, require updates to be deployed to each user's device individually. This can make maintenance and updates more time-consuming and costly, especially for apps with a large user base.

V. Advantages

- Offline mode: PWAs can be cached by the web browser and used even when offline.
- Improved performance: Because PWAs utilize the so-called service workers, which are JavaScript files that run separately from the main browser thread and proactively control the caching of assets, they can deliver much better performance than traditional web apps.
- No installation or manual updates required: All new features and bug fixes are available without any manual action required.
- Platform-specific features: PWAs can live on the user's home screen and deliver web push notifications that appear just like regular push notifications. They can run in full screen, change display orientation, start with a custom splash screen, access locational data, and much more.
- Low on data: PWAs are much smaller than mobile apps, and they require a lot less bandwidth than traditional web apps because they can take much better advantage of caching.
- App store independent: PWAs are app store-independent, which is great news for smaller businesses and independent app developers that don't want to pay Apple's annual fee or Google's lifetime fee. Of course, not depending on an app store also frees app developers to create any app they want without being shackled by Google's and Apple's app store policies and restriction

VI. Disadvantages

- Compatibility with iOS: Since iOS 11.3, it's been possible to run PWAs on Apple devices, but you can forget about compatibility with older devices. What's more, Apple doesn't allow PWAs to access many important features, including Touch ID, Face ID, ARKit, Bluetooth, serial, Beacons, altimeter sensor, and even battery information.
- Issues with legacy devices: PWAs have been around for just a few years, so it shouldn't come as a surprise that older mobile devices with outdated web browsers don't support them too well.
- PWAs can't do everything: As capable as PWAs are compared to traditional web apps, they can't do everything mobile apps can do. Their performance is also not as good as the performance of native apps, which has a lot to do with the fact that JavaScript is a single-threaded programming language.