PROGRESSIVE WEB APPS (PWA)

Saksham Sharma[1] Avni Bhardwaj [2]

1.Amity School of Engineering and Technology, Amity University Noida, Sector-125, Uttar Pradesh, Noida, India.

2.Amity School of Engineering and Technology, Amity University Noida, Sector-125, Uttar Pradesh, Noida, India.

Abstract : Progressive Web App is a term summarizing all the features allowing you to deliver mobile-like experiences. Not only is it strongly promoted by Google. It is also quickly being adopted by big companies like Twitter or the Washington Post. A progressive web application takes advantage of the latest technologies to combine the best of web and mobile apps. Think of it as a website built using web technologies but that acts and feels like an app. Recent advancements in the browser and in the availability of service workers and in the Cache and Push APIs have enabled web developers to allow users to install web apps to their home screen, receive push notifications and even work offline. Progressive web apps take advantage of the much larger web ecosystem, plug ins and community and the relative ease of deploying and maintaining a website when compared to a native application in the respective app stores.

For those of you who develop on both mobile and web, you'll appreciate that a website can be built in less time, that an API does not need to be maintained with backwards-compatibility (all users will run the same version of your website's code, unlike the version fragmentation of native apps) and that the app will generally be easier to deploy and maintain. According to a study, an average user spends 80% of his total time on apps on only three of his apps. The other apps just sit idle for most of this time consuming a precious portion of the memory. Moreover, it costs around ten times to develop an app rather than creating a website for the same. The cost can get much higher if you plan to develop and maintain separate code bases for different platforms like Android, iOS and the web.

Native App features that PWAs can use

- Push notifications
- Full Screen
- Offline working
- Splash screen is supported giving it a more app like feel

PWAs can make use of many more such features. The above points are only to give you a hint of what PWAs are capable of. However, there are some traditional features that only native apps enjoy as of now.

I. INTRODUCTION

1. PROBLEM STATEMENT:

To develop a Progressive Web Application to ease access of website into an application which provides features similar to native apps.

2. OBJECTIVES:

- To adept at back end involved for the project and progressing towards building successful PWA enabled website.
- Being able to develop a progressive web app having features and experience at par with native web apps.

3. METHODOLOGY TO BE ADOPTED:

- Getting acquainted with web technologies and application functionality.
- Deploying knowledge to create different modules, testing for errors and successfully implementing packages involved.

The term Progressive Web App as coined by Alex Russell and Frances Berriman. In Alex's words: Progressive Web Apps are just websites that took all the right vitamins.

It isn't a new framework or technology. It is a set of best practices to make a web application function similar to a desktop or mobile application. The dream is to have an experience so uniform and seamless that the user is unable to tell the difference between a Progressive Web App and a native mobile app.

Progressive web applications deliver user experiences through progressive enhancement. It essentially means that a PWA will perform the same functions on a new iPhone 8 as it would on an older generation iPhone. Sure, some features may not be available, but the app continues to work and perform like it should.

➢ Need for Progressive Web App?

Before we understand why we need a progressive web app, let's talk about some of the challenges we are facing today with native and web apps.

1. **Internet speed**: you may not realize this depending on where you live, but 60% of the world's population is still using 2G internet. Even in the US, some people have to use dialup to access internet.

2. **Slow website load:** Do you know how long a user waits to click the Close X button if a website is too slow? Three seconds! 53% of users abandon a website if it is too slow.

3. **High friction:** People don't want to install native apps. An average user installs 0 applications in a month.

4. **User engagement:** Users spend most of their time in native apps, but mobile web reach is almost three times that of native apps. Hence, most of the users are not actively engaged. However, users are spending 80% of their time on only their top three native apps.

➢ Native App features that PWAs can't use as of now:

1. No or highly restrictive access to different hardware sensors
2. Alarms
3. Phonebook Access
4. Modifying System Settings
5. PWAs are evolving quite fast and we can hope to see these features come to action pretty soon.

PWAs help solve these problems. There are multiple reasons for using a progressive web app, but here are some of the top capabilities it provides:

1. **F**ast: PWAs provide experiences that are consistently fast. From the moment a user downloads an app to the moment they start interacting with it, everything happens really fast. Because you can cache the data, it is extremely fast to start the app again even without hitting the network.

2. **I**ntegrated user experience: PWAs feel and behave like native apps. They sit in a user's home screen, send push notifications like native apps, and have access to a device's functionalities like native apps. The experience feels seamless and integrated.

3. **R**eliable experience: With the help of service workers, we can reliably paint a picture on a user's screen even when network has failed.

4. **E**ngaging: Because we can send notifications to a user, we can really drive the engagement up by keeping the user notified and engaged with the app.

➢ BUILDING PWA

Google has published a checklist of items for Progressive Web apps.

1. Web App Manifest

A sample manifest.json file

This is just a json file that gives meta information about the web app. It has information like the icon of the app (which a user sees after installing it in their app drawer), background color of the app, name of the app, short name, and so on. We can write this manifest file ourselves or we can use tools to generate one for us.

2. Service Workers:

Service Workers are event-driven workers that run in the background of an application and act as a proxy between the network and application. They are able to intercept network requests and cache information for us in the background. This can be used to load data for offline use. They are a javascript script that listens to events like fetch and install, and they perform tasks.

Service worker is where most of the magic of happens. It's nothing but JavaScript code that acts as programmable proxies solely responsible for intercepting and responding to network requests. Since it acts as a proxy and can be easily programmable, the application must be served over HTTPS to keep the data secure.

It's worth noting that the service worker caches the actual response, including all HTTP headers, rather than just the response data.

3. Icon:

This is used to provide an app icon when a user installs the PWA in their application drawer. A jpeg image will just be fine. The manifest tool I highlighted above helps in generating icons for multiple formats, and I found it very useful.

4. Served over HTTPS:

In order to be a PWA, the web application must be served over a secure network. With services like Cloudfare and LetsEncrypt, it is really easy to get an SSL certificate. Being a secure site is not only a best practice, it also establishes web application as a trusted site for users demonstrating trust and reliability, and avoiding middle man attacks.

5. App Manifest:

It's a JSON file that defines an app icon, how to launch the app (standalone, full-screen, in the browser etc), and any such related information. It's located in the root of your app. A link to this file is required on each page that has to be rendered.

It is added in the head section of the HTML page:

<link rel= manifest href= /manifest.json >

➢ HOW PWA IMPROVES BROWSING EXPERIENCE?

1. Fast and streamlined site:

Progressive Web Apps are based on new technology called service workers. Service workers are event driven scripts that have access to domain wide events. These are programmable proxies that sit between the user's tab and the wider Internet. They intercept and rewrite or fabricate network requests to allow very granular caching and offline support.

2. Better User experience:

Addition to building streamlined websites, service workers helps in improving user experience. Users visiting via browsers enjoy app-like user experience. When they come back, it loads nearly instantly, even on slow networks. Frequent users will be prompted to install the app and upgrade to a full version. AliExpress, a popular e-commerce website in America, Russia and Brazil built a cross-browser Progressive Web App to combine the best of their app with the broad reach of the web. AliExpress observes 104% increase in conversion rates and users now visit twice as many pages per session, and time spent per session increased an average of 74% across all browsers.

3. Increased customer engagement:

You must have observed while accessing websites with PWA features prompts the user to 'Allow 'sending of notifications. It's a subtle yet powerful message, particularly for social networking websites. As a user, tapping 'allow 'here will allow the website to notify you of messages or updates— even when you're not looking at the page.

4. Offline Access:

Thanks to service workers, offline access of previously visited website is possible. As described earlier service workers are the backbone of PWA as they empower push notification, content caching, background updates, offline functionality. Offline access is made possible by service workers because these are essentially.

➢ PROJECT DESCRIPTION:

- The Project focuses on the need to integrate web development and application development under one platform.
- The concept of single vendor e-commerce pwa has been initiated during the course of development of this project.

➢ SINGLE VENDOR E-COMMERCE

Single Vendor Marketplace can be used to represent a website where you have single vendor or seller selling their product to various customers. There are only two parties involved in this buying and selling process i.e. buyer and the seller. Single vendor marketplace websites are also known as Stand Alone Website.

A single vendor marketplace is a website where only one seller provides the goods or one product. Only the seller and the buyer are involved in such a platform's processes, so there are no third-party complications. Many people know this as Stand Alone Website. Because a single vendor marketplace cannot offer a wide range of goods and products, there is very limited traffic and the resource needs significant promotional efforts compared with a multi-vendor marketplace solution.

➢ Advantages of Single Vendor Marketplace

- Creating and maintaining a relationship with one supplier is easy when compared with two or more
- Administrative as well as other supplementary costs are decreased when you place an orders with just one supplier
- Maximize your volume leverage to achieve attractive pricing
- It's very easy to order and incorporate systems with a single supplier
- You may be capable of bargaining to receive small, regular deliveries and so develop an inventory control

➢ Factors to consider when deciding between single vendor vs. multi-vendor

1. Best-of-breed

Product is acknowledged (by the genuinely knowledgeable) as the best product of its type. Importantly, this more often than not refers to a specific module or product that focuses on one specific feature. The simple fact of the matter is, it's hard to be the best at everything. An integrated solution might be fantastic overall, but not every component module is going to be best-of-breed.

- Single vendor – it's very unlikely to get best-of-breed functionality for every feature with a single vendor. With careful selection, you should be able to find a single vendor will likely be the best in one-to-several areas. Make sure you do your homework if you take this option!
- Multi-vendor – this is basically the key advantage of choosing a multi-vendor solution.

2. Complexity

- Single vendor – should have the advantage – the single vendor only has to worry about integrating its own product with your pre-existing systems. That can be challenging, but it's at least a closed set of details to fuss over.
- Multi-vendor – the onus here is on you to integrate everything. With multiple systems and a wide tranche of details, very few outside vendors will be able to integrate everything into a single cohesive whole – at least, not without great.

3. Cost

There's no getting around it – pricing is going to be different across the two categories.

- Single vendor – typically, this will be lower in price, thanks to better volume discounts.
- Multi-vendor – price will be higher because you'll be forced to buy separate individual modules to constitute your solution.

4. Implementation difficulty

This is reference to the challenges that you'll likely encounter for installing and setting up whichever system you ultimately purchase.

- Single vendor – it will be generally be less challenging to install single-vendor solutions because it's only one set of programs/protocols to integrate into your existing systems.
- Multi-vendor – multi-vendor solutions typically will require more effort to put into place because you'll have to coordinate the integration of disparate (and potentially incompatible) systems.

5. Procurement effort

Systems don't just purchase themselves, right? The question is, how hard will it be to get them? The answers are pretty straightforward, however!

- Single vendor – (relatively) simple – you're only dealing with one vendor, after all!
- Multi-vendor – more effortful – you have a lot of different vendors with whom to negotiate contracts, and not every contract is going to run the same length and/or will be renewed at the same time, if at all!

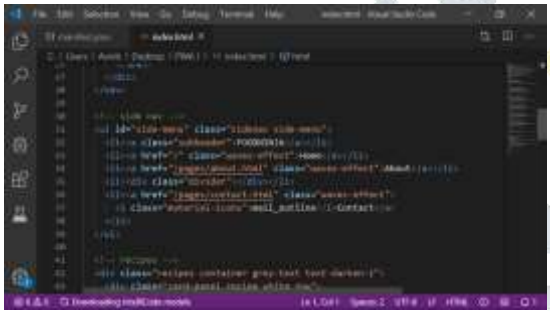
6. Product upgrades

We live in an age of ultra-rapid product updates and upgrades. What can we expect from our vendor packages?

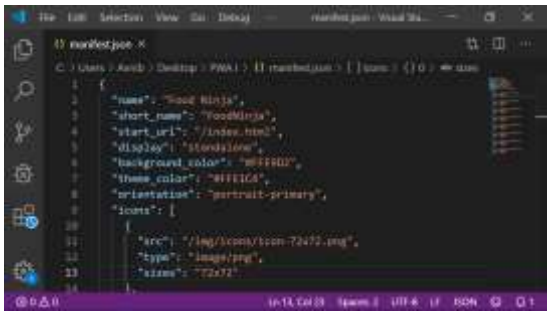
- Single vendor – you'll probably see major updates to single vendor solutions much more infrequently. The updates/upgrades will likely be much more comprehensive!
- Multi-vendor – because you're using and seeing smaller packages from each vendor, you should be getting upgrades more often. That could cause some problems with integration, however.

➢ SNAPSHOTS

1. HTML CODE:



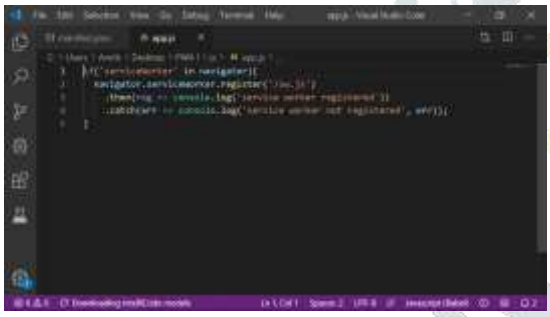
2. MANIFEST FILE:



3. SERVICE WORKER:

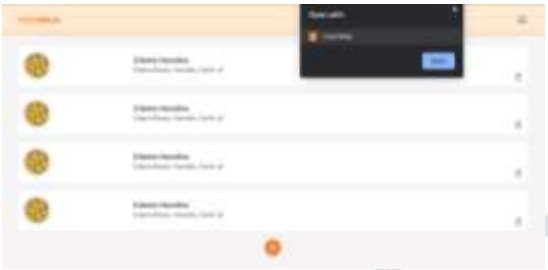
4. STYLESHEET:

5. JAVASCRIPT CODE:

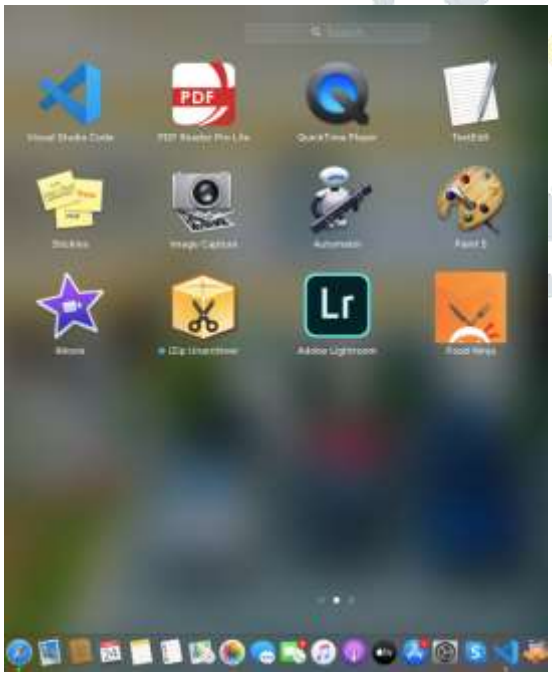


6. HOMEPAGE:

7. ADD TO HOMESCREEN



8. APPLICATION INSTALLATION



RESULTS AND DISCUSSION

For nearly two decades now (the first documented responsive design website was published by Audi around 2002), web developers have been adhering to the responsive web design approach to make websites display well on devices with large and small screens alike without the need to maintain a mobile-specific version of the same website.

Progressive web apps solve this problem through their use of service workers, creating a whole bag of fundamental design considerations, in addition to responsiveness, that all web developers and designers should keep in mind:

Dealing with change: The progressive aspect of PWAs means that more functionality can become available on subsequent visits (or even in real-time), and users should be properly informed of all changes using status notifications and other means.

Push notifications: One key feature of PWAs are push notifications. The ability of push notifications to increase user engagement is well documented at this point, but push notifications can also push users away if not handled correctly.

Different states: PWAs should be designed with different states in mind and work just as well offline as online. Other states include network problems, content still loading, or content loading only partially.

Perceived performance: Especially in developing markets such as India, where most people access the web from low-end devices, it's important to use loading skeleton screens and transition animations to let users know that they're making progress and thus improve perceived performance.

Accessibility: It goes without saying that PWAs should follow the Web Content Accessibility Guidelines (WCAG) 2.0, which define how to make web content more accessible to people with disabilities.

CONCLUSION AND RECOMMENDATIONS

PWA has a whole lot of Pros and cons to be considered. Progressive web apps features are interconnected and explain the reasons for their development.

Development savings

Specialists who follow the progressive web apps trend use a web stack for their development. This approach takes less effort and time, so it's more cost-efficient. The reason is that developers don't need to build the app for multiple platforms because a single progressive app can perform well on both Android and iOS and fit various devices.

Reduced installation friction

Discoverability, one of PWAs 'core features, increases their competitiveness over native apps. This advantage is especially meaningful considering that each additional step to download an app reduces the number of its potential users by 20 percent.

Easy updates

In addition to skipping the app store, surfing, and installation, PWA users free themselves from updating (or accepting the update offer) for the app each time a developer releases new versions. Users always have access to up-to-date solutions. This feature allows companies to avoid the problem referred to as software fragmentation when they have to maintain old versions of apps or risk the loss of users until they start the update.

Higher User Engagement

Researchers from comScore also found out that 80 percent of mobile users intentionally move apps to the home screen. So, the ability to be added to the home screen makes PWAs more competitive with web apps. There have to be other reasons people decide to try the app, of course. The frequency of use (61 percent), the simplicity of access (54 percent), and speed of access (49 percent) are the top factors influencing their decision. The chances for better distribution are, therefore, higher for PWAs. Push notifications also fuel user interest in the app.

Cons of PWAs

PWAs aren't only about pros: Their drawbacks are directly connected to the benefits.

Limited functionality and increased battery use compared to native apps

Despite their progressiveness, these are still web apps. Without access to device hardware, PWAs can't support such native-app typical features as fingerprint scanning, vicinity sensors, NFC, Bluetooth, geofencing, inter-app communications, and advanced camera controls. While app sharing via URL is convenient, it requires the connection, which drains a device battery faster than a native app.

Search traffic losses due to no presence on app stores

We mentioned that presence on app stores eliminates several steps users take before running an app, which reduces installation friction. Mobile web traffic can be redirected to an app store to showcase the app to users, while not being in an app store leads to potential traffic losses.

Examples of successfully developed PWAs

1.The Washington Post

The Washington Post was looking for efficient ways to deliver content to its audience, nearly 55 percent of which was accessing the content from mobile devices. Keeping in mind that users are likely to leave a website if the content loading takes more than three seconds, the publisher aimed to ensure that it appears instantly before the readers 'eyes. It also wanted to make articles available for offline reading.

At the Google I/O developer conference in 2016, the media introduced a PWA experience, made possible through the use of AMP, Accelerated Mobile Pages. AMP is a new standard for publishing content that provides immediate page load and content delivery for mobile users.

2.Shopify

Shopify, Canadian eCommerce platform, has launched the Litefy app to allow merchants to upgrade their online themes into progressive web apps.

3.Twitter

The popular social media platform with 328 million monthly users actively sharing, creating, and consuming information, developed the Twitter Lite PWA to increase engagement and reduce data consumption. The app has started working in May 2017. This step was particularly well-timed considering that more than 80 percent of users visit the platform from mobile devices.

Twitter Lite uses less than 3 percent of the device storage space, consumes up to 70 percent less data, and allows users to share tweets as quickly as possible.

IMPLICATIONS FOR FUTURE RESEARCH

The Project has been developed meeting all the requirements thoroughly.
It has been a substantial stride in the phase of digitalizing the traditional spreadsheet concept of maintaining company information. Yet there remain some aspects left unexplored with regard to taking the web development of the project to another level.
The following can be significant future implications:

a) Making User experience enriched with latest languages.
Example- Animating newly added content

b) Online deployment of the websites rather than just using it on intranet servers.

c) Tightening the security measures for any malicious attacks on the credential information on the portal.

In the future considering, the above-mentioned areas, the project can be carried further with its development and can contribute meaningfully to serve the organization with the ease of data access.

Development of progressive web apps can help you solve various challenges. Let's describe several situations when going progressive makes sense. First and foremost, building PWAs are about user engagement. Users who don't want to make extra clicks to download an app may be a significant part of your community. As people mostly rely on wireless network and mobile connection, they are likely to prefer websites and apps using less data and remaining at least partly functional offline. And PWAs have it all. Building a progressive web app is faster and cheaper, so you can use this app type to support a single or annual event like a music festival.

However, it's important to remember that functionality is PWAs 'weak spot. So, if native app-specific features (i.e. access to a camera or geolocation) aren't essential, then stick with a PWA. To summarize all of the above, things you should consider when making a choice are performance and functionality, development time frames and costs, as well as user experience.

REFERENCES

[1] https://developers.google.com/web/showcase/2017/twitter
[2] https://developers.google.com/web/showcase/2016/alibaba
[3] https://developers.google.com/web/showcase/2016/wapo
[4] https://brainhub.eu/blog/pwa-future/
[5] https://www.smashingmagazine.com/guide-pwa-progressive-web-applications/
[6] https://ionicframework.com/pwa
[7] https://www.appypie.com/faqs/pwa-minimum-requirements
[8] https://www.altexsoft.com/blog/engineering/progressive-web-apps/
[9] https://roobykon.com/blog/posts/122-multi-vendor-marketplace-vs-online-store
[10] https://www.webnexs.com/blog/difference-between-single-and-multivendor/