Chapter 1

INTRODUCTION

1.1 Progressive Web Apps

A Progressive Web App is an app that can provide additional features based on what the device supports, providing offline capability, push notifications, an almost native app look and speed, and local caching of resources. This technique was originally introduced by Google in 2015 and proves to bring many advantages to both the developer and the users.[2]

Developers have access to building almost-first-class applications using a web stack. This is always considerably easier and cheaper than building native applications, especially when considering the implications of building and maintaining cross-platform apps. [2]

Developers can benefit from a reduced installation friction, and at a time when having an app in the store does not actually bring anything in terms of discoverability for 99,99% of the apps, Google search can provide the same benefits if not more. [2]

Web applications are often described as being cross-platform. They are accessible from a multitude of different browsers, running on different operating systems. In 2014, the number of global users accessing the we on mobile devices surpassed those accessing it on a desktop. This shows that making your web applications mobile-friendly is more important now than ever. Companies often see the need to develop native mobile applications to overcome the limitations that the web as a platform imposes on mobile devices. In many cases, they have to develop their application for both the web, iOS, and Android. [1]

Developers have used web technologies to develop cross-platform mobile applications with tools like Cordova and PhoneGap for some time. These applications are installable from the respective operating systems application store, and run inside a native environment, with all features available to a native application. Taken out of this native environment, these applications fail to deliver this experience due to browser constraints. Progressive Web Applications (PWA) could solve this problem. [1]

1.2 Characteristics of Progressive Web Apps

It almost feels like working on a native app, as it offers the following characteristics [3]:

- **Progressive**

  Work for every user, regardless of browser choice, and so is compatible with devices of all specifications

- **Responsive**

  Fit any form factor: desktop, mobile, tablet, or forms yet to emerge so that the app can adapt to many screen sizes and aspect ratios.

- **Connectivity independent**

  Service workers allow work offline, or on low quality networks, and to make sure that it is responsive and seamless on low bandwidth.

- **App-like**

  Feel like an app to the user with app-style interactions and navigation. The user should feel that it is similar to a normal app from the PlayStore.

- **Fresh and updated**

  Always up-to-date thanks to the service worker update process, and any changes are instantaneous, similar to refreshing a web page.

- **Installable**

  Allow users to “keep” apps they find most useful on their home screen without the hassle of an app store.

- **Linkable**

  Easily shared via a URL and do not require complex installation. The web app is to be installed by the vendors’ website after requesting the user.

Thus, A PWA is a web application that aims to deliver a native-like user experience on a mobile device, such as offline support and push notifications.

1.3 Existing Methods

Since around 2005 web development technologies have shifted from static to dynamic documents driven by server (PHP, ASP.NET) and client-side tools, and responsive web design. Despite an early push for web-based apps based on these technologies on devices such as the 2007 iPhone, attempts at web-apps failed by comparison to native-apps. Native apps provided a better user experience and booted faster compared to having to load in a browser at runtime. Packaged resources and direct access to hardware allowed native apps to perform much faster and to provide more features. By the mid-2010s, however, continued enhancements in HTML5, CSS3, and JavaScript, significantly more capable and standards compliant web browsers, along with powerful processors such as the A10 and Snapdragon 821 made performant hybrid-apps a viable alternative. [2]

1.4 Problem Formulation

Many companies are facing the problem of developing different applications for different platforms. They often need to develop two mobile applications, one for iOS and one for Android. On top of that, they need a web application that works well on both a desktop and a mobile device. Web applications are somewhat limited today when compared to native mobile applications but are moving forward all the time. We want to see if a Progressive Web Application can compete with a native application when it comes to performance, specifically response time of accessing different parts of the phone's hardware. [1]

Web development is moving forward at a blazing pace all the time. Mobile access of the web has already surpassed desktop access and making web applications that works seamlessly on mobile devices is more important than ever. Many companies are facing a challenge when developing applications. [1]

Often, they need to develop for three different platforms - iOS, Android and the web. Cross-platform frameworks like Cordova, Xamarin and React Native have tried to solve this for the mobile platforms with a write once and use
everywhere approach. What if you could develop one application that worked on all these devices? This is what Progressive Web Applications wants to solve. This could save both time and money for many developers and companies that need to develop applications for all these platforms. [1]

1.5 Native Mobile Apps

Native mobile apps are the most obvious way to build a mobile app. Objective-C or Swift on iOS, Java /Kotlin on Android and C# on Windows Phone. Each platform has its own UI and UX conventions, and the native widgets provide the experience that the user expects. They can be deployed and distributed through the platform App Store. [2]

The main pain point with native apps is that cross-platform development requires learning, mastering and keeping up to date with many different methodologies and best practices. [2]

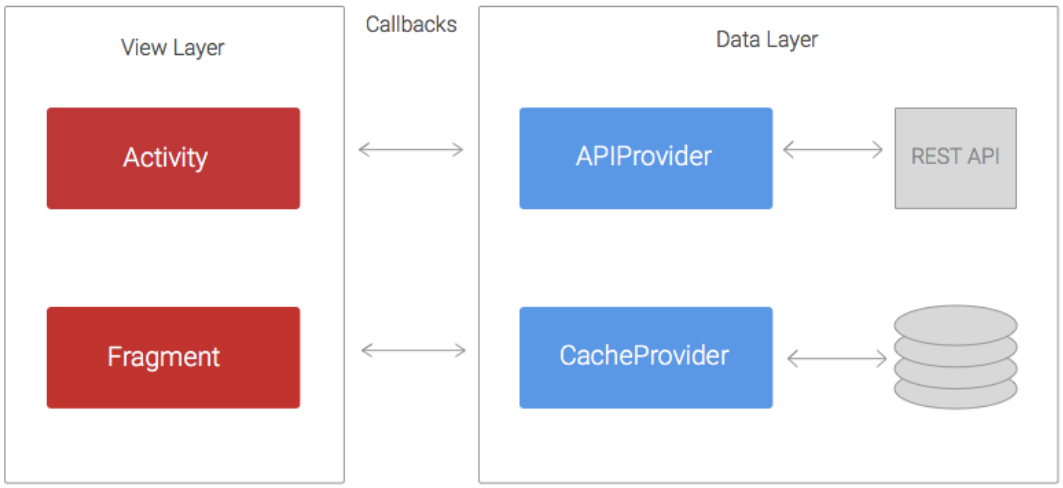

FIGURE 1.1 NATIVE ANDROID APP ARCHITECTURE

Chapter 2

ARCHITECTURE

2.1 Service Worker

A PWA must always work online as it is one of its main characteristics. Since the thing that allows the web app to work offline is the Service Worker, this implies that Service Workers are a mandatory part of a Progressive Web App.

A Service Worker is a JavaScript file that acts as a middleman between the web app and the network. Because of this it can provide cache services, speed the app rendering, and improve the user experience. It is a JavaScript worker that runs separately from the web page, so it cannot access the Document Object Model (DOM), a programming interface for HTML documents, directly. Instead, it communicates with the web page through an interface. Service workers allow developers to use features such as push notifications and offline functionality, two of the things that make a web application progressive. They also allow one to control how to handle network requests, for example, to serve cached content. Service workers are as of now supported by Firefox, Opera and Chrome. [1]

For security reasons, only HTTPS sites can make use of Service Workers, and this is part of the reason why a Progressive Web App must be served through HTTPS. Service Workers are not available on the device the first time the user visits the app. On the first visit the service worker is installed, and then on subsequent visits to separate pages of the site, this Service Worker will be called.

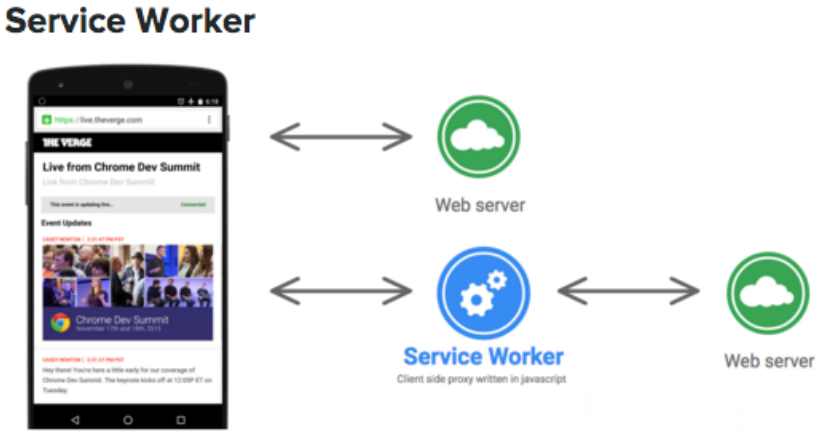

FIGURE 1.2 SERVICE WORKER

2.2 App Manifest

The App Manifest is a JSON file that you can use to provide the device information about your Progressive Web App. You add a link to the manifest in every header on each page of your web site: [4]

```
<link rel="manifest" href="/manifest.webmanifest">
```

This file will tell the device how to set:

• The name and short name of the app

• The icons' locations, in various sizes

• The starting URL, relative to the domain

• The default orientation

• The splash screen

The App Manifest gives the ability to control how your app appears to the user in areas where they would expect to see apps (for example, a mobile device's home screen), direct what the user can launch, and define its appearance at launch. [4]

Web app manifests provide the ability to save a site bookmark to a device's home screen. When a site is launched this way:

- It has a unique icon and name so that users can distinguish it from other sites.
- It displays something to the user while resources are downloaded or restored from cache.
- It provides default display characteristics to the browser to avoid too abrupt transition when site resources become available. [4]

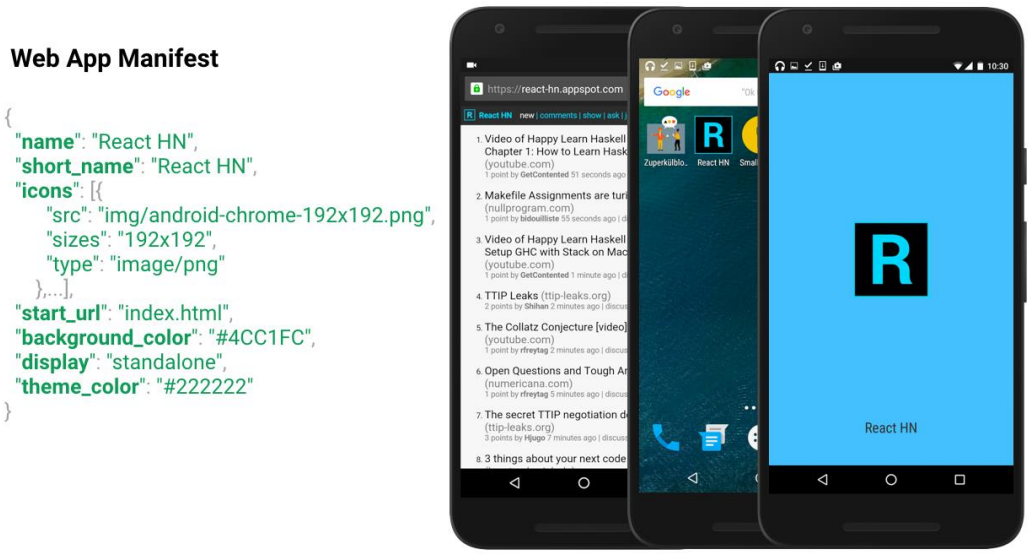

FIGURE 2.2 WEB APP MANIFEST

2.3 App Shell

The App Shell is not a technology but rather a design concept. It's aimed at loading and rendering the web app container first, and the actual content shortly after, to give the user a nice app-like impression. [2]

Some progressive web apps use an architectural approach called the App Shell Mode. For rapid loading, service workers store the Basic User Interface or "shell" of the responsive web design web application. This shell provides an initial static frame, a layout or architecture into which content can be loaded progressively as well as dynamically, allowing users to engage with the app despite varying degrees of web connectivity. Technically, the shell is a code bundle stored locally in the browser cache of the mobile device. [2]

Baseline Implementation:

- The Shell is cached using Service Worker
- Uses cached data from indexedDB or any web storage
- Updates cached view with online data when loaded
- Cached shell can load instantly on repeated visits.

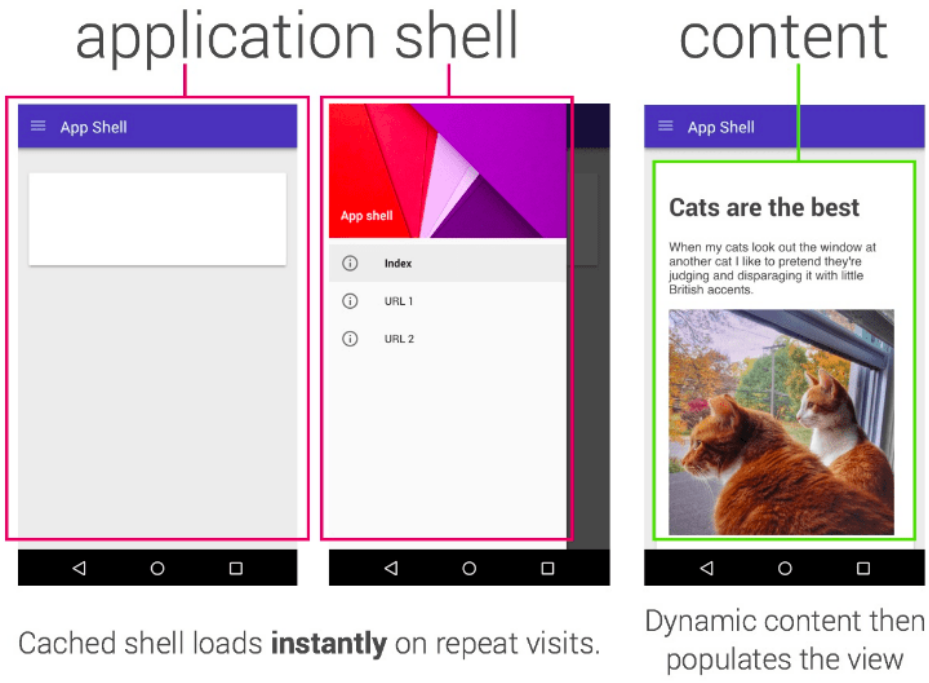

FIGURE 2.3 APP SHELL

2.4 Overall Architecture

The overall architecture of a Progressive Web App is as shown

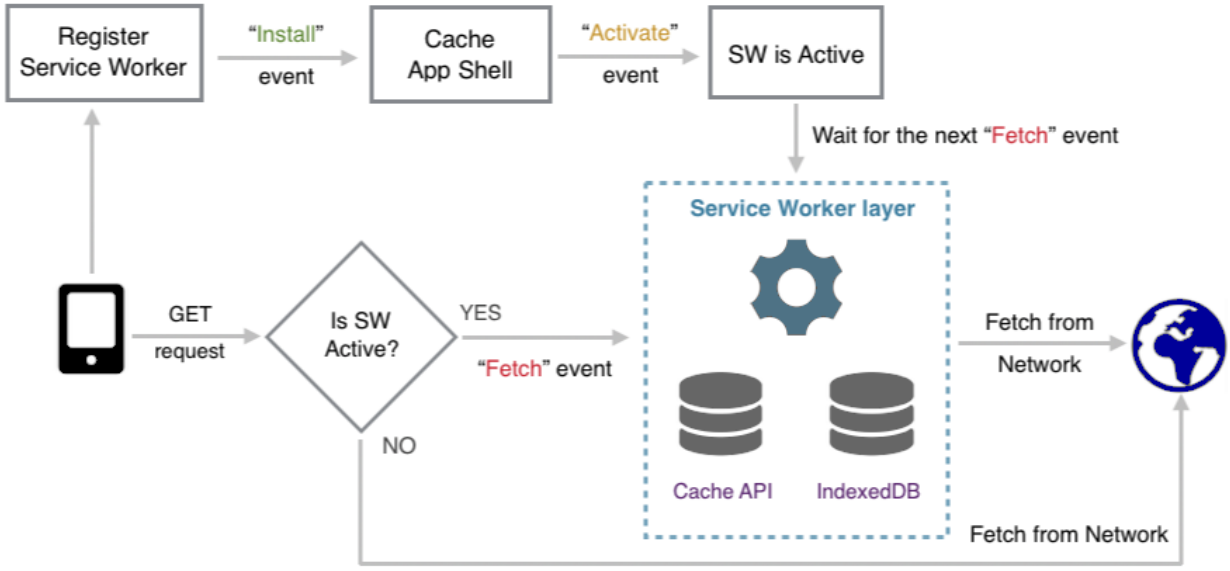

FIGURE 2.4 PROGRESSIVE WEB APP ARCHITECTURE

The experience architecture of the Progressive Web App is as shown below

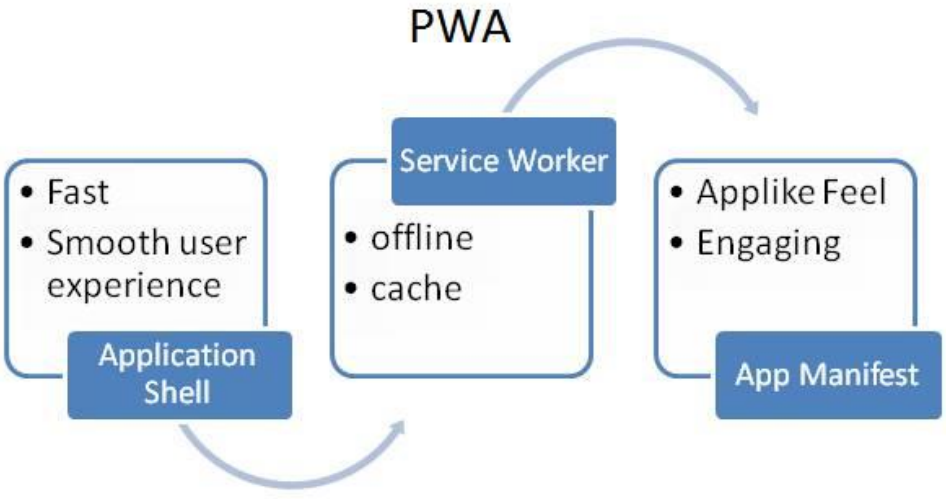

FIGURE 2.5 EXPERIENCE ARCHITECTURE

Chapter 3

ADVANTAGES & DISADVANTAGES

3.1 Advantages

Here are some of the merits of PWAs: [5]

- PWA are lighter. Native Apps can weigh 200MB or more, while a PWA could be in the range of the KBs.
- There's no native platform code, as the apps run on the browser, and not on the device.
- The cost of acquisition is lower (it's much more difficult to convince a user to install an app than to visit a website to get the first-time experience).
- Significantly less effort is needed to build and release updates.
- They have much more support for deep links than regular app-store apps.

3.2 Disadvantages

- Except Google Chrome, many browsers don't fully support PWA
- Limited to Android devices. PWAs are not available on iOS as of now
- Not available via Play Store. PWAs are installed by visiting a webpage and hence aren't available on app stores.
- Maybe unsafe. Spam or fraudulent websites may harm your devices by installing their PWA. [2]

3.3 Native Apps vs PWA

TABLE 3.1 NATIVE APPS VS PWA [1]

Native Apps	Progressive Web Apps
Easy to launch	No installation
Immediate value	Cross-platform
Offline	Easy deployment
Slow connections	Easy to update
Device Access	Links/ Bookmarks

Chapter 4

FEATURES

4.1 Offline

- Show content when on flaky networks
- Store data locally using service workers
- Caching pattern
    - Cache only or Network only
    - Try cache first and then network
    - Try network first and then cache

4.2 Background Sync

- App can request background sync
- Service worker triggers sync event when it is appropriate
- Also plans for periodic background sync

4.3 Push Notifications

- System level notifications like apps
- Register/receive push notification using Service Worker
- Ask to notify users with specific information
- Can send notifications even when page closed

4.4 Instant loading and smooth navigation

- For first-time visitors, load pages in <10s on 3G connections
- For repeat visitors, instant loading of page in <500 milliseconds
- Always scrolling at 60fps
- Content shouldn't jump as images are loaded

Chapter 5

RESULTS & DISCUSSIONS

Thus, Progressive Web Apps are user experiences that have the reach of the web, and are:

Reliable Load instantly and never show the dinosaur page, even in uncertain network conditions.

Fast Respond quickly to user interactions with silky smooth animations and no janky scrolling.

Engaging Progressive Web Apps are installable and live on the user's home screen, without the need for an app store and offer an immersive full screen.

This new level of quality allows Progressive Web Apps to earn a place on the user's home screen. Here is an example of Instagram as a Progressive Web App

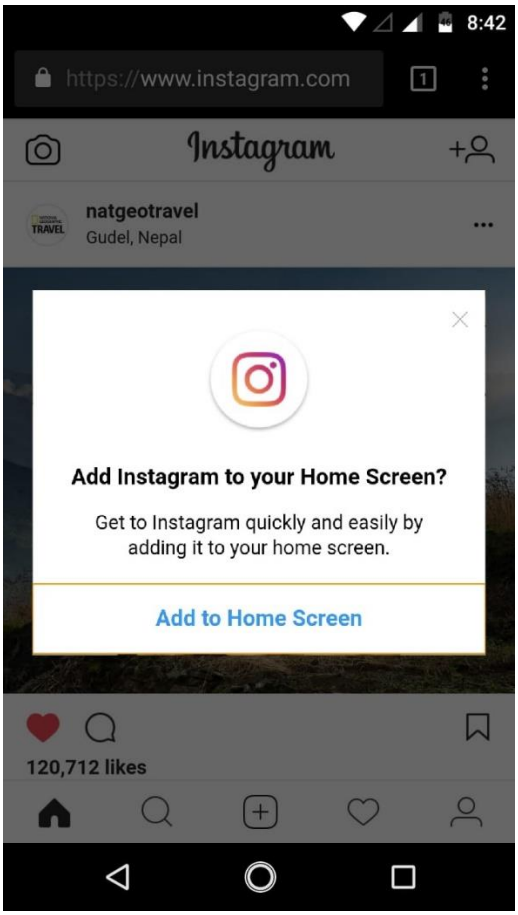

FIGURE 5.1 SCREEN 1

FIGURE 5.2 SCREEN 2

Chapter 6

CONCLUSION

Progressive Web Apps are the future of web development technologies and methods that allow us to perform our tasks on web browsers instead of our traditional native apps on mobile platforms.

This is an area where thoughtful application design and construction will give early movers a major advantage. Full Progressive App support will distinguish engaging, immersive experiences on the web from the “legacy web”. Progressive App design offers us a way to build better experiences across devices and contexts within a single codebase but it’s going to require a deep shift in our understanding and tools.

Building immersive apps using web technology no longer requires giving up the web itself. Progressive Apps are our ticket out of the tab, if only we reach for it.

REFERENCES

[1] “Comparing Progressive Web Applications with Native Android Applications” -an evaluation of performance when it comes to response time. **Rebecca Fransson**

[2] “An Introduction to Progressive Web Apps”. **Flavio Copes**

https://medium.freecodecamp.org/an-introduction-to-progressive-web-apps-6aa75f32816f

[3] “Progressive Web Apps” – Wikipedia | https://en.wikipedia.org/wiki/Progressive_Web_Apps

[4] “Progressive Web Apps – The next step in web app development”. **Rajat S**

https://hackernoon.com/progressive-web-apps-the-next-step-in-web-app-development-372235bf9a99

[5] “The Complete Guide to Progressive Web Apps”. **Flavio Copes**

https://flaviocopes.com/progressive-web-apps/

[6] “Progressive Web Apps”. **Nitheesh T Ganesh**

https://www.slideshare.net/iamntg/progressive-web-apps-72094520