

Parbat Thakur

# Evaluation and Implementation of Progressive Web Application

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

6 April, 2018

Author(s) Title	Parbat Thakur Evaluation and Implementation of Progressive Web Application
Number of Pages Date	34 pages 6 April, 2018
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Holvikivi Jaana, DSc.(tech), Principal Lecturer
<p>This thesis was carried out as a research project on Progressive Web Application. The main aim of this final year project was to evaluate and implement a progressive web application.</p> <p>This study was carried out by exploring relevant theory and developing a prototype PWA News App. React and React Material Web Components were used for the user interface whereas Web App Manifest, Service Worker, App Shell and Web Push Notification were implemented to make the app work offline, load fast even on flaky networks, install on home screen and send relevant push notification like a native app.</p> <p>The study demonstrates how PWA combines the best of web and native applications. Furthermore, it also shows some challenges that PWA is facing at the present scenario and states that PWA is still in its early stage of development. Meanwhile, Google is providing incentives and making a great effort in providing tutorials and support for it. Thus, with more browser and platform support, PWA might be the future of the mobile web.</p>	
Keywords	Progressive Web App, Service Worker, App Manifest, App shell Model, Push notification, React.js, Lighthouse

## Contents

1	Introduction	1
2	Web and Mobile Applications	3
2.1	Progressive web application	3
2.2	Native apps	5
2.3	Hybrid apps	5
3	Progressive Web App Components	6
3.1	Web App Manifest	6
3.2	Service Workers	10
3.2.1	Service Worker Life Cycle	12
3.3	Application shell Model	17
3.4	Web Push and Notifications	18
4	Technology Used	21
4.1	React.js	21
4.1.1	Create React App	24
4.2	Web Pack	25
4.3	Babel	26
4.4	Other Online Services	26
5	Implementation of PWA News App	27
5.1	Development Environment	27
5.2	Implementation	28
6	Results and Evaluation	31
7	Conclusions	33
8	References	34

## List of Abbreviations

PWA	Progressive Web Application
UI	User Interface
UX	User Experience
API	Application Programming Interface
HTTPS	Hypertext Transfer Protocol Secure
SDK	Software Development Kit
SSL	Secure Sockets Layer
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
URL	Uniform Resource Locator
SEO	Search Engine Optimization
DOM	Document Object Model
JSX	JavaScript XML
CLI	Command Line Interface
GPS	Global Positioning System
URL	Uniform Resource Locator

## 1 Introduction

Technological innovation always has an impact on how products and services are designed. In recent years, there has been unprecedented growth in the field of Web. With the introduction of new and enhancement of existing technologies in the world of web, things that were not possible before have now come into existence. The use of technologies is found in every field from education to health, research and agriculture. Most of the businesses have now adopted Internet as their business platform and the trend seems to be increasing. Thus, there is a need for developing fast, reliable, engaging and robust applications.

Native and Web are two major types of applications. Native apps are platform dependent (Android, iOS, Windows) built using specific programming languages and Software Development Kit, whereas Web apps are platform independent websites which in many ways look and feel like native. Web apps are run by a browser and typically written in HTML5. Native apps have full access to the device hardware while web apps still lack some of the access.

During the past few years, the use of native mobile applications has shown growth as compared to mobile web. Web apps are behind native apps in terms of performance, reliability and engagement. Businesses and developers often see the need to develop native mobile applications to overcome the limitations that the web as a platform imposes on mobile devices. However, native apps have its own shortcomings. When it comes to user's reachability, native apps are behind web apps. Native apps take more resources and time to develop, maintain and distribute applications for specific platforms. Also, native app publishing is a tedious process. In addition, to use the native app, users need to go through many steps which include signing up to the respective store, checking the memory, downloading, finalizing download by installing and finally opening to use it. A study revealed that, on average, an app loses about 20 % of its users for every step between the user's first contact and start to use. [1] Many users also find this process difficult and daunting. This is a huge disadvantage for both companies and developers.

Both the web and native platforms have their own shortcomings, so there was a need for a platform which can combines the capabilities and experiences of native apps with the

reach of web. Progressive Web App is simply that platform. Progressive Web Apps are user experience which have the reach of web and are reliable, fast and engaging. It combines the best of the web and the best of the native apps. It was first introduced in the Google I/O developer conference in May 2016 in San Francisco. It addressed most of the features lacked in web apps. The use of Service Worker, App shell, Web App Manifest and Push notification made PWA feel and behave like a native app. As PWA is a web app, it can be used in all platforms, reducing the cost and resources and increasing reachability.

This thesis was carried out as a research project on the topic 'Progressive web application'. The main objective of the thesis was to evaluate and implement Progressive web application and highlight its specific features. Thus a prototype, PWA News App was developed which illustrated how PWA is combining the best of native and web apps. The study also shows the features, benefits and future aspect of PWA.

## 2 Web and Mobile Applications

This chapter provides a brief theoretical background on web and mobile applications, i.e. Progressive Web Application, Native and Hybrid apps.

### 2.1 Progressive web application

Progressive Web Application is not a new technology or framework rather it is best practices which have been adopted by the web to give a native app like feeling to the user. Components such as Service worker, App Shell, Web App Manifest and Push Notification, which are discussed in the following section, work together to give native like feeling to PWA. It can be installed on the user's home screen with one tap. A user can enjoy full-screen experience without going through the hassle of the downloading process. It loads faster even on a flaky network and works offline. It gradually develops with interaction and sends relevant push notification thus increasing user engagement.

The popularity of PWA is growing at a very fast pace in the field of e-commerce, business, online news portal and other fields because of its peculiar characteristics listed below as follows:

- Progressive

PWA works for all user on all browser and builds up continuously; taking the benefits of features found in user's device and browser.

- Responsive

PWA's UI fit on all devices forms, factor and size: mobile, desktop and tablet. Responsive feature is achieved using the material design, fluid grid concepts, CSS3 media queries and flexible images. [2]

- Home screen Shortcut

Due to W3C web app manifest and service worker registration scope, search engines identify PWA as an application. It also increases the probability to be found easily on search engines compared to the native app.

- Native App-like

Implementation of design concept App-shell architecture makes PWA unique and divide application functionality from its content with least possible page refreshes to give native app look. PWA, when launched from the home screen, has an entirely native app-like look with a splash screen.

- Connectivity-independent

Implementation of service worker makes it able to work offline and give good performance even on a flaky network. PWA does not treat loss of connectivity as an error, but as an eventuality, which can be planned for, and handled with grace.

- Fresh

New content published gets an update once the user is connected to the Internet due to the service worker update process. Application shell and content, after it is cached always load from the local storage.

- Safe

Implementation of HTTPS connection and SSL certificate to serve the page is a must to prevent man-in-the-middle attacks, password intruding and making sure content is not manipulated.

- Push Notifications

Push API and Notifications API make the user more likely to revisit PWA by the user using push notifications. PWA can receive messages pushed from the server, which can be shown as a notification, and the user is notified about the updates. This helps for re-engagement and bringing the user back to the application. Progressive Web Apps' notifications have a completely native feel and are like those of native app notifications. [3]

PWAs have low friction as they are a product of the web, i.e. faster and cheaper to develop lowering the user acquisition costs and always up-to-date. However, PWA faces some challenges. It has issue with cross browser support. Google Chrome, Chrome for Android, Samsung Internet have good support for PWA while Firefox and Safari still lack good support. PWAs also have limited functionality as they cannot access all device-specific hardware. They can only support hardware that is supported by HTML5. [4]. It also lacks the cross application login support.



## 2.2 Native apps

Native apps are platform dependent (Android, iOS, Windows) built using the specific programming languages, SDK. Developers should stick with Java for Android, C# for windows and swift for iOS. Native apps have full access and take full advantage of device features such as access to the accelerometer, camera, compass, GPS, incorporate gestures and APIs, thus provide great UI, UX, performance and reliability. Native application is installed on the user's device via specific app stores (Apple's app store or Google play store).

Native apps are platform dependent thus cannot be deployed on multiple platform. The cost is high and development time is longer. A user must install the application via respective store on device to use thus set drawbacks if the user wants to use the application just one single time or periodically. Also, updating native app is tedious.

## 2.3 Hybrid apps

Hybrid apps are platform independent built using web technologies HTML5, CSS 3 and JavaScript wrapped inside a native container Cordova. Once the app is developed, it can be deployed on multiple platforms. Hybrid apps have access to most device features like native but when it comes to 3D, fluid animations, multi-touch, graphics, transition and gaming, their performance decreases. However, the cost and time of development is lower than native. Hybrid apps should also be installed on the device and need to be updated from time to time.

### 3 Progressive Web App Components

Building progressive web apps and meeting all the requirements based on Performance, Accessibility, Best Practices and SEO is quite challenging. To make that happen, all the components of PWA, i.e. Service Worker, Web App Manifest, Application Shell model, and Web Push notification need to be implemented with great care and work hand in hand. All these components are described in the following section.

#### 3.1 Web App Manifest

Web App Manifest is a simple JSON file containing information: name, short\_name, description, icons for different device resolution, start\_url, display mode, theme color of the application. The use of Web App Manifest installs the web app in the user home screen between the native apps. As a result, the user can get quick access and enjoy full-screen display like with the native app. [5] Listing 1 shows the manifest.json file for a web application.

```
{
  "short_name": "PWA app",
  "name": "PWA News App",
  "icons": [
    {
      "src": "favicon.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "./splash_images/splash-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff",
  "gcm_sender_id": "482941778795"
}
```

Listing 1. Web application manifest for PWA

As shown in listing 1, most of the terms used in manifest.json file are easily understandable and self-explanatory: name is what appears on the splash screen, short\_name is what appears on home screen, icons are used for the depiction of characteristics of an image with an array of object, which consists of scr, size, and type. Similarly, start\_url is URL to open when the icon is clicked; display controls the display mode the app launches in. The mode could be either standalone or full screen. The standalone mode opens the app without the browser's user interface, such as the location bar. [6] A web app with manifest file gives an app look as depicted in figure 1.



Figure 1. Combination of web site with manifest to give an app look. Copied from [7]

As illustrated in figure 1, Web app combines with manifest file so that the user can install app in their device home screen. Web App Manifest is linked with index.html of the page of an application so that the browsers can identify it as seen below in listing 2.

```
<head>  
<link rel="manifest" href="/manifest.json">  
</head>
```

Listing 2. Importing Web App Manifest in the head of the web page.

For the web app to be able to appear as install banner on sites and provide an app-like experience, the web app must fulfill the following criteria:

1. The site needs to be served over HTTPS.
2. The site needs to have a service worker registered.
3. The web app manifest file of the site should have at least the four mandatory fields name or short\_name (preferably both, start\_url, icons and display). [8]

When the above criteria are fulfilled, the browser determines and prompts a web app install banner as shown in figure 2.

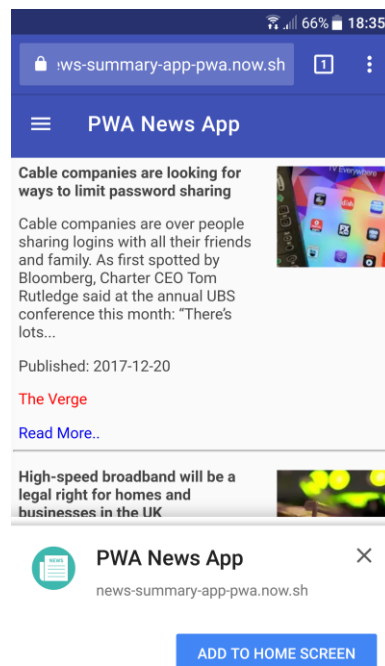


Figure 2. A Web app install banner on Chrome Android

As illustrated in Figure 2, a web app install banner is triggered by the browser when manifest file meets above mentioned criteria so the user can install app on home screen and enjoy full screen experience like a native app.

A web app manifest file can be verified manually by using the Manifest tab on the Application panel of Chrome DevTools, as shown in figure 3.

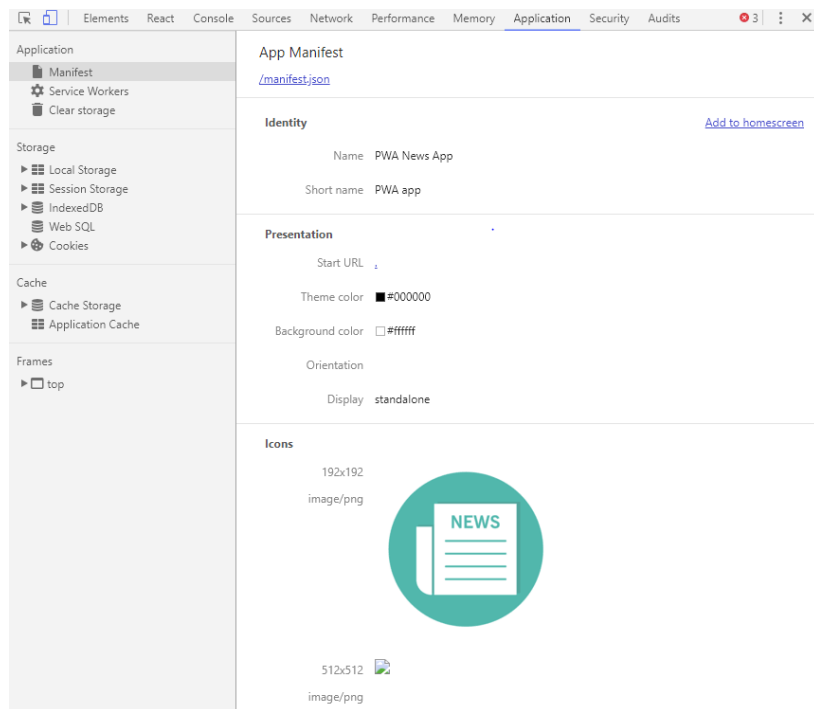


Figure 3. Web App Manifest in Chrome Developer Tools Application panel

Web app manifest has compatibility issue with browsers. Not all browser support manifest file. The compatibility of web app manifest with different browser can be checked by using online validation tool as shown in figure 4.

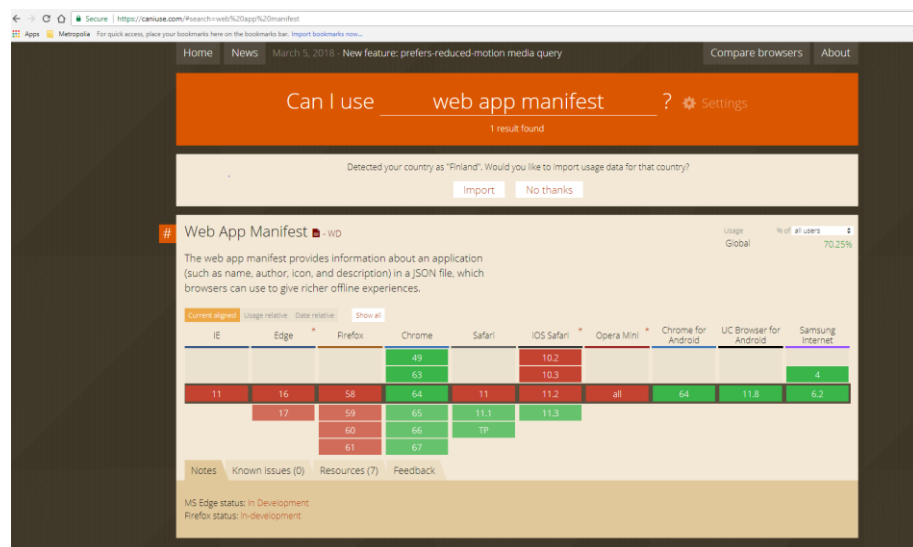


Figure 4. Result for Web App Manifest compatibility with different browser. [9]

As illustrated in figure 4, Web App Manifest works best for the Chrome, Chrome for Android, UC Browser for Android, Safari, iOS Safari have recently started supporting the manifest file whereas, Firefox and MS Edge have no support till date.

### 3.2 Service Workers

The web as it is today is rich, beautiful and useful but when users visit a web app; when they have poor connection or have lost their connectivity, they are shown the page 'there is no internet connection' as shown in figure 5.

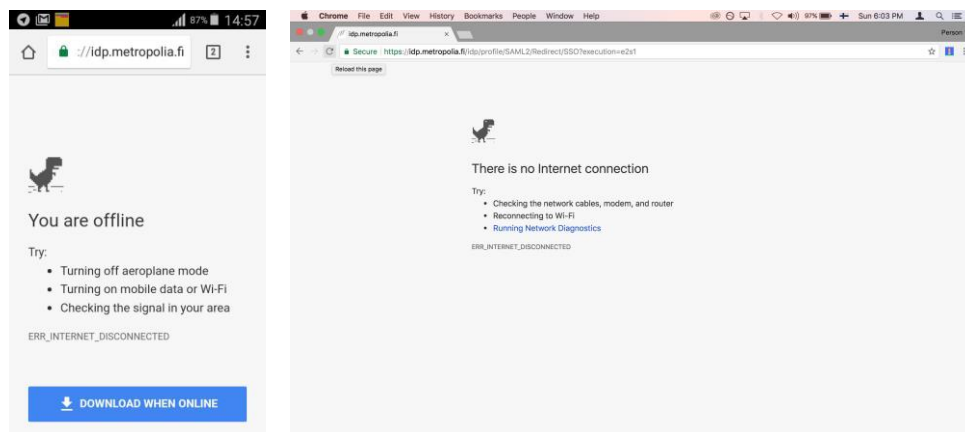


Figure 5. A web app in offline state in mobile and desktop view

As illustrated in figure 5, offline state of web app could not provide any useful information to users. But the introduction of a service worker has turned this error as something that can be handled with grace.

A Service Worker is an event-driven script that runs in the background separately from the webpage, reacts to events and intercept network requests of application or website with server and resources. It works as proxy between the network and the browser. It can run even when the application is closed, thus it serves to trigger events even when the site is closed. Features like push notifications and background sync are possible in web today because of Service Workers. In the future, the service worker shall back other cool features like periodic sync. [10] Since the service worker can intercept the request, modify content or even completely replace with new responses, only pages served over secure connections (HTTPS) can register a service worker. This is to protect users and

prevent man-in-the-middle attacks. [10]. Figure 6 shows the caching strategy of service worker.

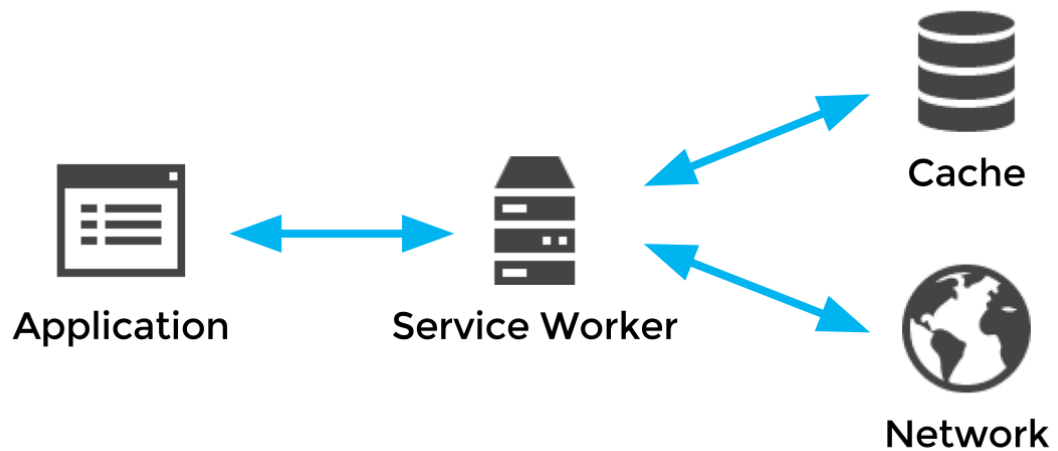


Figure 6. Service Worker caching strategy. Copied from [11]

As illustrated in figure 6, the service worker sits in between the application and the network and decides when it should get the content from cache and when to hit the network.

A simple news web app written without Service Worker will show standard “You are offline” message, when the connection is lost. However, an app, which has Service Worker running, can show cached GUI, run scripts and can even show the user cached contents from a previous visit as shown in the figure below. A well-written service worker can also log data sent to server and dispatch when the internet connection is available via sync events, and receive push notifications from the server via push events as shown in figure 7. [12]

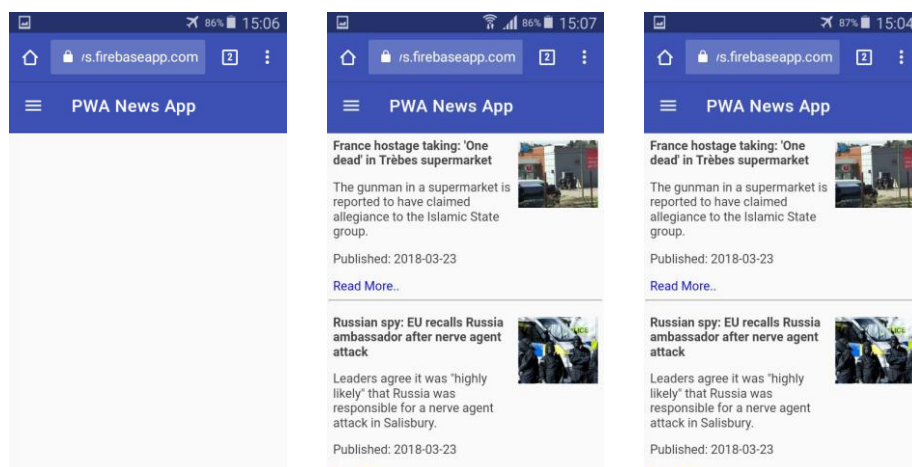


Figure 7. Service worker caching of the app shell, allowing it to load without the network [13]

As illustrated in figure 7, the service worker caches app shell and content when an application is loaded over the network and shows it to the users when they have poor connection or out of network.

Service Worker is taken as progressive enhancement of the application or webpage. Progressive enhancement improves the content and capabilities of an application. Thus, an application should be fully functional on any platform before adding service workers. A better experience can be gained by the user with compatible browsers with the application. Figure 8 depicts the compatibility of browsers for service workers.

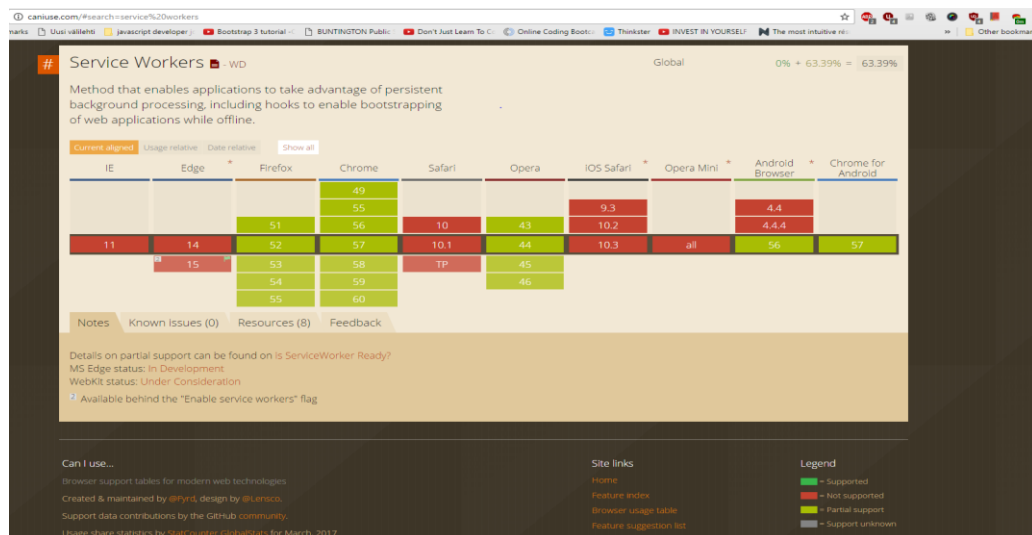


Figure 8. Compatibility of browsers for service workers. [9]

As shown in figure 8, most of the browsers such as Chrome, Firefox, Opera, Chrome for Android Safari, iOS Safari, Edge support Service Worker. Opera Mini, and Internet Explorer do not support service worker till date.

### 3.2.1 Service Worker Life Cycle

As service worker is an event-driven script, it has a short life span. It wakes up with events and runs only if it needs to process the event. Through Service Worker the developer can treat the network as enhancement, control caching of resources on a proper way. Control over the cached resources plays an important role in developing offline



application, which is one of the key features of PWA. Via service worker the webpage is available even offline and with cached data loads faster even on a flaky network. [13]

A service worker has a totally separated lifecycle from a web page. Figure 9 illustrates a simplified version of a service worker lifecycle on its first installation.

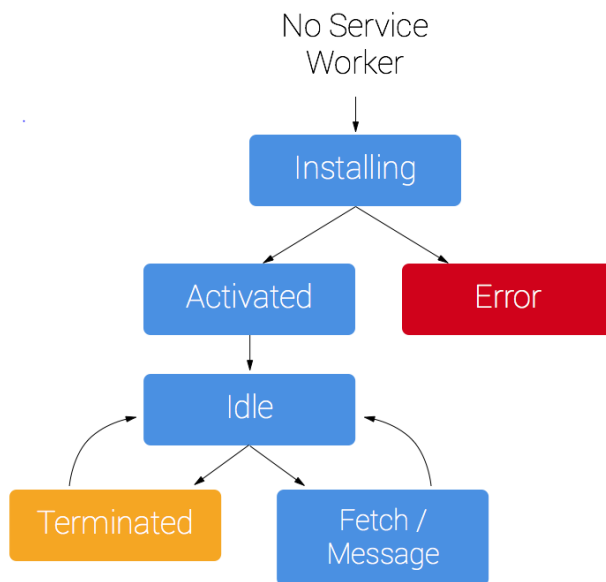


Figure 9. Simplified version of service worker lifecycle. Copied from [14]

As illustrated in figure 9, the service worker installation process begins with the registration of it in JavaScript, which declares the browser where the service worker JavaScript file is registered.

```

if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('/sw.js').then(function(registration) {
      // Registration was successful
      console.log('ServiceWorker registration successful with scope: ', registration.scope);
    }).catch(function(err) {
      // registration failed :(
      console.log('ServiceWorker registration failed: ', err);
    });
  });
}

```

### Listing 3. Browser support check for service worker

Listing 3 code makes sure whether the service worker API is accessible, and if it is there, the service worker at /sw.js is registered when the page is loaded. [10]

Once the registration process is completed, the browser starts to install the service worker by defining a callback for the install event and setup an environment to cache the files.

```
self.addEventListener('install', function(event) {  
  });
```

During the install event, service worker caches the static content, and when the caching is successful, the process of activation starts. If the caching process fails, then activation event gets terminated and service worker tries to install in the next attempt. Listing 4 illustrates the installing of a service worker.

```
var CACHE_NAME = 'my-site-cache-v1';  
var urlsToCache = [  
  '/',  
  '/styles/main.css',  
  '/script/main.js'  
];  
  
self.addEventListener('install', function(event) {  
  event.waitUntil(  
    caches.open(CACHE_NAME)  
      .then(function(cache) {  
        console.log('Opened cache');  
        return cache.addAll(urlsToCache);  
      })  
  );  
});
```

### Listing 4. Installing Service Worker

As shown in listing 4, shows the successful caching and installation process which triggers the activation event.

An activated service worker starts to receive fetch, push and sync events, when the user maneuvers to a different page as shown in Listing 5.

```
self.addEventListener('fetch', function(event) {
  event.respondWith(
    caches.match(event.request)
      .then(function(response) {
        // Cache hit - return response
        if (response) {
          return response;
        }
        return fetch(event.request);
      })
  );
});
```

**Listing 5. Matching Cached Value Response by Service Worker during fetch request**

Listing 5 shows the service worker responding to the fetch, push and sync events. As mentioned earlier, a service worker is event driven and has a lifecycle totally separated from a webpage, which can be manipulated as per need. A service worker can either be terminated to save memory or carry out the next step where it controls events, fetches and messages. [10]

After the completion of all the stages, a service worker registration can be checked by using the Service Worker tab on the Application panel of Chrome Dev Tools as shown in figure 10.

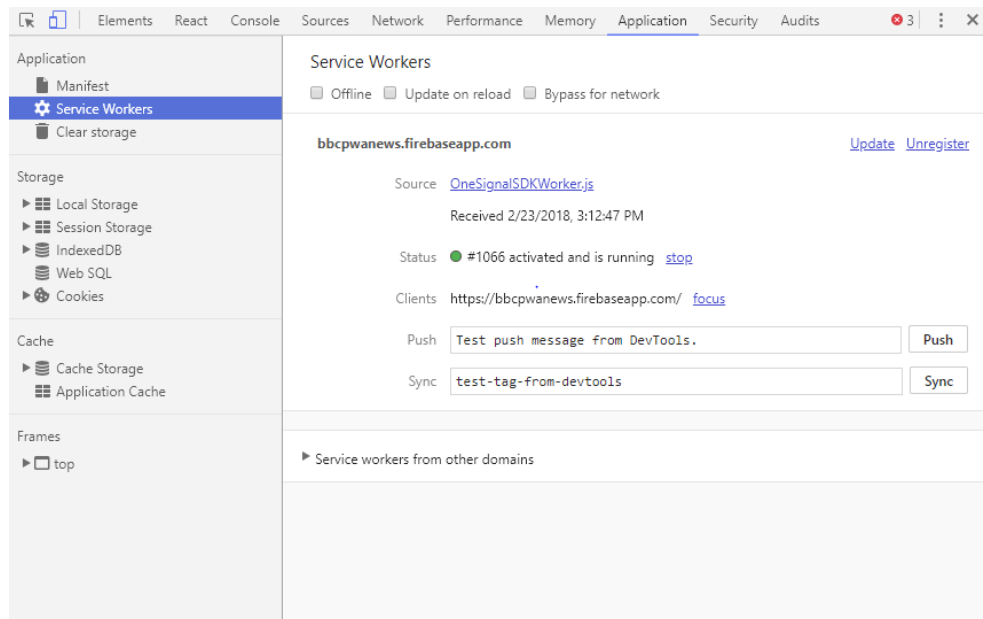


Figure 10. Service Worker Registration shown in Chrome Developer tool

As illustrated in figure 10, the registered service worker gets updated on reload, improves the offline experience and performance of the page by requesting only the raw data; presenting style and layout information from the data cached as shown in figure 11.

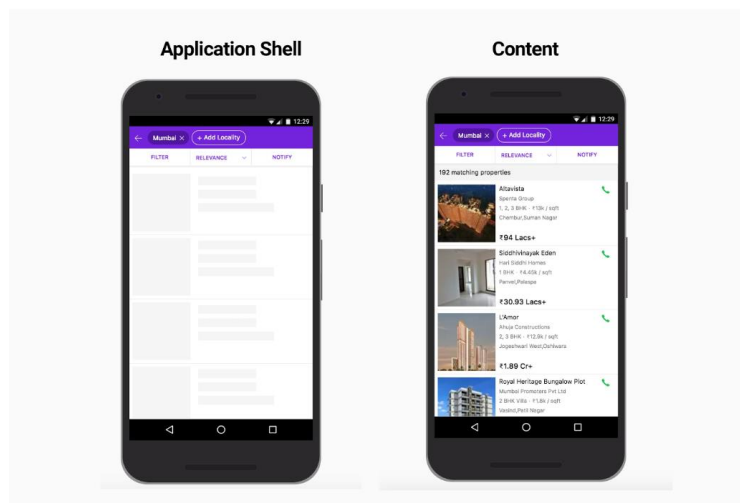
Path	Content-Type	Content-Le...	Time Cached
index.html	text/html; chars...	1,686	2/23/2018, 3:1...
static/css/main.68b82585.css	text/css; charset...	739	2/19/2018, 11:...
static/js/0.972293f2.chunk.js	application/java...	34,130	2/23/2018, 3:1...
static/js/1.ac8d81aa.chunk.js	application/java...	53,904	2/19/2018, 11:...
static/js/2.8b0c3f70.chunk.js	application/java...	53,755	2/19/2018, 11:...
static/js/3.d2630bac.chunk.js	application/java...	53,756	2/19/2018, 11:...
static/js/4.c9fa42ef.chunk.js	application/java...	53,759	2/19/2018, 11:...
static/js/5.33502253.chunk.js	application/java...	53,763	2/19/2018, 11:...
static/js/6.7d0c496a.chunk.js	application/java...	53,772	2/19/2018, 11:...
static/js/7.2c7033ef.chunk.js	application/java...	53,761	2/19/2018, 11:...
static/js/8.72b7ad6a.chunk.js	application/java...	53,760	2/19/2018, 11:...
static/js/9.ec121d54.chunk.js	application/java...	280	2/19/2018, 11:...
static/js/main.ff769941.js	application/java...	50,945	2/23/2018, 3:1...
static/media/bbc.1af813a7.svg	image/svg+xml	626	2/19/2018, 11:...
static/media/menu.c800523f.svg	image/svg+xml	209	2/19/2018, 11:...

Figure 11. Service Worker fetching raw and cached data in Chrome Developer tool

Figure 11 shows that, layout, style and static content is fetched from the cached data whereas dynamic content is fetched from the network which improves the loading speed and performance of web app.

### 3.3 Application shell Model

Application shell (architecture) is the minimum HTML, CSS and JavaScript needed to build basic representational User Interface of a PWA. It is one of the main factors that provide instant loading, smooth UX and good performance on repeat visit. An app shell is cached immediately which means that the shell files are loaded once over the network and then saved to the local device. Thus, whenever the user uses the app, the shell files are rendered from the local device's cache, which gives fast startup times. App shell model divides application into shell and content, as shown in figure 12.



Housing.com use an AppShell with placeholders for content. This is nice for improving perceived performance as content fills in these holders once fully loaded.

Figure 12. Shell and content description in app-shell architecture copied from. [14]

As illustrated in figure 12, the shell includes the static parts of the application needed to show the content. It keeps the UI local and gets all the content dynamically through an API. Thus, the caching of the whole shell ensures that even if there is no internet connection or connectivity is poor, the application loads at least the familiar UI and the user is not conferred with bare screen or a default connection failure message. [5]

An app shell is the same as the bundle of code that is published to an app store when publishing a native app. It is the most basic component essential to get app off the ground

but does not contain any data. The content is presented within the shell. Depending on the application type, the content can also be cached.

An app shell architecture focuses on speed, performance, instant loading and regular updates, like native app but these depend on how the app shell is designed and implemented. There are several constraints for designing App-shell architecture. However, a concrete implementation of an App-shell should at least include the following design principles:

- What needs to be on the screen instantly?
- What UI plug-in and components are indispensable to application?
- What auxiliary resources are required for the app shell? For instance, images, styles, JavaScript, etc. [14]

### 3.4 Web Push Notifications

Push notification is one of the key components of PWA. It had been missing from the web and was central to the gulf between native and web app.

A notification is a message that pops up on the user's device. It can either be triggered locally by an open application or be pushed from the server when the app is in idle state. A push notification provides timely updates of content and data, which the user selects to opt-in. It plays an important role in re-engaging users and bringing them back to apps repeatedly. [6] Push notification consists of: Notification API and Push API.

#### Notification API

The Notification API allows a web page, or a service worker, to create and control the display of system notification. Notifications appear on the device's UI (outside the browser) and thus exist outside of the context of any tab or browser window. Since, they are independent of any tabs or browser windows, they can be created even after a user has left the site.

The Notification API allows to display notifications to the users. A permission is needed from the user to display notifications.

```
Notification.requestPermission(function(status) {
  console.log('Notification permission status:', status);
});'
```

This event triggers the browser to ask for a permission from the user, as shown in figure 13.

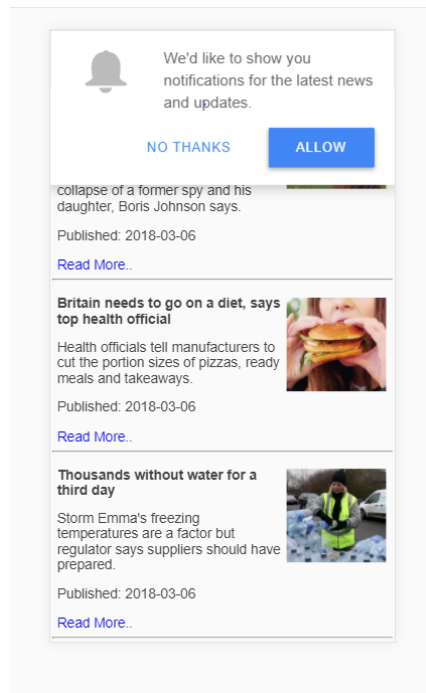


Figure 13. Push Notification Permission prompt

As shown in figure 13, user permission is needed to show notifications, and when a permission is granted, showNotification display the sytem notification as shown in listing 6.

```
function displayNotification() {
  if (Notification.permission == 'granted') {
    navigator.serviceWorker.getRegistration().then(function(reg) {
      reg.showNotification('Hello world!');
    });
  }
}
```

Listing 6. Notification using Service Worker

## Push API

The Push API allows handling messages that are pushed to the client from a server via the push service used by the browser. When service worker gets registered by the push service, it gives push events inside the service worker. On this event, service worker fetches data that needs to be shown in the notification and uses Notification API to display a notification. Push API also have issues with browser support as shown in figure 14.

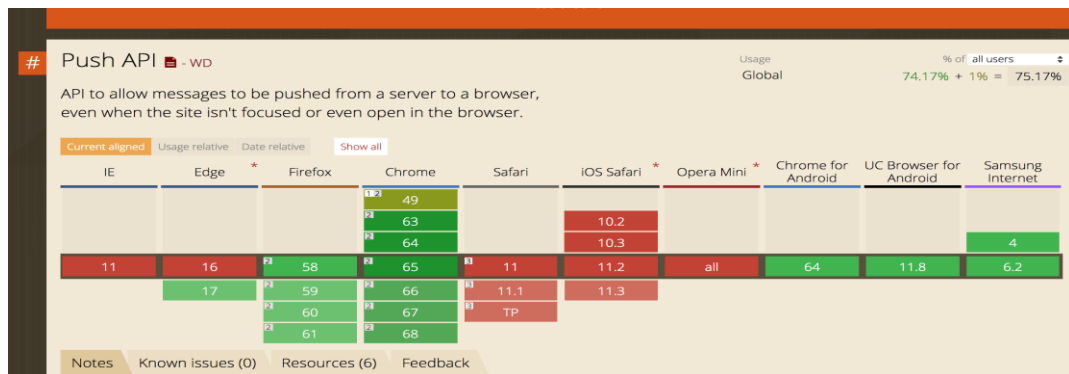


Figure 14. Browser support for push notifications [15]

As illustrated in figure 14, Chrome, Chrome for Android, Firefox, Opera and Edge have support for push notification whereas Safari does not have any support for the Push API. However, Safari has its own API for sending notifications to Safari users.



## 4 Technology Used

This section describes the technologies used to accomplish the project in this study such as React.js and ES6, Material UI components, Web Pack, Babel and Lighthouse. The tools used in the project were chosen with great care to decrease the boilerplate code and make the configuration process hassle free.

### 4.1 React.js

React is a JavaScript library, used for building UI. Facebook first introduced React in the year 2013. It was made open source two years later. React uses the virtual DOM for manipulating the UI as a result it gives blazing fast performance and making site most interactive as shown in figure 15. [16]

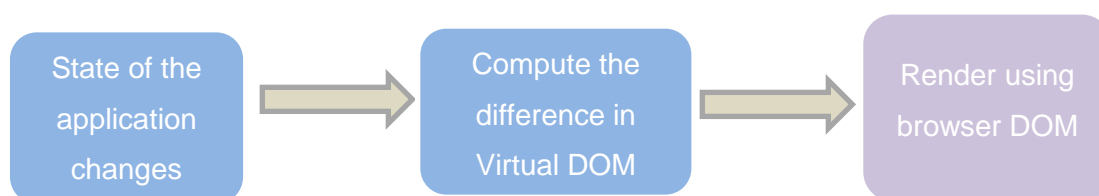


Figure 15. Rendering with the Virtual DOM in React.

As illustrated in figure 15, React keeps good track of an in-memory Virtual DOM. Facebook developed React specifically to solve one problem: building large application with data that changes over time. Thus, React adopted the declarative, Component-Based, Learn Once and Write Anywhere strategy. React makes it seamlessly easy to create interactive UIs. Designing simple views for each state in application, React smoothly updates and renders just the needed components when the data changes. Declarative views in React make code easy to predict and debug. [16]

React uses component-based structure. Component logic are written in JavaScript instead of templates which makes the easy passage of rich data through the web app and keeping state out of the DOM. Problems are solved by creating reusable components and when components get complex, they are broken down into smaller and simpler ones. React components are similar to JavaScript function. [16]

React is also used in the server side using NodeJs and power mobile apps using React Native. All these features collectively make React the most loved one among the developers compared to other frameworks and libraries. It has outdone its competitors AngularJs, Angular 2 and Ember within three years of its release as seen in Figure 16. Now, React is taken as a paradigm in the world of front-end web development. [16]

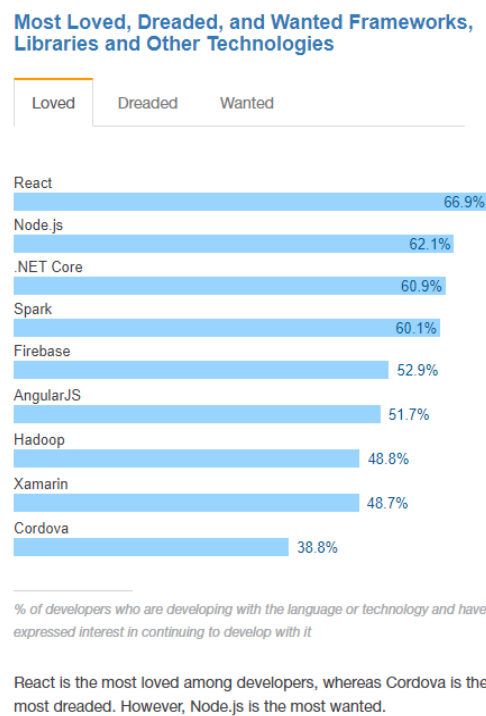


Figure 16. Most loved technologies of 2017 in web development [17]

As shown in Figure 16, React was the most loved technology in 2017 at 66.9% whereas the least loved one is Cordova with 38.8% only.

React uses an XML-like syntax JSX, a syntax extension to the ECMAScript to describe the component's DOM representation. It looks similar to that of the mark-up language HTML or XML but it is JavaScript centric. It makes code more readable and writing JSX gives the feel of writing HTML. It is based on separation of concerns rather than technology as it combines mark-up, style and behaviour of components in one file rather than having each separate files. However, it is not compulsory to use JSX in React application but the use of JSX helps for easy development, error handling and increase performance of React application. [18]

## React Components, Props and State

Components are like JavaScript functions, which allow splitting the UI into independent and reusable pieces. Components are the building blocks in a React application. React components are created with a method `React.createClass()`. Components accept arbitrary inputs called props and state. Props is the communication channel, which acts as a bridge between parent and child components. Props is used to display static immutable data while state data type is used to dynamic data. State is private and fully controlled by the component. [18] Listing 7 depicts the representation of components, props and state in React.

```
import React from 'react';
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      title: "Title from props",
      content: " Content from props"
    };
  }
  render() {
    return(
      <div>
        <Title titleProp={this.state.header}/>
        <Content ContentProp={this.state.content}/>
      </div>
    );
  }
}

class Title extends React.Component() {
  render() {
    return(
      <div>
        <h1>{this.props.titleProp}</h1>
      </div>
    );
  }
}

class Content extends React.Component() {
  render() {
    return(
      <div>
        <h1>{this.props.contentProp}</h1>
      </div>
    );
  }
}
```

```

    );
  }
}

}

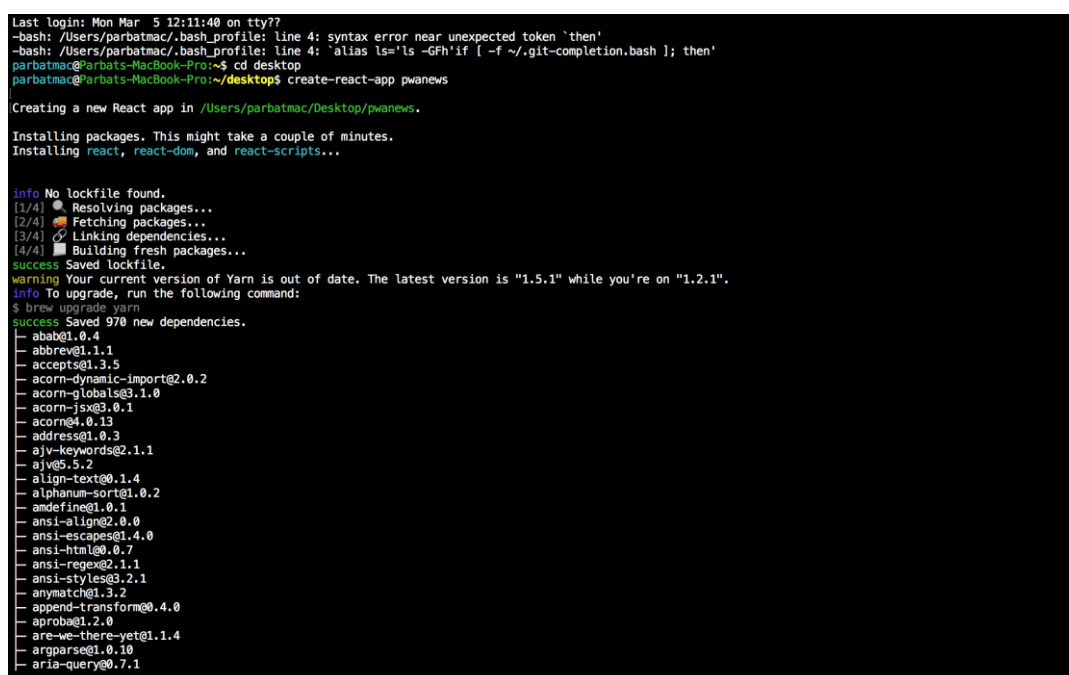
export default App;

```

## Listing 7. Representation of components, props and state in React

### 4.1.1 Create React App

Create react app is a tool to get started with React.js app development. Create react app with pre-configured Webpack, Babel and other necessary tools makes an app development process smooth, and hassle free. It also comes pre-configured with Service Worker, which is one of the key components of PWA. Create react app can be easily installed from the terminal with the command: `npm install -g create-react-app`, as shown in Figure 17.



```

Last login: Mon Mar  5 12:11:40 on tty??
-bash: /Users/parbatmac/.bash_profile: line 4: syntax error near unexpected token `then'
-bash: /Users/parbatmac/.bash_profile: line 4: `alias ls='ls -GFh'if [ -f ~/.git-completion.bash ]; then'
parbatmac@arbats-MacBook-Pro:~$ cd desktop
parbatmac@arbats-MacBook-Pro:~/desktop$ create-react-app pwanews

Creating a new React app in /Users/parbatmac/Desktop/pwanews.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
warning Your current version of Yarn is out of date. The latest version is "1.5.1" while you're on "1.2.1".
info To upgrade, run the following command:
$ brew upgrade yarn
success Saved 970 new dependencies.
├─ abab@1.0.4
├─ abbrev@1.1.1
├─ accepts@1.3.5
├─ acorn-dynamic-import@2.0.2
├─ acorn-globals@3.1.0
├─ acorn-jsx@3.0.1
├─ acorn@4.0.13
├─ address@1.0.3
├─ ajv-keywords@2.1.1
├─ ajv@5.5.2
├─ align-text@0.1.4
├─ alphanumeric-sort@1.0.2
├─ amdefine@1.0.1
├─ ansi-align@2.0.0
├─ ansi-escapes@1.4.0
├─ ansi-html@0.0.7
├─ ansi-regex@2.1.1
├─ ansi-styles@3.2.1
├─ anymatch@1.3.2
├─ append-transform@0.4.0
├─ aproba@1.2.0
├─ are-we-there-yet@1.1.4
├─ argparse@1.0.10
├─ aria-query@0.7.1

```

Figure 17. Installing create-react-app project

As illustrated in figure 17, `create-react-app pwanews` creates a new React app in `~/pwanews`. Changing directory to `pwanews` and running `npm start` runs react script starting the development server.

```
Compiled successfully!  
  
You can now view      in the browser.  
  
    http://localhost:   /  
    http://10.112.194.139:   /  
  
Note that the development build is not optimized.  
To create a production build, use yarn build.
```

Figure 18. Create-react-app compiled successfully

As shown in figure 18, after the successful compilation of create-react-app, it can be viewed in the browser by visiting `http://localhost:3000`.

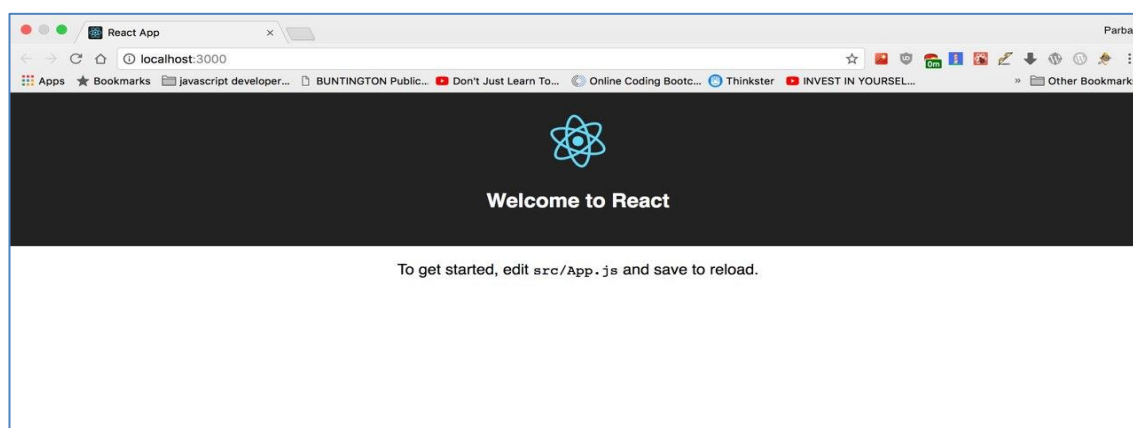


Figure 19. Default create-react-app project in local host

## 4.2 Web Pack

Webpack is a static module bundler for modern JavaScript applications. Modern JavaScript applications have various modules and stylesheets. These modules and stylesheets make the development process easy. However, during the time of deployment it creates hassle. When webpack is used, it processes the app and recursively builds a dependency graph, which includes every module needed by the application and packs all those modules into one or more bundles. Webpack improves the performance of the application. [19]

### 4.3 Babel

Babel is a JavaScript compiler. It transforms the new version of JavaScript code to the older version. In conjunction with Webpack, Babel transpiles ES6 and JSX to ES5. It is done to support the maximum number of users possible.

### 4.4 Other Online Services

#### RSS to JSON

It is an online service to convert an RSS feed into a REST API which then can be consumed by any applications which support fetching content from such API. The News app developed in the project receives its data from this service. The use of this service has allowed the app to run only as a Frontend web app.

#### Cloudinary

Cloudinary is the media management platform for web and mobile developers. Cloudinary is used as an image and video optimizing platform. Cloudinary played an important role in increasing the performance of PWA by optimizing the image size. One Signal and Zapier are used for implementing push notification.

## 5 Implementation of PWA News App

This chapter describes the implementation of the PWA news app prototype developed in this study to demonstrate the features and benefits of PWA. The application thus developed has all the components of PWA: Service Worker, Web App Manifest, App shell and Web push notification. Through the PWA news app, the user can browse news fetched by using BBC RSS newsfeed of World, Sports, Technology and Business, and get notified for the latest news via push notification, install app on the home screen with just one tap and use offline with UI and UX like a native app.

### 5.1 Development Environment

The application was developed in MacBookPro Early 2015 with the operating system macOS Sierra version 10.12.3 as the hardware. Node.js and create react app were downloaded and put together to build an environment for the PWA news app. Besides these, the following tools were also used in the project:

#### Visual Studio Code

Visual studio code is a free, lightweight and robust source code editor, which leverage the development of application through its features like embedded CLI and built-in git support. It is available for all the platforms: macOS. Windows and Linux. It has intelligent code completion, streamlined debugging and In-Product Source Control. It also comes with built-in support for Typescript, JavaScript and Node.js and extension for C#, C++, Java, Python, PHP and Go. Moreover, it also has runtime such as Unity and .NET.

NPM is used for the package manager for node modules. It is a fast, reliable and secure package manager. Likewise, Firebase was used for hosting the project. Firebase provided HTTPS connection while Git was used for the version control.

## 5.2 Implementation

The implementation phase of the project started by installing Node.js followed by create-react-app. Babel and web pack were preconfigured in create react app. Similarly, material UI components were imported.

The application flow of this project is quite simple: fetching the RSS news feed from the BBC news site and displaying the result. Figure 21 depicts the application structure of the PWA news project.

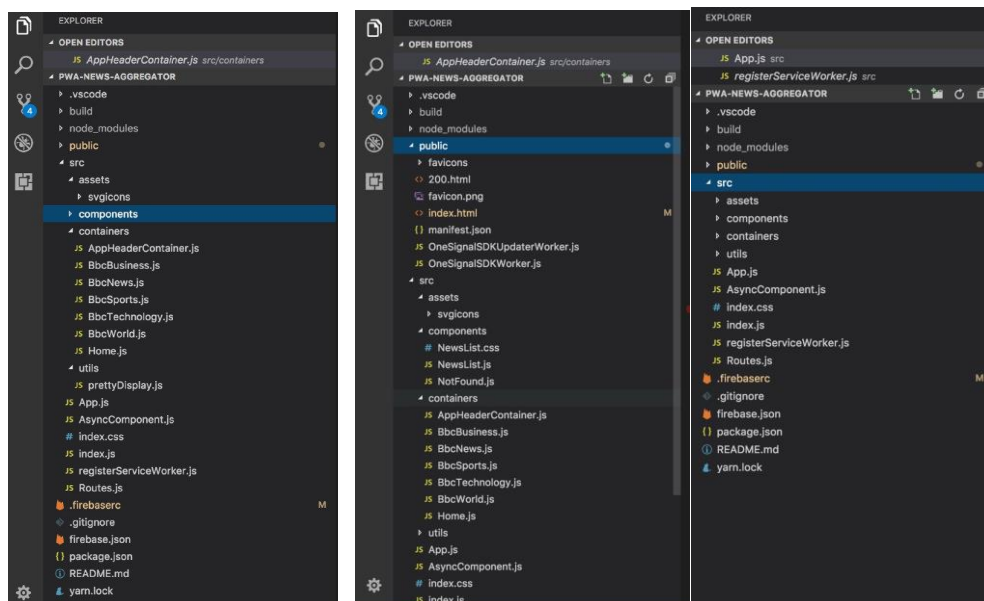


Figure 21. Application structure of the PWA news project

As shown in figure 21, the application folder structure contains all the necessary files for the development of the app. All files have been included in the parent directory PWA-NEWS-AGGREGATOR.

### Public

The public directory contains all the resources that are served directly without being additionally processed by webpack. It consists of index.html, manifest.json, favicons and files for the OneSignal. Index.html acts as the entry point for the web app. Manifest.json file contains all the configuration regarding how the PWA behaves when added to the home screen.



## Src

The src directory is the focal point of the create react app. It contains all the Javascript files which are later on processed by webpack. It consists of assets, components, containers, and utils. Utils consists of App.js, AsyncComponents.js, index.css, index.js, registerServiceWorker.js and Routes.js. App.js is the main JavaScript component whereas index.js just simply insert application into document. RegisterServiceWorker.js is associated with caching and updating files for the end-user and is responsible for the offline-caching and faster page load after the user's first visit to the application.

## Package.json

The 'package.json' file gives all the information regarding the app and its dependencies, which are used to build and run the app.

```
{
  "name": "news-summary-app-pwa",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "cloudinary-react": "^1.0.4",
    "material-ui": "^1.0.0-beta.23",
    "material-ui-icons": "^1.0.0-beta.17",
    "preact-material-components": "^1.3.3",
    "react": "^16.2.0",
    "react-dom": "^16.2.0",
    "react-router-dom": "^4.2.2",
    "react-scripts": "1.0.17",
    "rmwc": "^0.0.1-rc13",
    "serve": "^6.4.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "now-start": "serve --single ./build",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  }
}
```

Listing 8. package.json file of PWA news app

Figure 8 above is the 'package.json' file, which contains the detailed information of name, version, scripts and all the dependencies which was used in the project.

#### Node\_modules

Node\_modules contains all the packages but for this project, react and other dependencies specified in the package.json file are used.

#### .gitignore

This is the standard file, which is used by the version control git to resolve which files and directories to ignore.

## 6 Results and Evaluation

After the successful completion of the configuration, setup, coding, testing and debugging, the project was carried out successfully. A PWA news app was developed as the result.

The prototype news app thus developed followed all the guidelines and passed the test carried out with the Lighthouse tool. A lighthouse is an open source, automated tool used for auditing the web pages in terms of Accessibility, Progressive Web App, Performance and SEO. It can run in various ways as an extension for Chrome, either from the command line or as a Node module. Lighthouse runs a series of audits against the page of the given URL and generates a report on who the page performed. With the result, it also presents the failing audits as indicator on how the page score can be improved. Each audit has a reference document which explains the need for audit and ways to fix it.

PWA was hosted in firebase: <http://pwa-newsapp-8c76b.firebaseio.com> PWA news app URL was audited by the Lighthouse. Figure 22 illustrates the Lighthouse evaluation report for PWA news App.

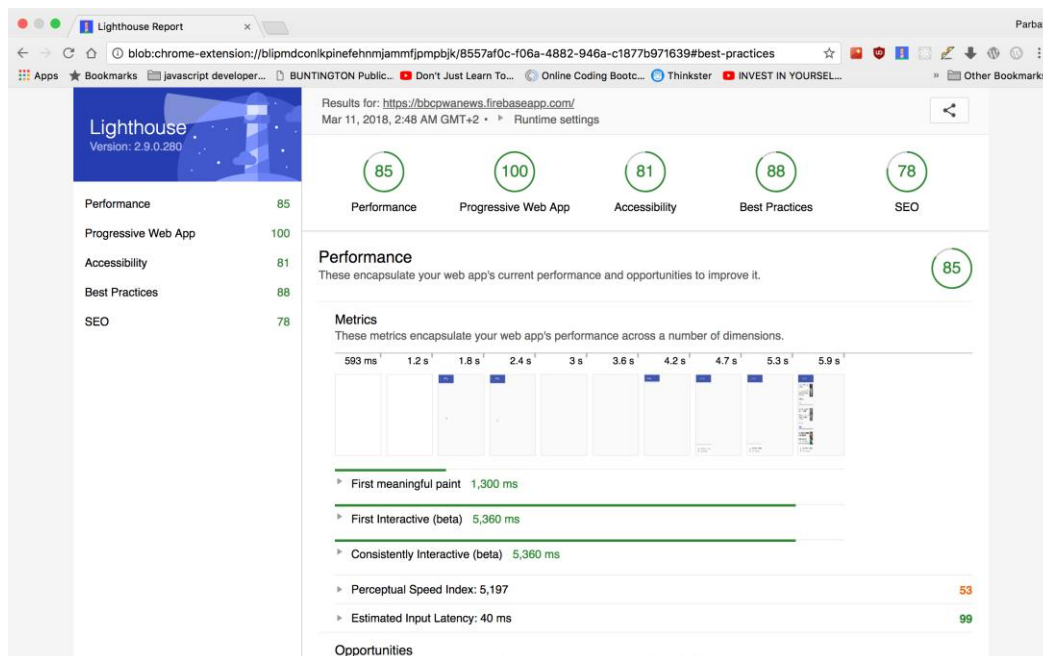


Figure 22. Lighthouse evaluation report for PWA news App

As shown in figure 22, the Lighthouse evaluation showed that the app is 100 percent progressive fulfilling all the requirement to be a Progressive Web app, with 85 percent score in performance, 81 in accessibility, 88 in best practices and 78 in SEO. The prototype was tested in a real device. The PWA news application in mobile and desktop view is shown in Figure 23.

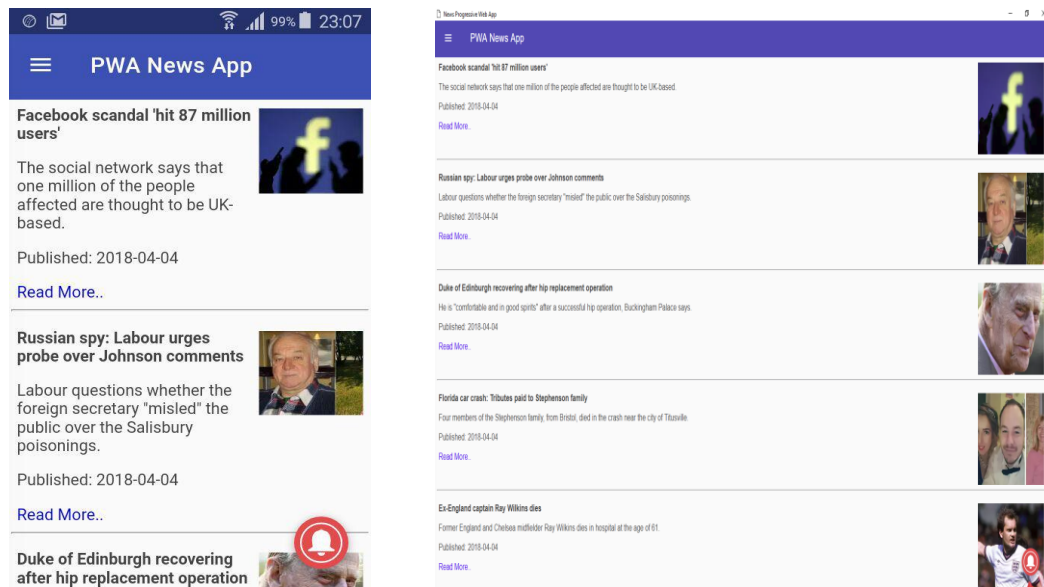


Figure 23. PWA news application in mobile and desktop view

The PWA news app thus developed have the reachability of the web and capabilities like a native application. PWA News App can work offline, install on home screen, launch in full screen mode without an omnibar, re-engage user by sending relevant push notification and load fast even on flaky networks.

## 7 Conclusion

The main goal of the thesis was to evaluate and implement a Progressive Web Application. A significant amount of time was invested in obtaining insight to the topic and its components. The pros and cons were analysed by comparing PWA to other applications.

After a detailed study of relevant theory and web app architecture, the focus was shifted towards the practical implementation. Thus, a prototype 'PWA News App' was developed which illustrated the implementation of Progressive Web App and its components. The prototype, thus developed feels and behaves like a native app. To sum up, the project was successfully carried out and the goals, which were set at the beginning of the project, were achieved.

PWA is not there to replace native apps. It is just a new approach of making web applications more like native. It is gaining popularity rapidly thanks to its features, easy development process, create one and deploy anywhere strategy. However, PWA is facing problems because of the issue in browser support for different functionalities. Firefox and Safari have some issues with App Manifest and Service Worker. Web App Manifest is in development status in Firefox and Safari has recently started support for Service worker in its recent version 11.1. However, Chrome, Chrome for Android, UC Browser for Android and Samsung Internet seem to have good support for all the functionalities.

Based on the research and practical implementation, it can be concluded that PWA may look like a buzz word around improved web standard but the positive impact it has brought to both businesses and users is undeniable. It has brought features and improvements for the mobile web which have so far been the exclusive domain of native apps. Thus, Companies like Flipkart, Housing.com, AliExpress, Financial Times, Google I/O, Twitter Lite, Forbes, and BookMyShow have implemented PWA and have experienced growth in their business. Additionally, Google is providing incentives and making great effort in providing tutorials and support for PWA. Thus, with more browser and platform support, PWA might be the future of mobile web.

## 8 References

- [1] Gabor Cselle, "Tales of Creation," 23 10 2012. [Online]. Available: <http://blog.gaborcselle.com/2012/10/every-step-costs-you-20-of-users.html>. [Accessed 02 03 2018].
- [2] Markov Danny, "Everything You Should Know About Progressive Web Apps," tutorialzine, 27 09 2016. [Online]. Available: <http://tutorialzine.com/2016/09/everything-you-should-know-about-progressive-web-apps/>. [Accessed 06 04 2017].
- [3] Farrugia Kevin, "A Beginner's Guide To Progressive Web Apps," 11 08 2016. [Online]. Available: <https://www.smashingmagazine.com/2016/08/a-beginners-guide-to-progressive-web-apps/>. [Accessed 04 04 2017].
- [4] Reshetilo Kateryna and Opanasenko Sergey, "Technology Org," 28 07 2017. [Online]. Available: <https://www.technology.org/2017/07/28/progressive-web-apps-vs-native-which-is-better-for-your-business/>. [Accessed 20 03 2018].
- [5] Osmani Addy, "The App Shell Model," Google, [Online]. Available: <https://developers.google.com/web/fundamentals/architecture/app-shell>. [Accessed 06 04 2017].
- [6] Ater Tal, Building Progressive Web Apps, O'Reilly.
- [7] Burtoft Jeff, "ThisHereWeb," 16 02 2015. [Online]. Available: <https://thishereweb.com/understanding-the-manifest-for-web-app-3f6cd2b853d6>. [Accessed 02 02 2018].
- [8] Gaunt Matt, "The Web App Manifest," [Online]. Available: <https://developers.google.com/web/fundamentals/web-app-manifest/>. [Accessed 02 03 2018].
- [9] Deveria Alexis, "Can I Use ?," [Online]. [Accessed 07 04 2017].
- [10] Gaunt Matt, "Service Workers: an Introduction," [Online]. Available: <https://developers.google.com/web/fundamentals/getting-started/primers/service-workers>. [Accessed 07 04 2017].
- [11] Google Developer, "Build an Offline Weather Web App," [Online]. Available: <https://codelabs.developers.google.com/codelabs/polymer-offline-weather/index.html?index=..%2F..%2Findex#5>. [Accessed 02 03 2018].
- [12] Teplý Jan Bc., "Hybrid mobile application for project," Czech Technical Univeristy, Prague, 2016.

- [13] Osmani Addy, "Getting started with Progressive Web Apps," 23 12 2015. [Online]. Available: <https://addyosmani.com/blog/getting-started-with-progressive-web-apps/>. [Accessed 07 04 2017].
- [14] LePage Pete, "Your First Progressive Web App," Google, [Online]. Available: <https://developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp/>. [Accessed 06 04 2017].
- [15] Deveria Alexis, "Canluse," [Online]. Available: <https://caniuse.com/#search=push%20notification>. [Accessed 03 03 2018].
- [16] Facebook Inc., "React, A JavaScript library for building user interface," Facebook, [Online]. Available: <https://reactjs.org/>. [Accessed 27 02 2018].
- [17] Stackoverflow, "Developer Survey Results 2017," [Online]. Available: <https://insights.stackoverflow.com/survey/2017#most-loved-dreaded-and-wanted>. [Accessed 28 02 2018].
- [18] Facebook Inc., "Introducing JSX," [Online]. Available: <https://reactjs.org/docs/introducing-jsx.html>. [Accessed 22 02 2018].
- [19] Webpack Org, "Webpack; bundle your scripts," [Online]. Available: <https://webpack.js.org/>. [Accessed 20 02 2018].
- [20] Mozilla Developer Network, "Web App Manifest," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/Manifest>. [Accessed 08 04 2017].
- [21] Burtoft Jeff, "Understanding the Manifest for Web App," 16 02 2015. [Online]. Available: <https://thishereweb.com/understanding-the-manifest-for-web-app-3f6cd2b853d6>. [Accessed 14 04 2017].
- [22] Google Progressive Web App Developers , "Case Studies," [Online]. Available: <https://developers.google.com/web/showcase/>. [Accessed 10 02 2018].

