PWA:

PWA (Progressive web apps) is a web application(not only mobile application) like other apps, but it gives you more mobile friendly experience, above all, it gives you the authority to install your apps on desktop and used them offline and get rid of google and apple playstore. The major difference b/w web and mobile application is that mobile application gives push notification and web application does not have. The major role is of serviceWorker. PWA is just the configuration

Website, PWA, native android apps

Website -> only view in the browser. Cannot view offline. Cannot get push notification from server

PWA -> same as website, but in mobile, we can create app, on clicking it icon, app open. Closed to mobile app in case of starting. We can view offline. Can get push notification from server. Its like converting website or web app into mobile app

Native apps -> do offline. Can get push notification from server

We see that mobile applications gives us the push notifications, but web applications donot. With PWA, web applications will also give push notifications

Implementation(Requirements):

1.seviceWorker(in src)

2.manifest.json(in public)

Manifest.json gives meta data, with this, we can handle our app behavior on different screen sizes.

serviceWorker -> register our application, and cash all the data that we want in offline mode. It is the feature of browser,

Code explanantion:

If service worker exist in browser(navigator detects all the browser features), register our serviceWorker.js file, and in serviceWorker.js, we provide push notification, offline and many other supports.

For live server experience, install live server extension in vs IDE, click on bottom port and go live

Working:

1. Create sw.js file in public folder, their we can provide all the stuff like for push notification, offline support etc.
2. Then we create swDev.js file in src folder which checks, If service worker exist in browser(navigator detects all the browser features), register our sw.js file.

Rough: -> register function give us the promise, so to handle it, we use then to get the result and catch to catch the error.

1. Now to check offline support, go onto inspect, then application, click on serviceWorker and click offline (and refresh), if we get the data, then pwa is their, if not, then there might be some mistake

Task how to get data when we offline:

The could which is responsible for that is cacheData part and this.addEventListener part.

In cacheData, we provide the name of the website.

1. In data.addAll array, we provide all the data that we needed when we get offline. For this open our application In browser, inspect, Networks and in Networks section, we got all the files that are responsible for the data. From their we click on those files whose data we want in offline mode, so we click on those files, under Response URL, copy its end part, like part which is starting from /static. We also need to put index. Files, / and api from which are getting data. So we can access that data offline. First visit all page online, so we can get them in cache, then click on offline and check whether they are working offline or not.

Rough: if facing problem, then in application part, unregister serviceWorker, in the same page under Cache,click on cache storage, delete app cache or website cache(not necessary), and refresh page again. And then make serviceWorker offline. If issue, we can also test it in incognito window.

Class # 18:

The major difference b/w web and mobile application is that mobile application gives push notification and web application does not have. The major role is of serviceWorker.

PWA and Push Notification in React from scratch:

Firebase is used for push notification. It is by google.

Incase of serviceWorker, when online, we get data from network and when offline, get data from serviceWorker.

Working:

Create firebase account, click on get started, new project, give project title, deselect analytics. It will give the config code

It is like system for messaging

Install firebase:

npm i --save firebase

Create firebase.js file

importing firebase

import firebase from 'firebase'

From firebase website

const config = {

    apiKey: "AIzaSyApEWq78NtkL1Nat2EjQTDo8JNUzICpwHg",

    authDomain: "fir-messaging-8a598.firebaseapp.com",

    databaseURL: "https://fir-messaging-8a598.firebaseio.com",

    projectId: "fir-messaging-8a598",

    storageBucket: "fir-messaging-8a598.appspot.com",

    messagingSenderId: "977640798956",

    appId: "1:977640798956:web:b46e60a6846115301103c9"

  };

Initializing firebase:

firebase.initializeApp(config)

importing in App.js:

import firebase from "./firebase";

getting permission from firebase

  const messaging = firebase.messaging();

  messaging.requestPermission().then(()=>{

    return messaging.getToken()

  }).then((token) => {

      console.log('token', token);

    });

Its like surge, used for deployment

Make firebase-messaging-sw.js file in public folder:

importScripts('https://www.gstatic.com/firebasejs/8.0.1/firebase-app.js')

importScripts('https://www.gstatic.com/firebasejs/8.0.1/firebase-messaging.js')

firebase.initializeApp({

    apiKey: "AIzaSyApEWq78NtkL1Nat2EjQTDo8JNUzICpwHg",

    authDomain: "fir-messaging-8a598.firebaseapp.com",

    databaseURL: "https://fir-messaging-8a598.firebaseio.com",

    projectId: "fir-messaging-8a598",

    storageBucket: "fir-messaging-8a598.appspot.com",

    messagingSenderId: "977640798956",

    appId: "1:977640798956:web:b46e60a6846115301103c9"

  })

firebase.messaging();

first we initialize for messaging, second for serviceWorker

It give us the token, this token is app specific. This token is used to send notification or message

We can verify this by going on to application and serviceWorker

Then go onto cloud messaging

OR https://console.firebase.google.com/u/0/project/fir-messaging-8a598/notification

and click on send your first mesaage

give title, text and then click on send text message, and then pass token their, click on + and send. We will get a notification.

That message for testing.

We send token to each client.

For deployment:

Instead of surge, we are deploying it on firebase

We are using firebase deployment to get notifications on phone

First write yarn build

Then in app folder, create another folder with name public\_

Copy build folder in that public\_

Install firebase:

npm install -g firebase-tools

then,

firebase init

with arrow keys, move to hosting, press space bar to select and then enter

then select use in existing project, select messagin path, which was fir messaging, then change public directory to build, then y, overwrite no.

After that firebase deploy.

If issue in getting project listing then;

firebase –P project\_id init