

PWA EXPERIMENT NO:07

AIM: To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.

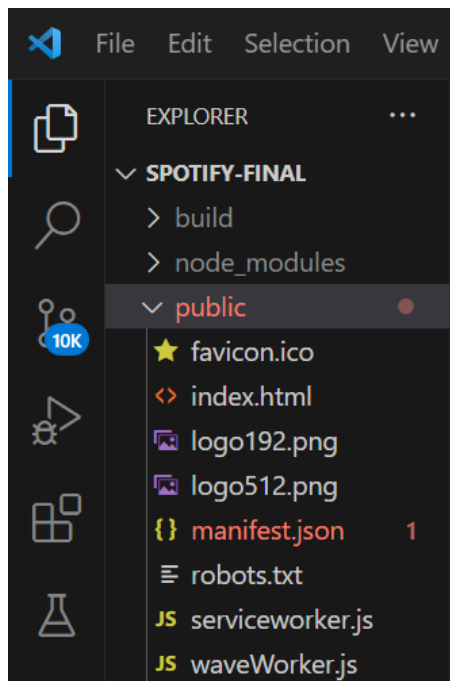
THEORY:

Regular Web App

A regular web app is a website that is designed to be accessible on all mobile devices such that the content gets fit as per the device screen. It is designed using a web technology stack (HTML, CSS, JavaScript, Ruby, etc.) and operates via a browser. They offer various native-device features and functionalities. However, it entirely depends on the browser the user is using. In other words, it might be possible that you can access a native-device feature on Chrome but not on Safari or Mozilla Firefox because the browsers are incompatible with that feature.

Progressive Web App

Progressive Web App (PWA) is a regular web app, but some extras enable it to deliver an excellent user experience. It is a perfect blend of desktop and mobile application experience to give both platforms to the end-users.



manifest.json:

```
{  
  "short_name": "React App",  
  "name": "Create React App Sample",  
  "icons": [  
    {  
      "src": "logo192.png",  
      "sizes": "192x192",  
      "type": "image/png"  
    },  
    {  
      "src": "logo512.png",  
      "sizes": "512x512",  
      "type": "image/png"  
    }  
  ]  
}
```

```

{
  "src": "favicon.ico",
  // "sizes": "64x64 32x32 24x24 16x16",
  "sizes": "64x64",
  "type": "image/x-icon"
},
{
  "src": "logo192.png",
  "type": "image/png",
  "sizes": "192x192"
},
{
  "src": "logo512.png",
  "type": "image/png",
  "sizes": "512x512"
}
],
"start_url": ".",
"display": "standalone",
"theme_color": "#000000",
"background_color": "#ffffff"
}

```

//Index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
      https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the 'public' folder during the build.

```

Only files inside the `public` folder can be referenced from the HTML.

Unlike `"/favicon.ico"` or `"favicon.ico"`, `"%PUBLIC_URL%/favicon.ico"` will work correctly both with client-side routing and a non-root public URL. Learn how to configure a non-root public URL by running ``npm run build``.

```
-->
<title>React App</title>
</head>
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.
```

You can add webfonts, meta tags, or analytics to this file.
The build step will place the bundled scripts into the `<body>` tag.

To begin the development, run ``npm start`` or ``yarn start``.
To create a production bundle, use ``npm run build`` or ``yarn build``.

```
-->
<script>
  window.addEventListener('load', () => {
    registerSW();
  });

  // Register the Service Worker
  async function registerSW() {
    if ('serviceWorker' in navigator) {
      try {
        await navigator
          .serviceWorker
          .register('serviceworker.js');
      }
      catch (e) {
        console.log('SW registration failed');
      }
    }
  }
</script>
</body>
</html>
```

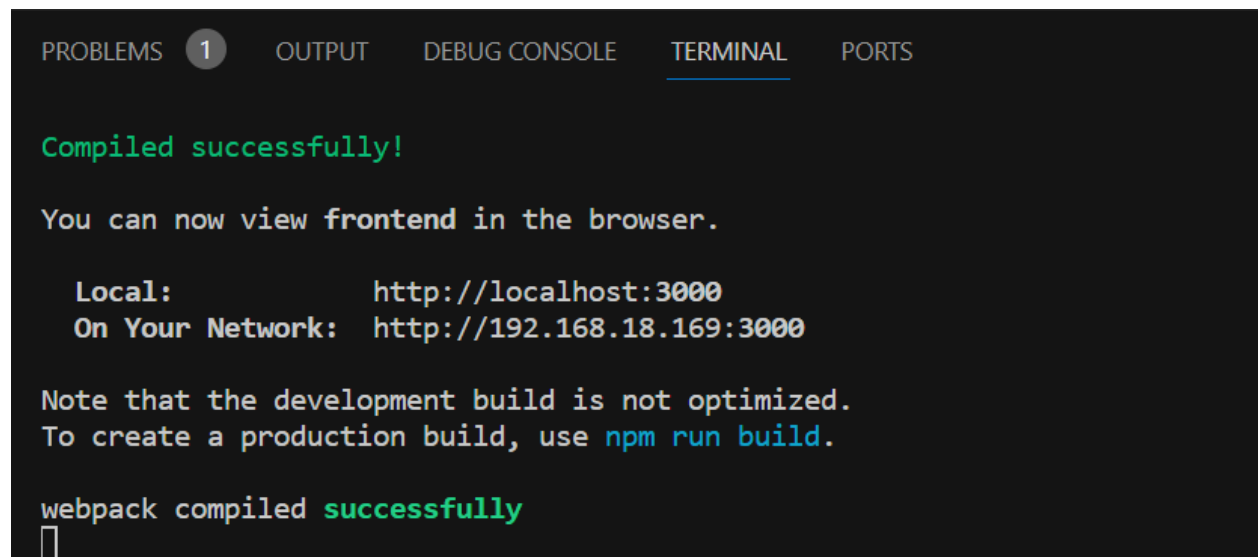
//serviceworker.js

```
var staticCacheName = "pwa";

self.addEventListener("install", function (e) {
  e.waitUntil(
```

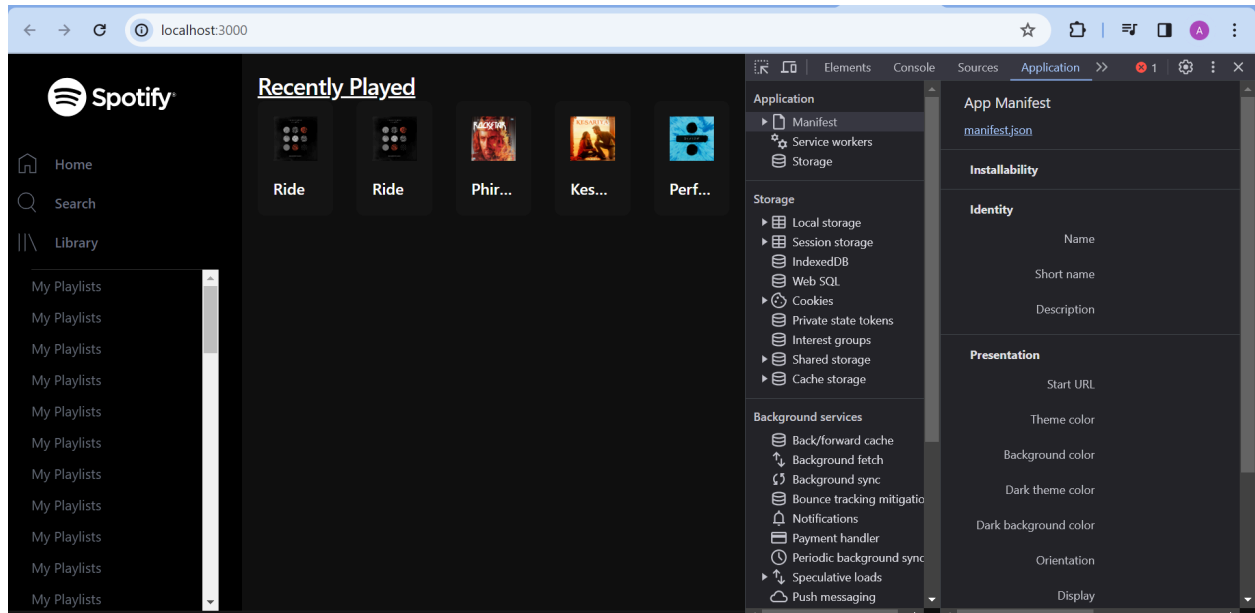
```
    caches.open(staticCacheName).then(function (cache) {  
      return cache.addAll([""]);  
    })  
  );  
});  
  
self.addEventListener("fetch", function (event) {  
  console.log(event.request.url);  
  
  event.respondWith(  
    caches.match(event.request).then(function (response) {  
      return response || fetch(event.request);  
    })  
  );  
});
```

>npm start

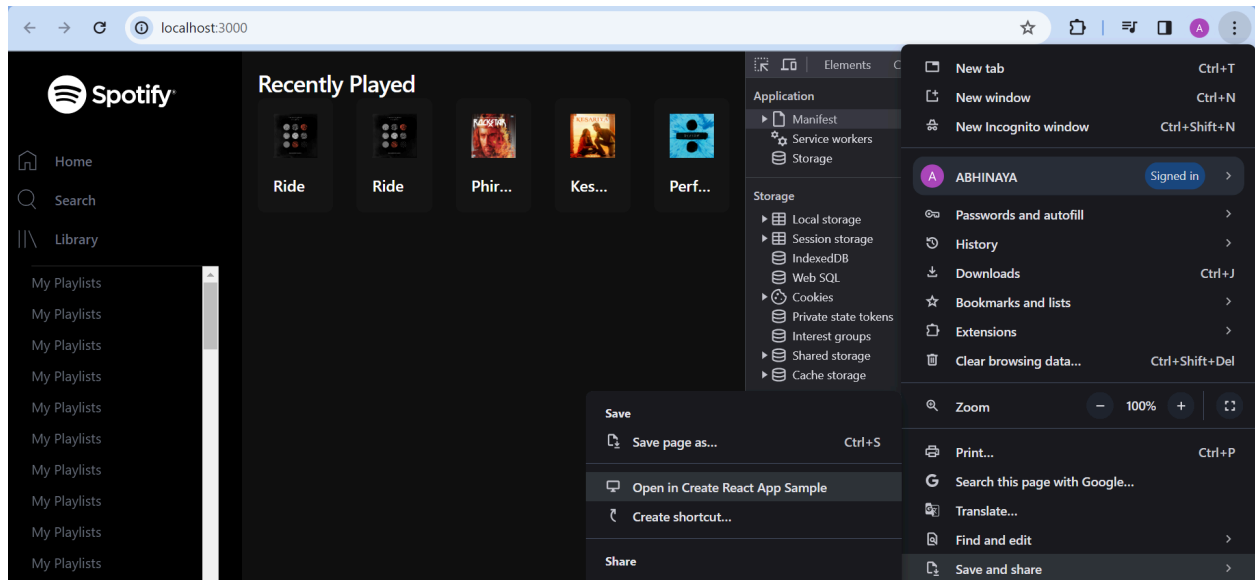
A screenshot of a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The 'PROBLEMS' tab has a small circle with the number '1' next to it. The terminal output shows a green message 'Compiled successfully!', followed by the text 'You can now view frontend in the browser.' Below this, it lists the local and network URLs: 'Local: http://localhost:3000' and 'On Your Network: http://192.168.18.169:3000'. A note follows: 'Note that the development build is not optimized. To create a production build, use npm run build.' At the bottom, it says 'webpack compiled successfully' in green, followed by a cursor icon.

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
Compiled successfully!  
  
You can now view frontend in the browser.  
  
Local: http://localhost:3000  
On Your Network: http://192.168.18.169:3000  
  
Note that the development build is not optimized.  
To create a production build, use npm run build.  
  
webpack compiled successfully  
█
```

//open Developer tools options



//click on 3 dots on top right corner of browser from app option install this site as an app



App icon will appear at the bottom

