

Experiment No: 7

Experiment No 7	
7: To write meta data of your Ecommerce PWA	
ROLL NO	03
NAME	Ansari Mohammed Sharjeel
CLASS	D15B
SUBJECT	MAD & PWA Lab
LO-MAPPE D	

Experiment 7

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.

Difference between PWAs vs. Regular Web Apps:

A Progressive Web is different and better than a Regular Web app with features like:

1. Native Experience

Though a PWA runs on web technologies (HTML, CSS, JavaScript) like a Regular web app, it gives user experience like a native mobile application. It can use most native device features, including push notifications, without relying on the browser or any other entity. It offers a seamless and integrated user experience that it is quite tough for one to differentiate between a PWA and a Native application by considering its look and feel.

2. Ease of Access

Unlike other mobile apps, PWAs do not demand longer download time and make memory space available for installing the applications. The PWAs can be shared and installed by a link, which cuts down the number of steps to install and use. These applications can easily keep an app icon on the user's home screen, making the app easily accessible to the users and helps the brands remain in the users' minds, and improving the chances of interaction.

3. Faster Services

PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the brands improve the user engagement and retention rate, which eventually adds value to their business.

4. Engaging Approach

As already shared, the PWAs can employ push notifications and other native device features more efficiently. Their interaction does not depend on the browser user uses. This eventually improves the chances of notifying the user regarding your services, offers, and other options related to your brand and keeping them hooked to your brand. In simpler words, PWAs let you maintain the user engagement and retention rate.

5. Updated Real-Time Data Access

Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.

In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.

6. Discoverable

PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

7. Lower Development Cost

Progressive web apps can be installed on the user device like a native device, but it does not demand submission on an App Store. This makes it far more cost-effective than native mobile applications while offering the same set of functionalities.

Steps involved:

1. Create a Manifest File:

In the root directory of your web application, create a file named manifest.json. Add the following content to your manifest.json file:

Code:

```
{
  "name": "E-Commerce App",
  "short_name": "Ecom",
  "description": "A Progressive Web App designed for shopping",
  "icons": [
    {
      "src": "/path/to/icon.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ],
  "start_url": "/",
```

```
"display": "standalone",  
"theme_color": "#ffffff",  
"background_color": "#ffffff"  
}
```

2. Link the Manifest File:

In the <head> section of your HTML file, add a link to the manifest.json file:

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

Replace /path/to/manifest.json with the actual path to your manifest.json file.

In this manifest file:

name: The name of your app.

short_name: A shorter name for your app.

description: A brief description of your app.

icons: An array of icon objects specifying the app's icons for different sizes and types.

start_url: The URL where your app should start when launched from the homescreen.

display: Specifies how your app should display (e.g., standalone, fullscreen, minimal-ui, browser).

theme_color: The color used for the app's theme (e.g., toolbar, splash screen).

background_color: The color of the splash screen background.

Conclusion:

Enabling the "add to homescreen" feature for your Ecommerce Progressive Web App (PWA) involves creating a manifest.json file with metadata that describes your app's name, icon, description, and other properties. This file allows users to add your PWA to their device's homescreen for quick and easy access.