NAME: HARSHITA DUBEY ROLL NO.:14
SUBJECT: MAD LAB CLASS:D15A/BATCH A

EXP 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

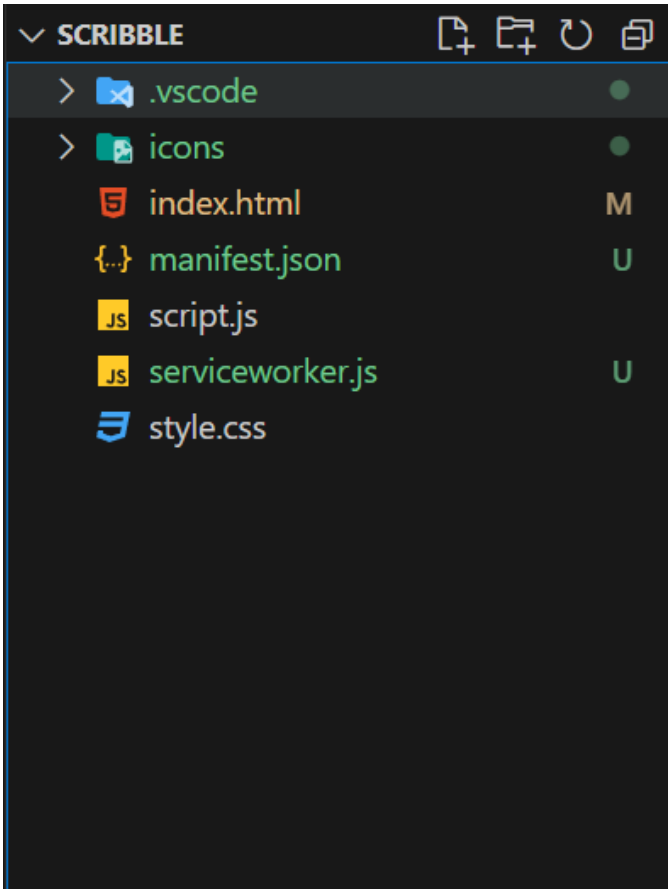
PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

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.

IMPLEMENTATION

STEP 1: Open any web application through any Text Editor eg:VSCODE.

STEP 2: Add few files to this. Folder structure of my application is:



CODE:

manifest.json

```
{
   "name":"Scribble",
   "short_name":"PWA",
   "start_url":"index.html",
   "display":"standalone",
   "background_color":"#5900b3",
   "theme_color":"black",
   "scope": ".",
   "description":"This is a PWA tutorial.",
   "icons":[
   {
   "src":"icons/ma.png",
   "sizes":"192x192",
   "type":"image/png"
   },
   {
   "src":"icons/twitter.png",
   "sizes":"512x512",
   "type":"image/png"
   }
 ]
}
```

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);
    })
  );
});
```

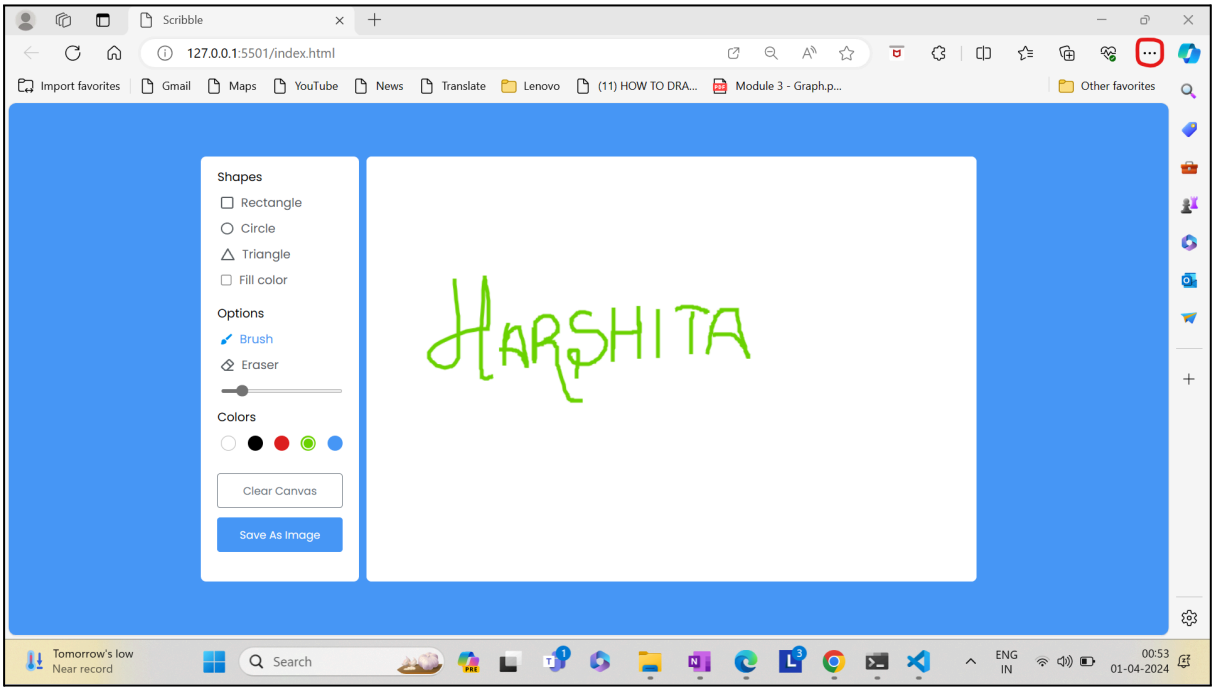
index.html

```
<!DOCTYPE html>

<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Scribble</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="script.js" defer></script>
    <!-- Manifest File link -->
<link rel="manifest"
href="manifest.json">
  </head>
  <body>
    <div class="container">
      <section class="tools-board">
        <div class="row">
          <label class="title">Shapes</label>
          <ul class="options">
            <li class="option tool" id="rectangle">
              <img src="icons/rectangle.svg" alt="">
              <span>Rectangle</span>
            </li>
            <li class="option tool" id="circle">
              <img src="icons/circle.svg" alt="">
              <span>Circle</span>
            </li>
            <li class="option tool" id="triangle">
              <img src="icons/triangle.svg" alt="">
              <span>Triangle</span>
            </li>
            <li class="option">
              <input type="checkbox" id="fill-color">
              <label for="fill-color">Fill color</label>
            </li>
```

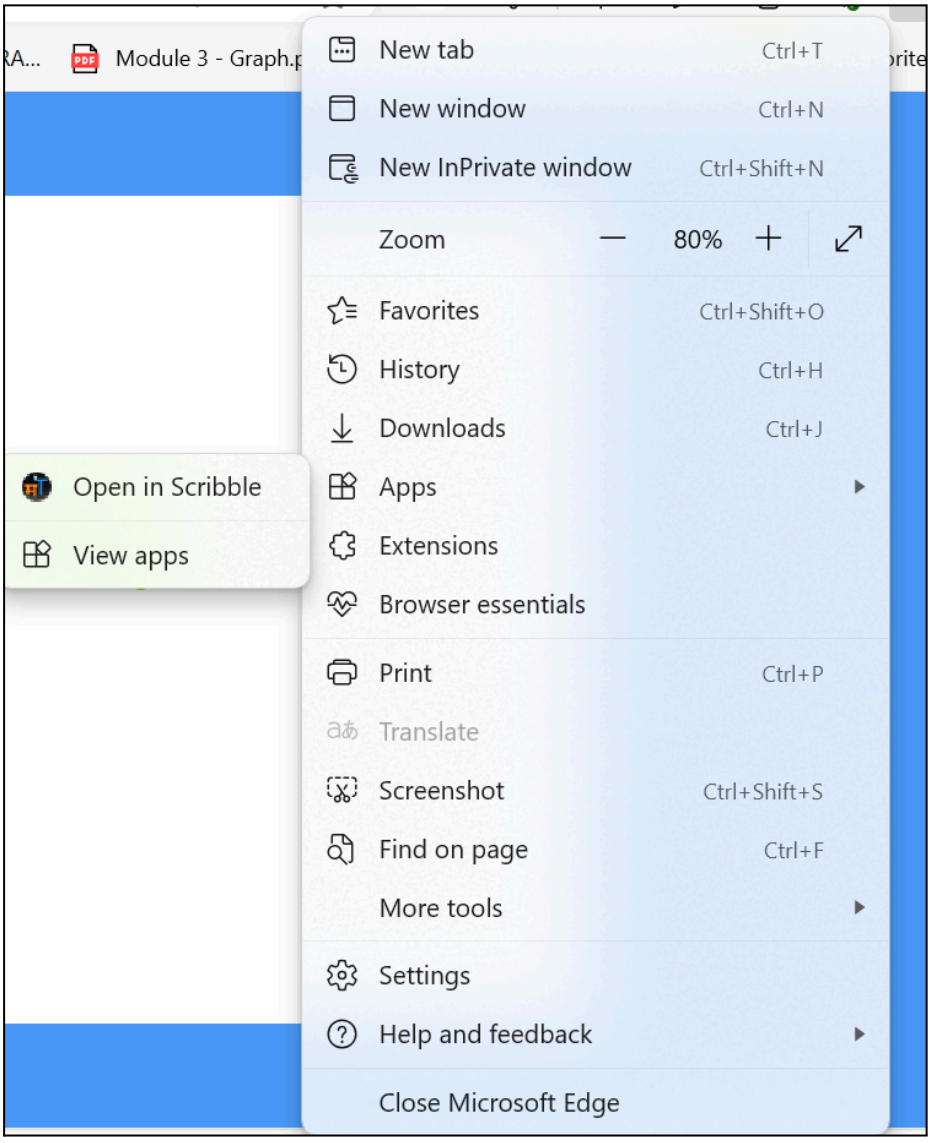
```
      </ul>
    </div>
    <div class="row">
      <label class="title">Options</label>
      <ul class="options">
        <li class="option active tool" id="brush">
          <img src="icons/brush.svg" alt="">
          <span>Brush</span>
        </li>
        <li class="option tool" id="eraser">
          <img src="icons/eraser.svg" alt="">
          <span>Eraser</span>
        </li>
        <li class="option">
          <input type="range" id="size-slider" min="1" max="30" value="5">
        </li>
      </ul>
    </div>
    <div class="row colors">
      <label class="title">Colors</label>
      <ul class="options">
        <li class="option"></li>
        <li class="option selected"></li>
        <li class="option"></li>
        <li class="option"></li>
        <li class="option">
          <input type="color" id="color-picker" value="#4A98F7">
        </li>
      </ul>
    </div>
    <div class="row buttons">
      <button class="clear-canvas">Clear Canvas</button>
      <button class="save-img">Save As Image</button>
    </div>
  </section>
  <section class="drawing-board">
    <canvas></canvas>
  </section>
</div>
<script>
  // Add event listener to execute code when page loads
  window.addEventListener('load', () => {
    // Call registerSW function when page loads
    registerSW();
```

```
    });

    // Register the Service Worker
    async function registerSW() {
      // Check if browser supports Service Worker
      if ('serviceWorker' in navigator) {
        try {
          // Register the Service Worker named 'serviceworker.js'
          await navigator.serviceworker.register('serviceworker.js');
        }
        catch (e) {
          // Log error message if registration fails
          console.log('SW registration failed');
        }
      }
    }
  </script>

  </body>
</html>
```

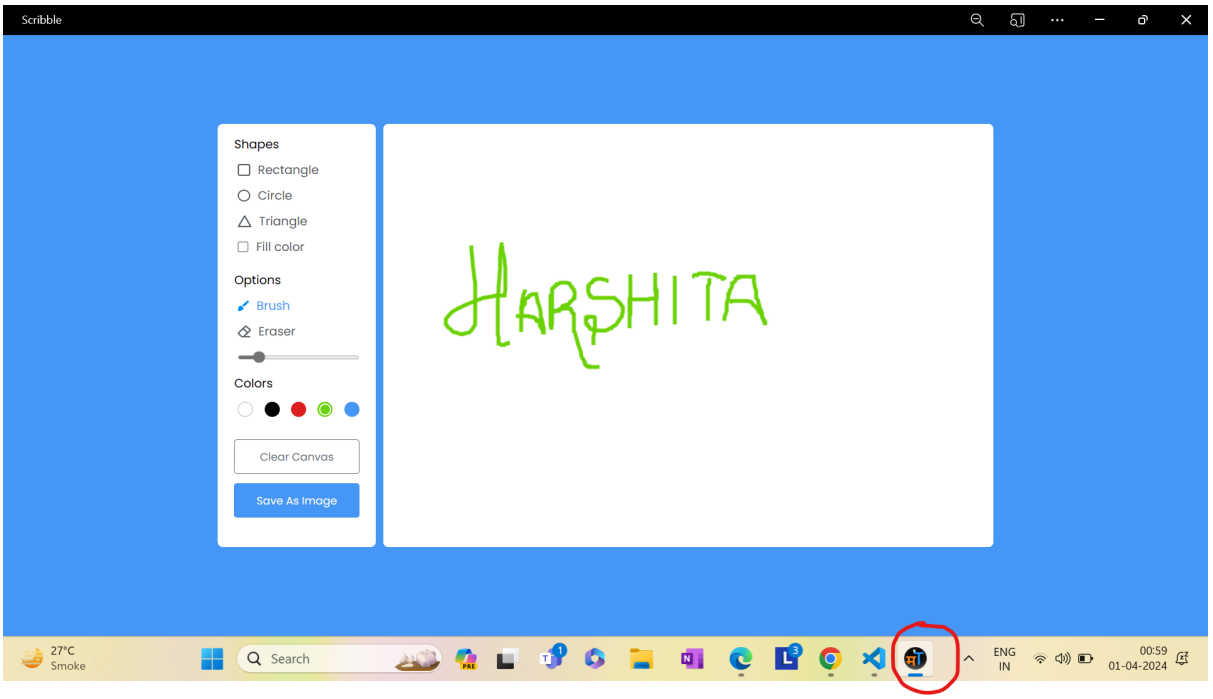
STEP 4 : Open Live Server and http://127.0.0.1:5501/index.html will be opened



STEP 5: Click on 3 dots on the top right corner of the browser → go to Apps → Click on Install Scribble.



App Icon will appear at the bottom:



CONCLUSION:

Hence, we learnt how to write a metadata of our E-commerce website PWA in a Web App Manifest File to enable add to homescreen feature.