Name: Aakanksha Suvarna Class: D15B Roll_no: 65

Experiment 7

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

Theory:

Progressive Web Apps (PWAs) are a type of web application built using web technologies such as HTML, CSS, and JavaScript, but they are designed to provide a user experience similar to native mobile apps. Here's the theory behind PWAs:

Progressive Enhancement:

PWAs are built using the concept of progressive enhancement, which means they are designed to work on any device or browser regardless of the user's network conditions. They leverage modern web capabilities to provide a seamless experience while gracefully degrading on older browsers or devices.

Responsive Design:

PWAs are designed to be responsive, meaning they can adapt and provide an optimal user experience across various devices, including desktops, smartphones, and tablets. This ensures that users can access the app regardless of the device they are using.

Service Workers:

One of the key features of PWAs is the use of service workers, which are JavaScript files that run in the background separate from the web page. Service workers enable features such as offline functionality, push notifications, and caching, allowing PWAs to work even when the user is offline or has a slow or unreliable network connection.

Difference between Progressive Web Apps (PWAs) and Regular Web Apps:

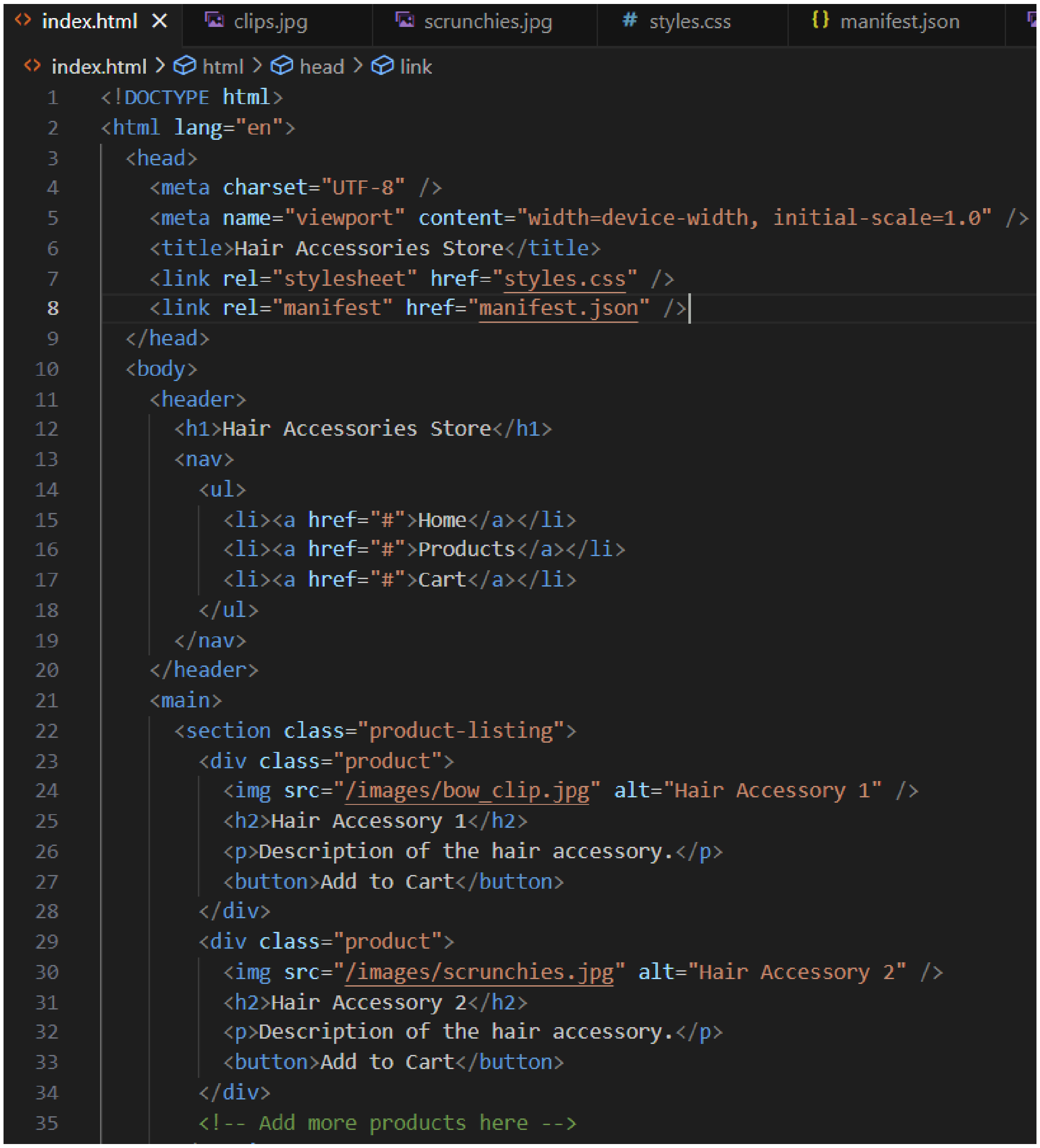
1. Offline Functionality: PWAs can work offline or with a limited network connection using service workers, while regular web apps typically require a constant internet connection to function.
2. Installation: PWAs can be installed on a user's device and accessed from the home screen, providing an app-like experience. Regular web apps are accessed through a web browser and do not have the option for installation.
3. Native-Like Features: PWAs can leverage native-like features such as push notifications, full-screen mode, and access to device hardware (e.g., camera, GPS) using web APIs, whereas regular web apps have limited access to such features.
4. Performance: PWAs are designed to be fast and responsive, providing a smooth user experience similar to native apps. Regular web apps may suffer from performance issues, especially on mobile devices with slower processors or limited memory.

Name: Aakanksha Suvarna Class: D15B Roll_no: 65

5. Discoverability: PWAs can be discovered and indexed by search engines, allowing users to find them through web searches. Regular web apps may have limited discoverability, especially if they are not optimized for search engines.

The below steps have to be followed to create a progressive web application:

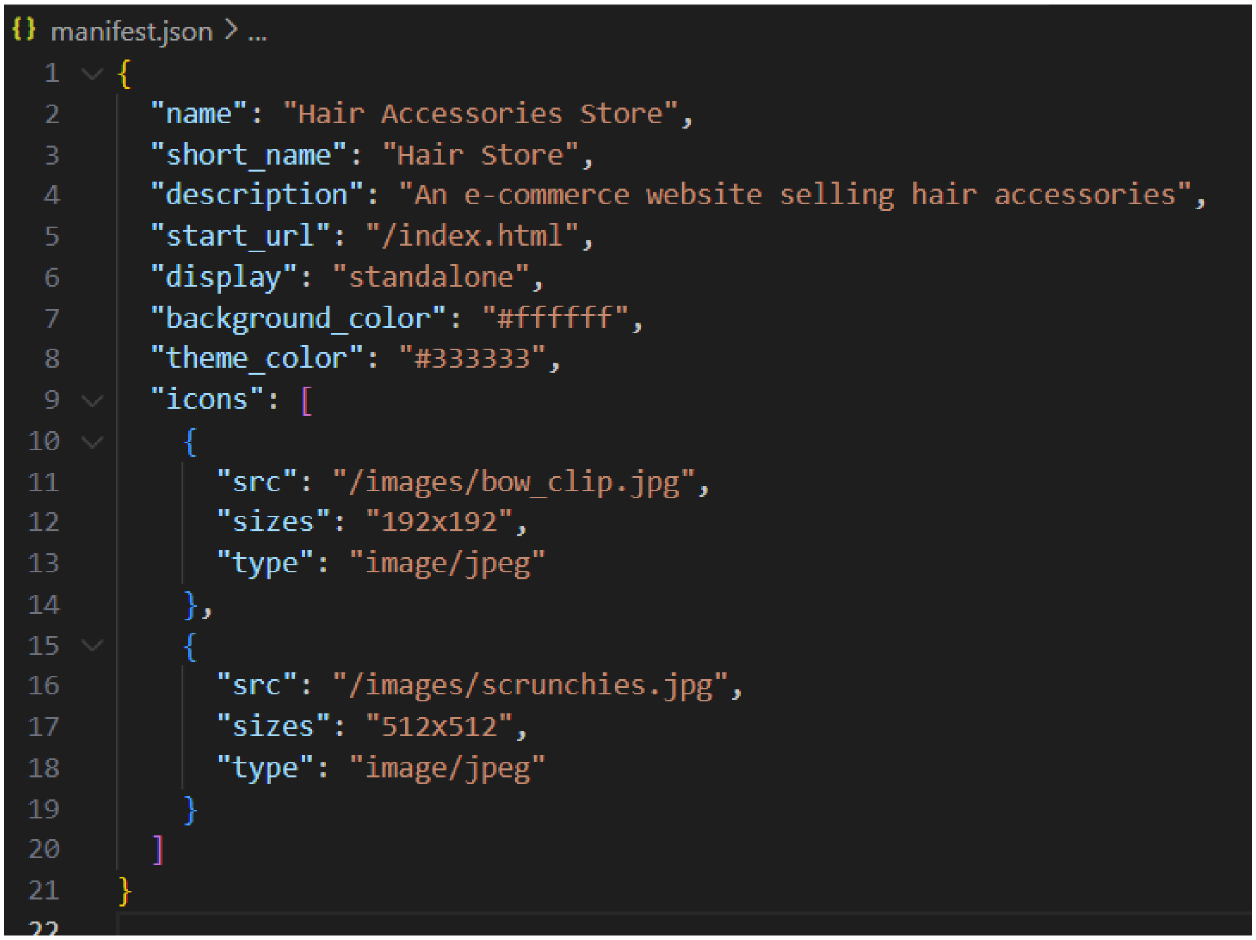
1. Step 1: Create an HTML page that would be the starting point of the application. This HTML will contain a link to the file named manifest.json. This is an important file that would be created in the next step.



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Hair Accessories Store</title>
    <link rel="stylesheet" href="styles.css" />
    <link rel="manifest" href="manifest.json" />
  </head>
  <body>
    <header>
      <h1>Hair Accessories Store</h1>
      <nav>
        <ul>
          <li><a href="#">Home</a></li>
          <li><a href="#">Products</a></li>
          <li><a href="#">Cart</a></li>
        </ul>
      </nav>
    </header>
    <main>
      <section class="product-listing">
        <div class="product">
          <img src="/images/bow_clip.jpg" alt="Hair Accessory 1" />
          <h2>Hair Accessory 1</h2>
          <p>Description of the hair accessory.</p>
          <button>Add to Cart</button>
        </div>
        <div class="product">
          <img src="/images/scrunchies.jpg" alt="Hair Accessory 2" />
          <h2>Hair Accessory 2</h2>
          <p>Description of the hair accessory.</p>
          <button>Add to Cart</button>
        </div>
        <!-- Add more products here -->
```

Name: Aakanksha Suvarna Class: D15B Roll_no: 65

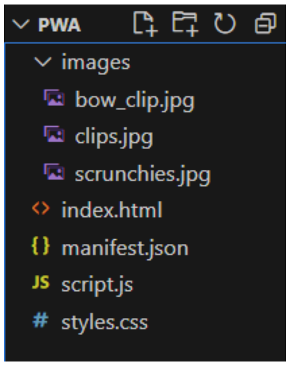
Step 2: Create a manifest.json file in the same directory. This file basically contains information about the web application. Some basic information includes the application name, starting URL, theme color, and icons. All the information required is specified in the JSON format. The source and size of the icons are also defined in this file.



manifest.json > ...

```json
{
  "name": "Hair Accessories Store",
  "short_name": "Hair Store",
  "description": "An e-commerce website selling hair accessories",
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#333333",
  "icons": [
    {
      "src": "/images/bow_clip.jpg",
      "sizes": "192x192",
      "type": "image/jpeg"
    },
    {
      "src": "/images/scrunchies.jpg",
      "sizes": "512x512",
      "type": "image/jpeg"
    }
  ]
}
```

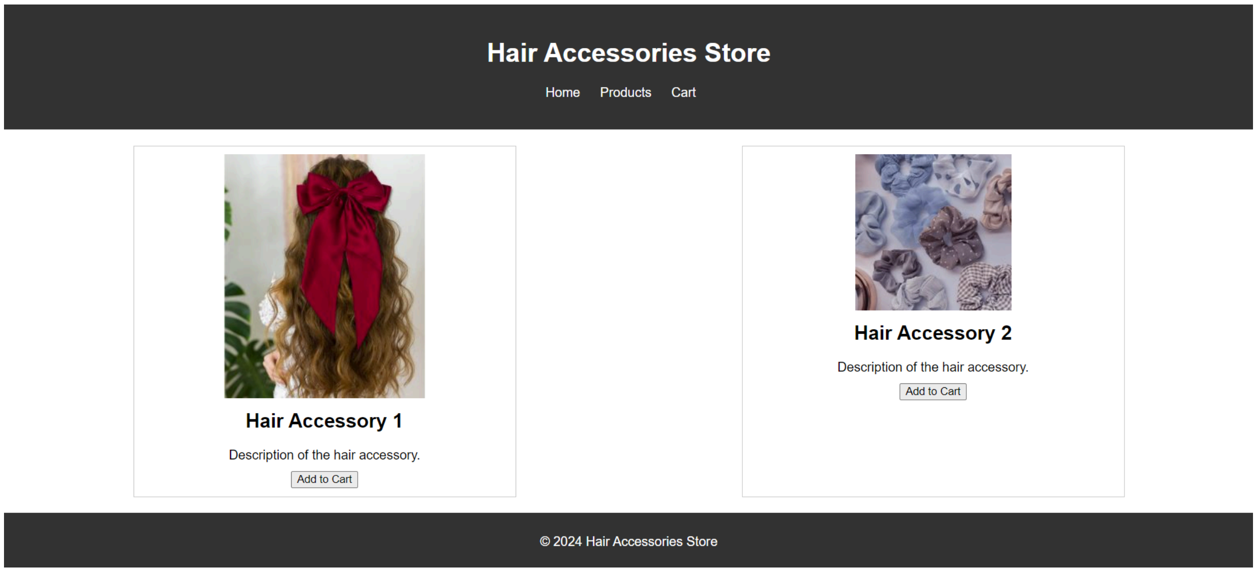
Step 3: Create a new folder named images and place all the icons related to the application in that folder. It is recommended to have the dimensions of the icons at least 192 by 192 pixels and 512 by 512 pixels. The image name and dimensions should match that of the manifest file.



```
PWA
  images
    bow_clip.jpg
    clips.jpg
    scrunchies.jpg
  index.html
  manifest.json
  script.js
  styles.css
```

Name: Aakanksha Suvarna Class: D15B Roll_no: 65

Step 4: Serve the directory using a live server so that all files are accessible



Hair Accessories Store

Home Products Cart

Hair Accessory 1

Description of the hair accessory.

Add to Cart

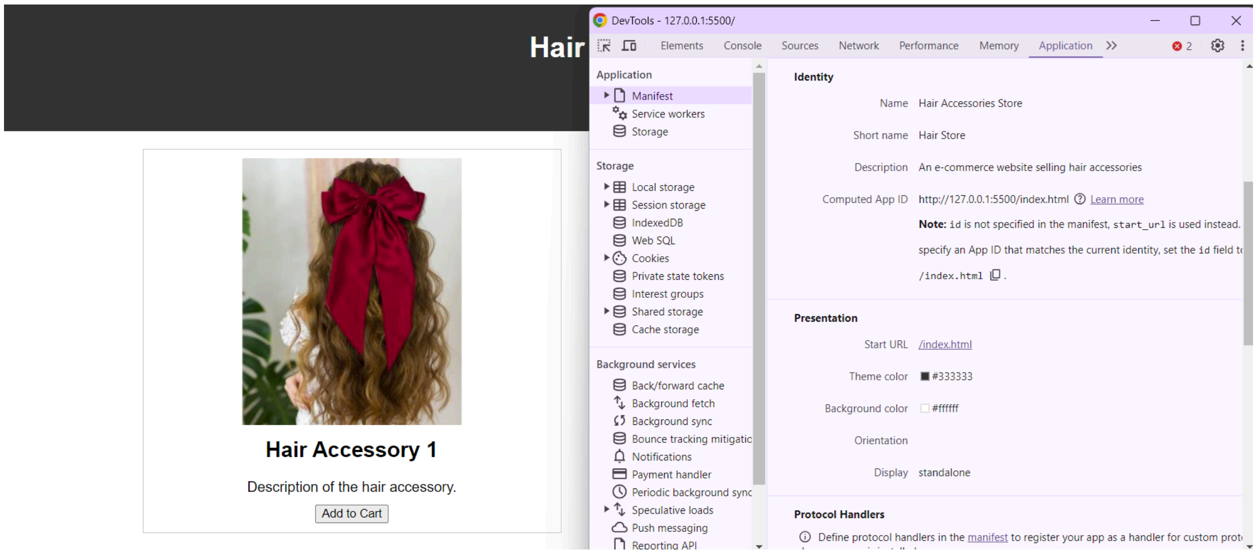
Hair Accessory 2

Description of the hair accessory.

Add to Cart

© 2024 Hair Accessories Store

Step 5: Open the index.html file in Chrome navigate to the Application Section in the Chrome Developer Tools. Open the manifest column from the list.



Identity

Name Hair Accessories Store

Short name Hair Store

Description An e-commerce website selling hair accessories

Computed App ID http://127.0.0.1:5500/index.html Learn more

Note: `id` is not specified in the manifest, `start_url` is used instead.

specify an App ID that matches the current identity, set the `id` field t

`/index.html` .

Presentation

Start URL /index.html

Theme color ■ #333333

Background color □ #ffffff

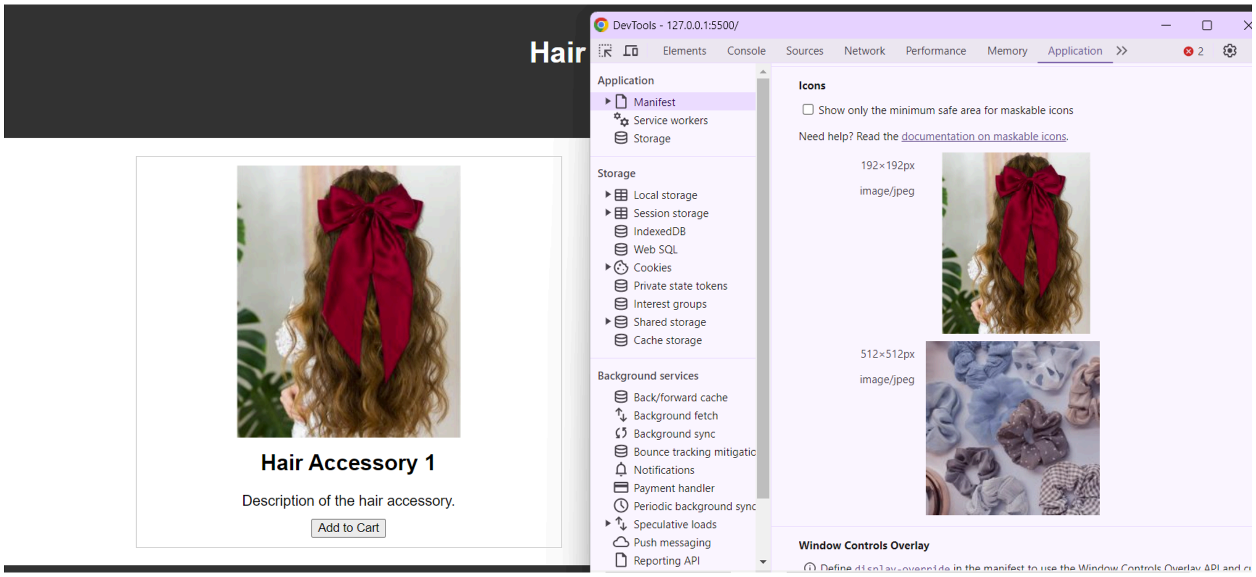
Orientation

Display standalone

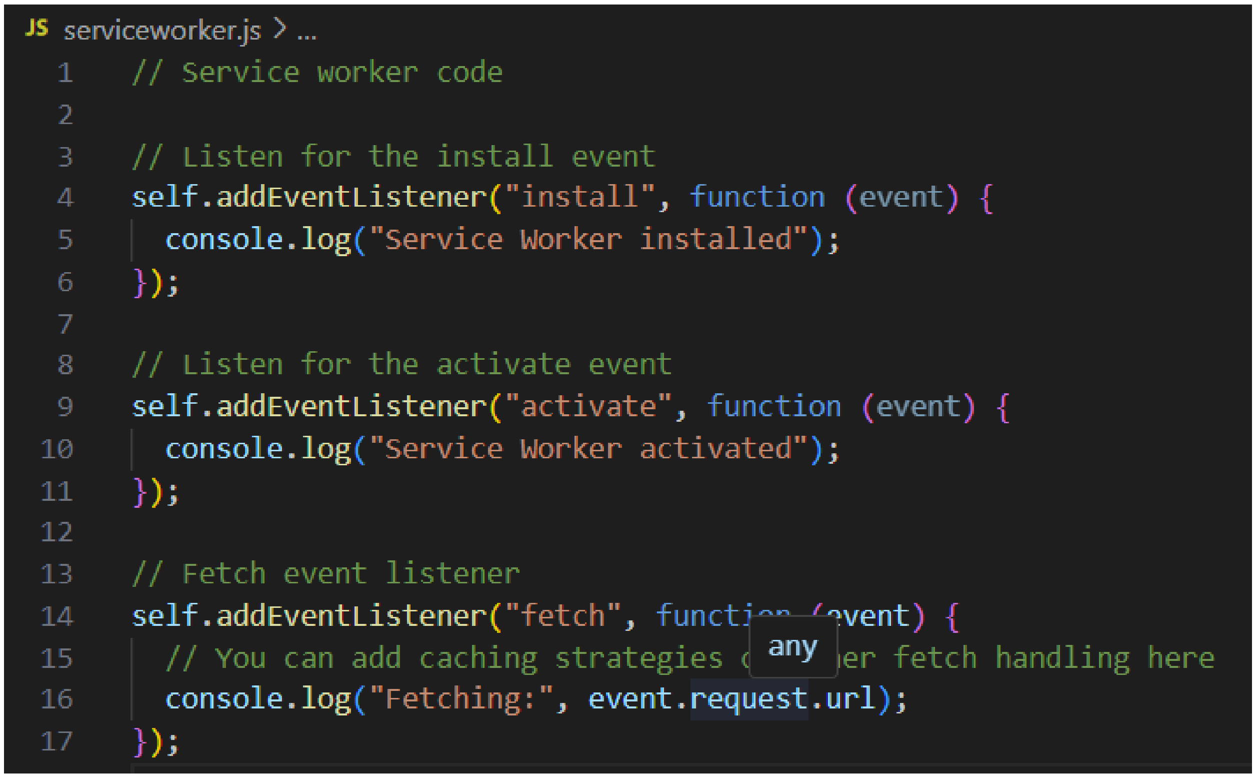
Protocol Handlers

Define protocol handlers in the manifest to register your app as a handler for custom prot

Name: Aakanksha Suvarna Class: D15B Roll_no: 65



Step 6: Under the installability tab, it would show that no service worker is detected. We will need to create another file for the PWA, that is, serviceworker.js in the same directory. This file handles the configuration of a service worker that will manage the working of the application.



serviceworker.js > ...

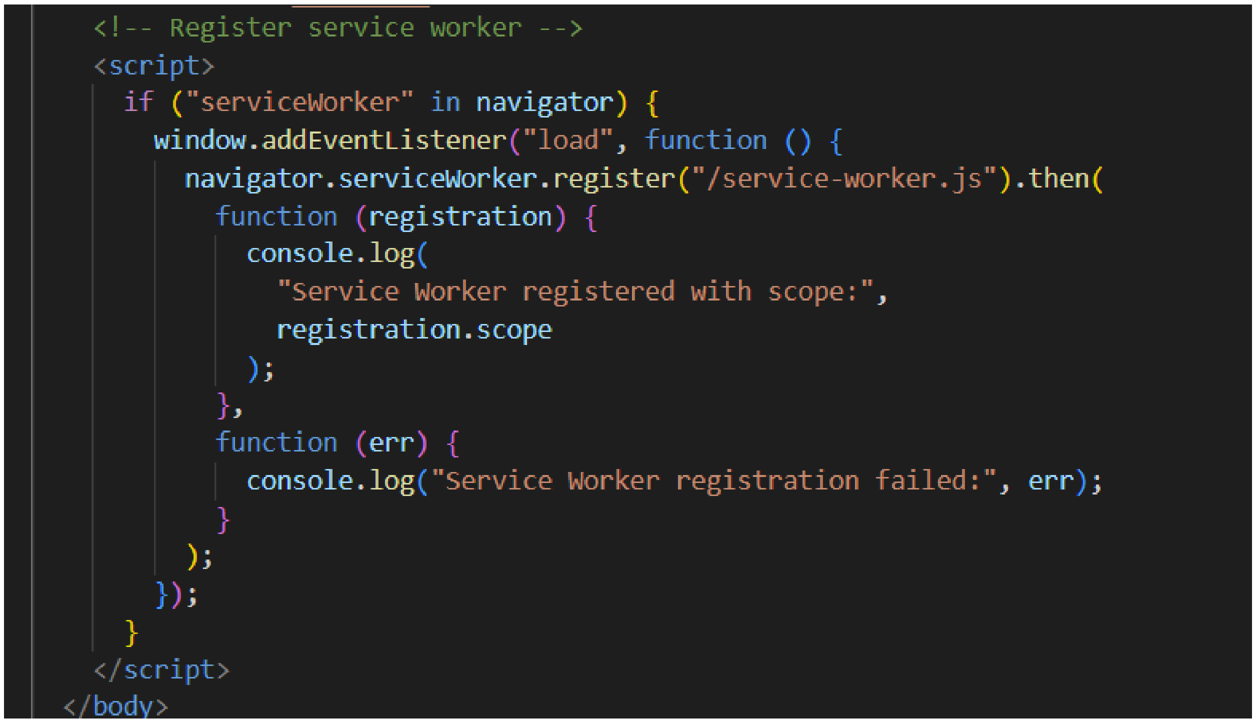
```javascript
// Service worker code

// Listen for the install event
self.addEventListener("install", function (event) {
  console.log("Service Worker installed");
});

// Listen for the activate event
self.addEventListener("activate", function (event) {
  console.log("Service Worker activated");
});

// Fetch event listener
self.addEventListener("fetch", function (event) {
  // You can add caching strategies or other fetch handling here
  console.log("Fetching:", event.request.url);
});
```

Step 7: The last step is to link the service worker file to index.html. This is done by adding a short JavaScript script to the index.html created in the above steps. Add the below code inside the script tag in index.html.



```
    <!-- Register service worker -->
    <script>
      if ("serviceWorker" in navigator) {
        window.addEventListener("load", function () {
          navigator.serviceWorker.register("/service-worker.js").then(
            function (registration) {
              console.log(
                "Service Worker registered with scope:",
                registration.scope
              );
            },
            function (err) {
              console.log("Service Worker registration failed:", err);
            }
          );
        });
      }
    </script>
  </body>
```

Installing the application: Navigating to the Service Worker tab, we see that the service worker is registered successfully and now an install option will be displayed that will allow us to install our app. Click on the install button to install the application. The application would then be installed, and it would be visible on the desktop. For installing the application on a mobile device, the Add to Home screen option in the mobile browser can be used. This will install the application on the device.

Conclusion: Thus writing metadata for the PWA, especially for an eCommerce application, is crucial for enabling features like the "add to homescreen" functionality. By crafting a well-structured manifest.json file with accurate metadata properties such as name, description, icons, and colors, developers can enhance the accessibility and user experience of their PW