Name: Aastha Jajoo **Class: D15B** **Roll No: 23**

Experiment No: 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):

Progressive Web Apps (PWAs) are web applications that leverage modern web capabilities to offer a user experience similar to native mobile apps. They are built using standard web technologies like HTML, CSS, and JavaScript but are designed to work on any platform with a standards-compliant browser, including desktop and mobile devices.

Regular web app

Regular web apps, on the other hand, are accessed and used through a web browser on various devices, such as desktops, laptops, tablets, and mobile phones. While they also use web technologies, they may lack some of the advanced features and user experience enhancements found in PWAs.

How Progressive Web Apps (PWAs) differ from regular web apps?

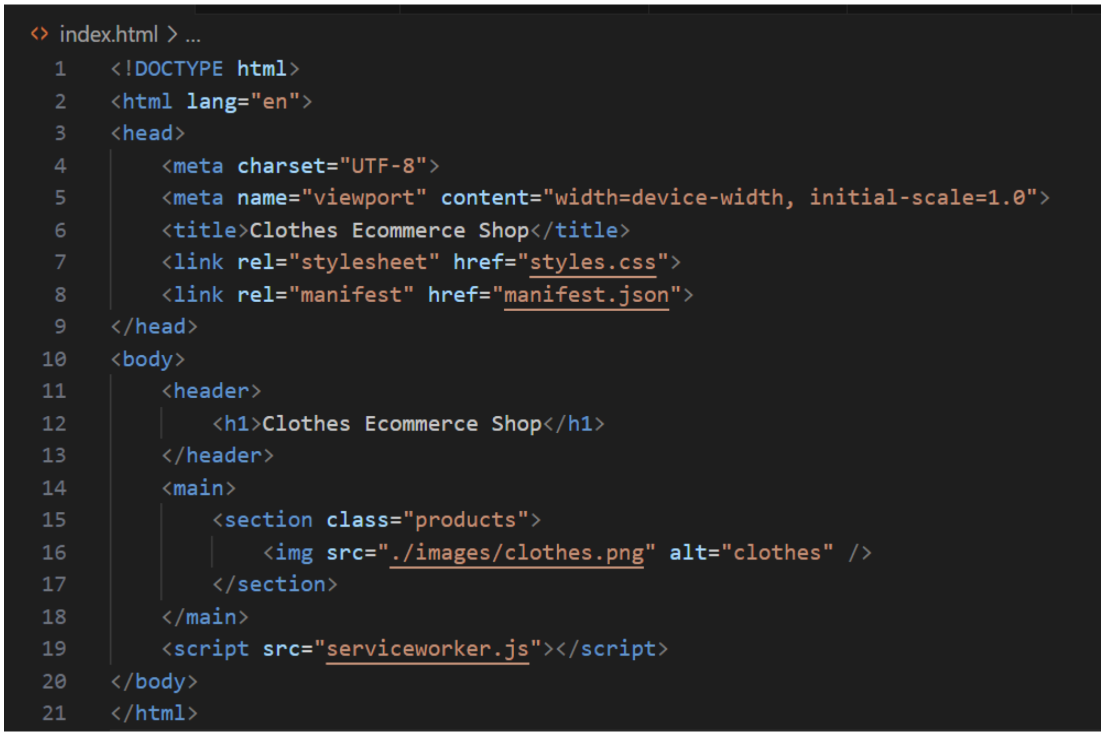
Progressive Web Apps (PWAs) differ from regular web apps in several key aspects, which contribute to providing a more app-like experience for users. Here's how PWAs differ from regular web apps:

1. **Offline Functionality:** PWAs can work offline or with a poor internet connection, thanks to service workers. This enables features like caching of assets and data, allowing PWAs to continue functioning even when the user is offline. Regular web apps typically require an active internet connection to function properly.
2. **App-like User Interface:** PWAs can be installed on the device's homescreen and launched in full-screen mode, giving them a native app feel. They can also utilize features like push notifications, providing a more immersive and engaging user experience compared to regular web apps.
3. **Improved Performance:** PWAs are optimized for performance, offering faster loading times and smoother interactions. Techniques like lazy loading, code splitting, and caching help minimize loading times and reduce data transfer, resulting in a more responsive user experience.
4. **Discoverability and Shareability:** PWAs are discoverable through web search engines and can be shared via URL links. This makes it easy for users to discover and share PWAs with others. Regular web apps may have limited discoverability and shareability compared to PWAs.
5. **Cross-platform Compatibility:** PWAs are designed to work across different platforms and devices, ensuring a consistent user experience. They utilize responsive design principles to adapt to various screen sizes and orientations, making them a versatile solution for reaching users on different devices.

Name: Aastha Jajoo **Class: D15B** **Roll No: 23**

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

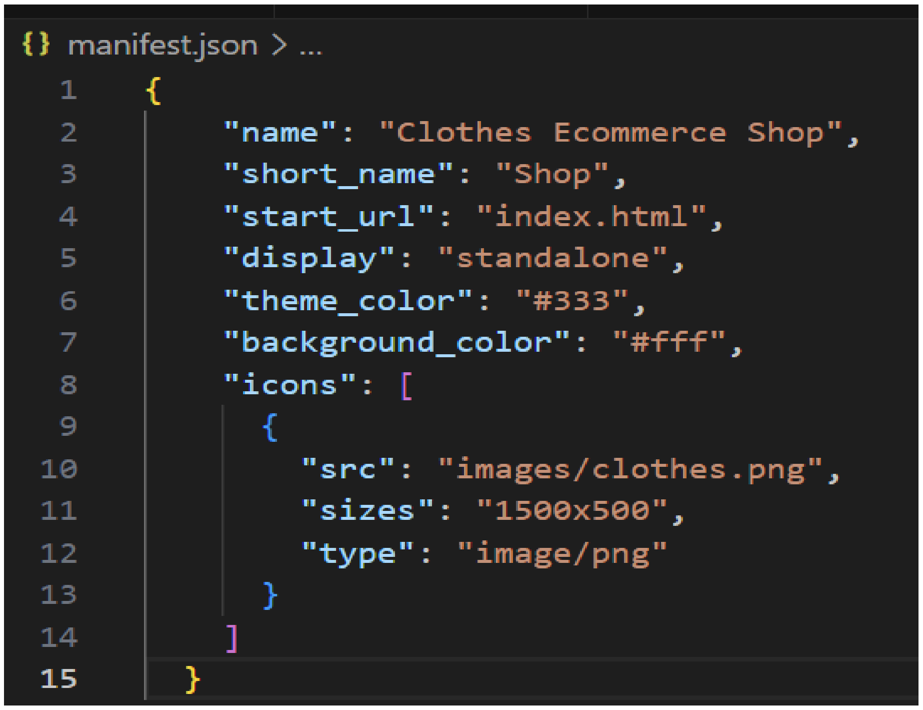
Step 1: Create an HTML page that would be the starting point of the application. This HTML will contain a link to the file named manifest.json. This is an important file that would be created in the next step.



index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Clothes Ecommerce Shop</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="manifest" href="manifest.json">
</head>
<body>
    <header>
        <h1>Clothes Ecommerce Shop</h1>
    </header>
    <main>
        <section class="products">
            <img src="./images/clothes.png" alt="clothes" />
        </section>
    </main>
    <script src="serviceworker.js"></script>
</body>
</html>
```

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.

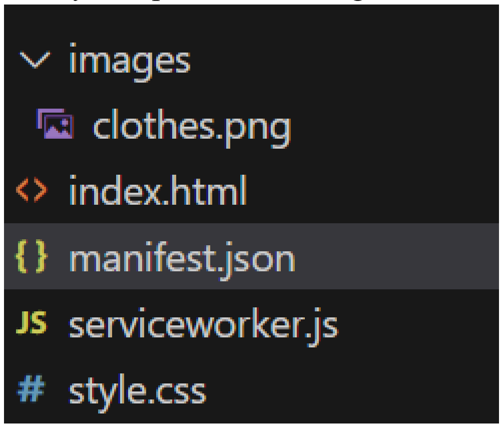


manifest.json > ...

```json
{
    "name": "Clothes Ecommerce Shop",
    "short_name": "Shop",
    "start_url": "index.html",
    "display": "standalone",
    "theme_color": "#333",
    "background_color": "#fff",
    "icons": [
      {
        "src": "images/clothes.png",
        "sizes": "1500x500",
        "type": "image/png"
      }
    ]
  }
```

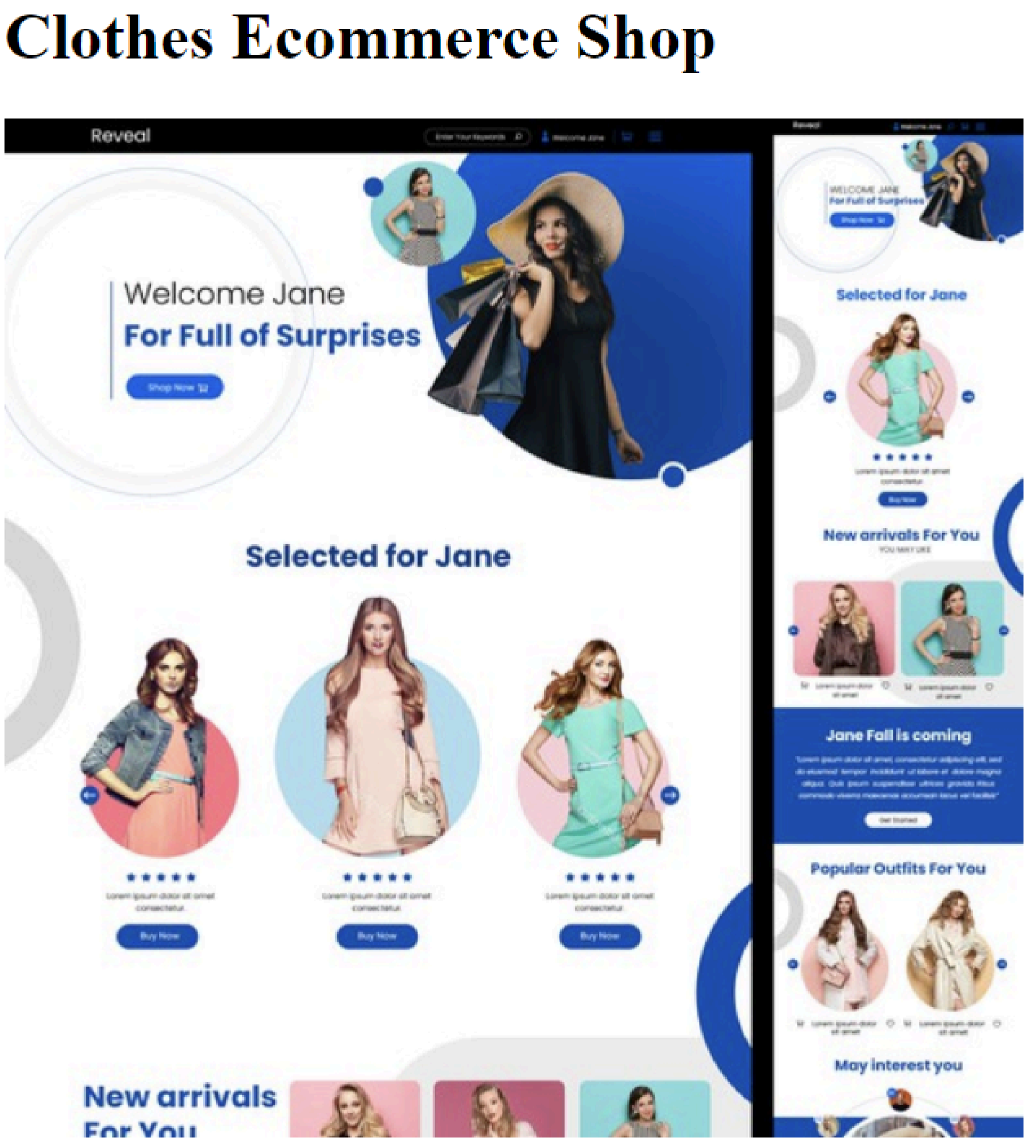
Name: Aastha Jajoo **Class: D15B** **Roll No: 23**

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.



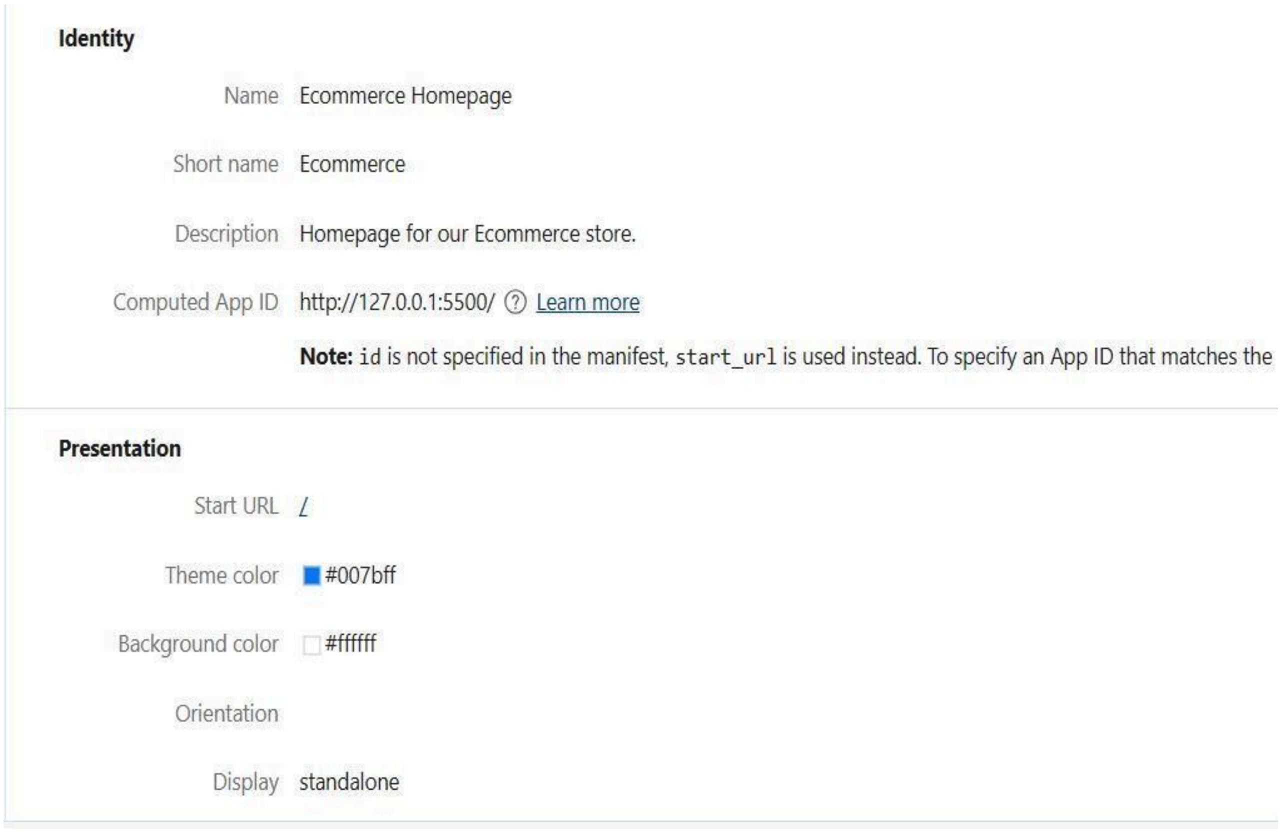
```
images
  clothes.png
index.html
manifest.json
serviceworker.js
style.css
```

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



Name: Aastha Jajoo **Class: D15B** **Roll No: 23**

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.



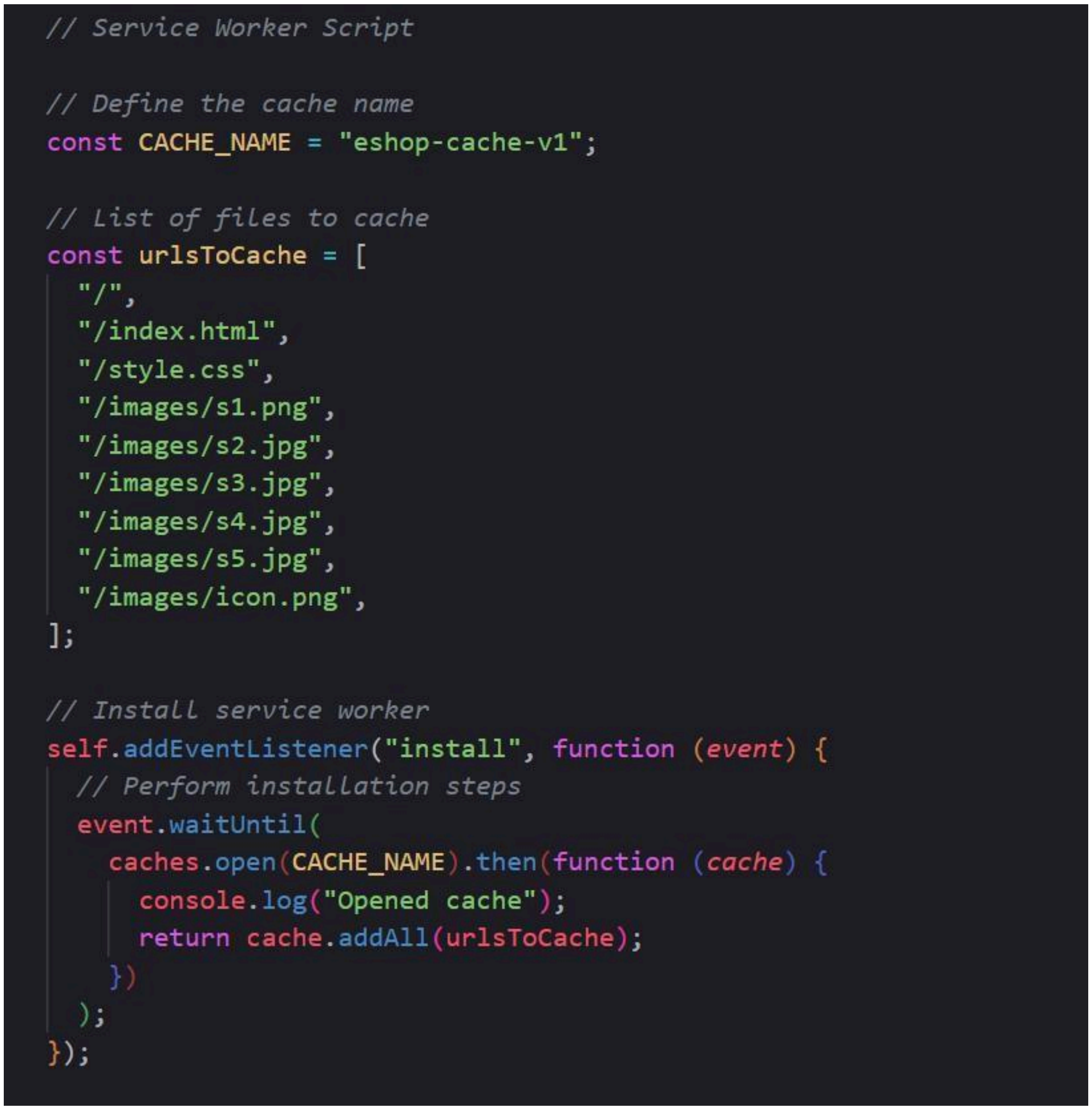
Identity

Name	Ecommerce Homepage
Short name	Ecommerce
Description	Homepage for our Ecommerce store.
Computed App ID	http://127.0.0.1:5500/ ⓘ Learn more
	Note: `id` is not specified in the manifest, `start_url` is used instead. To specify an App ID that matches the

Presentation

Start URL	/
Theme color	🟦 #007bff
Background color	⬜ #ffffff
Orientation	
Display	standalone

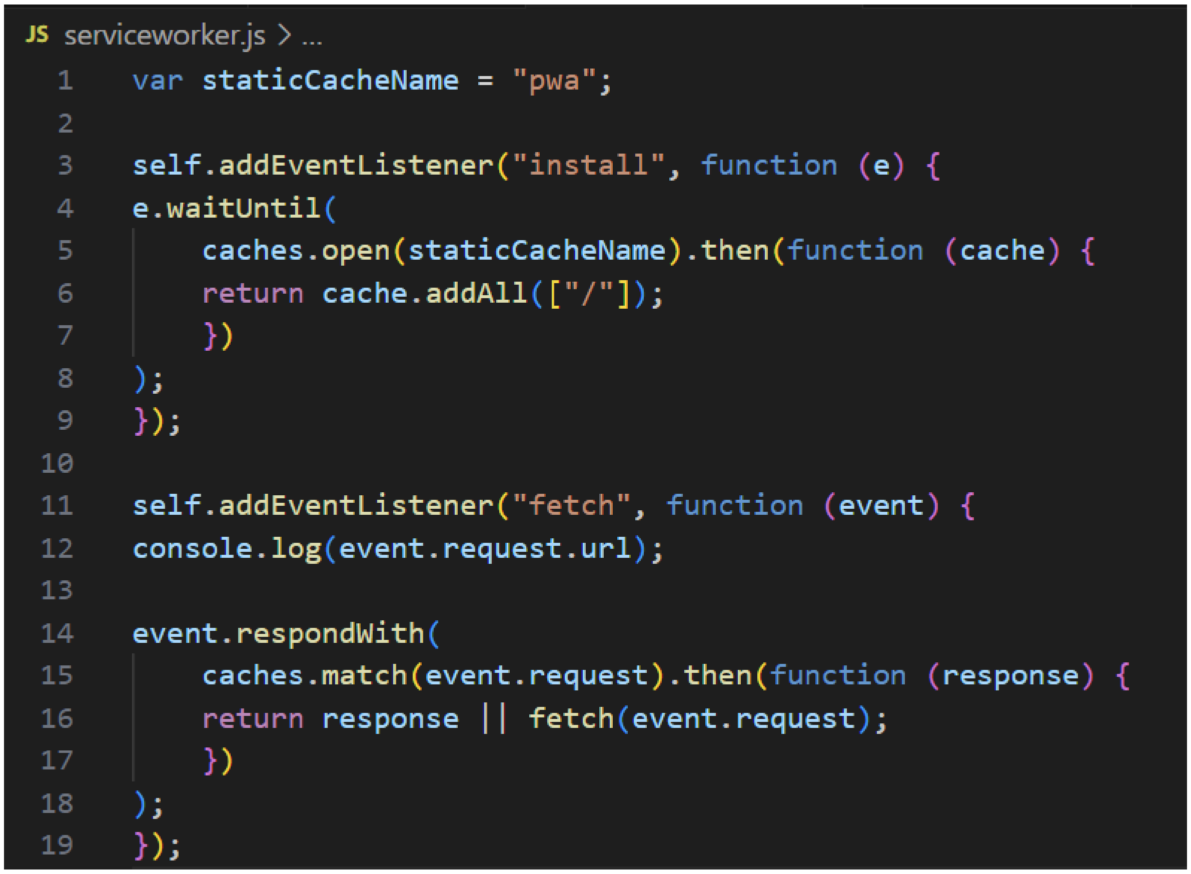
Step 6: Under the installability tab, it would show that no service worker is detected. We will need to create another file for the PWA, that is, serviceworker.js in the same directory. This file handles the configuration of a service worker that will manage the working of the application.



```javascript
// Service Worker Script

// Define the cache name
const CACHE_NAME = "eshop-cache-v1";

// List of files to cache
const urlsToCache = [
  "/",
  "/index.html",
  "/style.css",
  "/images/s1.png",
  "/images/s2.jpg",
  "/images/s3.jpg",
  "/images/s4.jpg",
  "/images/s5.jpg",
  "/images/icon.png",
];

// Install service worker
self.addEventListener("install", function (event) {
  // Perform installation steps
  event.waitUntil(
    caches.open(CACHE_NAME).then(function (cache) {
      console.log("Opened cache");
      return cache.addAll(urlsToCache);
    })
  );
});
```

Name: Aastha Jajoo **Class: D15B** **Roll No: 23**

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.



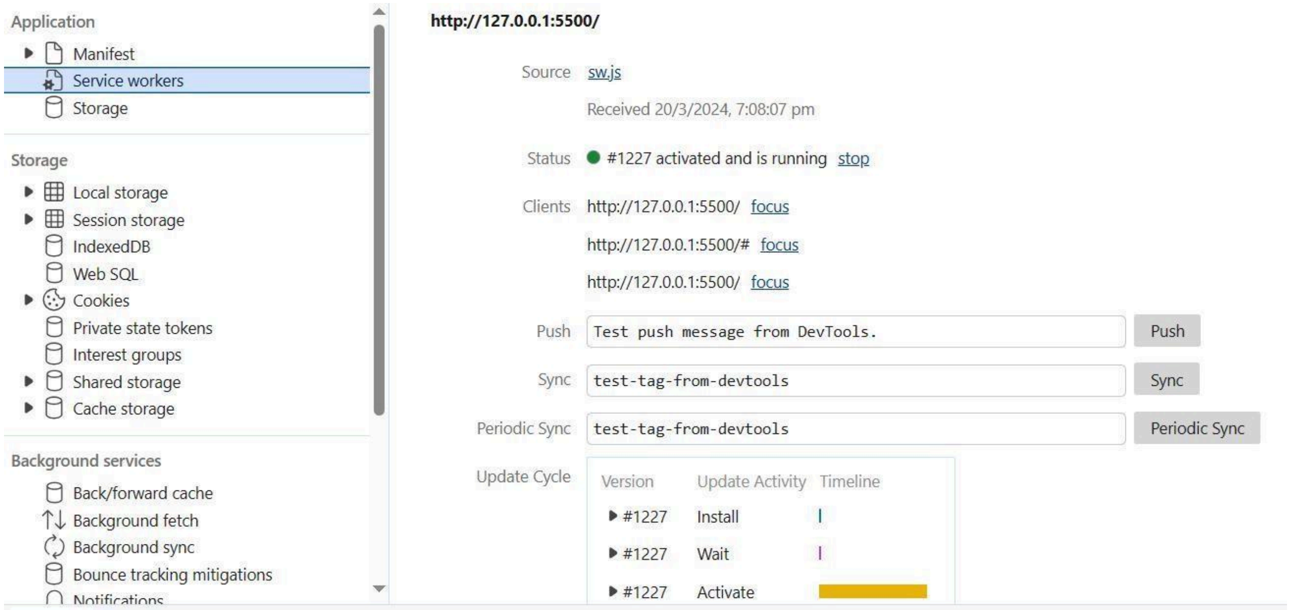
serviceworker.js > ...

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

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.



http://127.0.0.1:5500/

Source sw.js

Received 20/3/2024, 7:08:07 pm

Status ● #1227 activated and is running stop

Clients http://127.0.0.1:5500/ focus
http://127.0.0.1:5500/# focus
http://127.0.0.1:5500/ focus

Push `Test push message from DevTools.`

Sync `test-tag-from-devtools`

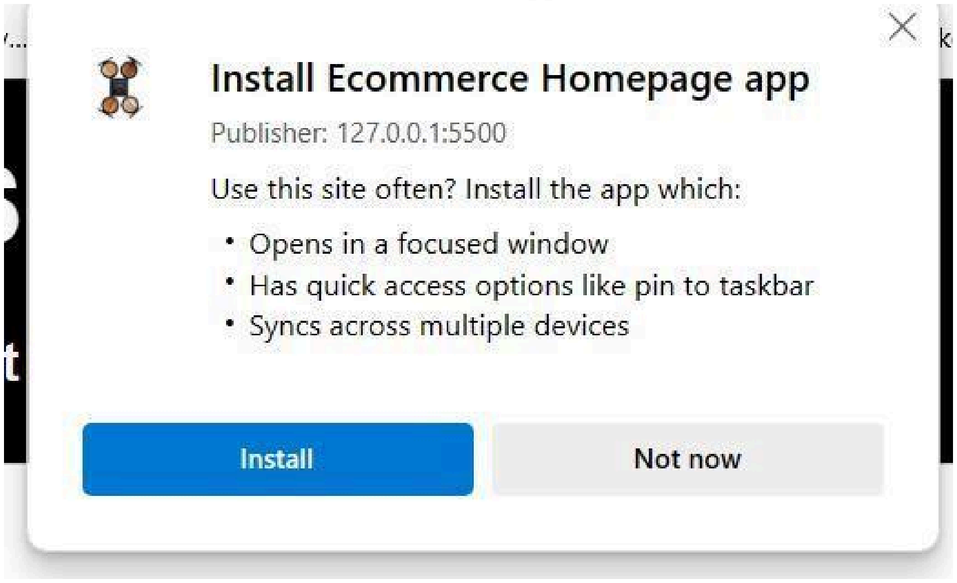
Periodic Sync `test-tag-from-devtools`

Update Cycle

Version	Update Activity	Timeline
#1227	Install	
#1227	Wait	
#1227	Activate	

Name: Aastha Jajoo **Class: D15B** **Roll No: 23**

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.



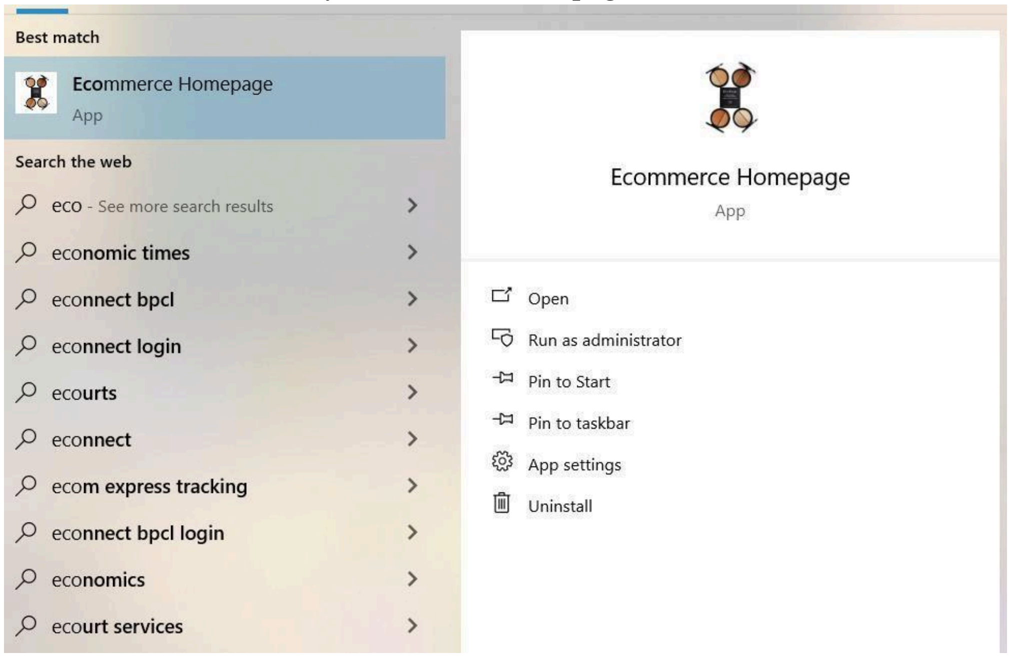
Install Ecommerce Homepage app

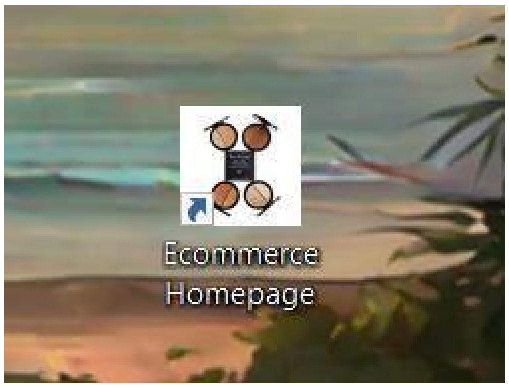
Publisher: 127.0.0.1:5500

Use this site often? Install the app which:

- Opens in a focused window
- Has quick access options like pin to taskbar
- Syncs across multiple devices

Now we have successfully installed this webpage:





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 PWA.