Veydant Sharma D15C 50

EXPERIMENT 7

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

Theory:

A regular web application is a website designed using standard web technologies like HTML, CSS, and JavaScript. It runs inside a web browser and is accessible across various devices and screen sizes. These apps are responsive and can be used on desktops, tablets, and smartphones. However, regular web apps are limited by the browser's capabilities and often require a continuous internet connection to function properly. Features such as offline access, push notifications, and native app-like experiences are generally not supported or are inconsistently available across different browsers.

A Progressive Web App (PWA) is an enhanced version of a regular web application. It combines the best features of web and native mobile applications. PWAs are built using the same core web technologies but offer additional functionalities such as offline support, background sync, push notifications, and installation on the user's device home screen. These features provide a native app-like experience while still being served through the web.

One of the key aspects of a PWA is the ability to be added to the home screen of a user's device, similar to native apps. This is made possible by writing metadata in a special file called the Web App Manifest. The manifest is a JSON file that contains important information about the app, such as its name, icons, theme color, background color, display mode, and start URL. When a valid manifest is linked in the HTML of the web app and a service worker is registered, the browser can prompt the user with an "Add to Home Screen" option. This allows the user to install the app without visiting an app store.

In summary, PWAs provide a better user experience compared to regular web apps by offering improved performance, offline capabilities, native-like interactions, and ease of installation. Writing and linking the correct metadata in the manifest file is an essential step in enabling the "Add to Home Screen" feature and transforming a standard web app into a PWA.

Code:

manifest.json

Veydant Sharma D15C 50

This is a JSON file that defines the metadata of a Progressive Web App, such as name, icons, colors, and display mode.
It enables the "Add to Home Screen" feature by telling the browser how the app should look and behave when installed.

```
{
   "name": "ShopEase Ecommerce App",
   "short_name": "ShopEase",
   "start_url": "index.html",
   "scope": "./",
   "display": "standalone",
   "background_color": "#ffffff",
   "theme_color": "#4CAF50",
   "description": "A simple ecommerce PWA demo app",
   "icons": [
     {
       "src": "https://picsum.photos/192",
       "sizes": "192x192",
       "type": "image/png"
     },
     {
       "src": "https://picsum.photos/512",
       "sizes": "512x512",
       "type": "image/png"
     }
   ]
 }
```

service-worker.js
This is a JavaScript file that runs in the background and handles caching, offline support, and background tasks.
It is essential for making the PWA work offline and improving performance by loading cached resources.

```
self.addEventListener('install', function(e) {
   console.log('Service Worker installing.');
   self.skipWaiting();
 });
```

```
  self.addEventListener('activate', function(e) {
    console.log('Service Worker activating.');
  });

  self.addEventListener('fetch', function(e) {
    // Cache-first strategy for offline support
    e.respondWith(
      caches.match(e.request).then(function(response) {
        return response || fetch(e.request);
      })
    );
  });
```

main.js

```
let deferredPrompt;

if ('serviceWorker' in navigator && 'SyncManager' in window) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('sw.js')
      .then(reg => {
        console.log('Service Worker registered!', reg);

        if ('sync' in reg) {
          reg.sync.register('sync-order')
            .then(() => {
              console.log('Sync event registered');
            })
            .catch(err => {
              console.error('Sync registration failed:', err);
            });
        }
      })
      .catch(err => {
        console.error('Service Worker registration failed:', err);
      });
  });
} else {
  console.error('Service Worker or SyncManager not supported.');
}
```

Veydant Sharma D15C 50

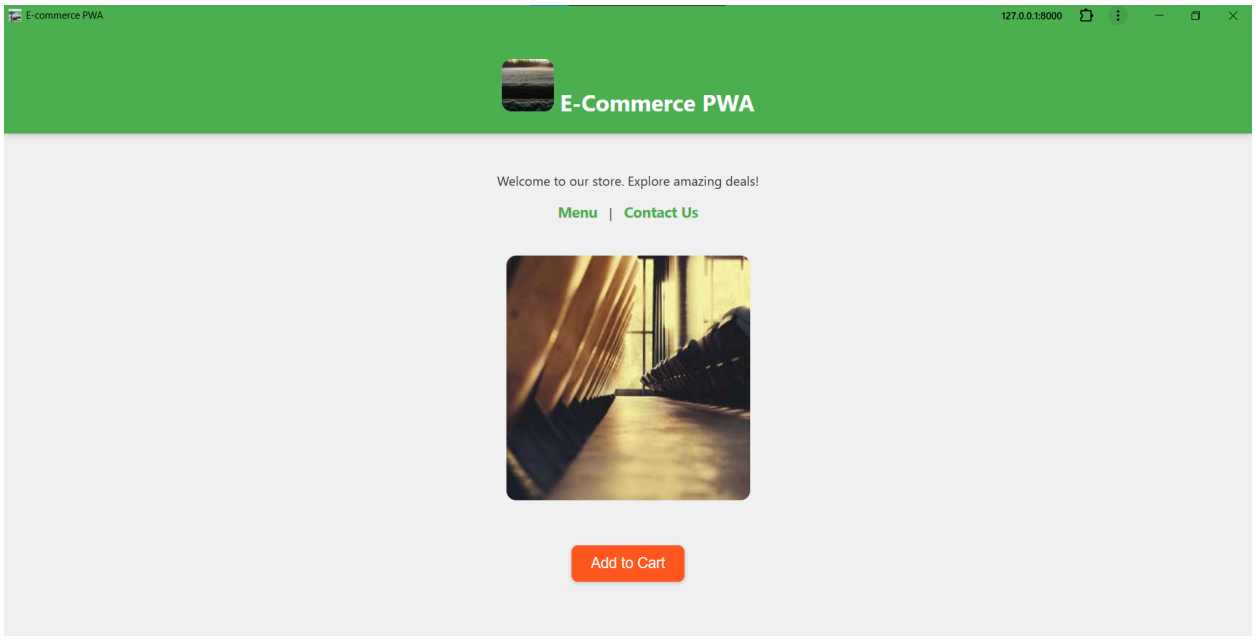
```
// ADD TO HOME SCREEN SETUP
window.addEventListener('beforeinstallprompt', (e) => {
  e.preventDefault(); // Stop automatic prompt
  deferredPrompt = e;

  const installBtn = document.createElement('button');
  installBtn.textContent = ' Install App';
  installBtn.style.cssText = 'position:fixed;bottom:20px;right:20px;padding:10px
20px;background:#4CAF50;color:white;border:none;border-radius:5px;z-index:1000;';
  document.body.appendChild(installBtn);

  installBtn.addEventListener('click', () => {
    installBtn.remove(); // Hide the button
    deferredPrompt.prompt(); // Show prompt

    deferredPrompt.userChoice.then(choice => {
      if (choice.outcome === 'accepted') {
        console.log('User accepted the A2HS prompt');
      } else {
        console.log('User dismissed the A2HS prompt');
      }
      deferredPrompt = null;
    });
  });
});
```

Output:



Conclusion:

In this experiment, we successfully wrote and linked a Web App Manifest file to our E-commerce web application, enabling the “Add to Home Screen” feature. We also implemented a basic service worker to support offline functionality. These components together transformed the standard web app into a Progressive Web App (PWA), offering users a native app-like experience with improved performance, offline support, and installability. This experiment demonstrates the foundational steps required to create a reliable and user-friendly PWA.