

EXPERIMENT NO. 7**Name:-** Tejas Gunjal**Class:-** D15A**Roll:No: -** 18

Aim: To write metadata of your E-commerce Progressive Web App (PWA) in a Web App Manifest file to enable the “Add to Home Screen” feature.

Theory :

A **Progressive Web App (PWA)** is a web application that combines the best features of both websites and mobile apps. PWAs are designed to be fast, reliable, and engaging, offering features such as offline access, push notifications, and the ability to be installed on a user's device.

One of the key features of a PWA is the **"Add to Home Screen" (A2HS) functionality**, which allows users to install the web app on their mobile or desktop devices, just like a native app. To enable this feature, a **Web App Manifest** file is required.

PWAs leverage modern web technologies to deliver an app-like experience, eliminating the need for users to download applications from app stores. They provide seamless performance across different devices and platforms, ensuring accessibility and responsiveness. By utilizing service workers for caching, PWAs can function even in low or no-network conditions, making them an excellent choice for enhancing user engagement and retention.

WEB APP MANIFEST

The **Web App Manifest** is a JSON file (manifest.json) that provides important metadata about the PWA, such as its name, icons, colors, and display settings. This file is essential for enabling the **"Add to Home Screen"** feature and ensuring the PWA appears like a native app when installed.

Importance of the Web App Manifest:

1. **Enables App Installation** – Allows users to install the PWA on their home screen or desktop.
2. **Defines App Appearance** – Controls how the app is displayed (standalone, fullscreen, or minimal UI).
3. **Enhances User Experience** – Provides a consistent theme, splash screen, and branding.
4. **Supports Cross-Platform Compatibility** – Works on multiple devices and browsers.

Key Components of a Web App Manifest File (manifest.json)

1. name and short_name

- Defines the full name and a shorter version of the app name used in installation prompts.

2. start_url

- Specifies the URL to open when the app is launched.

3. display

- Defines how the PWA will appear (e.g., standalone, fullscreen, minimal UI, or browser mode).

4. background_color and theme_color

- Controls the splash screen and browser theme when the app is launched.

5. icons

- Provides different icon sizes for various devices.

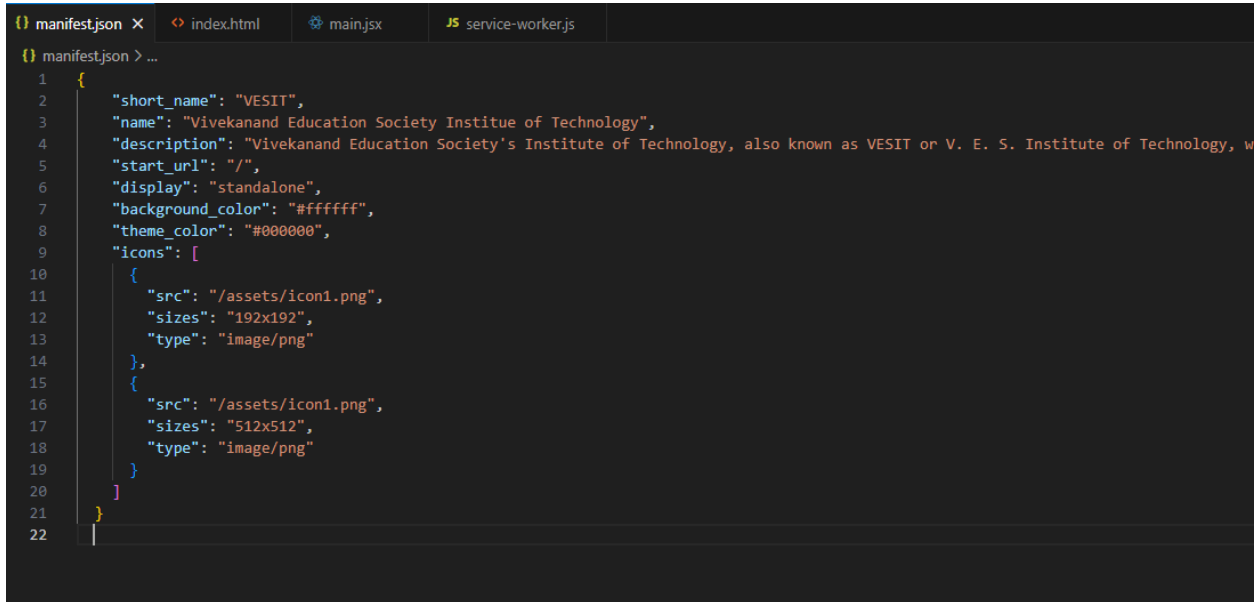
Advantages of Adding a PWA to Home Screen

1. **Fast Access** – Users can launch the app directly from their home screen.
2. **Offline Support** – PWAs can work offline using Service Workers.
3. **Push Notifications** – Engage users with real-time notifications.
4. **Better Performance** – Cached assets make loading faster.
5. **No App Store Required** – Users can install the app without visiting an app store.

Steps to “Add to HomeScreen” feature

1) Create a Web App Manifest File

Create a manifest.json file in your project’s root directory.
Add metadata required for PWA installation



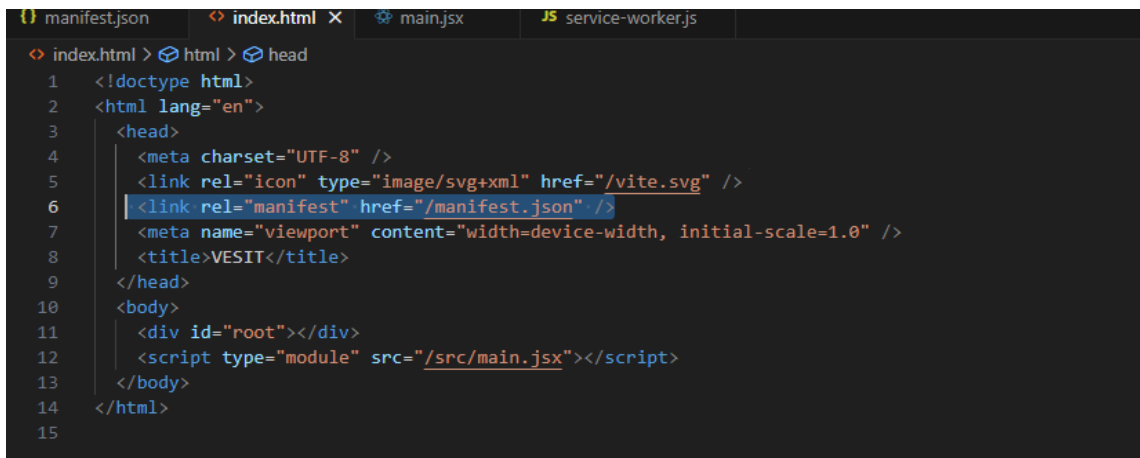
```

1  {
2    "short_name": "VESIT",
3    "name": "Vivekanand Education Society Institute of Technology",
4    "description": "Vivekanand Education Society's Institute of Technology, also known as VESIT or V. E. S. Institute of Technology, w
5    "start_url": "/",
6    "display": "standalone",
7    "background_color": "#ffffff",
8    "theme_color": "#000000",
9    "icons": [
10     {
11       "src": "/assets/icon1.png",
12       "sizes": "192x192",
13       "type": "image/png"
14     },
15     {
16       "src": "/assets/icon1.png",
17       "sizes": "512x512",
18       "type": "image/png"
19     }
20   ]
21 }
22

```

2) Link the Manifest in index.html

Open index.html and add this line inside the <head> tag



```

1  <!doctype html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6      <link rel="manifest" href="/manifest.json" />
7      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
8      <title>VESIT</title>
9    </head>
10   <body>
11     <div id="root"></div>
12     <script type="module" src="/src/main.jsx"></script>
13   </body>
14 </html>
15

```

3) Register a Service Worker

Create a new file service-worker.js and add the following code to enable caching

```
var staticCacheName = "pwa-v1"; // Versioned cache name
```

```
self.addEventListener("install", function (e) {
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll([
        "/",
        "/index.html",
        "/style.css",
        "/script.js",
        "/logo.png",
        "/icon1.png",
        "/offline.html" // Offline fallback page
      ]);
    })
  );
});
```

```
self.addEventListener("activate", function (event) {
  var cacheWhitelist = ["staticCacheName"];

  event.waitUntil(
    caches.keys().then(function (cacheNames) {
      return Promise.all(
        cacheNames.map(function (cacheName) {
          if (cacheWhitelist.indexOf(cacheName) === -1) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
```

```
self.addEventListener("fetch", function (event) {
```

```

event.respondWith(
  caches.match(event.request).then(function (response) {
    if (response) {
      return response; // Serve from cache
    }

    return fetch(event.request).catch(function () {
      return caches.match("/offline.html"); // Serve offline fallback
    });
  })
);
});

self.addEventListener('message', (event) => {
  if (event.data.action === 'skipWaiting') {
    self.skipWaiting();
  }
});

```

4) **Register this service worker in index.html**

```

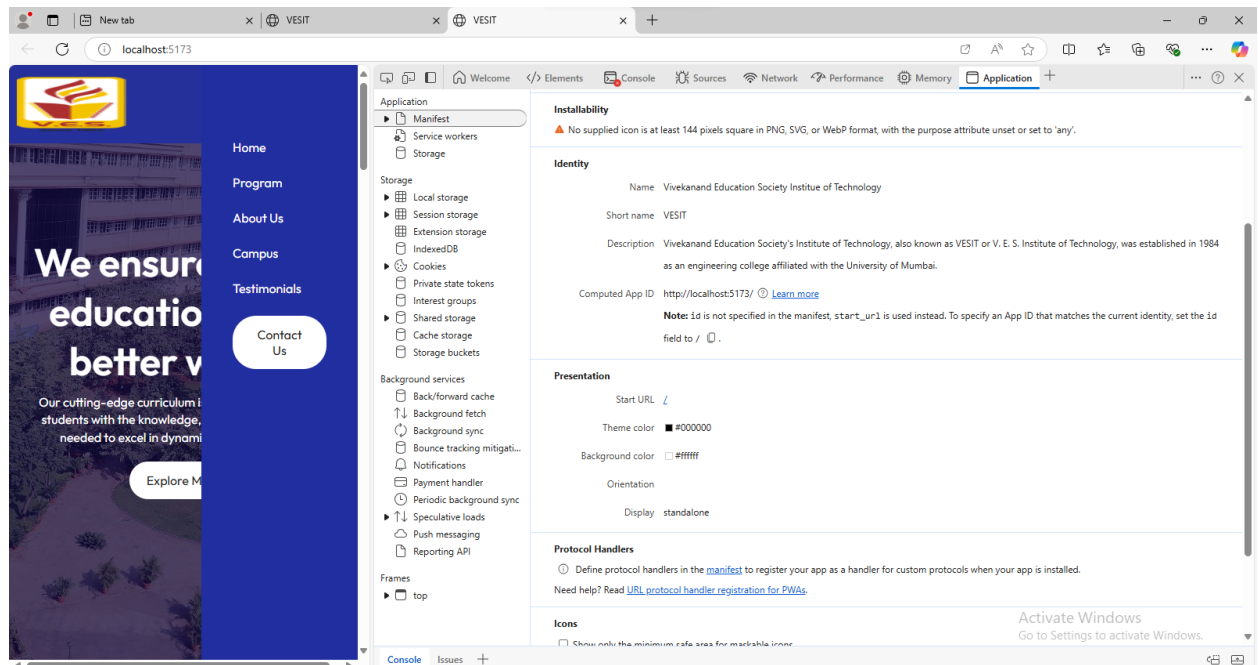
// src/index.js (or src/main.jsx)
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker
      .register('/service-worker.js')
      .then((registration) => {
        console.log('Service Worker registered with scope:', registration.scope);
      })
      .catch((error) => {
        console.log('Service Worker registration failed:', error);
      });
  });
}

```

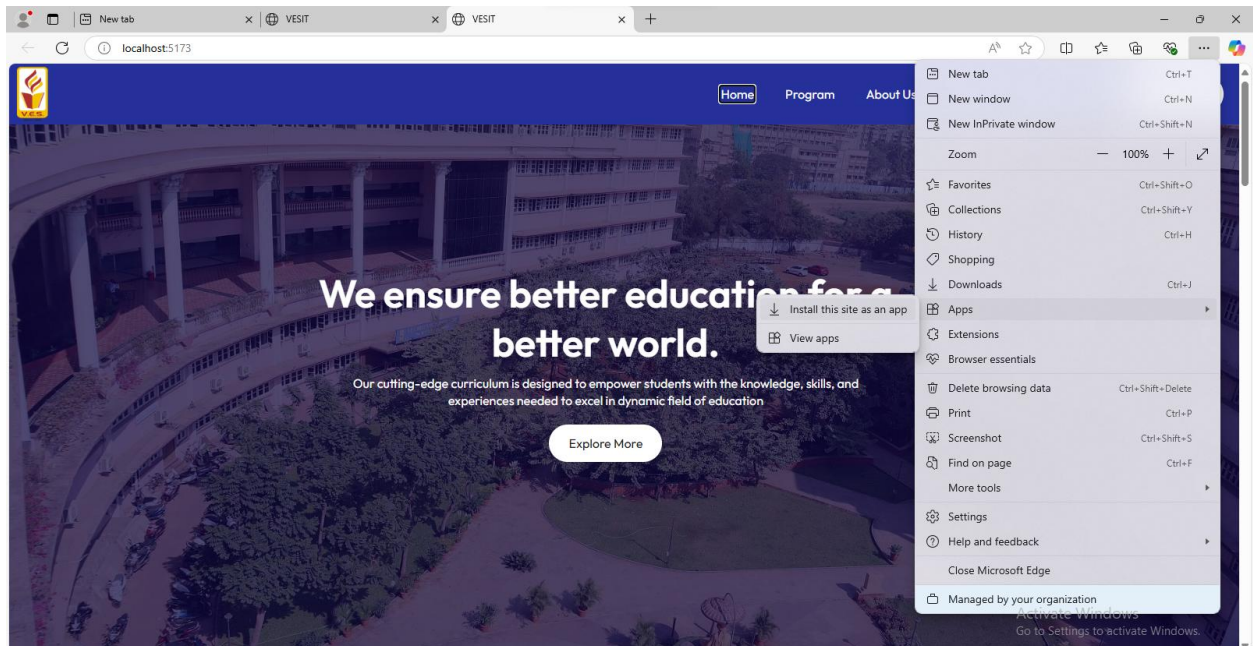
5) Test the PWA Installation



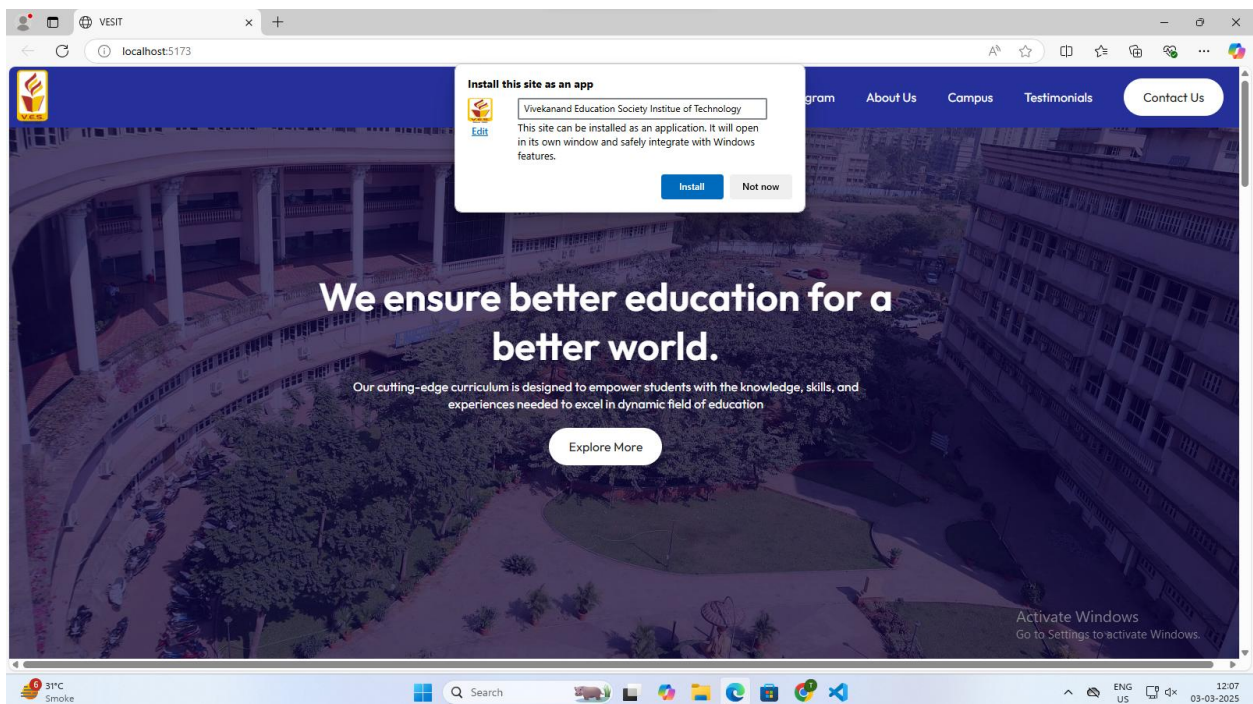
5.1) Open Google Chrome and go to Developer Tools > Application > Manifest to verify your manifest file.



5.2) Installing the PWA Using Microsoft Edge's "Add to Home Screen" Feature



5.3) App icon along with name of the app will appear



5.4) PWA Successfully Installed

Users can choose to **pin the app to the taskbar, Start menu, or create a desktop shortcut** for easy access.

