MAD & PWA Lab
Journal

Experiment No.	07
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	20
Name	Madhura Jangale
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Theory:

A Progressive Web Application (PWA) is a type of web application that offers an enhanced user experience similar to native mobile apps. It is built using standard web technologies (HTML, CSS, JavaScript) but incorporates modern features like offline functionality, push notifications, and home screen installation.
PWAs aim to bridge the gap between traditional websites and mobile apps by providing fast, reliable, and engaging experiences across all devices.

Evolution of Web Apps to PWAs

Web applications have evolved significantly over the years:

- Traditional Websites – Early websites were static and lacked interactivity.
- Dynamic Web Apps – Introduced AJAX and JavaScript for better user interactions.
- Single Page Applications (SPA) – Frameworks like React, Angular, and Vue improved performance but still relied on internet connectivity.
- Progressive Web Apps (PWA) – Introduced offline functionality, app-like experiences, and better performance, making them competitive with native apps.

Core Principles of PWA

PWAs follow specific principles to ensure a high-quality experience:

1. Progressive – Works for all users, regardless of the browser or device.
2. Responsive – Adapts to various screen sizes and device types.
3. Connectivity Independent – Functions even with limited or no internet access.
4. App-like Feel – Provides an immersive experience similar to native apps.
5. Secure – Uses HTTPS to protect data and user privacy.
6. Discoverable – Indexed by search engines, making them easy to find.
7. Re-engageable – Supports push notifications to keep users engaged.
8. Installable – Users can add the app to their home screen without an app store.
9. Linkable – Easily shared via a URL without requiring installation.

Key Technologies Behind PWAs

PWAs rely on modern web technologies to deliver enhanced functionality:

a) Service Workers

- A background script that enables caching, push notifications, and offline functionality.
- It helps store web assets so that users can access content even without the internet.

b) Web App Manifest

- A JSON file that defines the app's metadata, such as name, icon, theme color, and display mode.
- Enables home screen installation and native-like appearance.

c) HTTPS (Secure Context)

- Ensures data privacy and security by encrypting communication between the app and server.
- Required for service workers and push notifications.

d) Responsive Design

- Uses CSS media queries and flexible layouts to ensure compatibility across different devices and screen sizes.

Benefits of PWAs

a) Improved Performance

- PWAs load faster due to caching and preloading mechanisms.
- Minimizes server requests by storing data locally.

b) Works Offline

- Cached content allows users to browse even without an internet connection.
- Useful for news websites, e-commerce platforms, and social media apps.

c) No App Store Dependency

- PWAs do not require approval from app stores like Google Play or Apple App Store.
- Users can install PWAs directly from the browser, reducing friction in the adoption process.

d) Cost-Effective

- A single PWA works across multiple platforms, reducing development and maintenance costs.

- Eliminates the need for separate native apps for iOS and Android.

e) Increased User Engagement

- Features like push notifications help re-engage users and improve retention rates.
- Users can receive updates and alerts even when the app is not open.

f) SEO-Friendly

- Unlike native apps, PWAs are indexed by search engines, making them discoverable.
- Better ranking potential in Google Search results.

Implementation:

Step 1: Add a link to the file named manifest.json in index.html.

<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

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.

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "faviconn.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logoo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logoo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

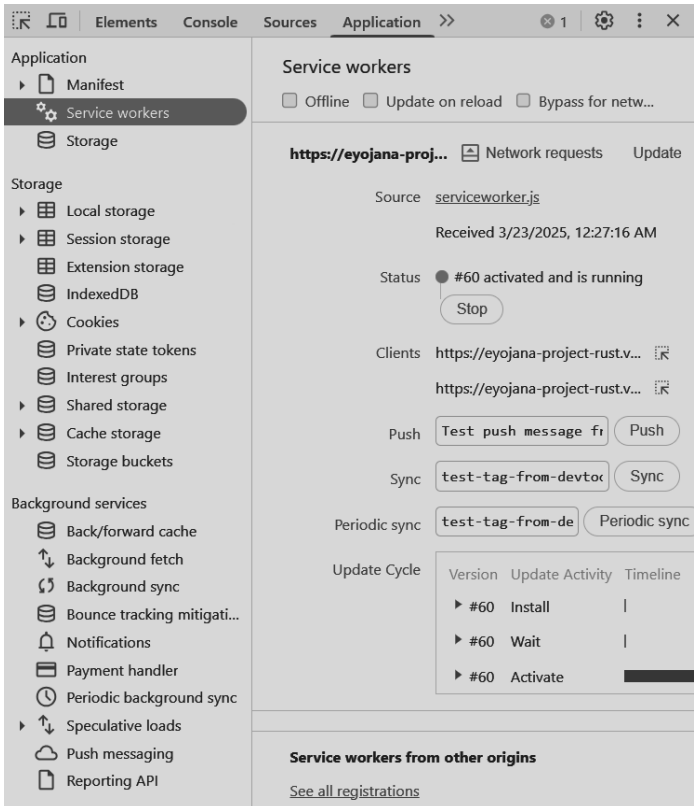
Step 3: We will need to create another file for the PWA, that is, serviceworker.js in the same directory. This file handles the configuration of a service worker that will manage the working of the application.

```
var staticCacheName = "eyojana";

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

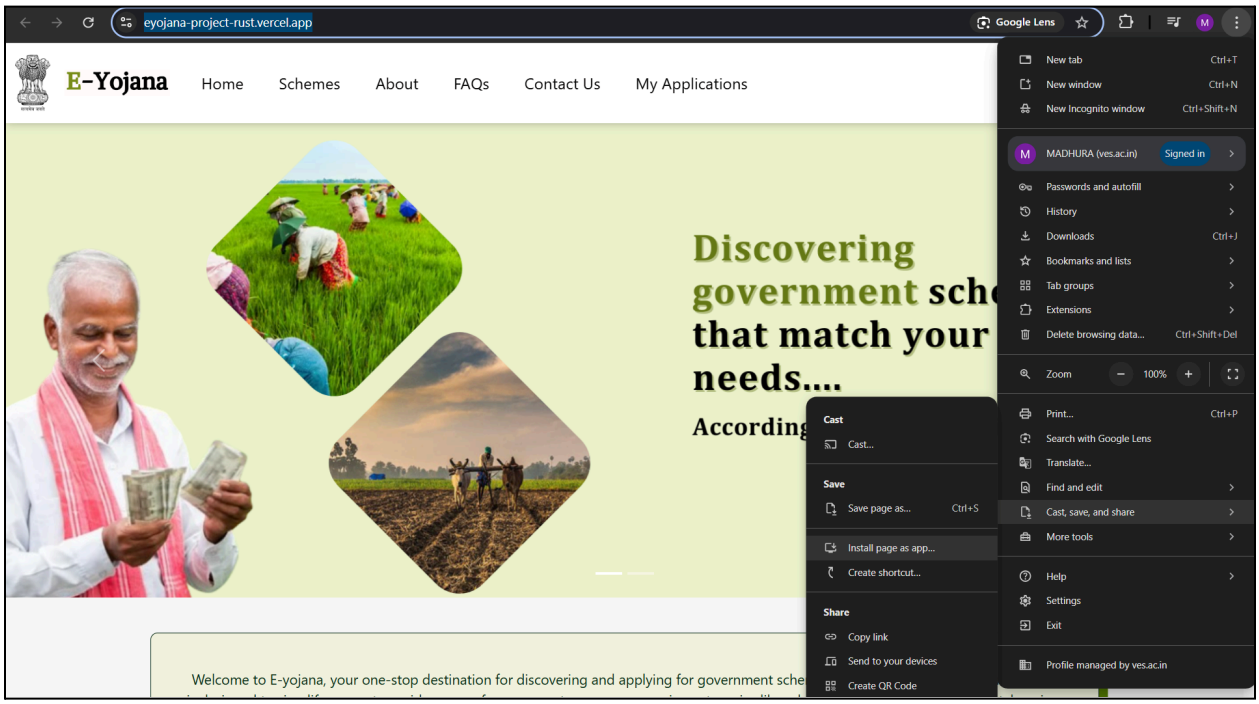
Step 4: link the service worker file to index.html. Add the below script tag in it.

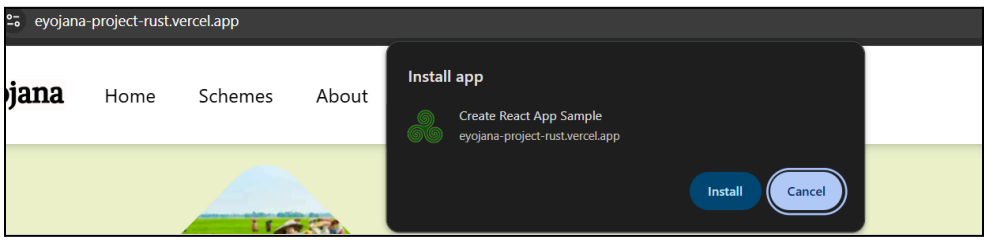
```
<script>
        window.addEventListener('load', () => {
        registerSW();
        });

        // Register the Service Worker
        async function registerSW() {
        if ('serviceWorker' in navigator) {
                try {
                await navigator
                                .serviceWorker
                                .register('serviceworker.js');
                }
                catch (e) {
                console.log('SW registration failed');
                }
        }
        }
    </script>
```

Step 5: Open website -> inspect -> Application -> Service workers



Step 6: Install the application





App installed!