# Experiment No: 8

**Aim**: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

**Theory**:

- **Service Worker**

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

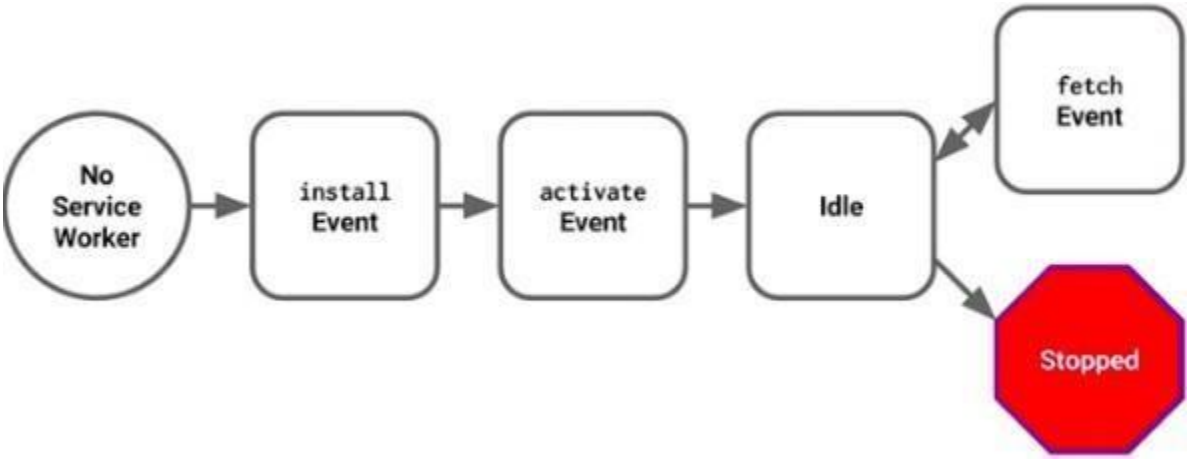
1. A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
2. Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
3. The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.

- **Uses of Service Workers**
    a. **You can dominate Network Traffic:** You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.
    b. **You can Cache:** You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.
    c. **You can manage Push Notifications:** You can manage push notifications with Service Worker and show any information message to the user.
    d. **You can Continue:** Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

- **Limitations of Service Workers**

    a. You can’t access the **Window:** You can’t access the window, therefore, You can’t manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.
    b. You can’t work it on **80 Port:** Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

- **Service Worker Cycle**




A service worker goes through three steps in its life cycle:

- Registration
- Installation ● Activation

## Code:

### 1. flutter_service_worker.js

```
const CACHE_NAME = 'estore-cache-v2';
const urlsToCache = [
  '/',
  '/index.html',
  '/js/app.js',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png',
  '/offline.html'
];

// Install
self.addEventListener('install', event => {
event.waitUntil(
    caches.open(CACHE_NAME).then(cache => cache.addAll(urlsToCache))
  );
  self.skipWaiting();
});

// Activate
self.addEventListener('activate', event => {
event.waitUntil(     caches.keys().then(keys
=>       Promise.all(keys.map(key => {
if (key !== CACHE_NAME) {
return caches.delete(key);
        }
      }))
```

```
    )
  );
  self.clients.claim();
});

// Fetch
self.addEventListener('fetch', event => {
event.respondWith(
    caches.match(event.request).then(cachedResponse => {
      // Serve from cache if available
      if (cachedResponse) return cachedResponse;

      // Try fetching from network
      return fetch(event.request).catch(() => {
        // Fallback only for navigation (HTML page loads)
if (event.request.mode === 'navigate') {           return
caches.match('/offline.html');
        }
      });
    })
  );
});
```

**2. index.html**

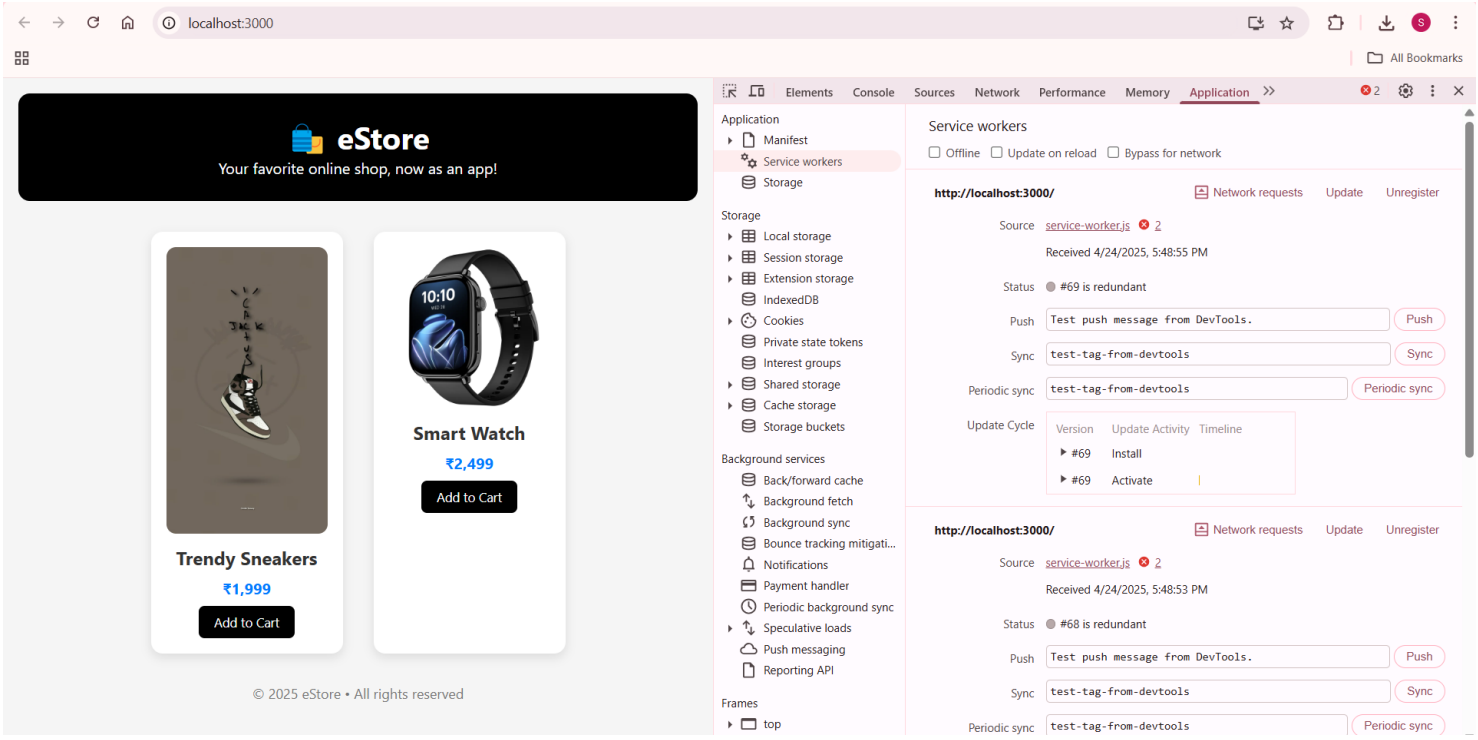
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="theme-color" content="#000000" />
  <meta name="description" content="An awesome eCommerce PWA for all your shopping needs." />
  <meta name="keywords" content="eCommerce, shopping, PWA, web app, buy, cart" />
  <meta name="author" content="Your Name or Store" />
  <meta property="og:title" content="My eCommerce Store" />
  <meta property="og:description" content="Shop the latest products on our PWA!" />
  <meta property="og:type" content="website" />
  <meta property="og:image" content="icons/web-app-manifest-512x512.png" />
  <meta property="og:url" content="https://your-site-url.com" />

  <link rel="manifest" href="manifest.json" />
  <link rel="icon" href="icons/web-app-manifest-192x192.png" />
  <title>My eCommerce PWA</title>
</head>
<body>
  <h1>  Welcome to My eCommerce universe!</h1>
  <p>Enjoy offline shopping experience and add us to your home screen!</p>
```

```
  <script src="js/app.js"></script>
  <script>
    if ('serviceWorker' in navigator) {
navigator.serviceWorker.register('service-worker.js')
.then(() => console.log("  Service Worker registered"))
        .catch(err => console.error("  Service Worker registration failed", err));
    }
  </script>
</body>
</html>
```

**Output:**

- After code update in flutter_service_worker.js a serviceworker got created in the background.
- And its status is #2701 activated and running means it is actively running in the background.




**Conclusion:**

In this experiment, we successfully implemented and registered a service worker for our eCommerce PWA. This enables offline functionality, caching essential files, and managing network traffic. The service worker ensures users can continue browsing even without an internet connection, enhancing the app's performance and reliability. With the service worker activated and running in the background, we've improved the app's offline capabilities and overall user experience.