# EXP 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:**

A Service Worker is a background script that runs separately from the web page and enables key Progressive Web App (PWA) features such as offline access, background sync, and push notifications.

Coding and Registering a Service Worker:

- The service worker is written in a separate JavaScript file (usually service-worker.js).
- It is registered in the main HTML or JavaScript file using navigator.serviceWorker.register().
- This registration allows the browser to recognize and manage the service worker.

Installation and Activation:

- Install Phase: This is the first step when the browser detects a new service worker. In this phase, static assets (HTML, CSS, JS, images) are cached using caches.open() and cache.addAll() for offline availability.
- Activate Phase: After installation, the service worker moves to the activate phase. Here, old caches can be cleaned up using caches.keys() and caches.delete() to ensure only the latest resources are stored.

Role in an E-Commerce PWA:

- Ensures offline functionality so users can browse cached product pages without internet.
- Improves performance through cached responses.
- Provides a reliable user experience, especially in low or no network areas.

## Code:

## flutter_service_worker.js

```
self.addEventListener('install', (event) => {
  console.log('[Service Worker] Install event');
  self.skipWaiting(); // Activate worker immediately
});

self.addEventListener('activate', (event) => {
  console.log('[Service Worker] Activate event');
        // Cleanup old caches here if needed
});

self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
        return response || fetch(event.request);
        })
        );
});
```

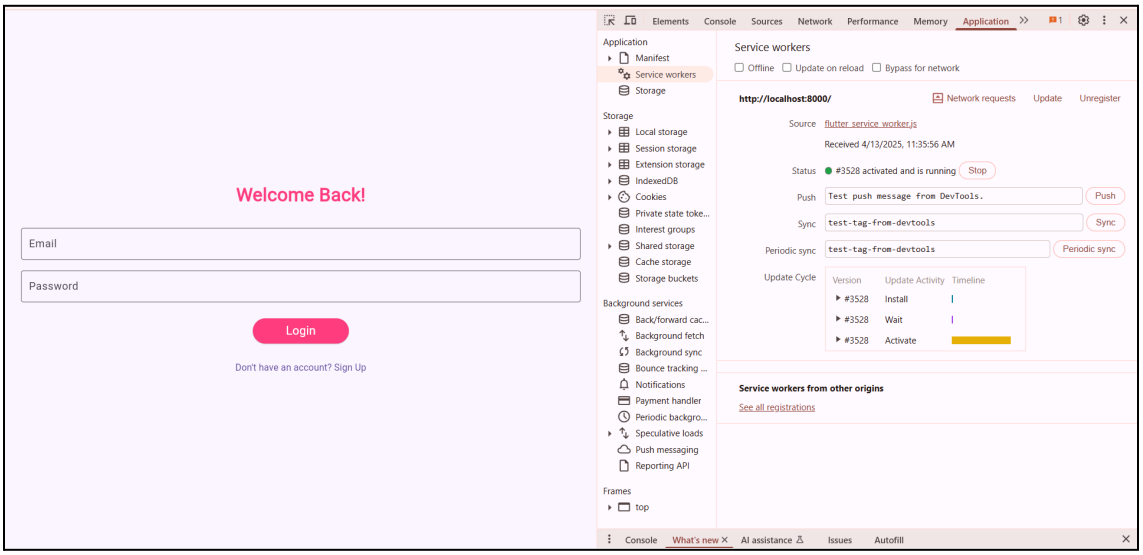
## index.html

```
<!DOCTYPE html>
<html>
<head>
  <!--
    If you are serving your web app in a path other than the root, change the
    href value below to reflect the base path you are serving from.

    The path provided below has to start and end with a slash "/" in order for
    it to work correctly.

    For more details:
    * https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base

    This is a placeholder for base href that will be replaced by the value of
    the `--base-href` argument provided to `flutter build`.
  -->
  <base href="$FLUTTER_BASE_HREF">

  <meta charset="UTF-8">
  <meta content="IE=Edge" http-equiv="X-UA-Compatible">
  <meta name="description" content="A new Flutter project.">

  <!-- iOS meta tags & icons -->
  <meta name="apple-mobile-web-app-capable" content="yes">
```

```
  <meta name="apple-mobile-web-app-status-bar-style" content="black">

  <meta name="apple-mobile-web-app-title" content="moodlog">

  <link rel="apple-touch-icon" href="icons/Icon-192.png">

  <meta name="google-signin-client_id"
content="15128894708-pqnk893c2cnslsldu7g1ghk33dech5vm.apps.googleusercontent.com">



  <script src="https://accounts.google.com/gsi/client" async defer></script>



  <!-- Favicon -->

  <link rel="icon" type="image/png" href="favicon.png"/>



  <title>Moodlog</title>

  <link rel="manifest" href="manifest.json">

</head>

<body>

  <script src="flutter_bootstrap.js" async></script>

</body>

<script>

  if ('serviceWorker' in navigator) {

            window.addEventListener('load', function () {

     navigator.serviceWorker.register('flutter_service_worker.js')

            .then(function (registration) {

            console.log('Service Worker registered with scope:', registration.scope);

            }).catch(function (error) {

            console.log('Service Worker registration failed:', error);
```

```
        });

        });

    }

</script>

</html>
```

**Output:**




**Conclusion:**

In this experiment, a service worker was successfully coded, registered, installed, and activated for the PWA. This enhanced the app's performance by enabling offline access, faster loading, and improved reliability, ensuring a seamless user experience even with limited or no internet connectivity.