Experiment number 08

MAD and PWA ab

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

Theory:

Service Worker:-

A Service Worker is a script that runs in the background of a web application, separate from the main browser thread. It enables powerful features like offline support, background sync, and push notifications by intercepting network requests and caching resources. Service workers are essential for building Progressive Web Apps (PWAs), allowing them to function reliably even with limited or no internet connectivity. They follow a lifecycle of installation, activation, and fetching, and must be served over HTTPS for security reasons. Overall, service workers enhance performance, reliability, and user experience in modern web applications.

Installation and Activation Process:-

The installation and activation process of a service worker involves two main phases: **installation** and **activation**. During the installation phase, the browser downloads the service worker script and executes it, typically caching essential assets for offline use. If the installation is successful, the service worker moves to the activation phase, where it takes control of the app and clears any outdated caches. This ensures the app uses the latest cached resources. The service worker only becomes active when there are no active pages using the old version, ensuring a smooth and controlled update. This process is essential for maintaining a reliable and up-to-date PWA experience.

Code:-

service_worker.js

```
//service-worker.js
const CACHE_NAME = 'news-app-cache-v1';
const FILES_TO_CACHE = [
  '/',
  '/index.html',
  '/offline.html',
  '/main.dart.js',
  '/manifest.json',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png'
];
```

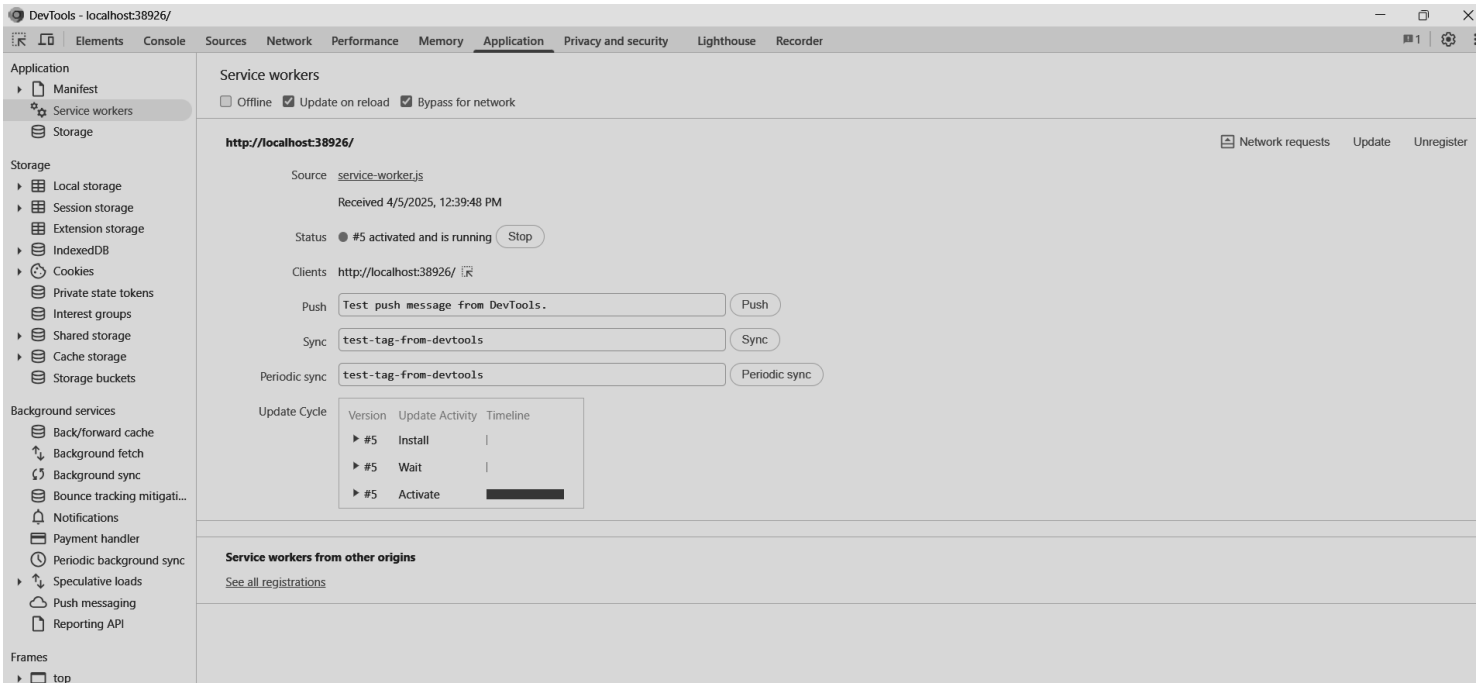
```
// Install event
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME).then(cache => cache.addAll(FILES_TO_CACHE))
  );
});

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

// Fetch event
self.addEventListener('fetch', event => {
  if (event.request.method !== 'GET') return;

  event.respondWith(
    caches.match(event.request).then(cached => {
      if (cached) {
        console.log('[Service Worker] Serving from cache:', event.request.url);
        return cached;
      }

      return fetch(event.request)
        .then(response => {
          return caches.open(CACHE_NAME).then(cache => {
            cache.put(event.request, response.clone());
            console.log('[Service Worker] Fetched and cached:', event.request.url);
            return response;
          });
        })
        .catch(() => {
          console.log('[Service Worker] Fetch failed, serving offline page.');
          return caches.match('/offline.html');
```

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

//  Sync event
self.addEventListener('sync', event => {
  console.log('[Service Worker] Sync event fired with tag:', event.tag);
  if (event.tag === 'sync-news') {
    event.waitUntil(sendQueuedArticles());
  }
});

// 📨 Push event
self.addEventListener('push', event => {
  if (event.data) {
    const data = event.data.json();
    if (data.method === 'pushMessage') {
      const options = {
        body: data.message,
        icon: '/icons/icon-192x192.png',
        badge: '/icons/icon-192x192.png'
      };
      event.waitUntil(
        self.registration.showNotification("News App", options)
      );
    }
  }
});

function sendQueuedArticles() {
  return Promise.resolve(console.log(" Background sync triggered: sending queued articles."));
}

service_worker_reistration.js
if ('serviceWorker' in navigator && 'SyncManager' in window) {
    window.addEventListener('load', () => {
      navigator.serviceWorker
        .register('/service-worker.js')
        .then(reg => {
          console.log('Service Worker registered ');

          // Register sync event
```

```
      reg.sync.register('sync-news');
    })
    .catch(err => console.error('Service Worker registration failed ', err));
});
}
```

Github Link:-
https://github.com/Ishan2611/News_Edge/commit/79c2f094459dcee84a9362d7b313abf9019edfe4

Output:



This shows the Service Workers section of the Chrome DevTools Application tab. It displays a service worker running at http://localhost:38926/, with the file service-worker.js. The service worker is currently activated and running, with options to stop, push messages, sync, and trigger periodic sync. It also shows the update cycle, indicating the stages: Install, Wait, and Activate. This panel is useful for testing and debugging service worker behaviors in Progressive Web Apps (PWAs), allowing developers to simulate offline mode, push notifications, and background sync features.

Conclusion:-

In this experiment, we successfully coded and registered a service worker for the News-App

PWA, enabling offline capabilities and enhancing performance. The process involved writing a service worker script, caching essential assets during the install phase, and cleaning up outdated caches during activation. This ensured that users could access the app even without an internet connection. Registering the service worker correctly allowed it to run in the background and handle network requests efficiently. Through this, we gained hands-on experience with the core functionality of PWAs. Overall, the experiment highlighted the importance of service workers in delivering a reliable and seamless user experience.