

Name : Harsh Jadhav

Roll No 21

DN : D15B

20

### MPL Experiment No 9.

\* Aim :- To implement service worker event like fetch event for my app.

\* Theory :-

Service worker is a script that works on browser background without user interaction independently.

Also it reassembles a proxy that works on user guide

Things to note about service worker :-

• A service worker is a ~~request~~ programmable network proxy that lets you control how network requests from your page are handled.

• Service workers only over HTTPS. Because service worker can intercept network requests and modify responses "man in the middle" attack could be very bad.

\* Fetch Event :-

You can track and manage page through traffic with this event. You can check existing cache memory, cache, manage "cache first" request and restore.

a response that you want.

### → Sync Event :-

Background Sync is a web API that is used to delay a process until the Internet connection is available. We can adopt this definition to real world there is an browser and we want to send an email with this tool.

### → Push Event :-

This is the event that handles push notifications are received from server, you can apply any method with received data.

We can check it by using following method:  
Notification.requestPermission().

### \* Conclusion :-

The implementation of fetch, sync, push events in a PWA service worker significantly enhances the functionality, performance and user experience of modern web application.

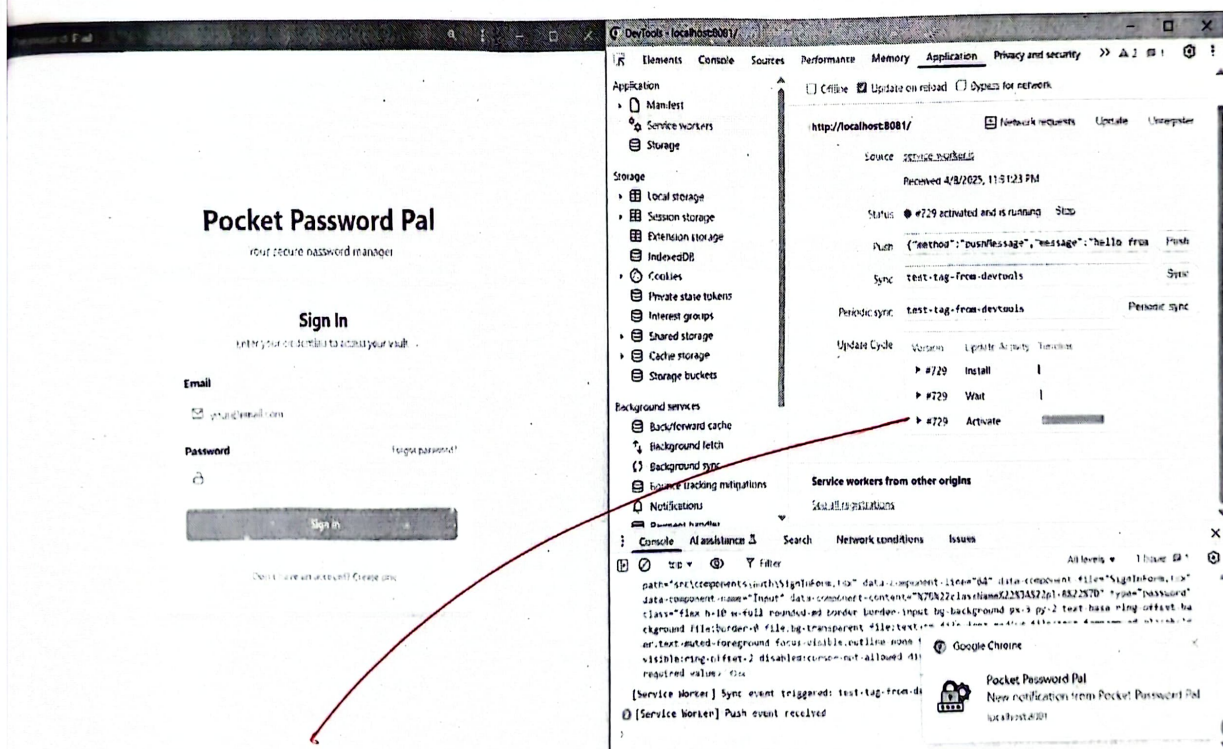


## Push Event:

```
// Push notification event
self.addEventListener('push', event => {
  console.log('[Service Worker] Push event received');
  const data = event.data ? event.data.json() : {};

  const options = {
    body: data.body || 'New notification from Pocket Password Pal',
    icon: '/icons/icon-192x192.png',
    badge: '/icons/icon-72x72.png',
    data: {
      url: data.url || '/'
    }
  };

  event.waitUntil(
    self.registration.showNotification(data.title || 'Pocket Password Pal', options)
  );
});
```



## Fetch Event:

```
// Fetch event - respond with cache first, then network
self.addEventListener('fetch', event => {
  console.log(`[Service Worker] Fetch event for: ${event.request.url}`);

  if (event.request.url.includes('supabase.co')) {
    console.log('[Service Worker] Skipping cache for Supabase API call');
  }
});
```

```

return;
)
event.respondWith(
  caches.match(event.request)
    .then(response => {
      if (response) {
        console.log('[Service Worker] Cache hit:', event.request.url);
        return response;
      }
      console.log('[Service Worker] Cache miss, fetching:', event.request.url);
      return fetch(event.request).then(response => {
        if (!response || response.status !== 200 || response.type !== 'basic') {
          console.log('[Service Worker] Invalid response, not caching');
          return response;
        }
        const responseToCache = response.clone();
        caches.open(CACHE_NAME)
          .then(cache => {
            cache.put(event.request, responseToCache);
            console.log('[Service Worker] Cached new response:', event.request.url);
          });
        return response;
      });
    });
);
});

```

The screenshot shows a web browser window with the 'Pocket Password Pal' login page on the left and the Chrome DevTools console on the right.

**Pocket Password Pal Login Page:**

- Header: Pocket Password Pal, Your secure password manager
- Section: Sign In
- Text: Enter your email to test and access your vault
- Form: Email (placeholder: your@email.com), Password (placeholder: your password)
- Buttons: Sign In, Forgot password?

**Chrome DevTools Console:**

- Left sidebar: Application, Storage, Background services, Service workers from other origins.
- Right pane: Network requests, showing a request from 'http://localhost:8081/'.
- Console: Shows logs for the Service Worker, including 'Cache hit' and 'Cache miss' messages.

## Sync Event:

```
// Background sync event - for offline data syncing
self.addEventListener('sync', event => {
  console.log(`[Service Worker] Sync event triggered: ${event.tag}`);
  if (event.tag === 'sync-passwords') {
    event.waitUntil(syncPasswords());
  }
});

// Function to sync passwords from IndexedDB to Supabase when online
async function syncPasswords() {
  console.log('[Service Worker] Syncing passwords from offline storage');
  // Your IndexedDB + Supabase logic goes here
}
```

The image shows a web application interface on the left and a service worker management interface on the right.

**Web Application Interface (Left):**

- Pocket Password Pal**  
Your secure password manager
- Sign In**  
Enter your credentials to access your vault
- Email:** your@email.com
- Password:** (masked with dots)
- Sign In** button
- Don't have an account? Create one

**Service Worker Management Interface (Right):**

- Application:** Manifest, Service workers, Storage
- Storage:** Local storage, Session storage, Extension storage, IndexedDB, Cookies, Private state tokens, Interest groups, Shared storage, Cache storage, Storage buckets
- Background services:** Back/forward cache, Background fetch, Background sync, Bounce tracking mitigations, Notifications, Document handler
- Service workers from other origins:** See all registrations
- Console:** All assistance, Search, Network conditions, Issues
- Console Log:** [Service Worker] Sync event triggered: test-tag-from-devtools