# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 09 |
| Experiment Title. | To implement Service worker events like fetch, sync and push for E-commerce PWA |
| Roll No. | 37 |
| Name | Nihal Nagdev |
| Class | D15B |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

**Aim :**

To implement Service Worker events like fetch, sync, and push for the E-commerce PWA.

**Theory:**

Service workers provide powerful features that allow web applications to operate reliably, even under unreliable network conditions. Key events such as fetch, sync, and push enhance the performance, offline capability, and user engagement of a Progressive Web App (PWA).

1. Fetch Event

The fetch event allows the service worker to intercept network requests made by the PWA. This is commonly used for caching strategies, such as:

- Cache-first: Serve content from cache, then update in the background.
- Network-first: Try to fetch from the network, fall back to cache if offline.

This is useful for delivering product pages, images, or static assets quickly and reliably.

2. Sync Event

The sync event, especially background sync, helps the app manage tasks when connectivity is restored. For instance, if a user submits an order or review while offline, the service worker can save it locally and send it once the connection is re-established.

- Requires registering a sync task using registration.sync.register('tag-name').

3. Push Event

The push event enables the PWA to receive push notifications from a server, even when the app is not open. This is ideal for eCommerce apps to send:

- Order updates
- Promotional offers ● Cart reminders

It requires integration with a push service and permission from the user.

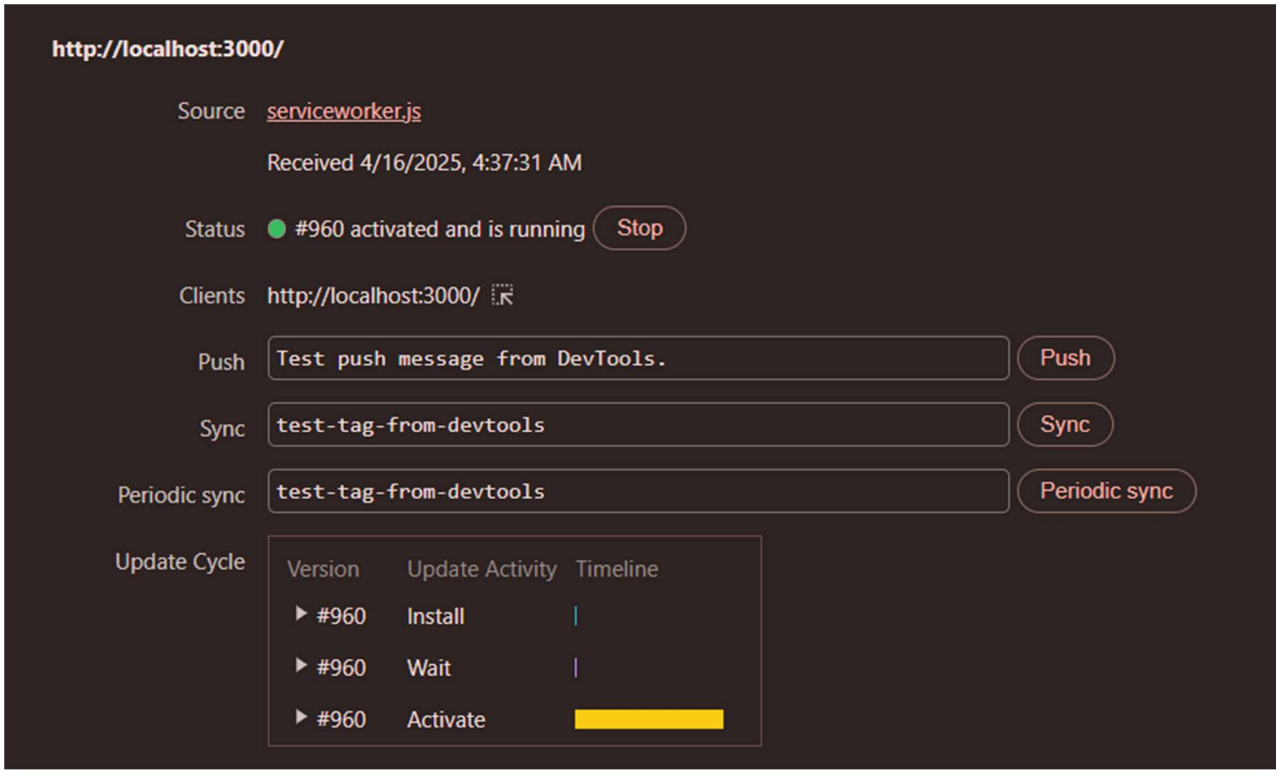
**Code:**

```
self.addEventListener('install', function(event) {
event.waitUntil(     caches.open('flashcard-cache-
v1').then(function(cache) {       return cache.addAll([
       '/',
       '/static/manifest.json',
       '/static/js/script.js',
       '/static/images/icon-128x128.png',
       '/static/images/icon-512x512.png'
     ]);
   })
 );
});

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

self.addEventListener('activate', function(event)
{   const cacheWhitelist = ['flashcard-cache-v1'];
event.waitUntil(
caches.keys().then(function(cacheNames) {
return Promise.all(
cacheNames.map(function(cacheName) {
if (!cacheWhitelist.includes(cacheName)) {
return caches.delete(cacheName);
         }
       })
     );
   })
 );
});

// Background Sync - For syncing offline-created flashcards or
decks self.addEventListener('sync', function(event) {   if (event.tag
```

```
=== 'sync-flashcards') {
event.waitUntil(syncFlashcardsToServer());
  }
});

async function syncFlashcardsToServer() {   const cards = await
getUnsyncedFlashcards(); // Replace with IndexedDB logic   for (const card of
cards) {     try {       const res = await fetch('/api/cards', {         method: 'POST',
body: JSON.stringify(card),         headers: {
          'Content-Type': 'application/json'
        }       });       if (res.ok) {
await markCardAsSynced(card.id);
      }
    } catch (err) {       console.error('Sync failed for
card', card.id, err);
    }
  }
}

// Push Notifications - For reminders or study tips
self.addEventListener('push', function(event) {
const data = event.data ? event.data.json() : {};
const options = {
    body: data.body || 'Time to study your
flashcards!',     icon: '/static/images/icon-
128x128.png',     badge: '/static/images/icon-
128x128.png',     data: {       url: data.url || '/'
    }
  };

  event.waitUntil(     self.registration.showNotification(data.title || 'Flashcard
Reminder', options)
  );
});

self.addEventListener('notificationclick', function(event)
{   event.notification.close();   event.waitUntil(
clients.openWindow(event.notification.data.url)
```

```
  );
});

// Demo IndexedDB stubs — replace with real logic async function
getUnsyncedFlashcards() {   return [{ id: 1, question: 'What is PWA?',
answer: 'Progressive Web App' }]; }

async function markCardAsSynced(id) {   console.log(`✅
Flashcard ${id} marked as synced.`);
}
```

**Screenshots:**




## Conclusion

Implementing fetch, sync, and push events in the service worker of an E-commerce PWA significantly enhances the app's usability, resilience, and user engagement. These features enable the PWA to function offline, perform background tasks, and communicate with users proactively, providing a seamless and efficient shopping experience.