Name: Atif Ansari Roll no: 04 Class: D15B

MPL Experiment - 9

Aim: To implement Service Worker events like Fetch, Sync, and Push for an E-commerce Progressive Web App (PWA).

Theory: A Service Worker is a background script that runs independently in the browser without direct user interaction. It acts as a network proxy, allowing developers to manage caching, track network requests, and enable offline-first web applications using the Cache API.

Key Characteristics of Service Workers:

- Operate independently in the background, enabling offline capabilities.
- Function as programmable network proxies to handle network requests efficiently.
- Require HTTPS for security, preventing man-in-the-middle attacks.
- Do not maintain a global state and rely on IndexedDB for persistent data storage.
- Use Promises extensively for asynchronous operations.

Service Worker Events:

1. Fetch Event:

- Used to track and manage network traffic.
- Implements caching strategies like "Cache First" and "Network first" to optimize performance and offline access.
- Cache First: Returns cached data if available;

otherwise, fetches from the network.

- Network first: Tries fetching from the network, first; if unsuccessful, it retrieves cached data.

2. Sync Event:

- Ensures tasks complete even when the internet is temporarily unavailable.
- Data is stored in IndexedDB and processed when the connection is available.

3. Push Event:

- Handles push notifications sent from a server to the user's device.
- The Notification.requestPermission(); method is used to request permission for displaying notifications.

Conclusion:

By Implementing Service Worker events like fetch, Sync, and Push in an E-commerce PWA improves offline support, reliability, and user engagement. Fetch handles caching for better performance, Sync ~~mess~~ manages tasks when offline, and Push enables real-time notifications. These features make the app faster, more interactive and usable even without internet access.

Code Implementation of Experient 9:

index.html:

```
<section class="sync-section">
      <div class="sync-card">
        <h3><i class="fas fa-sync-alt"></i> Try Background Sync</h3>
        <p class="sync-subtitle">Test how SneakCart stays smart — even offline!</p>
        <form id="dummyForm" class="sync-form">
          <input type="text" placeholder="Enter test data..." id="dummyInput" required />
          <button type="submit" class="sync-button">
            <i class="fas fa-paper-plane"></i> Submit
          </button>
        </form>
        <div id="status" class="sync-status">Waiting for input...</div>
      </div>
    </section>
```

service-worker.js:

```
const CACHE_NAME = "sneakcart-cache-v3";
const urlsToCache = [
  "/",
  "/index.html",
  "/offline.html",
  "/style.css",
  "/script.js",
  "/products/shoe1.png",
  "/products/shoe2.jpg",
  "/icons/icon-192.png",
  "/icons/icon-512.png"
];

// Install Event
self.addEventListener('install', event => {
  console.log('[SW] Install event');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        console.log('[SW] Caching all files');
        return cache.addAll(urlsToCache);
      })
      .then(() => self.skipWaiting())
  );
});
```

```
// Activate Event
self.addEventListener('activate', event => {
  console.log('[SW] Activate event');
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.map(cache => {
          if (cache !== CACHE_NAME) {
            console.log('[SW] Deleting old cache:', cache);
            return caches.delete(cache);
          }
        })
      );
    })
    .then(() => self.clients.claim())
  );
});

// Fetch Event (Cache with Network Fallback)
self.addEventListener('fetch', event => {
  // Skip non-GET requests
  if (event.request.method !== 'GET') return;

  event.respondWith(
    caches.match(event.request)
      .then(cachedResponse => {
        // Return cached response if found
        if (cachedResponse) {
          console.log(`[SW] Serving from cache: ${event.request.url}`);
          return cachedResponse;
        }

        // Otherwise fetch from network
        return fetch(event.request)
          .then(networkResponse => {
            // Cache the new response if successful
            if (networkResponse && networkResponse.status === 200) {
              const responseToCache = networkResponse.clone();
              caches.open(CACHE_NAME)
                .then(cache => cache.put(event.request, responseToCache));
            }
            return networkResponse;
          })
          .catch(() => {
            // If both fail, show offline page for HTML requests
```

```
          if (event.request.headers.get('accept').includes('text/html')) {
            return caches.match('/offline.html');
          }
        });
      })
  );
});

// Sync Event
self.addEventListener('sync', event => {
  if (event.tag === 'sync-form') {
    console.log('[SW] Background sync triggered');
    event.waitUntil(
      (async () => {
        // Simulate sync process
        await new Promise(resolve => setTimeout(resolve, 1500));

        // Get all clients to show sync complete message
        const clients = await self.clients.matchAll();
        clients.forEach(client => {
          client.postMessage({
            type: 'sync-complete',
            data: localStorage.getItem('syncData') || 'No data'
          });
        });

        console.log('[SW] Background sync completed');
      })()
    );
  }
});

// Push Event
self.addEventListener('push', event => {
  const data = event.data ? event.data.json() : {};

  if (data.method === 'pushMessage') {
    const title = data.title || '🛍️ SneakCart Update';
    const options = {
      body: data.message || 'New deals available!',
      icon: '/icons/icon-192.png',
      badge: '/icons/icon-192.png'
    };
```

```
    event.waitUntil(
      self.registration.showNotification(title, options)
    );
  }
});

// Message Event (for communication from page)
self.addEventListener('message', event => {
  if (event.data && event.data.method === 'pushMessage') {
    self.registration.showNotification(
      event.data.title || '🛒 SneakCart',
      {
        body: event.data.message,
        icon: '/icons/icon-192.png'
      }
    );
  }
});
```

Script.js:

```
// Service Worker Registration
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('service-worker.js')
      .then(reg => {
        console.log('✅ Service Worker registered!', reg.scope);

        // Set up sync after SW registration
        setupBackgroundSync(reg);
        setupPushDemo();
      })
      .catch(err => {
        console.error('❌ Service Worker registration failed:', err);
      });
  });
}

// Background Sync Setup
function setupBackgroundSync(swReg) {
  const form = document.getElementById('dummyForm');
  if (!form) return;

  form.addEventListener('submit', function(e) {
```

```
    e.preventDefault();
    const input = document.getElementById('dummyInput');
    const status = document.getElementById('status');

    // Store data in localStorage for demo
    localStorage.setItem('syncData', input.value);

    // Register sync
    swReg.sync.register('sync-form')
      .then(() => {
        status.textContent = '🔄 Sync registered! Will complete when online.';
        status.style.color = 'var(--primary)';
        input.value = '';
      })
      .catch(err => {
        status.textContent = '❌ Sync registration failed';
        status.style.color = 'var(--danger)';
        console.error('Sync registration failed:', err);
      });
  });
}

// Push Notification Demo
function setupPushDemo() {
  const pushButton = document.createElement('button');
  pushButton.className = 'sync-button';
  pushButton.innerHTML = '<i class="fas fa-bell"></i> Test Push Notification';
  pushButton.onclick = triggerTestPush;

  const syncSection = document.querySelector('.sync-section');
  if (syncSection) {
    syncSection.appendChild(pushButton);
  }
}

function triggerTestPush() {
  navigator.serviceWorker.ready
    .then(reg => {
      reg.active.postMessage({
        method: "pushMessage",
        message: "🎉 New deal on Sneakers! 50% OFF today only!"
      });
      alert('Test push notification sent! Check your notifications.');
    })
```

```
  .catch(err => {
    console.error('Push test failed:', err);
  });
}
```

Output: