Assignment No.: 09
Title: Implementation of Fetch, Sync, and Push Events using Service Workers in E-commerce PWA
Name: Harshit Raheja
Class: D15B
Roll Number: 45

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

Theory:
A Service Worker is a background script that works independently of the main web page, enabling features like offline access, background synchronization, and push notifications. Acting as a programmable network proxy, it enhances the reliability and performance of Progressive Web Apps (PWAs).

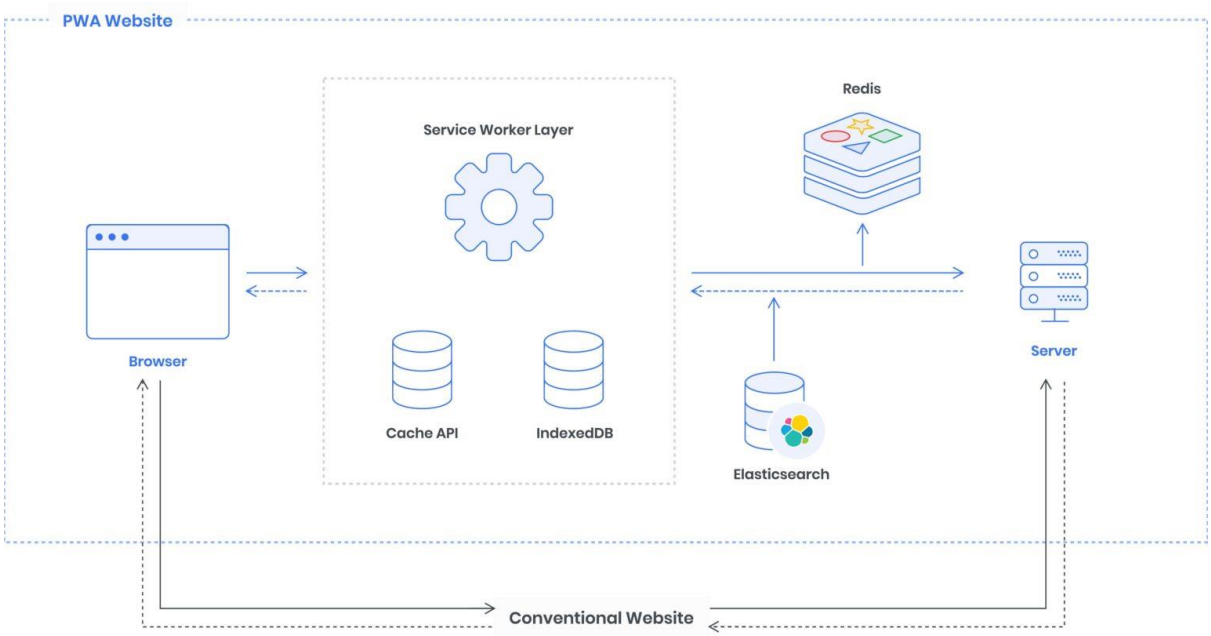
Key Characteristics of Service Workers:

- Only function over HTTPS or localhost (for development)
- Use Promises extensively for async operations
- Do not have access to the DOM (communicate via postMessage)
- Become idle when not in use and restart when needed

Fetch Event:
The fetch event allows the Service Worker to intercept network requests.

- **Cache First Strategy:** If a cached response is available, return it. Otherwise, fetch from the network.
- **Network First Strategy:** Try to fetch from the network first; if it fails, fall back to the cache.

Sync Event (Background Sync):
Enables delayed tasks to run when connectivity is restored.



Example:

1. Data (e.g., unsent emails) is saved to IndexedDB when offline.
2. Service Worker registers a sync event.
3. When the device goes online, the SW sends the stored data.

Push Event:
Allows the app to receive push notifications even when it's not active.

- Requires user permission via Notification.requestPermission()
- Triggered using the push event in the service worker
- Displays notifications using self.registration.showNotification()

Code (sw.js):

```
var filesToCache = ['/', '/menu', '/contactUs', '/offline.html'];
var preLoad = function () {
  return caches.open("offline").then(function (cache) {
    return cache.addAll(filesToCache);
  });
};
var checkResponse = function (request) {
  return new Promise(function (fulfill, reject) {
    fetch(request).then(function (response) {
```

```
        if (response.status !== 404) {
          fulfill(response);
        } else {
          reject();
        }
      }, reject);
    });
};
self.addEventListener('fetch', function (event) {
  event.respondWith(
    checkResponse(event.request).catch(function () {
      return caches.match(event.request);
    })
  );
});
self.addEventListener('sync', function (event) {
  if (event.tag === 'sync-data') {
    event.waitUntil(
      // Logic to retrieve and send data from IndexedDB
      console.log("Sync successful!")
    );
  }
});
self.addEventListener('push', function (event) {
  if (event && event.data) {
    var data = event.data.json();
    if (data.method === "pushMessage") {
      event.waitUntil(
        self.registration.showNotification("Omkar Sweets Corner", {
          body: data.message
        })
      );
    }
  }
});
```

Output:

- **Fetch:** Intercepts and responds with cached or network data
- **Sync:** Executes background tasks when network returns
- **Push:** Displays push notifications from the server