Name : Vivek Gupta
Div : D15B
Roll No : 19

MPL Practical 09

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

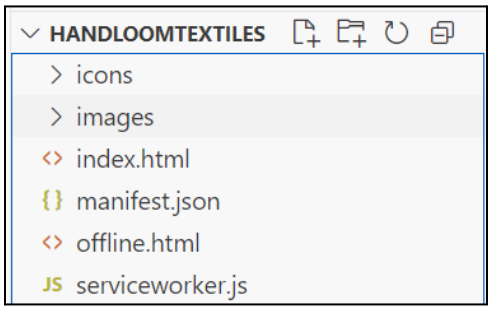
Theory:
Service workers are powerful scripts that run in the background of Progressive Web Apps (PWAs), enabling features like offline access, background sync, and push notifications. In this experiment, we focused on three key service worker events: fetch, sync, and push.

1. Fetch Event
The fetch event allows the service worker to intercept network requests. In our code, we used it to serve cached files when the network is unavailable, helping our E-commerce website work offline. This improves performance and reliability by loading essential resources from cache.

2. Sync Event
The sync event enables background data synchronization when the user regains internet connectivity. In our implementation, we registered a sync task named 'sync-data'. Once the device is back online, the service worker gets triggered and performs background tasks like logging or syncing cart data—ensuring smooth user experience even after going offline.

3. Push Event
The push event is used to receive and display push notifications sent from the server. In our code, we handled this event by showing a browser notification with custom text. We also included error handling to support both JSON and plain text messages.

Folder Structure:



index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- Theme Color -->
  <meta name="theme-color" content="#9e5c39" />
  <title>Maharashtrian Handloom & Textiles</title>
```

```
  <style>

  </style>
  <!-- Web Manifest -->
  <link rel="manifest" href="manifest.json" />
 </head>
 <body>

  <script>
    if ("serviceWorker" in navigator) {
     window.addEventListener("load", () => {
      navigator.serviceWorker
       .register("/serviceworker.js")
       .then((registration) => {
        console.log("✅ Service Worker registered! Scope:", registration.scope);

        // 🔔 Request Push Notification Permission
        if ("PushManager" in window) {
         Notification.requestPermission().then((permission) => {
          if (permission === "granted") {
           console.log("🔔 Push notifications granted.");
          } else {
           console.log("🔕 Push notifications denied.");
          }
         });
        }

        // 🔄 Register Background Sync
        if ("SyncManager" in window) {
         navigator.serviceWorker.ready.then((swReg) => {
          swReg.sync.register("sync-data").then(() => {
           console.log("🔄 Sync registered");
          }).catch((err) => {
           console.log("❌ Sync registration failed:", err);
          });
         });
        }
       })
       .catch((error) => {
        console.log("❌ Service Worker registration failed:", error);
       });
     });
    }
   </script>
 </body>
</html>
```

offline.html

```
<!-- offline.html -->
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>Offline</title>
 <style>
  body {
   font-family: Arial;
   text-align: center;
   padding: 50px;
   background: #f5f5f5;
  }
 </style>
</head>
<body>
 <h2>You are offline</h2>
 <p>Please check your internet connection and try again.</p>
</body>
</html>
```

serviceworker.js

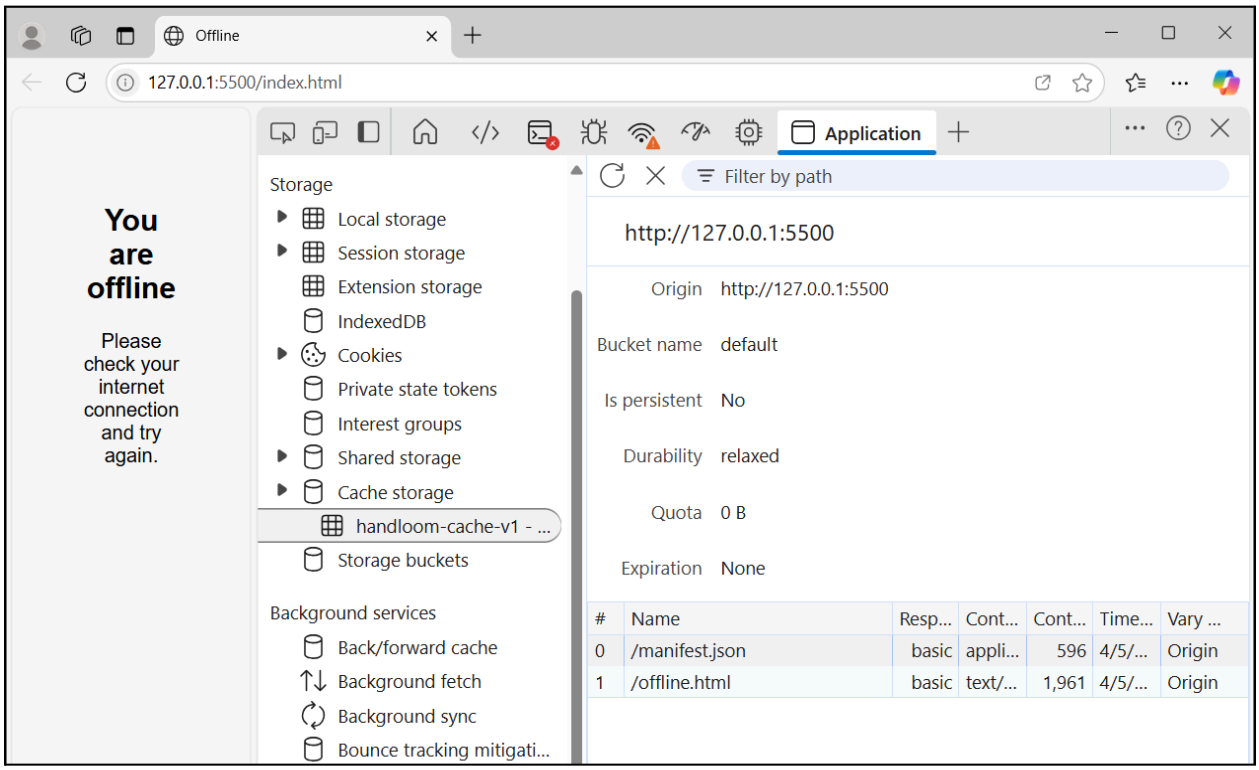
```
const CACHE_NAME = "handloom-cache-v1";
const FILES_TO_CACHE = [
 "/",
 "index.html",
 "manifest.json",
 "icons/icon-192x192.png",
 "icons/icon-512x512.png",
 "images/Artisan Village.webp",
 "images/hero.webp",
 "images/Himroo Shawl.webp",
 "images/Karvati Saree.webp",
 "images/Kolhapuri Chappals.webp",
 "images/Mashru Silk Fabric.webp",
 "images/Narali Patali Fabric.webp",
 "images/Paithani Saree.webp",
 "images/place.webp",
 "images/place1.webp",
 "images/place2.webp",
 "images/placeholder.webp",
 "images/placeholder1.webp",
 "images/placeholder2.webp",
 "images/Traditional Craft.webp",
 "images/Weaving Workshop.webp",
 "offline.html",
```

```
];

// Install Event
self.addEventListener("install", (event) => {
 console.log("[ServiceWorker] Install");
 event.waitUntil(
  caches.open(CACHE_NAME).then((cache) => {
   console.log("[ServiceWorker] Caching files");
   return cache.addAll(FILES_TO_CACHE);
  })
 );
});

// Activate Event
self.addEventListener("activate", (event) => {
 console.log("[ServiceWorker] Activate");
 event.waitUntil(
  caches.keys().then((keyList) =>
   Promise.all(
    keyList.map((key) => {
     if (key !== CACHE_NAME) {
      console.log("[ServiceWorker] Removing old cache", key);
      return caches.delete(key);
     }
    })
   )
  )
 );
 return self.clients.claim();
});

// Enhanced Fetch Event
self.addEventListener("fetch", (event) => {
 console.log("[ServiceWorker] Fetch", event.request.url);
 const requestURL = new URL(event.request.url);

 // If request is same-origin, use Cache First
 if (requestURL.origin === location.origin) {
  event.respondWith(
   caches.match(event.request).then((cachedResponse) => {
    return ( cachedResponse ||
     fetch(event.request).catch(() => caches.match("offline.html"))
    );
   })
  );
 } else {
  // Else, use Network First
  event.respondWith(
```

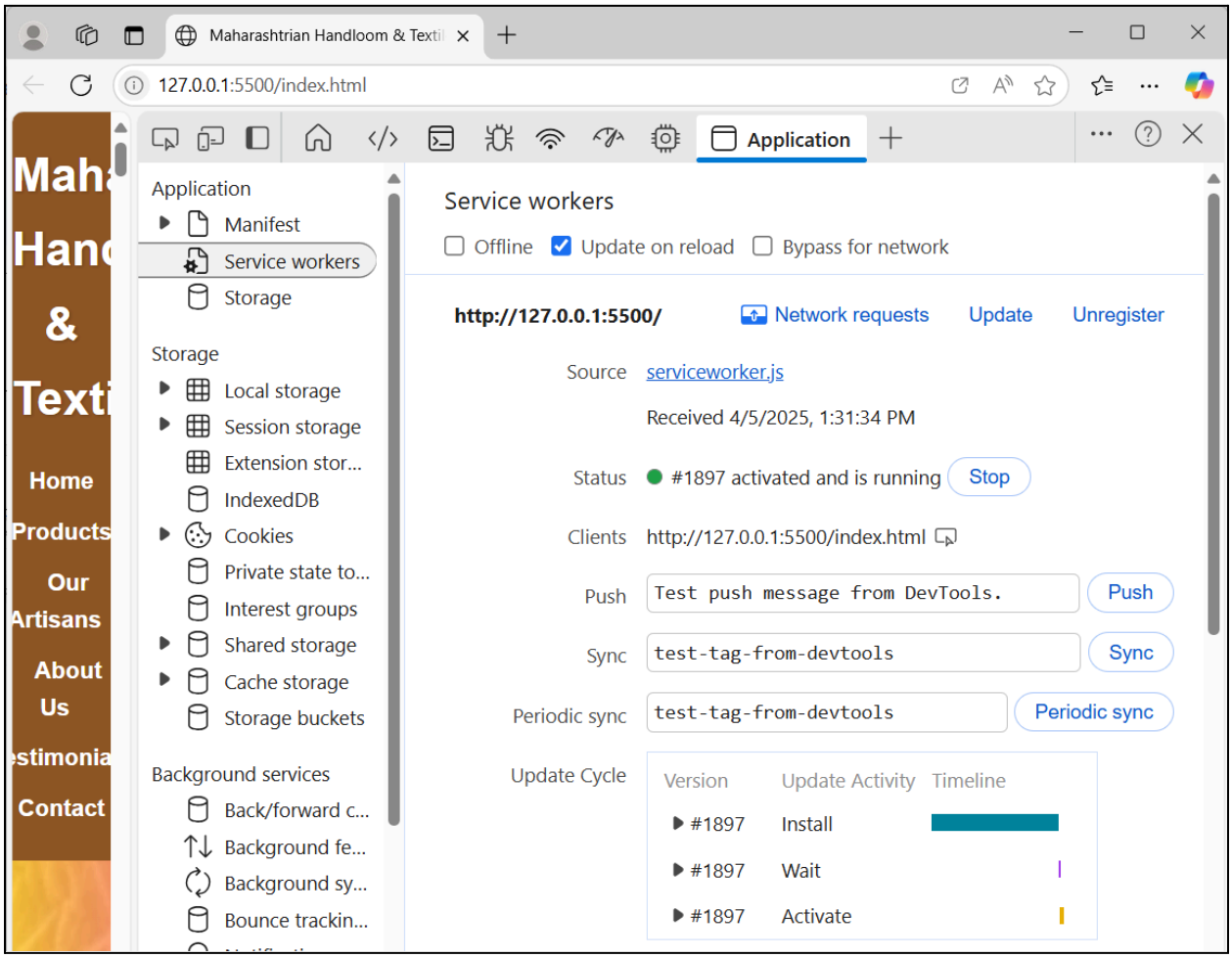
```
    fetch(event.request)
      .then((response) => {
        return response;
      })
      .catch(() =>
        caches.match(event.request).then((res) => {
          return res || caches.match("offline.html");
        })
      )
  );
 }
});

// Sync Event (simulation)
self.addEventListener("sync", (event) => {
 if (event.tag === "sync-data") {
  event.waitUntil(
    (async () => {
      console.log("Sync event triggered: 'sync-data'");
      // Here you can sync data with server when online
    })()
  );
 }
});

// Push Event
self.addEventListener("push", function (event) {
 if (event && event.data) {
  let data = {};
  try {
    data = event.data.json();
  } catch (e) {
    data = {
      method: "pushMessage",
      message: event.data.text(),
    };
  }

  if (data.method === "pushMessage") {
    console.log("Push notification sent");
    event.waitUntil(
      self.registration.showNotification("Maharashtrian Handloom", {
        body: data.message,
      })
    );
  }
 }
});
```

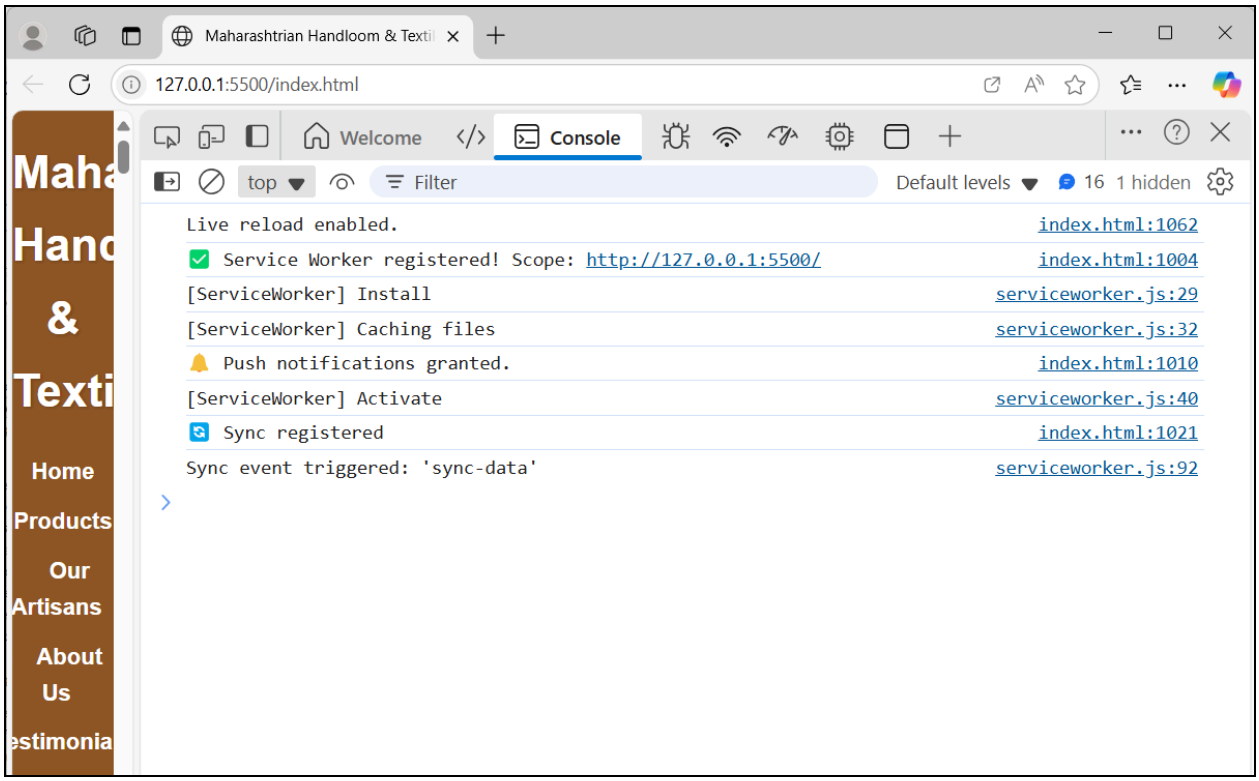
Screenshot

1. Test: Fetch Event (Offline Support)

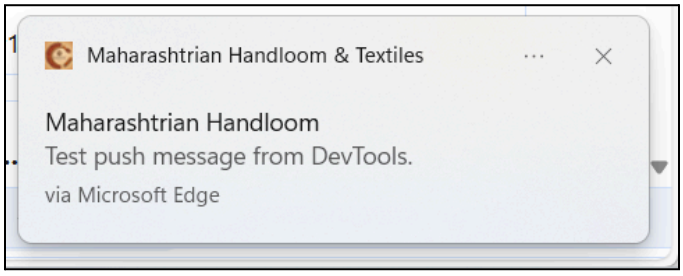


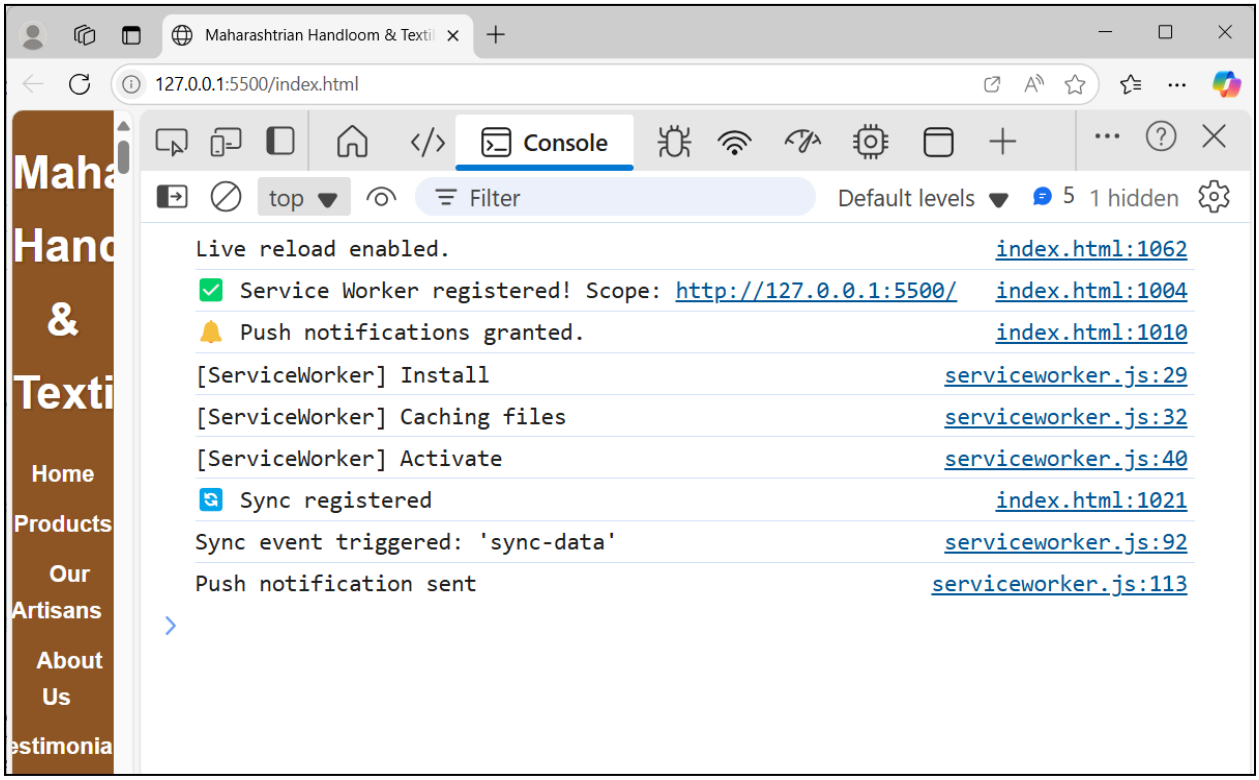
2. Test: Background Sync Event





3. Test: Push Notification Event





Conclusion

In this experiment, we successfully implemented fetch, sync, and push events in the service worker for our *Maharashtrian Handloom & Textiles* PWA. While testing push notifications, we initially faced a JSON parsing error, but we resolved it by updating the code to handle both JSON and plain text payloads properly.