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

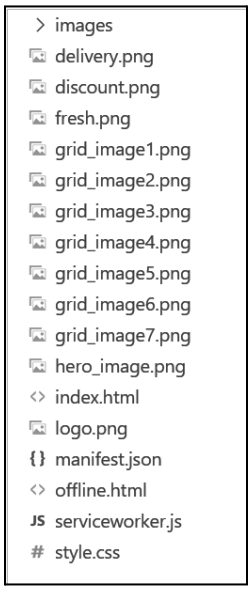
Theory:

In Progressive Web Apps (PWAs), Service Workers play a major role in enabling offline capabilities, background data synchronization, and push notifications. The three key events implemented in this experiment are fetch, sync, and push.

- The fetch event is triggered whenever a network request is made. In our Restaurant PWA, we used this to serve cached files when the user is offline and to ensure faster load times using a Cache First strategy for same-origin requests and Network First for others.
- The sync event allows background synchronization when the user comes back online. We registered a sync event (sync-data) which, in a real-world scenario, could be used to sync orders or updates with a server once connectivity is restored.
- The push event is used for push notifications. In our implementation, we requested permission for notifications and displayed a custom message when a push event is triggered, simulating a real-time alert or update from the server.

This experiment helped us understand how service workers enhance the user experience in PWAs by enabling offline support, background tasks, and real-time communication—all essential for modern E-commerce applications.

Folder Structure:



Code:

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <meta name="theme-color" content="#007BFF" />
    <title>Restaurant Website</title>
    <link rel="stylesheet" href="style.css" />
    <link rel="manifest" href="manifest.json" />
    <script
      src="https://kit.fontawesome.com/7a4b62b0a4.js"
      crossorigin="anonymous"
    ></script>
  </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>
```

serviceworker.js

```
const CACHE_NAME = "cooking-cache-v2";
const FILES_TO_CACHE = [
  "/index.html",
  "/style.css",
  "/manifest.json",
  "/offline.html",
  "/images/app.png",
  "/images/big.png",
  "delivery.png",
  "discount.png",
  "fresh.png",
  "grid_image1.png",
  "grid_image2.png",
  "grid_image3.png",
  "grid_image4.png",
  "grid_image5.png",
  "grid_image6.png",
  "grid_image7.png",
  "hero_image.png",
  "logo.png"
];


// 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(
```

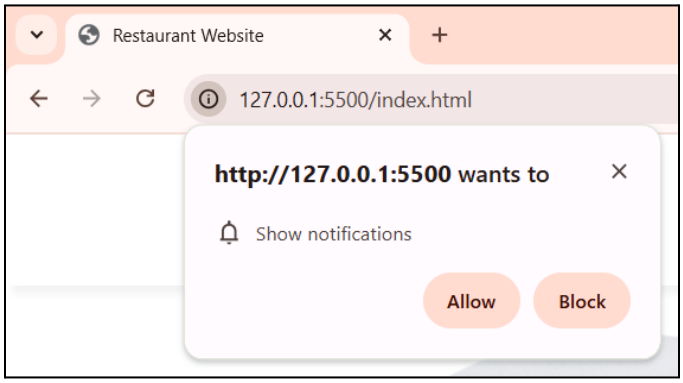
```
        fetch(event.request)
          .then((response) => {
            return response;
          })
          .catch(() =>
            caches.match(event.request).then((res) => {
              return res || caches.match("offline.html");
            })
          )
      );
    }
  });

// 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("Restaurant", {
```

```
            body: data.message,
          })
        );
      }
    }
  });
```

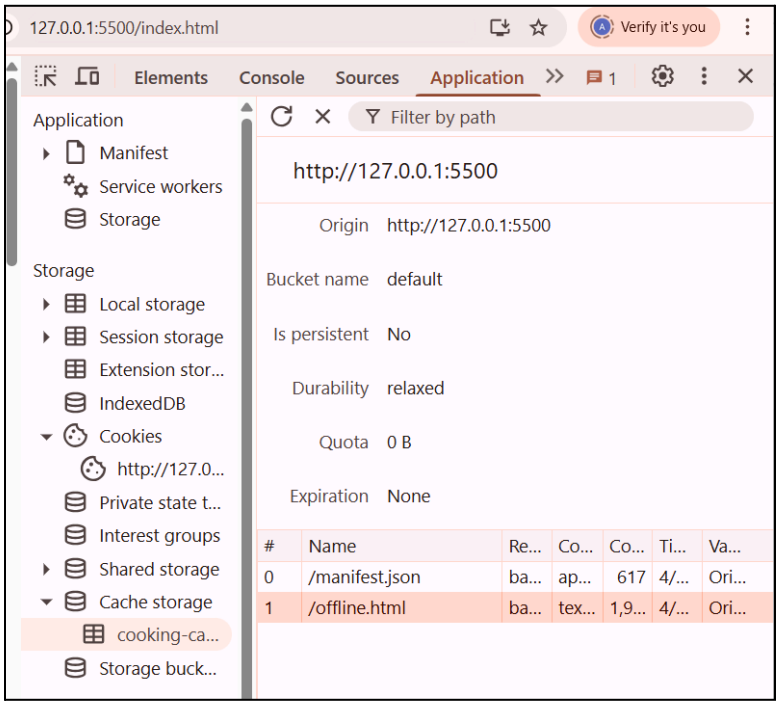
offline.html

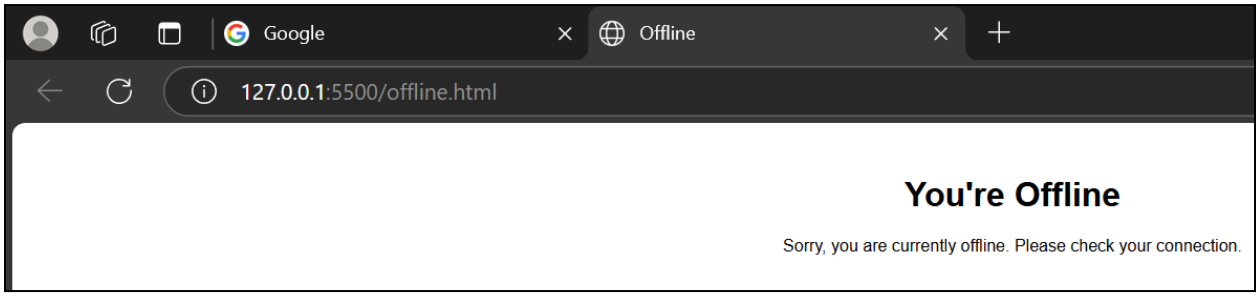
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Offline</title>
    <style>
        body { font-family: Arial, sans-serif; text-align: center; margin:
50px; }
    </style>
</head>
<body>
    <h1>You're Offline</h1>
    <p>Sorry, you are currently offline. Please check your connection.</p>
</body>
</html>
```

Screenshot:

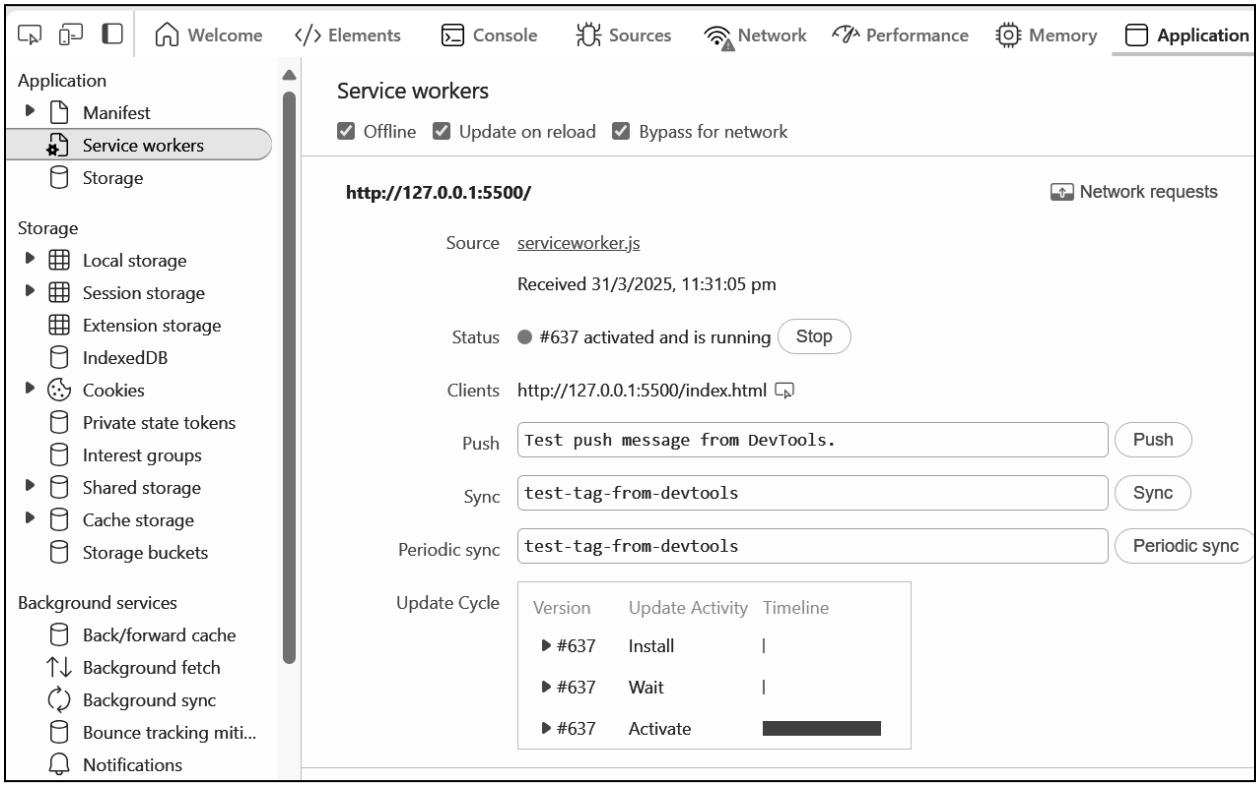


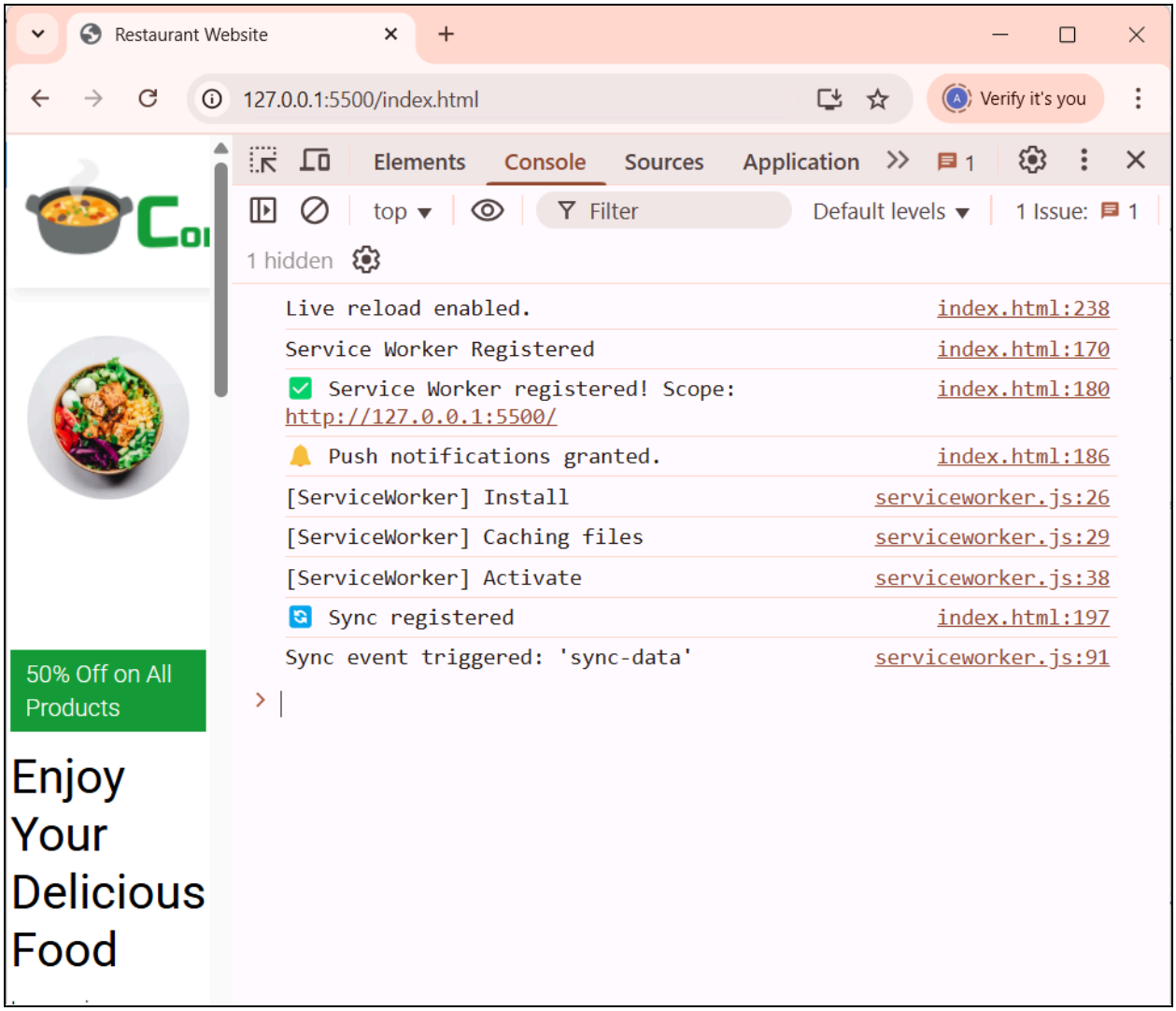
Fetch


127.0.0.1:5500/index.html
Verify it's you
Elements
Console
Sources
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension stor...
IndexedDB
Cookies
http://127.0...
Private state t...
Interest groups
Shared storage
Cache storage
cooking-ca...
Storage buck...
Filter by path
http://127.0.0.1:5500
Origin http://127.0.0.1:5500
Bucket name default
Is persistent No
Durability relaxed
Quota 0 B
Expiration None
Name Re... Co... Co... Ti... Va...
0 /manifest.json ba... ap... 617 4/... Ori...
1 /offline.html ba... tex... 1,9... 4/... Ori...

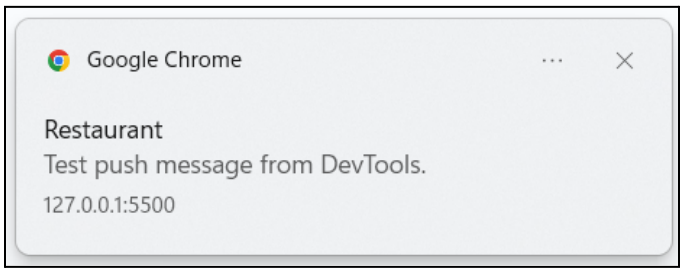

Google
Offline
127.0.0.1:5500/offline.html
You're Offline
Sorry, you are currently offline. Please check your connection.

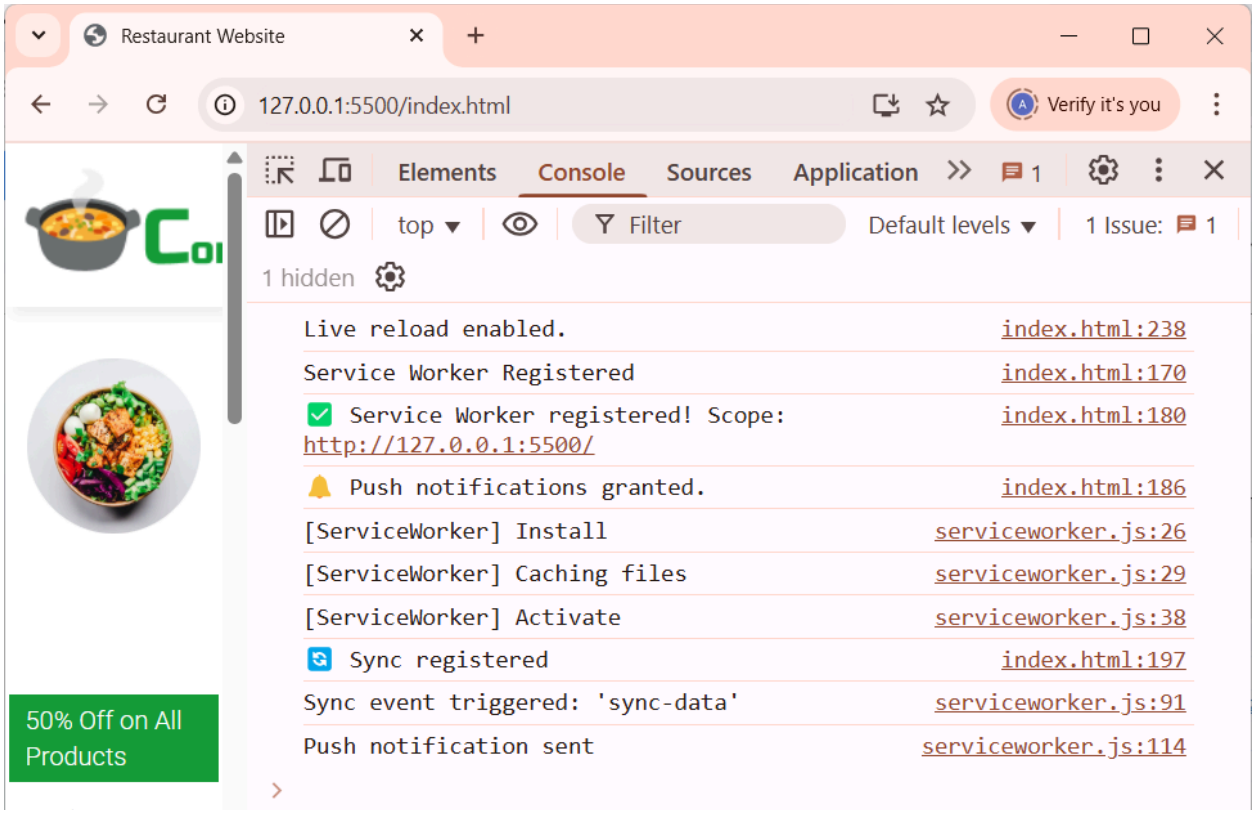
Sync


Welcome
Elements
Console
Sources
Network
Performance
Memory
Application
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 miti...
Notifications
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Source serviceworker.js
Received 31/3/2025, 11:31:05 pm
Status #637 activated and is running Stop
Clients http://127.0.0.1:5500/index.html
Push Test push message from DevTools. Push
Sync test-tag-from-devtools Sync
Periodic sync test-tag-from-devtools Periodic sync
Update Cycle
Version Update Activity Timeline
#637 Install
#637 Wait
#637 Activate


Restaurant Website
127.0.0.1:5500/index.html
Verify it's you
Elements
Console
Sources
Application
top
Filter
Default levels
1 Issue: 1
1 hidden
Live reload enabled. index.html:238
Service Worker Registered index.html:170
Service Worker registered! Scope: http://127.0.0.1:5500/ index.html:180
Push notifications granted. index.html:186
[ServiceWorker] Install serviceworker.js:26
[ServiceWorker] Caching files serviceworker.js:29
[ServiceWorker] Activate serviceworker.js:38
Sync registered index.html:197
Sync event triggered: 'sync-data' serviceworker.js:91
50% Off on All Products
Enjoy Your Delicious Food

Push


Google Chrome
Restaurant
Test push message from DevTools.
127.0.0.1:5500



Conclusion

In this experiment, we successfully implemented fetch, sync, and push events in the service worker to enable offline access, background sync, and push notifications for our Restaurant PWA. Initially, we faced issues with incorrect file paths and sync event registration, but we resolved them by carefully checking cache file references and ensuring the service worker was ready before registering sync.