Practical 9

Name: Yash Rahate
Class: D15B
Roll No.: 44

Aim:

To implement Service worker events like fetch, sync and push for Netflix clone 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.

In this experiment, we enhanced our Netflix clone Progressive Web App (PWA) by implementing advanced Service Worker events — **fetch**, **sync**, and **push**.

- The **install** and **activate** events handle caching of files and removal of old caches.
- The **fetch** event intercepts network requests and serves cached resources for better performance and offline support.
- The **sync** event simulates background synchronization when the app regains connectivity.
- The **push** event listens for push notifications and displays them, improving user engagement.

These events help in making PWAs more reliable, fast, and interactive even in low or no network conditions.

Code:

Index.html

```
<script>
  if ('serviceWorker' in navigator && 'SyncManager' in window) {
    window.addEventListener('load', () => {
      navigator.serviceWorker.register('serviceworker.js')
        .then(reg => {
          console.log("✅ Service Worker registered:", reg.scope);

          // Optional: Register a sync event
          reg.sync.register('sync-tag').then(() => {
            console.log('🔁 Sync registered');
          }).catch(err => {
            console.warn('❌ Sync registration failed:', err);
```

```
        });

      }).catch(err => {
        console.error("❌ Service Worker registration failed:", err);
      });
  });

  // Request notification permission
  Notification.requestPermission().then(permission => {
    if (permission === "granted") {
      console.log("🔔 Notifications allowed");
    }
  });

} else {
  console.warn("⚠️ Service Worker or Background Sync not supported.");
}
</script>
```

Serviceworker.js

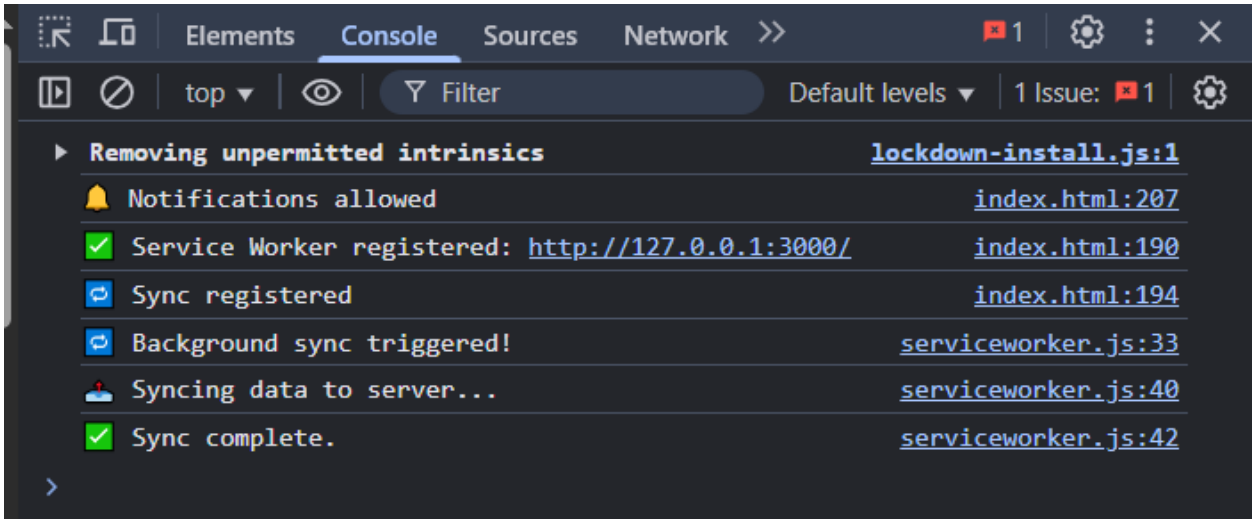
```
const CACHE_NAME = "netflix-clone-cache-v2";
const urlsToCache = [
  "index.html",
  "style.css",
  "app.js",
  "icons/icon-192x192.png",
  "icons/icon-512x512.png"
];

// Install event: cache assets
self.addEventListener("install", (event) => {
  console.log("📦 Installing service worker...");
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => cache.addAll(urlsToCache))
      .catch((err) => console.error("❌ Cache add failed", err))
  );
```

```
});

// Fetch event: serve from cache or network
self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request)
      .then((response) => {
        return response || fetch(event.request);
      })
  );
});

// Sync event: background sync logic (mocked)
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-tag") {
    console.log("🔁 Background sync triggered!");
    event.waitUntil(syncDataWithServer());
  }
});

async function syncDataWithServer() {
  // Simulate sending queued data to server
  console.log("📤 Syncing data to server...");
  await new Promise(resolve => setTimeout(resolve, 1000));
  console.log("✅ Sync complete.");
}

// Push event: show notification
self.addEventListener("push", (event) => {
  const data = event.data?.text() || "Default Push Message";
  const options = {
    body: data,
    icon: "icons/icon-192x192.png",
    badge: "icons/icon-192x192.png"
  };
  event.waitUntil(
    self.registration.showNotification("🔥 Push Notification, Welcome the Yash Rahate Netflix ",
options)
```

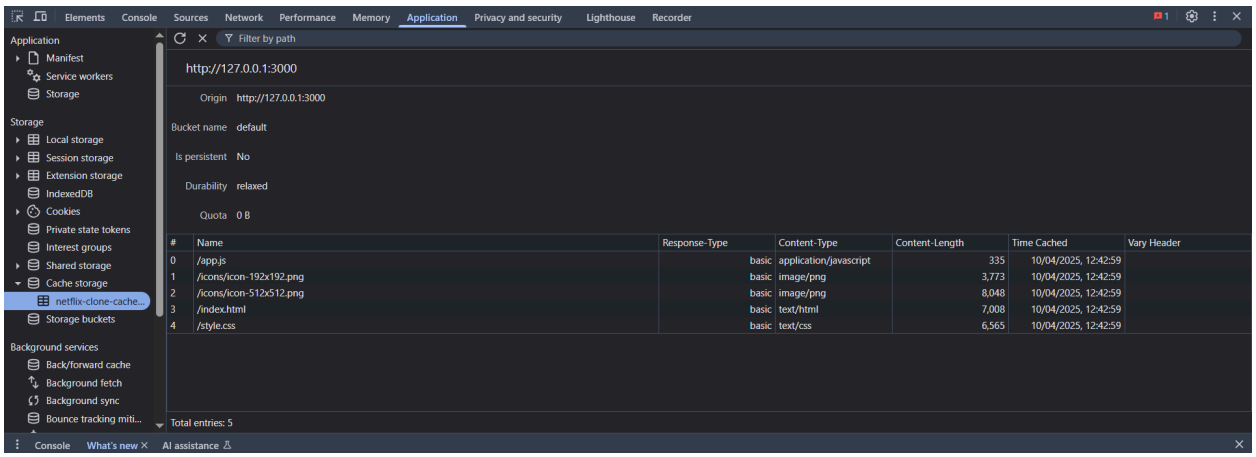
```
  );
});
```

Output:

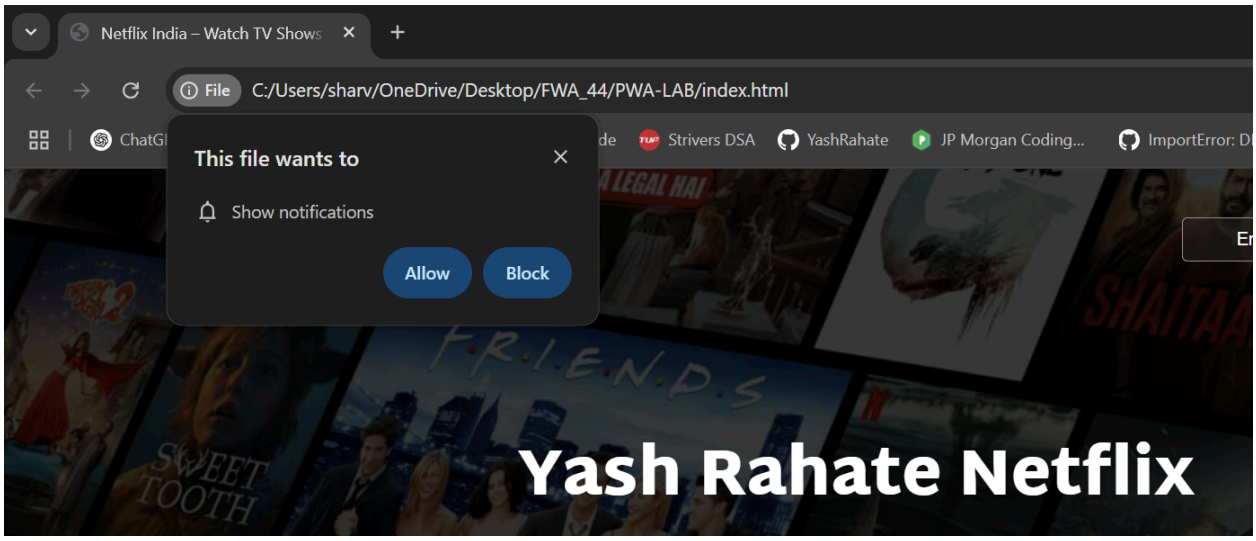
Working of service worker



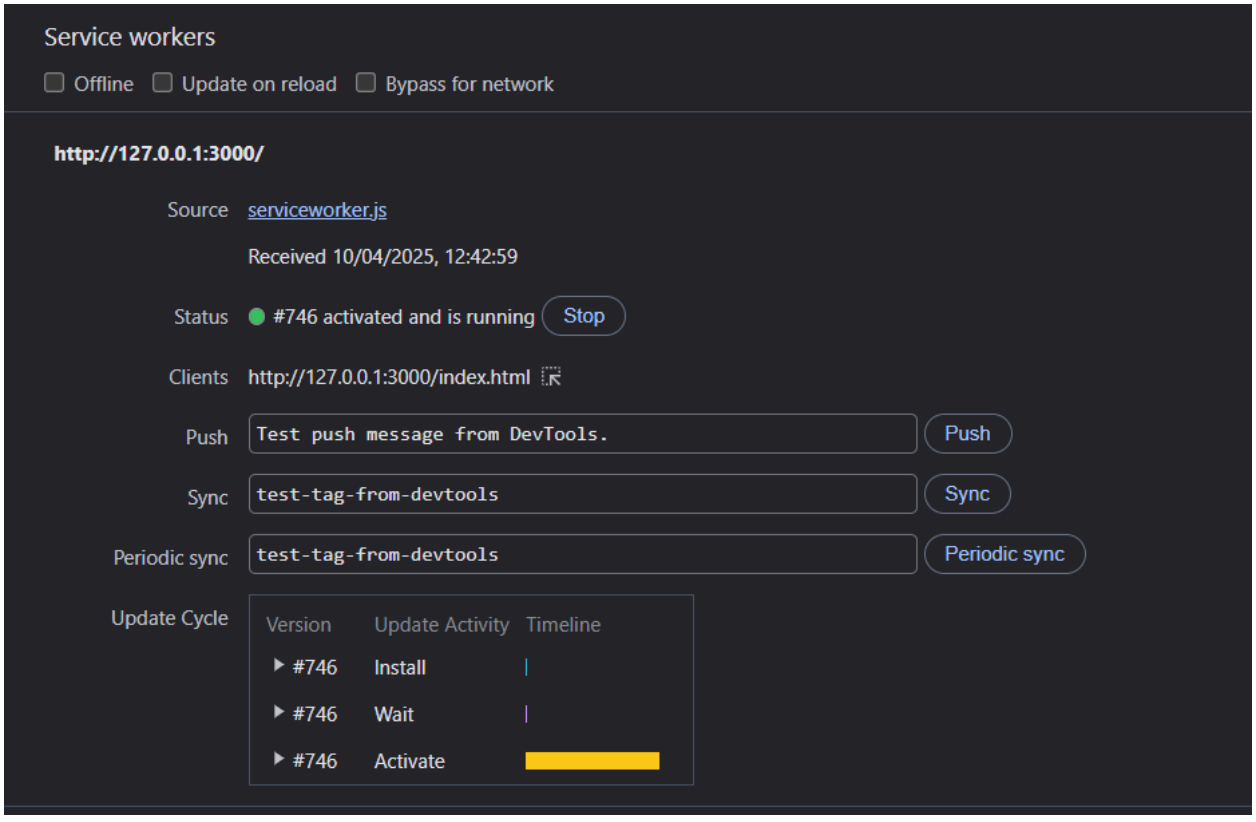
Cache

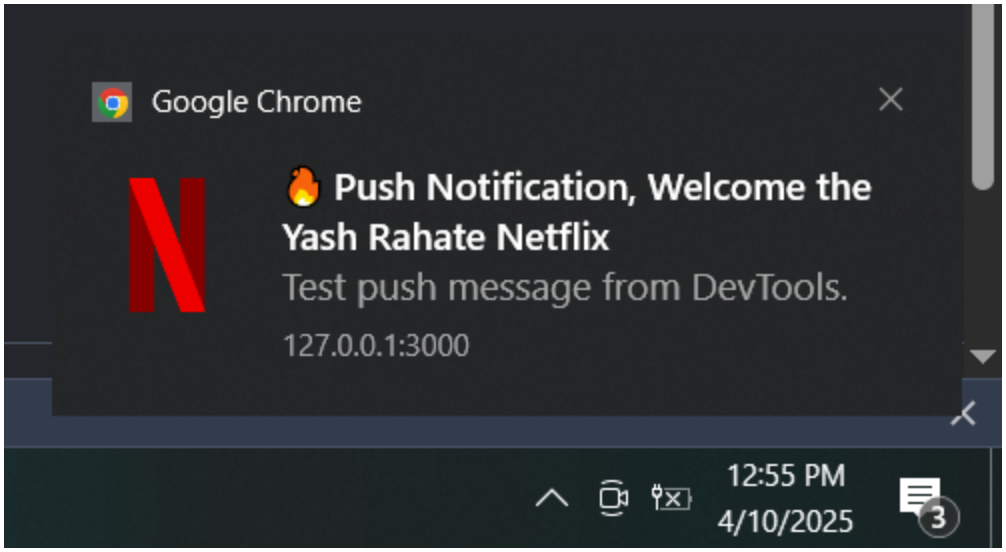


Sending push notification.



Test: Push Notification sent





Conclusion:

By implementing fetch, sync, and push events in the Service Worker, we made our Netflix clone PWA capable of working offline, syncing data in the background, and sending push notifications. This ensures a better user experience and highlights the power of PWAs in building modern web applications.