

Name: Sahil Motiramani
Div: D15C
Roll no: 35

Experiment-9

1. Aim

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

2. Basic Description

Progressive Web Apps (PWAs) leverage service workers to provide app-like experiences on the web. Service workers act as programmable proxies that enable features like offline caching, background data sync, and push notifications.

This experiment focuses on three key events supported by service workers:

1. Fetch Event

- The fetch event is triggered whenever the PWA makes a network request. It allows the service worker to intercept requests and serve cached responses.
- This improves performance and ensures offline functionality by loading resources from the cache if the network is unavailable.

2. Sync Event

- The sync event enables **background synchronization** when the user is offline.
- Using the Background Sync API, we can delay actions (like sending form data or syncing products) until the user is back online.
- This improves reliability in poor or intermittent network conditions.

3. Push Event

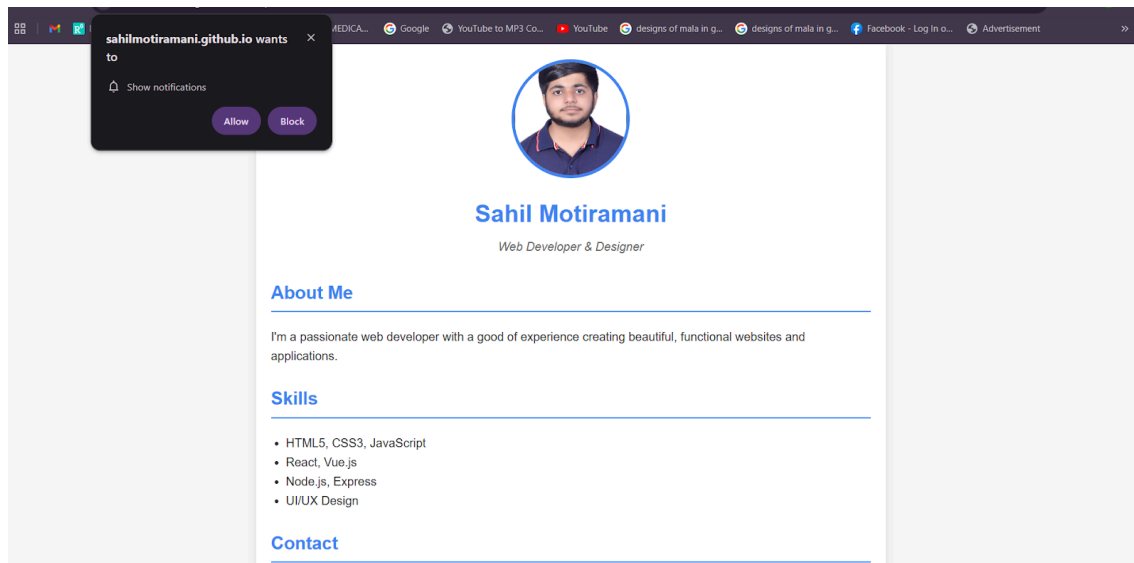
- The push event allows the server to push notifications to the user even when the app is not open.
- It's useful for user engagement through marketing messages, order status updates, or reminders.
- The service worker displays the notification using the `showNotification()` method after receiving data from the push server.

Concepts Covered:

- Role and lifecycle of service workers (install, activate, fetch)
- Intercepting and responding to network requests using cache
- Registering and handling background sync tasks
- Receiving and displaying push notifications
- Importance of user permissions for notifications

Github: <https://github.com/SahilMotiramani/MPL-Profile.git>

3. Output:



4. Conclusion

In this experiment, we successfully implemented three critical service worker events—fetch, sync, and push—to enhance the E-commerce PWA's reliability and engagement. By enabling offline access, background data synchronization, and push messaging, the PWA becomes more user-friendly, robust, and app-like.

This implementation demonstrates how PWAs can provide seamless experiences even in unreliable network conditions, making them a powerful alternative to native mobile apps.