Experiment 9

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

Theory:

1. Network-First Strategy:
   - In the network-first strategy, the application attempts to fetch resources from the network (server) first.
   - If the network request fails or the resource is not available, the application falls back to using a cached version of the resource.
   - This strategy ensures that the application always fetches the latest version of resources from the server whenever possible but gracefully handles offline scenarios by using cached resources.

2. Cache-First Strategy:
   - In the cache-first strategy, the application attempts to retrieve resources from the cache first before making a network request.
   - If the resource is found in the cache, it is returned immediately without making a network request.
   - This strategy is suitable for scenarios where resources are relatively static and can be safely cached for extended periods, reducing the need for frequent network requests.

3. Fetch Event:
   - The `fetch` event is fired whenever the browser makes a network request for a resource, such as an HTML page, CSS file, JavaScript file, image, or API endpoint.
   - Developers can intercept fetch requests using service workers to implement custom caching strategies, handle network errors, or modify responses before they are delivered to the browser.

4. Push Event:
   - The `push` event is fired when a push notification is received by the service worker from a push service.
   - Push notifications allow web applications to deliver real-time updates and alerts to users, even when the browser is not actively in use.
   - When a push event is received, the service worker can process the notification data and display a notification to the user, providing timely information or updates.

5. Sync Event:
   - The `sync` event is triggered when the browser regains connectivity after being offline and is able to synchronize data with the server.
   - Sync events are useful for implementing background synchronization tasks, such as sending queued requests or updating local data with server changes, when the browser is back online.
   - Service workers can listen for sync events and perform synchronization tasks in the background, even when the web application is not actively being used.

Code:
Push event:

```
self.addEventListener('push', function (event) {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method == "pushMessage") {
            console.log("Push notification sent");
            event.waitUntil(
                self.registration.showNotification("My watch", {
                    body: data.message
                })
            );
        }
    }
});
```

Event Listener for Background Sync Registration:

```
document.querySelector("button").addEventListener("click",async(
)=>{
    var swRegistration = await
navigator.serviceWorker.register("sw.js");
    swRegistration.sync.register("helloSync").then(function(){
        console.log("helloSync success[main.js");
    })
})
```

Event Listener for sw.js:

```
self.addEventListener('sync', event => {
    if (event.tag === 'helloSync') {
        console.log("helloSync[sw.js]");
    }
});
```

Fetch:

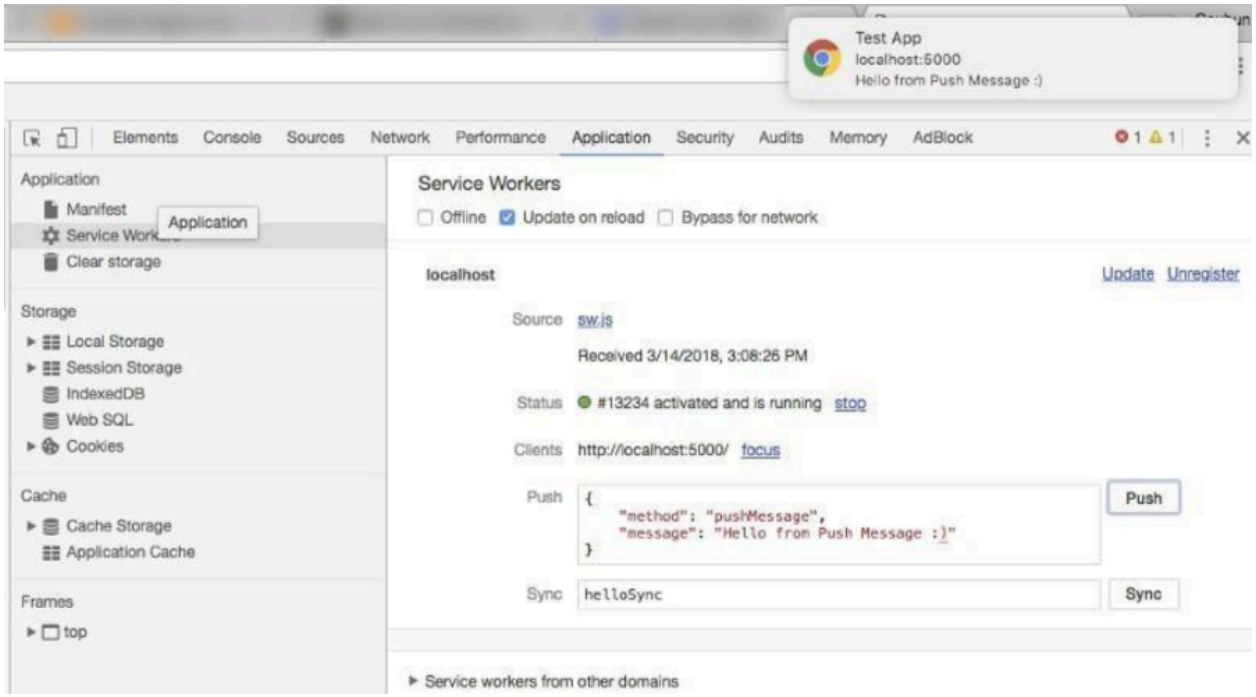
```
self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request)
            .then(response => {
                if (response) {
                    return response;
                }
                return fetch(event.request);
            })
    );
});
```

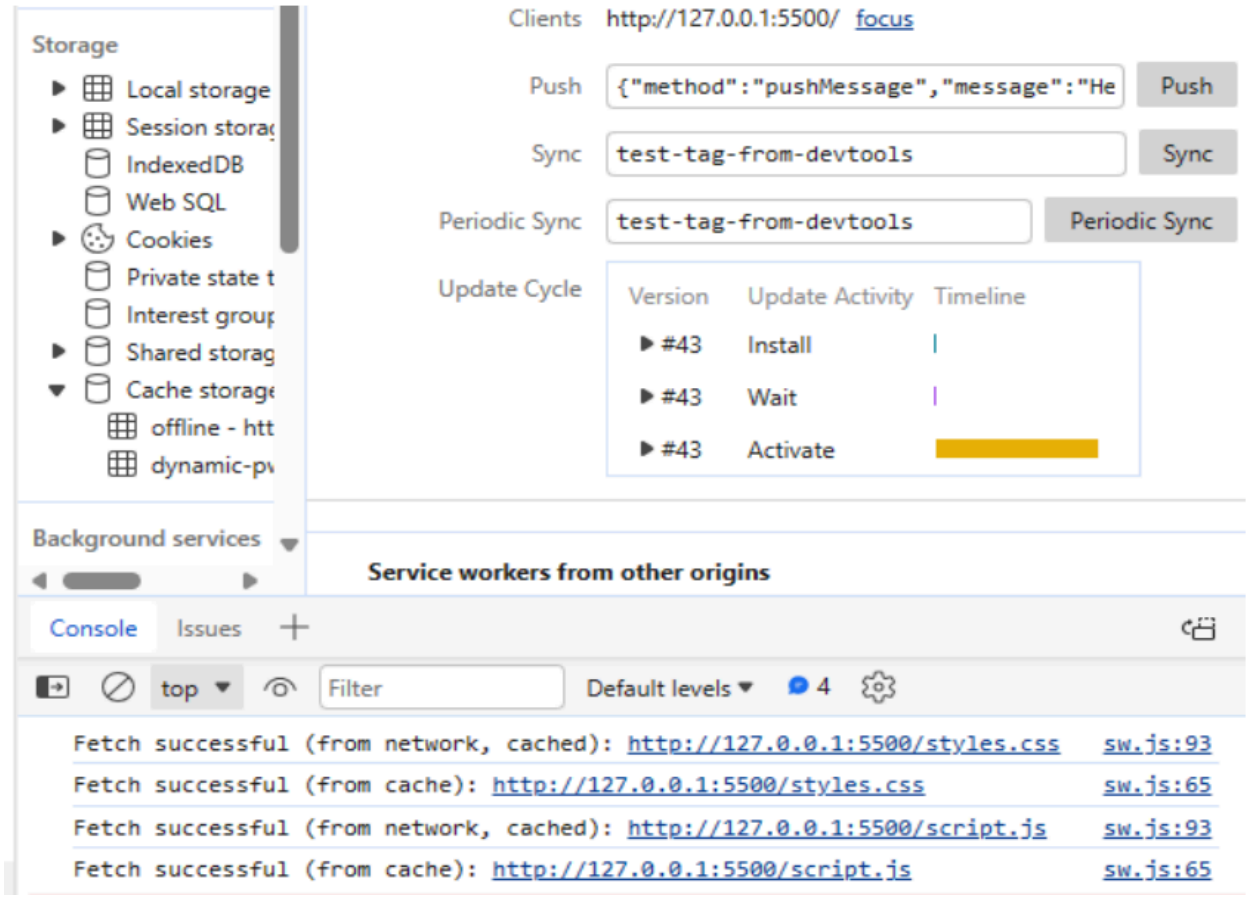
Network First:

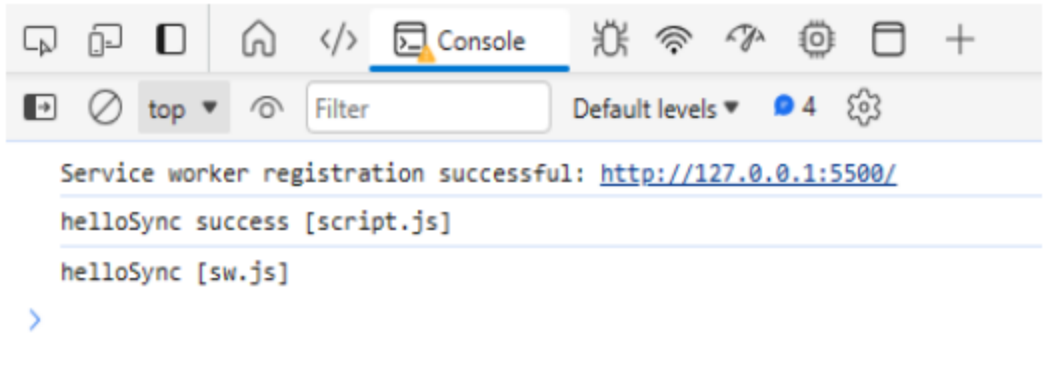
```
async function networkFirst(req){
    const cache = await caches.open("pwa-dynamic");
    try{
        const res = await fetch(req);
        cache.put(req,res.clone());
        return res;
    }
    catch(error){
        const cachedResponse = await.cache.match(req);
        return cachedResponse || await
caches.match("./noconnection.json")
    }
    }
```

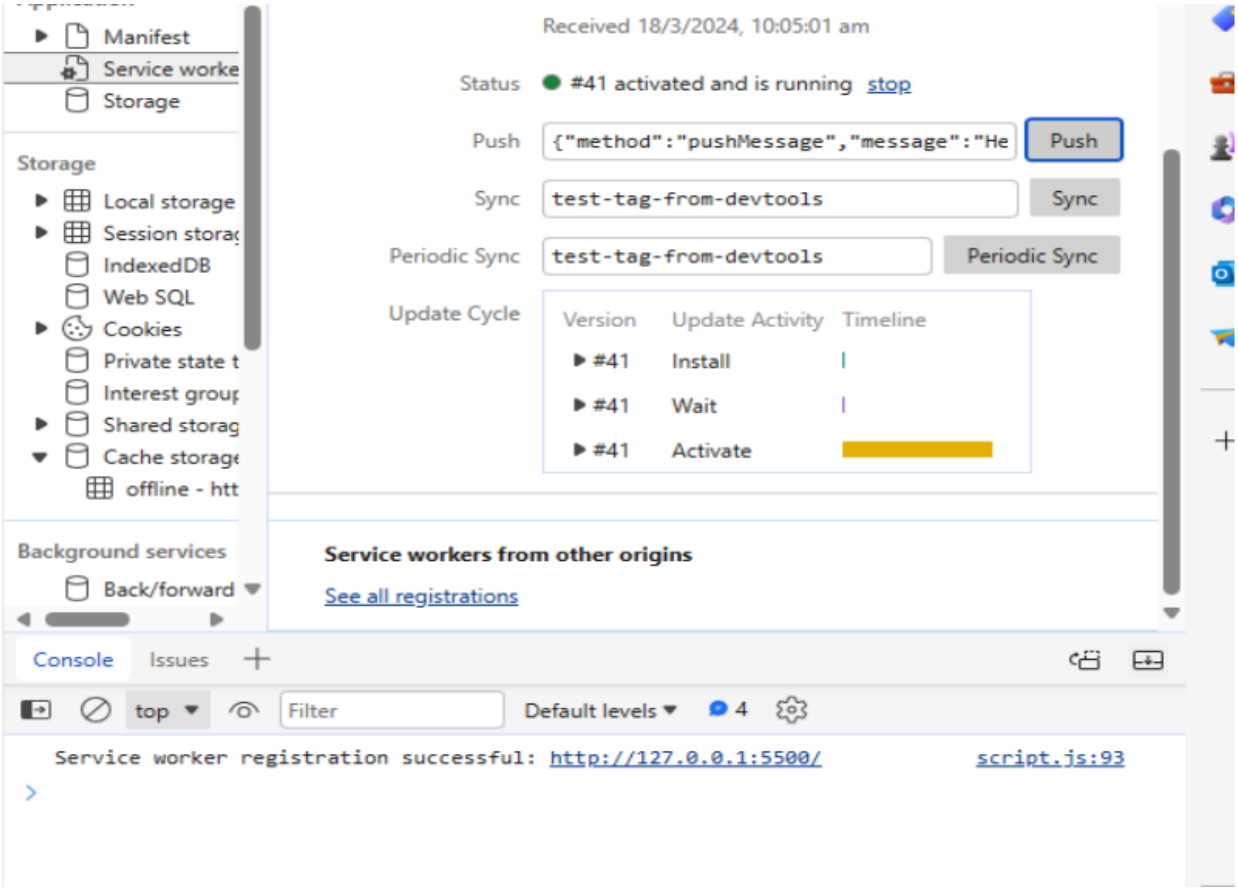
Cache First:

```
async function cacheFirst(req){
    return await caches.match(req) || fetch(req);
}
```


Test App
localhost:5000
Hello from Push Message :)
Elements Console Sources Network Performance Application Security Audits Memory AdBlock
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
Application Cache
Frames
top
Service Workers
Offline Update on reload Bypass for network
localhost
Update Unregister
Source sw.js
Received 3/14/2018, 3:08:26 PM
Status #13234 activated and is running stop
Clients http://localhost:5000/ focus
Push
{
"method": "pushMessage",
"message": "Hello from Push Message :)"
}
Push
Sync helloSync
Sync
Service workers from other domains


Storage
Local storage
Session storag
IndexedDB
Web SQL
Cookies
Private state t
Interest group
Shared storag
Cache storage
offline - htt
dynamic-pv
Background services
Clients http://127.0.0.1:5500/ focus
Push {"method":"pushMessage","message":"He Push
Sync test-tag-from-devtools Sync
Periodic Sync test-tag-from-devtools Periodic Sync
Update Cycle
Version Update Activity Timeline
#43 Install
#43 Wait
#43 Activate
Service workers from other origins
Console Issues
top Filter Default levels 4
Fetch successful (from network, cached): http://127.0.0.1:5500/styles.css sw.js:93
Fetch successful (from cache): http://127.0.0.1:5500/styles.css sw.js:65
Fetch successful (from network, cached): http://127.0.0.1:5500/script.js sw.js:93
Fetch successful (from cache): http://127.0.0.1:5500/script.js sw.js:65





Conclusion : We have understood and successfully implemented events like fetch , push and sync for our ecommerce PWA.