

## Practical 9

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

### Theory:

#### Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications, and develop “offline first” web applications with Cache API.

Things to note about Service Workers:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.
- Service workers make extensive use of promises, so if you're new to promises, then you should stop reading this and check out Promises, an introduction.

#### Fetch Event

You can track and manage page network traffic with this event. You can check existing cache, manage “cache first” and “network first” requests and return a response that you want.

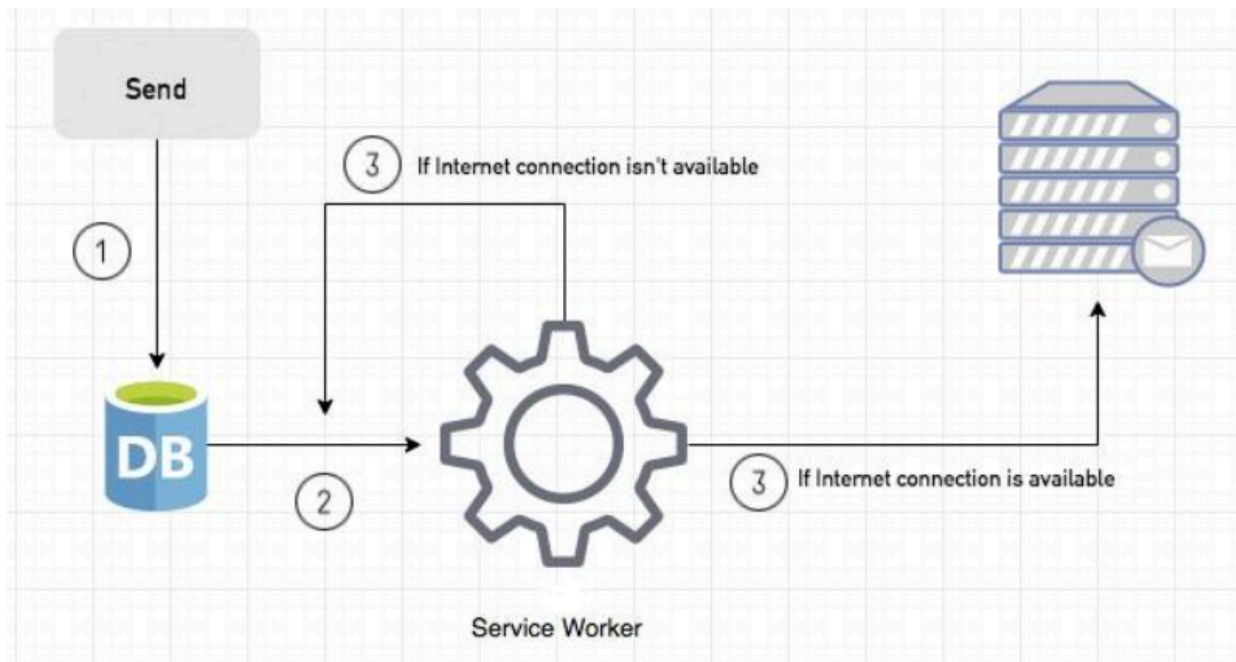
Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request's and current location's origin are the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted external URL, this is called “networkFirst”.

- CacheFirst - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- NetworkFirst - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new response will be cached and returned. But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

## Sync Event

Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are writing e-mail content and we didn't realize it. When completing the writing, we click the send button.

A sample is in the following for this case.



1. When we click the “send” button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. If the Internet connection is available, all email content will be read and sent to Mail Server.

If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

## Push Event

This is the event that handles push notifications that are received from the server. You can apply any method with received data. “Notification.requestPermission();” is the necessary line to show notification to the user. If you don't want to show any notification, you don't need this line. You can use Application Tab from Chrome Developer Tools for testing push notification.

**Code:**

```
//serviceworker
const cacheName = 'ecommerce-pwa-v1';
var filesToCache = [
  '/',
  '/index.html',
  '/app.js',
  '/serviceworker.js'
  // Add more files and assets here as needed
];
self.addEventListener('install', event => {
  event.waitUntil(preLoad());
});
self.addEventListener("fetch", function(event){
  event.respondWith(checkResponse(event.request).catch(function(){
    console.log("Fetch from cache successful!")
    return returnFromCache(event.request);
  }));
  console.log("Fetch successful!");
  event.waitUntil(addToCache(event.request));
});
self.addEventListener('sync', event=>{
  if (event.tag ==='syncMessage'){
    console.log("Sync successful!")
  }
});
self.addEventListener("push", function(event) {
  if (event && event.data) {
    try {
      var data = event.data.json();
      if (data && data.method === "pushMessage") {
        console.log("Push notification sent");
        self.registration.showNotification("Kiddoshield", { body: data.message });
      }
    } catch (error) {
      console.error("Error parsing push data:", error);
    }
  }
});
var preLoad = function(){
  return caches.open("offline").then(function(cache){
    return cache.addAll(filesToCache);
  });
};
```

```

var checkResponse = function (request) {
  return new Promise(function (fulfill, reject) {
    fetch(request)
      .then(function (response) {
        if (response.status !== 404) {
          fulfill(response);
        } else {
          reject(new Error("Response not found"));
        }
      })
      .catch(function (error) {
        reject(error);
      });
  });
};

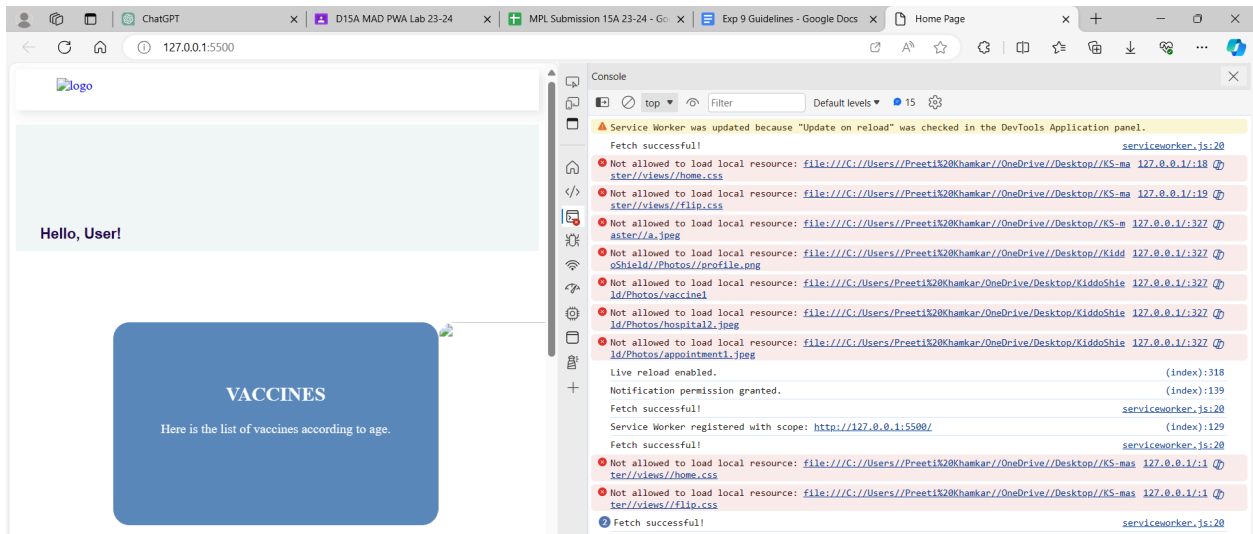
var returnFromCache = function(request){
  return caches.open("offline").then(function(cache){
    return cache.match(request).then(function(matching){
      if(!matching || matching.status == 404){
        return cache.match("offline.html");
      }
      else{
        return matching;
      }
    });
  });
};

var addToCache = function(request) {
  return caches.open("offline").then(function(cache) {
    return fetch(request).then(function(response) {
      return cache.put(request, response.clone()).then(function() {
        return response;
      });
    });
  });
};

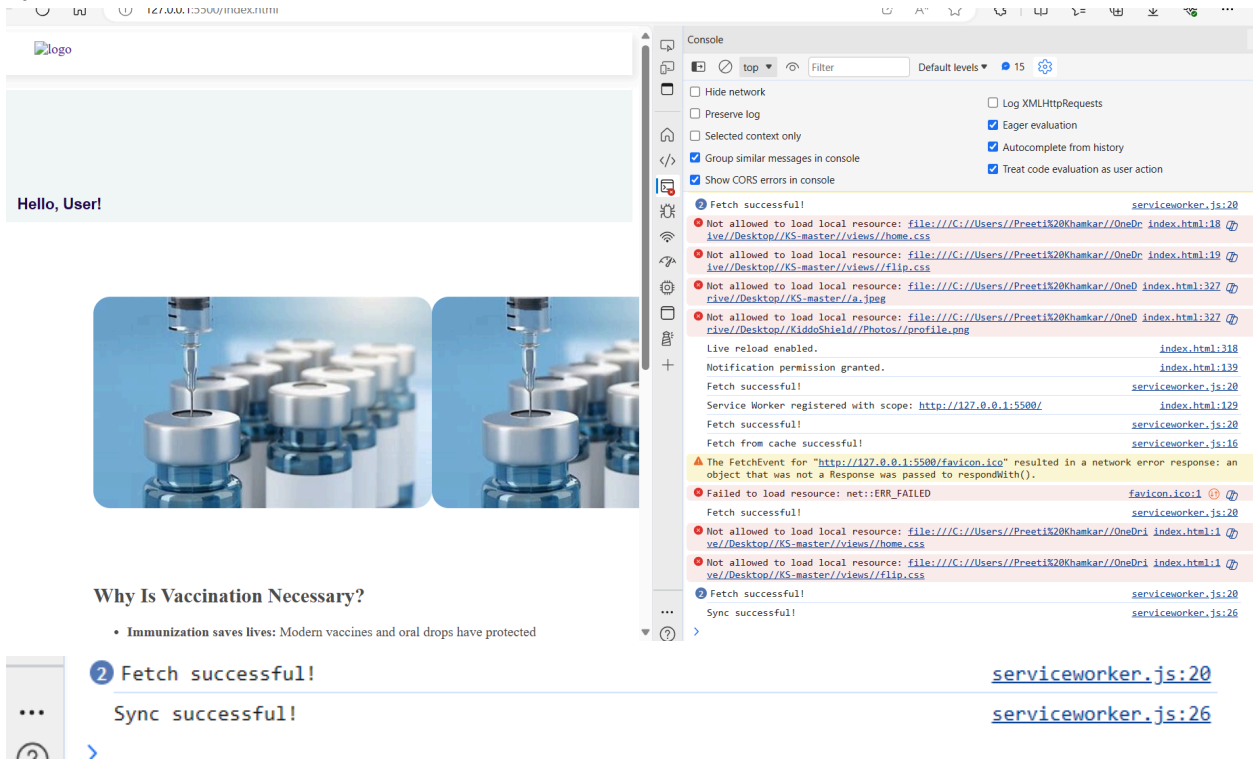
```

## Output:

### Fetch:



### Sync:




### Push:

127.0.0.1:5500/index.html

Logo

Hello, User!



### Why Is Vaccination Necessary?

- Immunization saves lives: Modern vaccines and oral drops have protected

Application

- Manifest
- Service workers
- Storage
  - Local storage
  - Session storage
  - IndexedDB
  - Web SQL
  - Cookies
  - Private state tokens
  - Interest groups
  - Shared storage
  - Cache storage
- Background services
  - Back/forward cache
  - Background fetch
  - Background sync
  - Bounce tracking mitigation
  - Notifications
  - Payment handler
  - Periodic background sync
  - Speculative loads
  - Push messaging
  - Reporting API
- Frames
  - top

Service workers

☐ Offline ☒ Update on reload ☐ Bypass for network

http://127.0.0.1:5500/ Network requests Update Unregister

Source: serviceworker.js  
Received 26/3/2024, 9:27:53 am

Status: #4530 activated and is running [stop](#)

Clients: [http://127.0.0.1:5500/index.html](#) [focus](#)  
[http://127.0.0.1:5500/index.html](#) [focus](#)

Push: { "method": "pushMessage", "message": "test-tag-from-devtools" } [Push](#)

Sync: syncMessage [Sync](#)

Periodic Sync: test-tag-from-devtools [Periodic Sync](#)

Update Cycle

Version	Update Activity	Timeline
#4530	Install	
#4530	Wait	
#4530	Activate	

Service workers from other origins

[See all registration](#) KiddoShield

KiddoShield  
Test push notification  
via Microsoft Edge

ve//Desktop//KS-master//views//+lip.css

- 2 Fetch successful! [serviceworker.js:20](#)
- Sync successful! [serviceworker.js:26](#)
- Push notification sent [serviceworker.js:35](#)

**Conclusion:** Implemented service worker events like fetch, sync, and push for E-commerce PWA.