EXPERIMENT NO: - 09

Name:- Tejas Gunjal **Class:-** D15A **Roll:No: -** 18

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 Worker:

- 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.

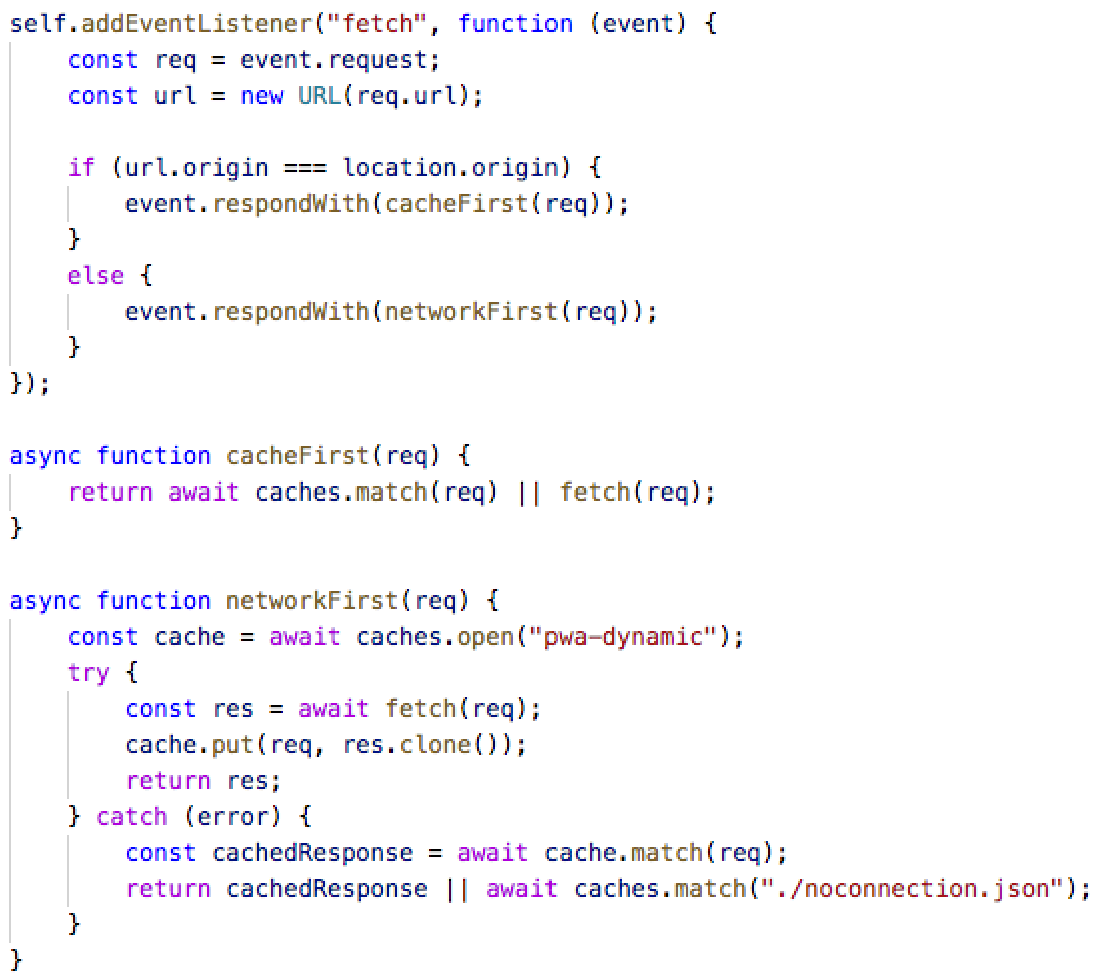
Fetch Event

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.



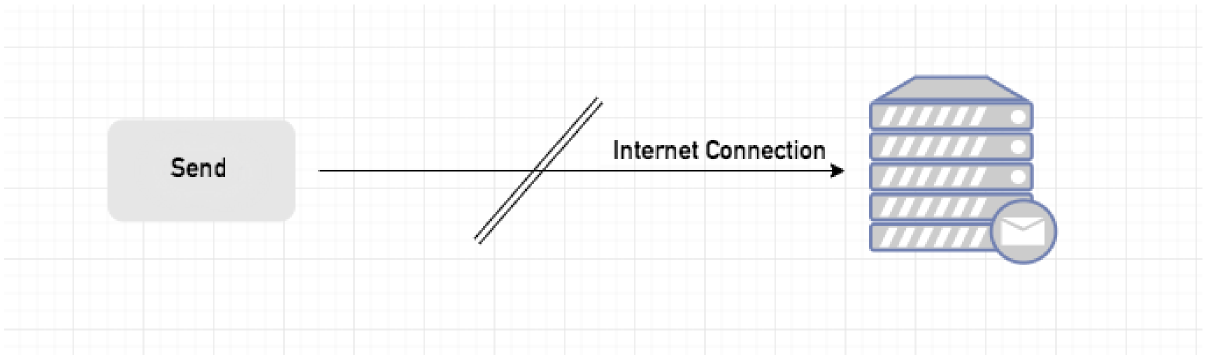
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

    if (url.origin === location.origin) {
        event.respondWith(cacheFirst(req));
    }
    else {
        event.respondWith(networkFirst(req));
    }
});

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

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");
    }
}
```

Sync Event

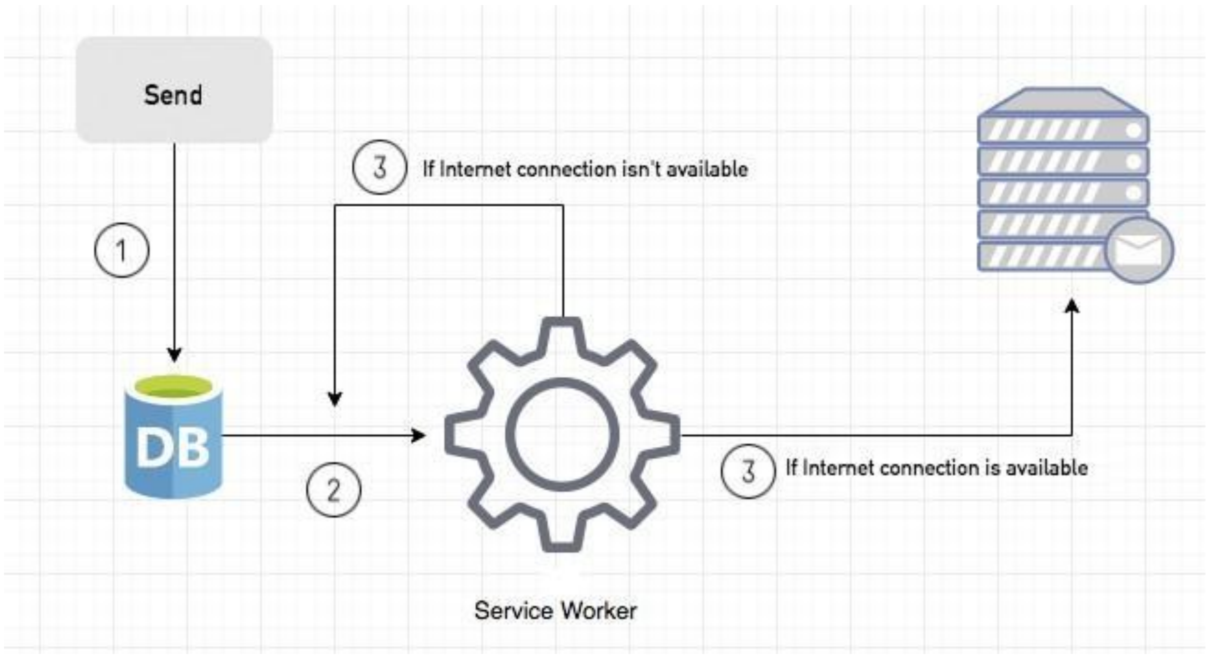
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

The following view shows the classical process of sending email to us. If the Internet Connection is broken, we can't send any content to Mail Server.



Here, you can create any scenario for yourself. A sample is in the following for this case.

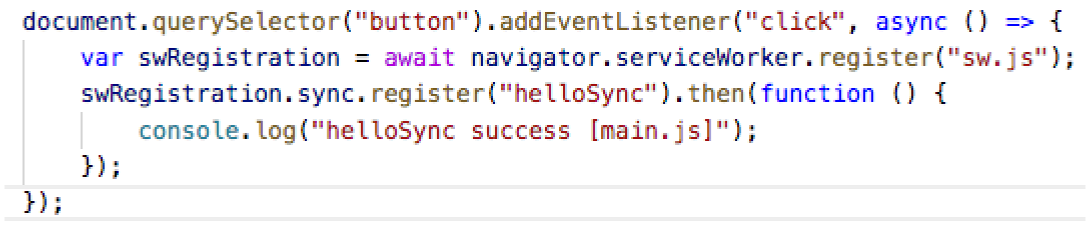

Send
1
3 If Internet connection isn't available
DB
2
3 If Internet connection is available
Service Worker

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.

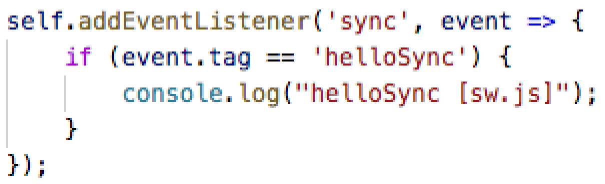
   You can see the working process within the following code block.

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]");
    }
});
```

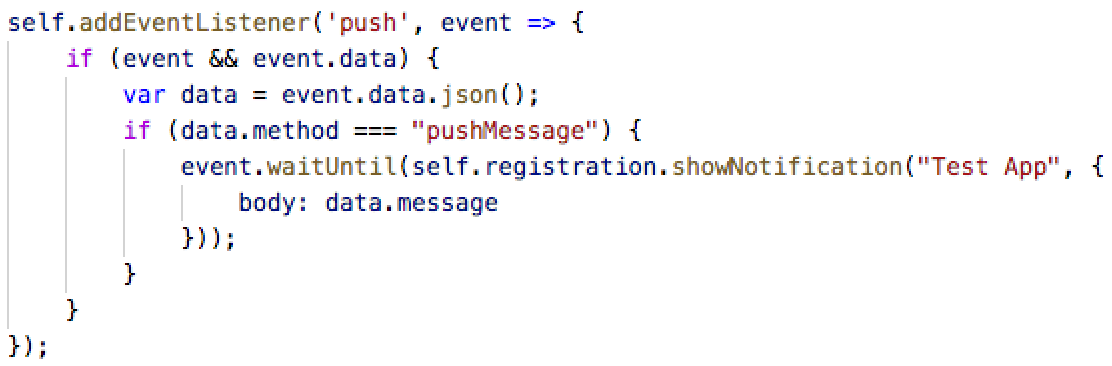
Push Event

This is the event that handles push notifications that are received from the server. You can apply any method with received data.

We can check in the following example.

"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.

In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has "method" and "message" properties. If the method value is "pushMessage", we open the information notification with the "message" property.



```
self.addEventListener('push', event => {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method === "pushMessage") {
            event.waitUntil(self.registration.showNotification("Test App", {
                body: data.message
            }));
        }
    }
});
```

You can use Application Tab from Chrome Developer Tools for testing push notification.

Code: -

service-worker.js

```
var staticCacheName = "pwa-v1"; // Versioned cache name

// Install event: Caching static assets
self.addEventListener("install", function (e) {
  console.log("Service Worker: Installing...");
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      console.log("Service Worker: Caching files...");
      return cache.addAll([
        "/",
        "/index.html",
        "/styles.css",
        "/app.js",
        "/logo.png",
        "/icon.png",
        "/offline.html" // Offline fallback page
      ]);
    })
  );
});

// Activate event: Cleanup old caches
self.addEventListener("activate", function (event) {
  console.log("Service Worker: Activating...");
  var cacheWhitelist = [staticCacheName];

  event.waitUntil(
    caches.keys().then(function (cacheNames) {
      return Promise.all(
        cacheNames.map(function (cacheName) {
          if (!cacheWhitelist.includes(cacheName)) {
            console.log("Service Worker: Deleting old cache:", cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    }).then(() => {
      console.log("Service Worker: Now controlling all clients.");
      self.clients.claim();
    })
  );
});
```

```
// Fetch event: Serve from cache, then network, then offline fallback
self.addEventListener("fetch", function (event) {
  console.log("Fetching:", event.request.url);
  event.respondWith(
    caches.match(event.request).then(function (response) {
      if (response) {
        console.log("Fetch successful (from cache):", event.request.url);
        return response; // Serve from cache
      }
      return fetch(event.request, { credentials: "include" })
        .then((networkResponse) => {
          console.log("Fetch successful (from network):", event.request.url);
          return networkResponse;
        })
        .catch(() => {
          console.log("Fetch failed, serving offline page.");
          return caches.match("/offline.html");
        });
    })
  );
});

// Push Notification Event
self.addEventListener("push", function (event) {
  console.log("Push event received.");
  const data = event.data ? event.data.json() : { message: "Default notification" };

  const options = {
    body: data.message,
    icon: "/logo.png"
  };

  event.waitUntil(
    self.registration.showNotification("VESIT", options).then(() => {
      console.log("Push Notification displayed successfully.");
    })
  );
});

// Background Sync Event
self.addEventListener("sync", function (event) {
  console.log("Sync event received:", event.tag);

  if (event.tag === "syncMessage") {
    event.waitUntil(
      (async () => {
        console.log("Processing sync event...");

        // Simulate an actual sync operation (like posting data to server)
        try {
          let response = await fetch("/sync-endpoint", { method: "POST" });
          console.log("Sync request sent:", response.status);
        } catch (error) {
          console.error("Sync request failed:", error);
        }

        // Show notification
        await self.registration.showNotification("VESIT", {
          body: "Sync successful!",
          icon: "/logo.png"
        });

        console.log("Sync event handled successfully.");
      })()
    );
  }
});

// Message event for skipWaiting
self.addEventListener("message", (event) => {
  if (event.data.action === "skipWaiting") {
    console.log("Skipping waiting phase...");
    self.skipWaiting();
  }
});
```

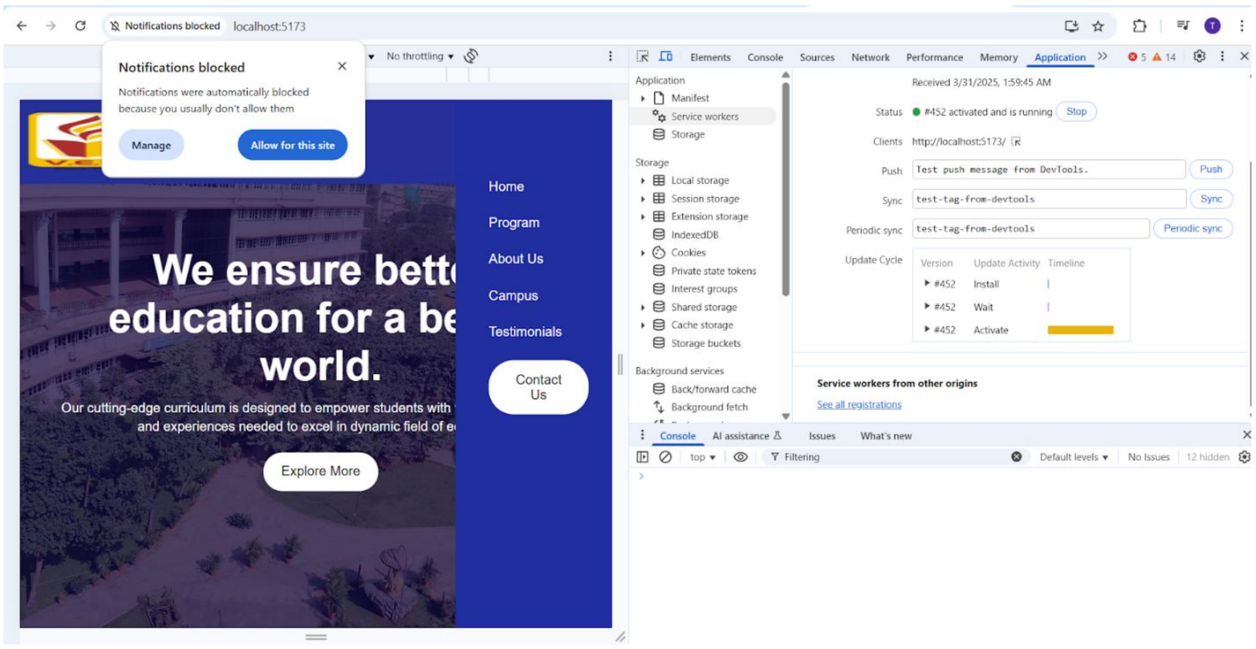
Main.jsx

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import App from "./App.jsx";
import "./index.css";

createRoot(document.getElementById("root")).render(
  <StrictMode>
    <App />
  </StrictMode>
);

// Register Service Worker
if ("serviceWorker" in navigator) {
  window.addEventListener("load", () => {
    navigator.serviceWorker
      .register("/service-worker.js")
      .then((registration) => {
        console.log("Service Worker registered successfully with scope:", registration.scope);

        // Request Notification Permission
        Notification.requestPermission().then((permission) => {
          if (permission === "granted") {
            console.log("Notification permission granted.");
          } else {
            console.warn("Notification permission denied.");
          }
        });

        if ('serviceWorker' in navigator && 'SyncManager' in window) {
          navigator.serviceWorker.ready.then(registration => {
              return registration.sync.register('syncMessage')
                  .then(() => console.log('Sync event registered'))
                  .catch(err => console.error('Sync registration failed', err));
          });
      }


        // Push Notification Setup
        navigator.serviceWorker.ready.then((reg) => {
          reg.pushManager.subscribe({
            userVisibleOnly: true,
            applicationServerKey: "BD13Iv9g2jOfJ-
7JtItR7smZV7rk5DgFWfB43GLh7HgjrATPzJKDS8jCGYFNnaTutJM-
5oN885EjYB0k1CfOG2s" // Replace with actual VAPID key
          }).then((subscription) => {
            console.log("Push Notification Subscription successful:", subscription);
          }).catch((error) => {
            console.error("Push Notification Subscription failed:", error);
          });
        });

        // Background Sync Setup
        navigator.serviceWorker.ready.then((reg) => {
          if ("sync" in reg) {
            reg.sync.register("syncMessage").then(() => {
              console.log("Background sync registered successfully.");
            }).catch((error) => {
              console.error("Background sync registration failed:", error);
            });
          } else {
```

```
        console.warn("SyncManager is not supported.");
      }
    });

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

OUTPUT: -

➢ **Notification Permission**



Notifications blocked

Notifications were automatically blocked because you usually don't allow them

Manage | Allow for this site

localhost:5173

Received 3/31/2025, 1:59:45 AM

Status ● #452 activated and is running Stop

Clients http://localhost:5173/

Push Test push message from DevTools. Push

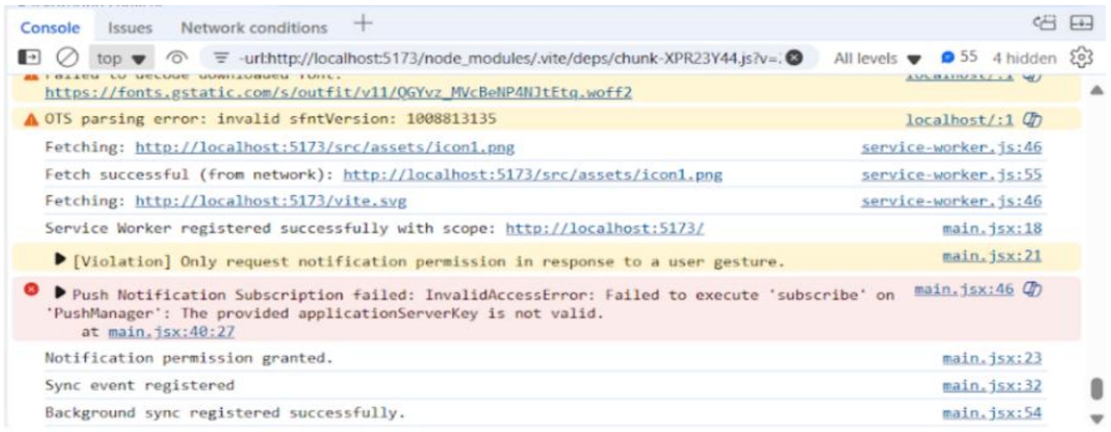
Sync test-tag-from-devtools Sync

Periodic sync test-tag-from-devtools Periodic sync

```
Update Cycle  Version  Update Activity  Timeline
              ▸ #452   Install
              ▸ #452   Wait
              ▸ #452   Activate
```

Service workers from other origins

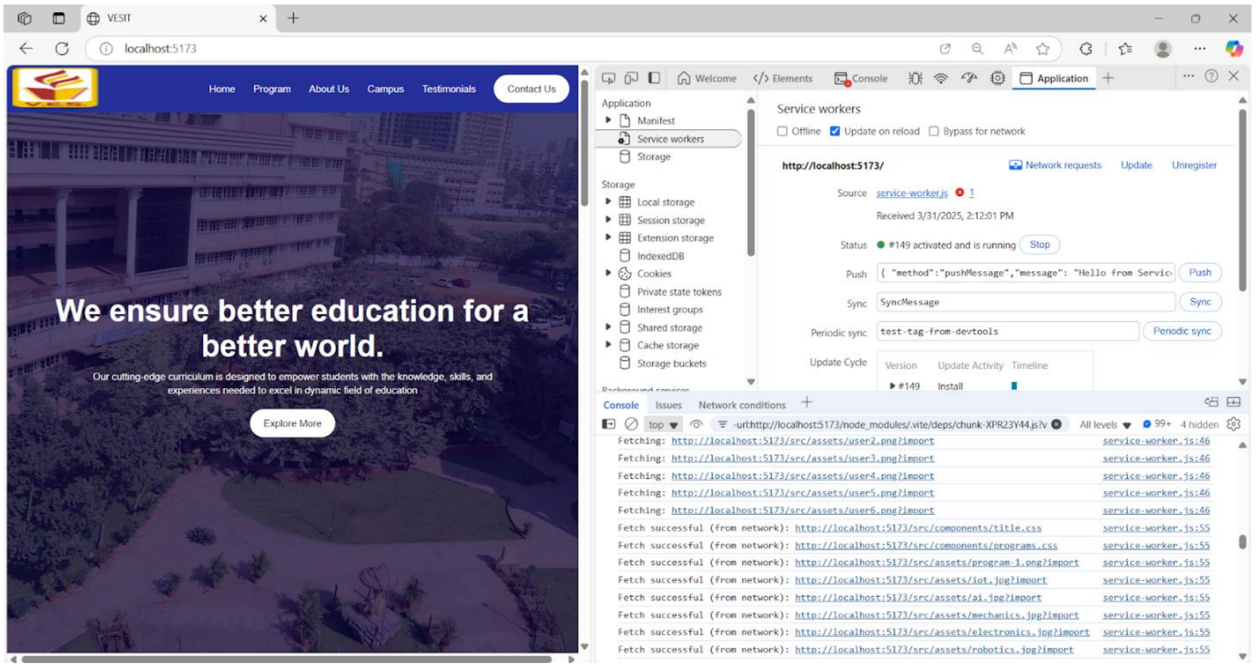
See all registrations



Console
Issues
Network conditions
top
-url:http://localhost:5173/node_modules/.vite/deps/chunk-XPR23Y44.js?v=.
All levels
55
4 hidden

```
https://fonts.gstatic.com/s/outfit/v11/QGYvz_MVcBeNP4NJtEtq.woff2
OTS parsing error: invalid sfntVersion: 1008813135                               localhost/:1
Fetching: http://localhost:5173/src/assets/icon1.png                       service-worker.js:46
Fetch successful (from network): http://localhost:5173/src/assets/icon1.png service-worker.js:55
Fetching: http://localhost:5173/vite.svg                                   service-worker.js:46
Service Worker registered successfully with scope: http://localhost:5173/          main.jsx:18
 ▶ [Violation] Only request notification permission in response to a user gesture.  main.jsx:21
 ▶ Push Notification Subscription failed: InvalidAccessError: Failed to execute 'subscribe' on  main.jsx:46
'PushManager': The provided applicationServerKey is not valid.
    at main.jsx:40:27
Notification permission granted.                                                   main.jsx:23
Sync event registered                                                              main.jsx:32
Background sync registered successfully.                                           main.jsx:54
```

➢ Fetch Event



Service workers

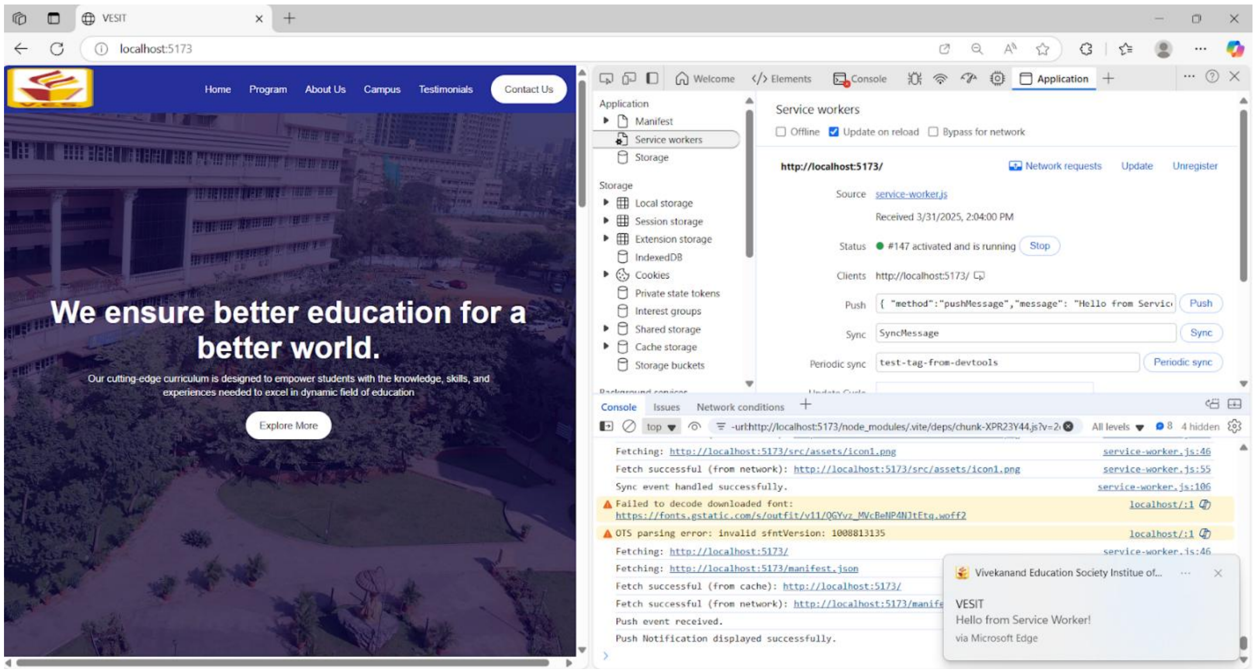
Offline · Update on reload · Bypass for network

http://localhost:5173/

Source: service-worker.js
Received 3/31/2025, 2:12:01 PM
Status: #149 activated and is running
Push: { "method":"pushMessage","message": "Hello from Servic
Sync: SyncMessage
Periodic sync: test-tag-from-devtools
Update Cycle: Version #149 · Update Activity: Install

```
Fetching: http://localhost:5173/src/assets/user2.png?import                       service-worker.js:46
Fetching: http://localhost:5173/src/assets/user3.png?import                       service-worker.js:46
Fetching: http://localhost:5173/src/assets/user4.png?import                       service-worker.js:46
Fetching: http://localhost:5173/src/assets/user5.png?import                       service-worker.js:46
Fetching: http://localhost:5173/src/assets/user6.png?import                       service-worker.js:46
Fetch successful (from network): http://localhost:5173/src/components/title.css   service-worker.js:55
Fetch successful (from network): http://localhost:5173/src/components/programs.css   service-worker.js:55
Fetch successful (from network): http://localhost:5173/src/assets/program-1.png?import   service-worker.js:55
Fetch successful (from network): http://localhost:5173/src/assets/iot.jpg?import   service-worker.js:55
Fetch successful (from network): http://localhost:5173/src/assets/ai.jpg?import   service-worker.js:55
Fetch successful (from network): http://localhost:5173/src/assets/mechanics.jpg?import   service-worker.js:55
Fetch successful (from network): http://localhost:5173/src/assets/electronics.jpg?import   service-worker.js:55
Fetch successful (from network): http://localhost:5173/src/assets/robotics.jpg?import   service-worker.js:55
```

➢ Push Event



Service workers

Offline · Update on reload · Bypass for network

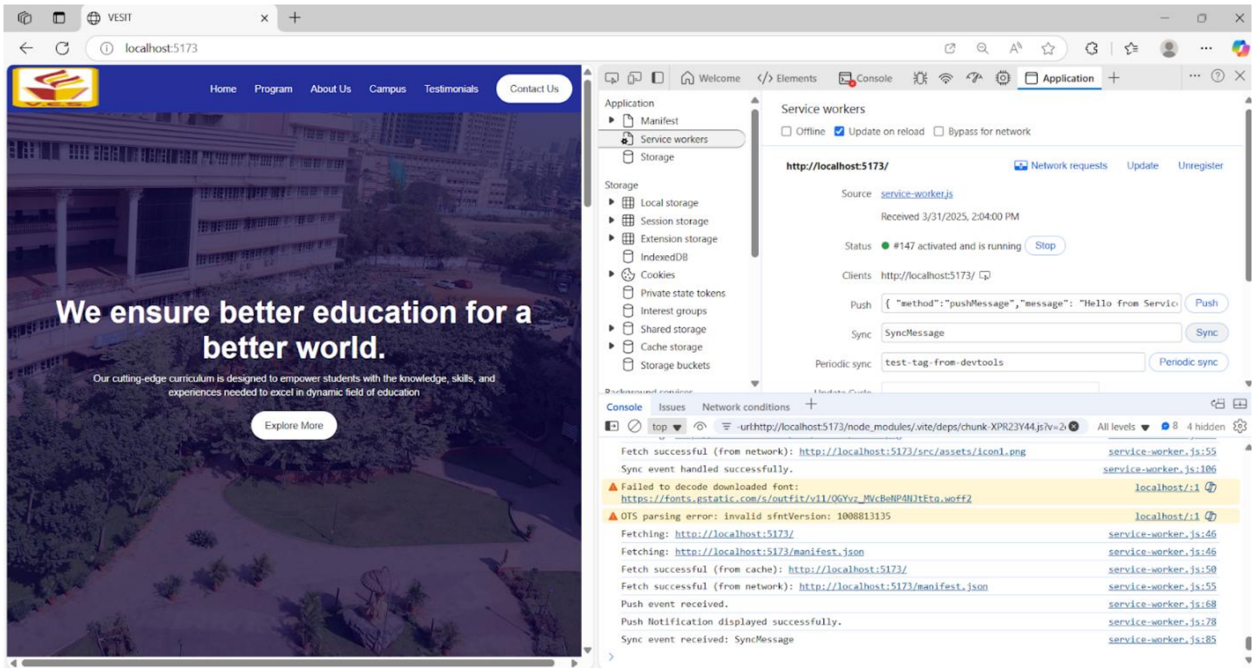
http://localhost:5173/

Source: service-worker.js
Received 3/31/2025, 2:04:00 PM
Status: #147 activated and is running
Clients: http://localhost:5173/
Push: { "method":"pushMessage","message": "Hello from Servic
Sync: SyncMessage
Periodic sync: test-tag-from-devtools

```
Fetching: http://localhost:5173/src/assets/icon1.png                              service-worker.js:46
Fetch successful (from network): http://localhost:5173/src/assets/icon1.png       service-worker.js:55
Sync event handled successfully.                                                  service-worker.js:106
⚠ Failed to decode downloaded font:                                               localhost/:1
https://fonts.gstatic.com/s/outfit/v11/QGYvz_MVcBeNP4NJtEtq.woff2
⚠ OTS parsing error: invalid sfntVersion: 1008813135                              localhost/:1
Fetching: http://localhost:5173/                                                  service-worker.js:46
Fetching: http://localhost:5173/manifest.json
Fetch successful (from cache): http://localhost:5173/
Fetch successful (from network): http://localhost:5173/manife
Push event received.
Push Notification displayed successfully.
```

Vivekanand Education Society Institue of...
VESIT
Hello from Service Worker!
via Microsoft Edge

➢ Sync Event



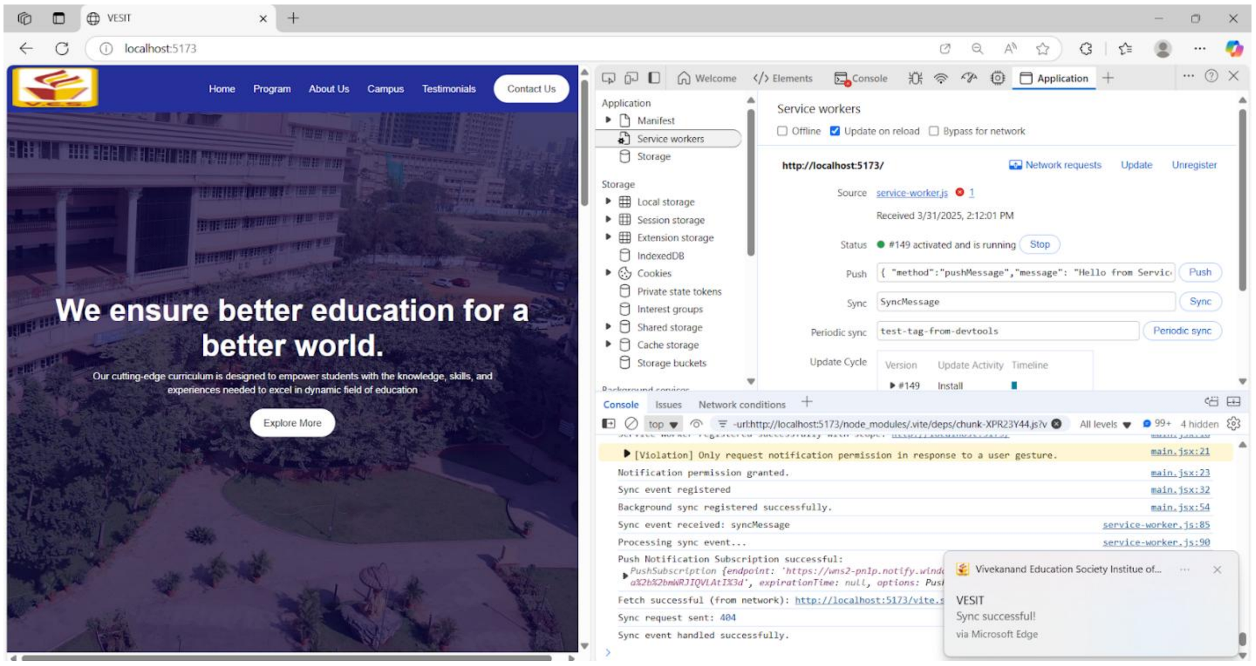
Service workers

☐ Offline ☑ Update on reload ☐ Bypass for network

http://localhost:5173/ — Network requests · Update · Unregister

Source	service-worker.js
	Received 3/31/2025, 2:04:00 PM
Status	#147 activated and is running (Stop)
Clients	http://localhost:5173/
Push	{ "method":"pushMessage","message": "Hello from Servic (Push)
Sync	SyncMessage (Sync)
Periodic sync	test-tag-from-devtools (Periodic sync)

```
top ▼   -url:http://localhost:5173/node_modules/.vite/deps/chunk-XPR23Y44.js?v=2   All levels ▼   8   4 hidden
Fetch successful (from network): http://localhost:5173/src/assets/icon1.png       service-worker.js:55
Sync event handled successfully.                                                 service-worker.js:106
⚠ Failed to decode downloaded font:                                                      localhost/:1
https://fonts.gstatic.com/s/outfit/v11/QGYvz_MVcBeNP4NJtEtq.woff2
⚠ OTS parsing error: invalid sfntVersion: 1008813135                                     localhost/:1
Fetching: http://localhost:5173/                                                  service-worker.js:46
Fetching: http://localhost:5173/manifest.json                                     service-worker.js:46
Fetch successful (from cache): http://localhost:5173/                             service-worker.js:50
Fetch successful (from network): http://localhost:5173/manifest.json              service-worker.js:55
Push event received.                                                              service-worker.js:68
Push Notification displayed successfully.                                         service-worker.js:78
Sync event received: SyncMessage                                                  service-worker.js:85
```



Service workers

☐ Offline ☑ Update on reload ☐ Bypass for network

http://localhost:5173/ — Network requests · Update · Unregister

Source	service-worker.js ⊗ 1
	Received 3/31/2025, 2:12:01 PM
Status	#149 activated and is running (Stop)
Push	{ "method":"pushMessage","message": "Hello from Servic (Push)
Sync	SyncMessage (Sync)
Periodic sync	test-tag-from-devtools (Periodic sync)
Update Cycle	Version: #149 · Update Activity: Install · Timeline

```
top ▼   -url:http://localhost:5173/node_modules/.vite/deps/chunk-XPR23Y44.js?v   All levels ▼   99+   4 hidden
▶ [Violation] Only request notification permission in response to a user gesture.       main.jsx:21
Notification permission granted.                                                        main.jsx:23
Sync event registered                                                                   main.jsx:32
Background sync registered successfully.                                                main.jsx:54
Sync event received: syncMessage                                                  service-worker.js:85
Processing sync event...                                                          service-worker.js:90
Push Notification Subscription successful:
▶ PushSubscription {endpoint: 'https://wns2-pn1p.notify.windo
a%2b%2bmWRJIQVLAtI%3d', expirationTime: null, options: Push
Fetch successful (from network): http://localhost:5173/vite.s
Sync request sent: 404
Sync event handled successfully.
```

Vivekanand Education Society Institue of...
VESIT
Sync successful!
via Microsoft Edge