

EXPERIMENT- 10

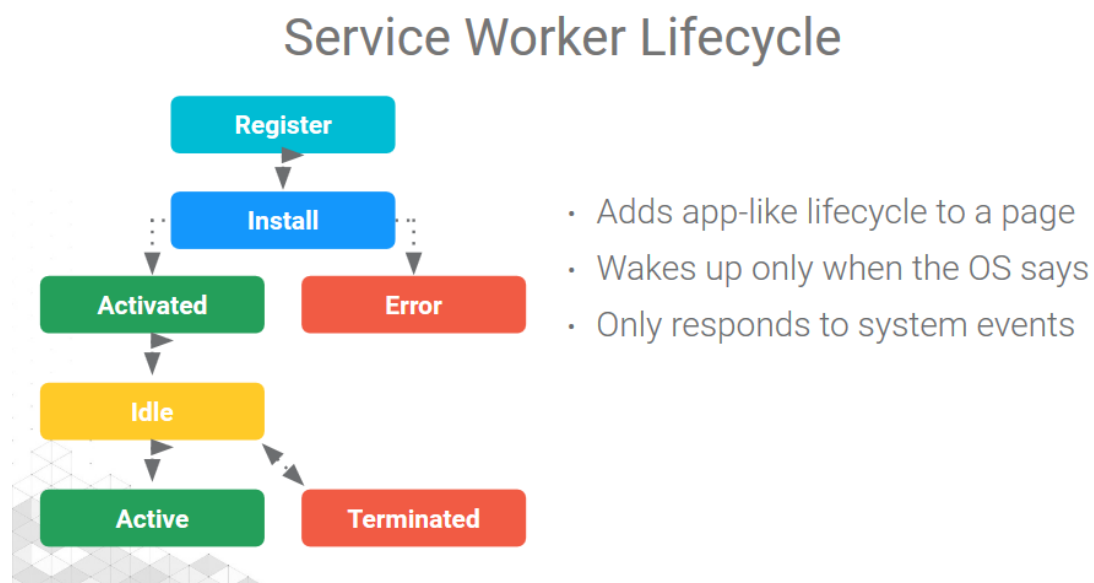
Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA.

Lab Outcome: Develop and Analyze PWA Features and deploy it over app hosting solutions.

Theory:

1. Explain service worker life cycle.

A service worker is a JavaScript file that runs separately from the main browser thread and acts as a programmable proxy between the web application, the browser, and the network. It is an important component of progressive web applications (PWAs), enabling them to work offline and provide a faster and smoother user experience. The service worker has its own life cycle, consisting of four different states:



1. **Registration:** This is the first stage of the service worker life cycle, where the service worker is registered with the browser. Once the service worker is registered, it is then installed.
2. **Installation:** During this stage, the browser checks the service worker script for changes, and if there are any, it is downloaded and installed. Once installed, the service worker is activated.
3. **Activation:** In this stage, the service worker takes control of the pages it is associated with. It can now intercept network requests, respond with cached assets, and even modify requests and responses. After activation, the service worker can still be updated and changed.

4. **Idle:** This is the final stage of the service worker life cycle. The service worker remains in the idle state until there is an event to handle, such as a fetch event or a push notification. Once an event occurs, the service worker moves back to the activation state and continues to handle events until it is unregistered.

The service worker life cycle enables web applications to perform actions even when the user is offline. By intercepting network requests, the service worker can serve cached content, reducing the time it takes to load the web application. It can also push notifications to users, even when the web application is not open, making it a powerful tool for engaging users and improving their experience.

2. **What is pushed notification?**

A push notification is a short message that appears as a pop-up on your desktop browser, mobile home screen, or in your device notification centre from a mobile app. Push notifications are typically opt-in alerts that display text and rich media, like images or buttons, which enable a user to take a specific action.

Organizations use push notifications as a marketing or communication channel, but they can also be used as a security mechanism. There are two types of push notifications:

1. **Web-based notifications**, also termed “web-push” notifications, can appear on your desktop or mobile device. Any site can send a push notification through supported operating systems (OS) and browsers. Notifications can pop-up on different areas of a screen or view, dependent on your browser or OS type.
2. **App-based notifications** are what most users deem a push notification. This type of notification is created in-app. From a user's device, whether on a mobile or desktop device, a recipient must typically opt-in first. These notifications or alerts are more typically pops-up on mobile devices. Apps offer these as a way to either create greater in-app user engagement, and open-rates, or to compel a lead to take a specific action.

Push notifications can be cloud-based or app-based, and are built to work with a server that provides the notification. An API can enable push notifications from cloud services as app and web push services. Once an organization requests a push notification, an API calls this service and sets the message in place to be delivered.

A push notification arrives on your mobile home screen or your lock screen. It can also appear as a notification on your app icon, or on your desktop home screen when launching your browser and also while in use.

Types of push notifications:

There are many ways of applying push notifications, but these are irrespective of marketing strategies, and are conventional to the channel in general. These include:

1. Reminders such as in-cart actions, sign-ons and next-step actions
2. Updates, including news-related or relevant brand information
3. Deals like calls-to-action (CTAs) for sales, specials or subscription opt-ins
4. Authentications like security-based, one-time pass-codes
5. PSA notifications such as civic information and weather alerts

Benefits of push notifications:

There are several benefits to using push notifications. However, these are tightly tied to how well an organization plans and executes its use of them. There are many reasons organizations benefit from using push notifications. These include the following:

- Higher open-rates than email
- Automates marketing campaigns and communication
- Similar to SMS messaging (short message service), a device, browser or app does not need to be powered on to send a notification
- Generates increased user satisfaction and enhances user experience
- Creates opportunities for more user interactions—and sales
- Achieves real-time responsiveness
- Enables user-centric customization for opt-ins and opt-outs
- Provides behavior analytics that can inform content strategy

The most significant benefit for push notification recipients is that the channel is a user-centric medium. Recipients can receive information on their terms and in their preferred space. They can also change device notification settings, or unsubscribe to notifications. This latitude of options counteracts notification fatigue, but also compels app publishers and organizations to create the most relevant content for a recipient.

Practical & Outcome:

Step 1: Take the app from the previous experiment and add the service worker file in the public folder.

Serviceworker.json:

```
var staticCacheName = "pwa";

self.addEventListener("install", function (e) {
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll(["/"]);
    })
  );
});


self.addEventListener("fetch", function (event) {
  console.log(event.request.url);

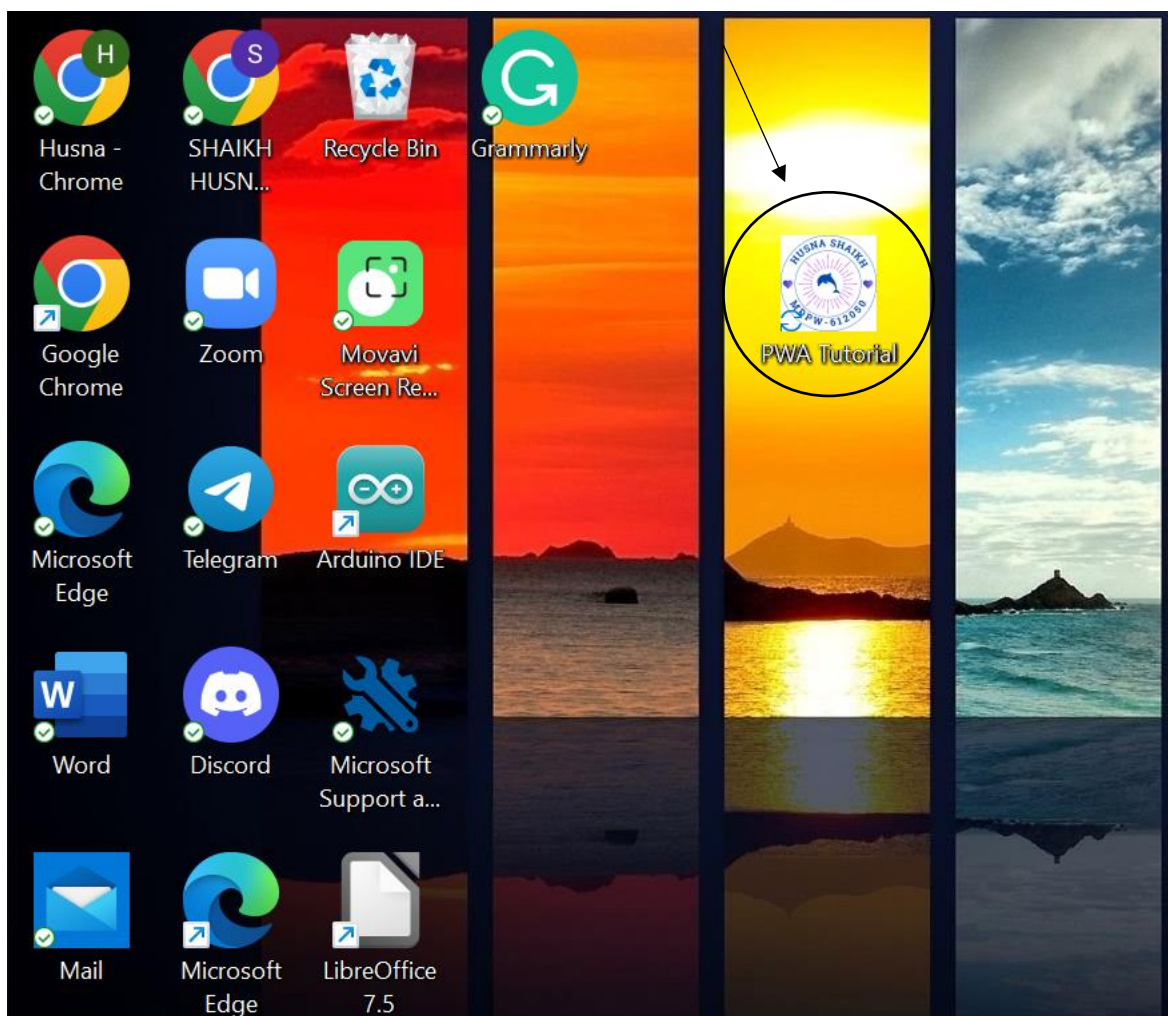
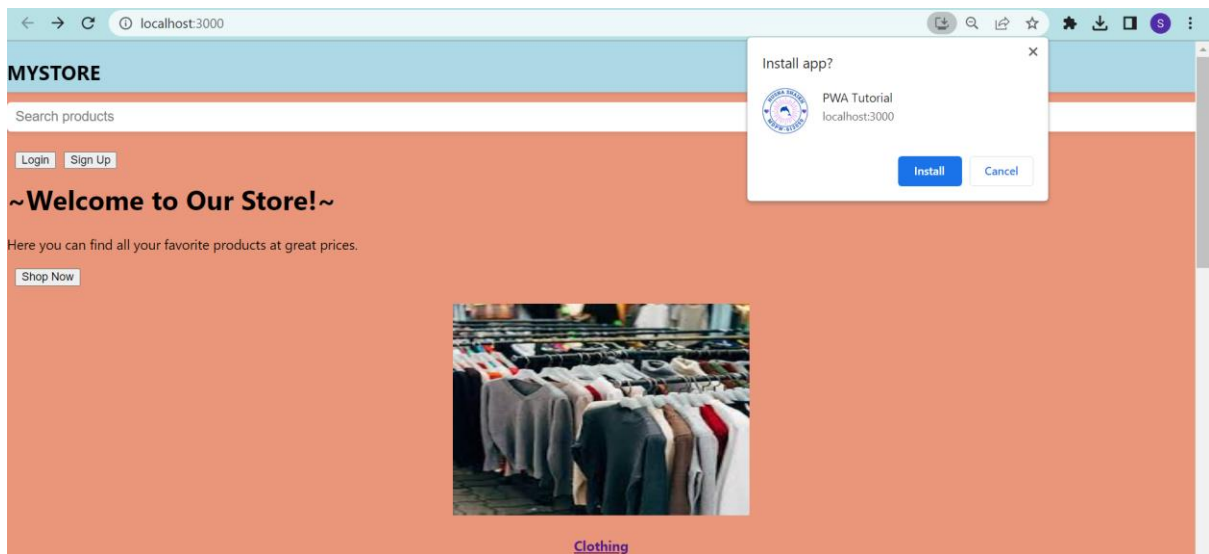
  event.respondWith(
    caches.match(event.request).then(function (response) {
      return response || fetch(event.request);
    })
  );
});
```

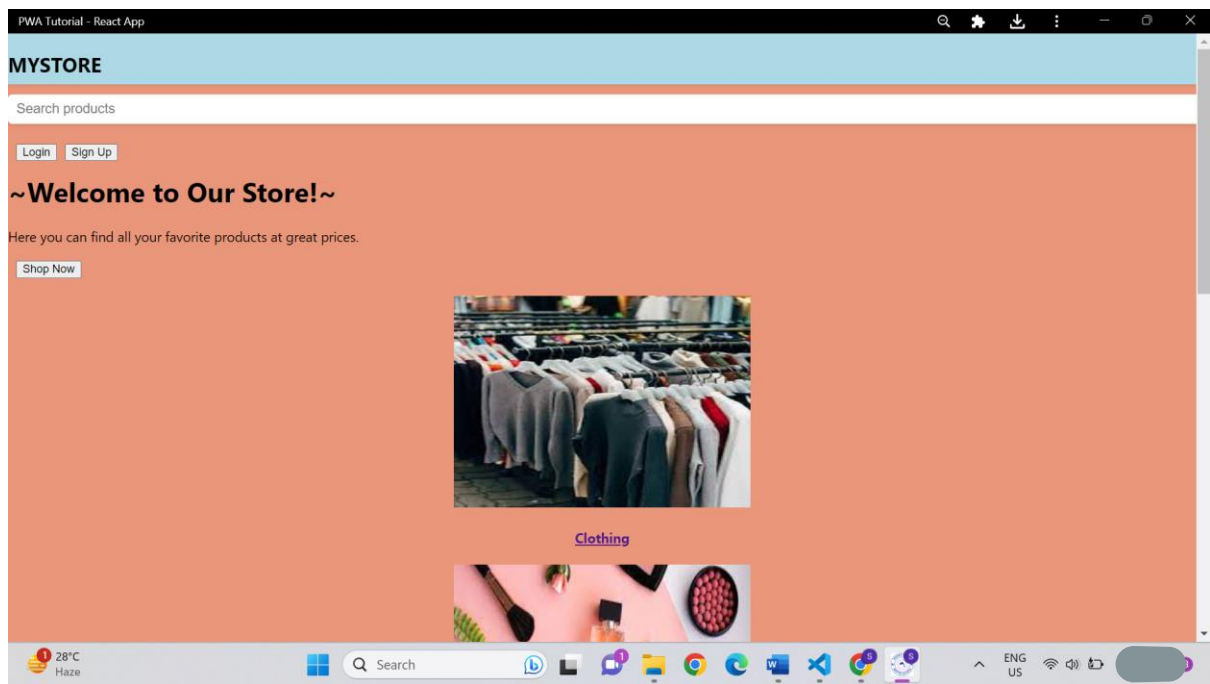
Step 2: Add the following script in the index.html file.

```
<script>
  window.addEventListener('load', () => {
    registerSW();
  });

  // Register the Service Worker
  async function registerSW() {
    if ('serviceWorker' in navigator) {
      try {
        await navigator
          .serviceWorker
          .register('serviceworker.js');
      }
      catch (e) {
        console.log('SW registration failed');
      }
    }
  }
</script>
```

Step 3: You will get the  icon on your web page, click on it and install the PWA on your desktop.





Tools used: Visual Studio Code, React, serviceworker.json file

Conclusion:

From this experiment we have learned how to add the serviceworker.json file and `<script>` inside the index.html file in order to download out PWA onto our desktop.