Experiment No. 8

Aim : To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

Theory :

Service Worker

Service Worker is a powerful script that operates in the background of a browser, independently of user interaction. It acts as a network proxy, intercepting outgoing HTTP requests made by your web application. With Service Worker, you can manage network traffic, handle push notifications, and develop "offline first" web applications using the Cache API.

- **Network Proxy:** Service workers intercept all outgoing HTTP requests, allowing you to handle these requests. For example, you can serve content from a local cache if available, improving performance and providing a better user experience.
- **Offline Capabilities:** Service workers enable offline functionality by caching essential application resources such as HTML, CSS, JavaScript, and images. When a user is offline, the service worker can retrieve the requested content from the cache, ensuring a seamless experience even without an internet connection.
- **HTTPS Requirement:** For security reasons, service workers require HTTPS connections. This ensures secure communication between the service worker, your application, and the server.

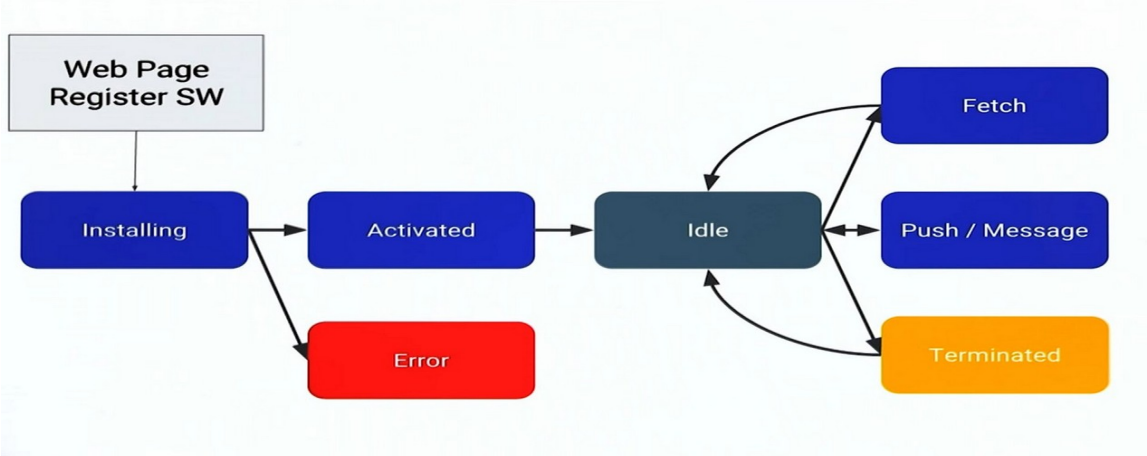
What can we do with Service Workers?

- **Network Traffic Control:** Manage all network traffic of the page and manipulate requests and responses. For example, you can respond to a CSS file request with plain text or an HTML file request with a PNG file. You can also respond with the requested content.

- **Caching:** Cache any request/response pair with Service Worker and Cache API, allowing you to access offline content anytime.

- **Push Notifications:** Manage push notifications with Service Worker, enabling you to show informational messages to the user.
- **Background Processes:** Even when the internet connection is broken, you can start any process with the Background Sync feature of Service Worker.

Service Worker Cycle



Steps for coding and registering a service worker for your E-commerce PWA completing the install and activation process:

1. Create the Service Worker File (sw.js):

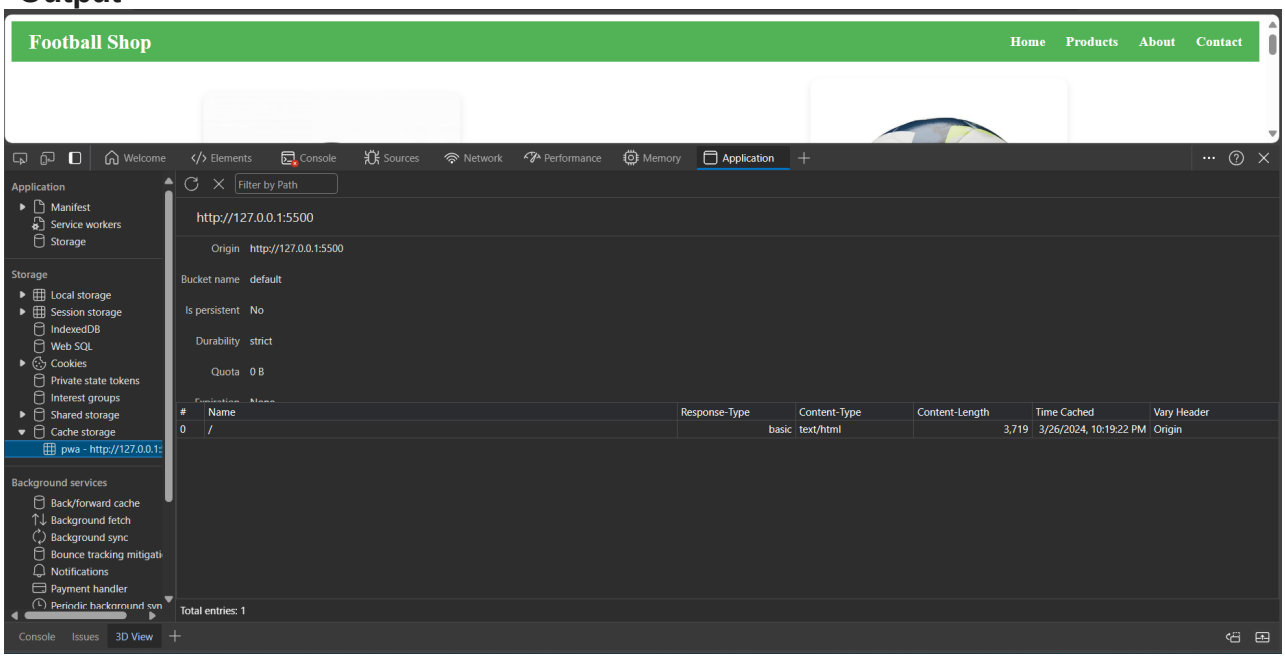
```
JS sw.js > self.addEventListener('install') callback > then() callback
self.addEventListener('install', function(event) {
    event.waitUntil(
      caches.open('offline')
        .then(function(cache) {
          return cache.addAll([
            '/',
            '/index.html',
          ]);
        })
    );
  });

  self.addEventListener('fetch', function(event) {
    event.respondWith(
      fetch(event.request)
        .catch(function() {
          return caches.match(event.request)
            .then(function(matching) {
              return matching || caches.match('offline.html');
            });
        })
    );
  });
```

2. Register the Service Worker:

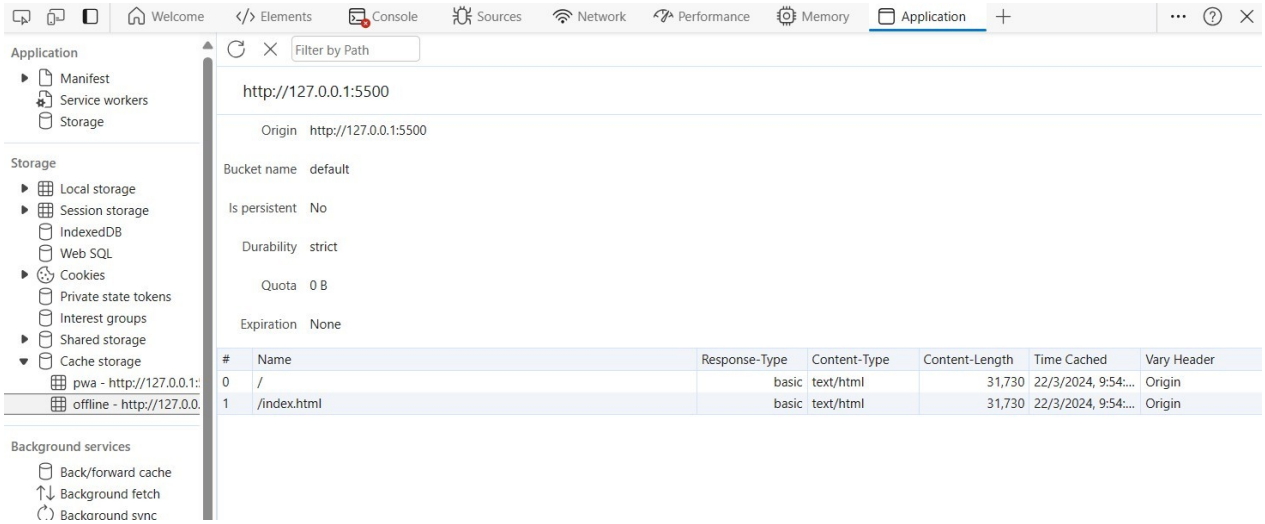
In your main JavaScript file (e.g., main.js or app.js), add the following code:

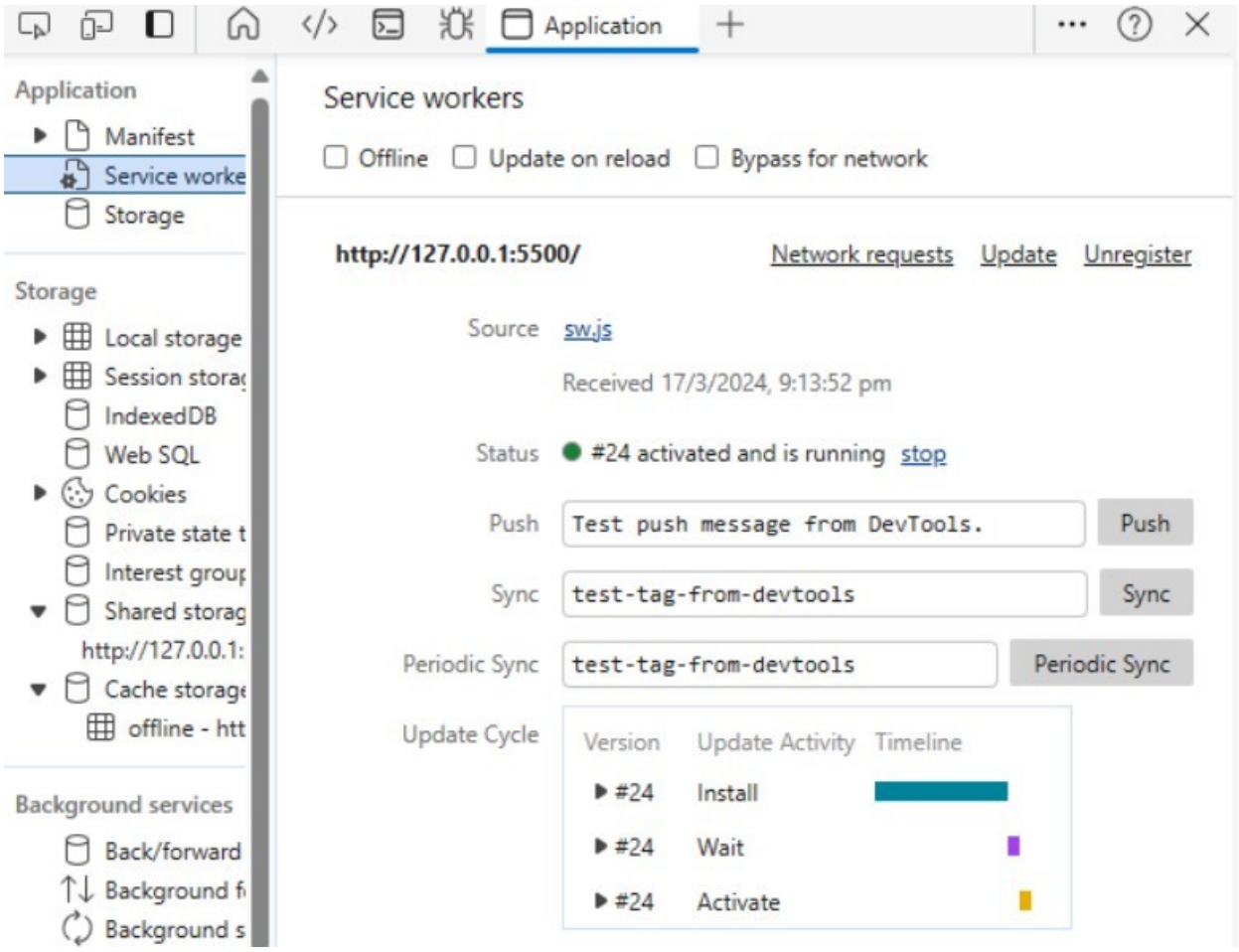
```
async function registerSW() {
  if ('serviceWorker' in navigator) {
    try {
      await navigator
            .serviceWorker
            .register('serviceworker.js');
    }
    catch (e) {
      console.log('SW registration failed');
    }
  }
}
```

Output



Pranav Chindage pwa exp Roll no:





Conclusion : I have understood and successfully registered a service worker, and completed the install and activation process for a new service worker for the E-commerce PWA.