

**Name : Anushka Karhadkar**

**Class: D15B**

**Roll no.: 28**

### **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**

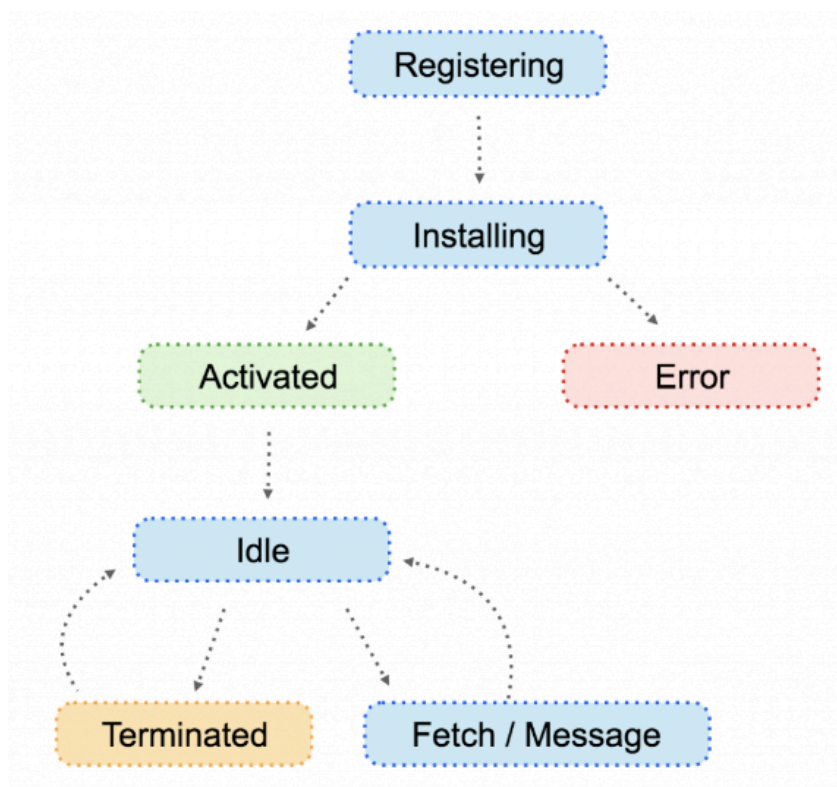
Service Worker is a powerful script that operates in the background of a browser, independently of user interaction. It acts as a network proxy, intercepting outgoing HTTP requests made by your web application. With Service Worker, you can manage network traffic, handle push notifications, and develop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



## 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 (serviceworker.js):

```
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => cache.addAll(urlsToCache))
  );
});

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

2. Register the Service Worker:

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

```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/service-worker.js')
      .then(registration => {
        console.log('Service Worker registered with scope:', registration.scope);
      })
      .catch(error => {
        console.error('Service Worker registration failed:', error);
      });
  });
}
```

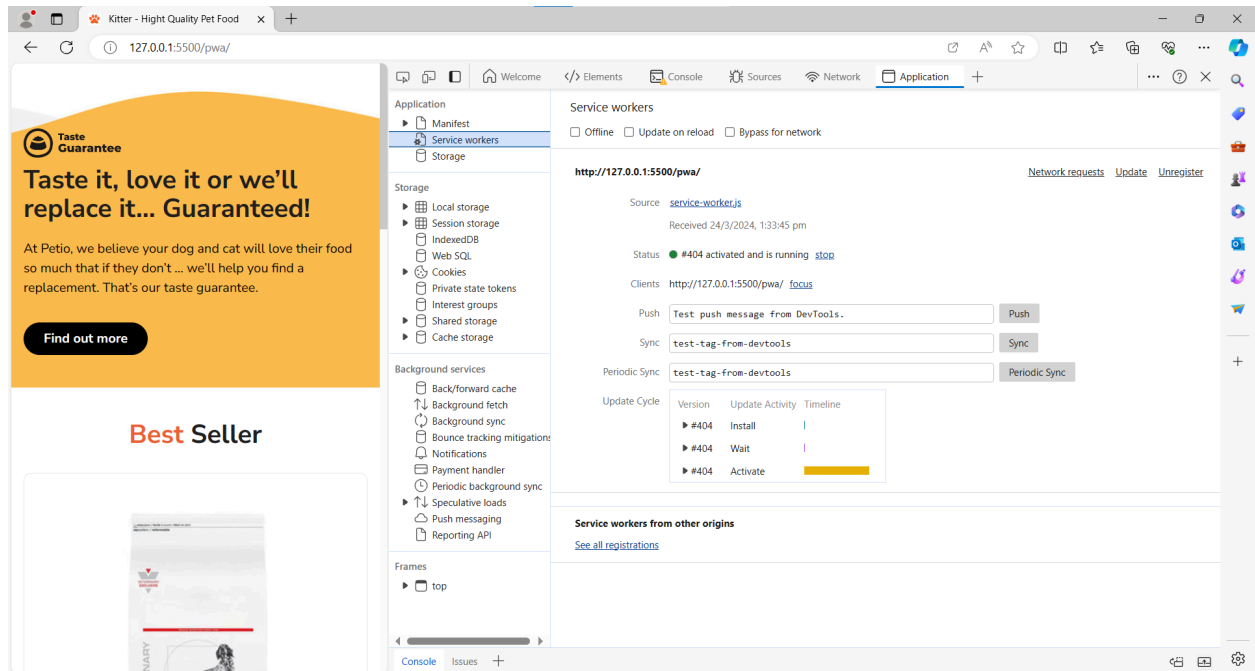
The screenshot displays a web browser window with the URL `http://127.0.0.1:5500/pwa/`. The page content includes a 'Taste Guarantee' section with the text 'Taste it, love it or we'll replace it... Guaranteed!' and a 'Best Seller' section featuring a product image. The browser's developer tools are open, showing the 'Application' tab. The 'Manifest' section lists the following resources:

- Manifest:**
  - Manifest
  - Service workers
  - Storage
- Storage:**
  - Local storage
  - Session storage
  - IndexedDB
  - Web SQL
  - Cookies
  - Private state tokens
  - Interest groups
  - Shared storage
  - Cache storage
- Cache storage:**
  - kitter-cache-v1 - `http://127.0.0.1:5500/pwa/`
- 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

The 'Cache storage' section shows a table of cached entries for the 'kitter-cache-v1' cache:

#	Name	Response-Type	Content-Type	Content-Length	Time Cached	Vary Header
0	/pwa/	basic	text/html	10,748	24/3/2024, 1...	Origin
1	/pwa/assets/css/style.css	basic	text/css	18,269	24/3/2024, 1...	Origin
2	/pwa/assets/images/cta-banner.png	basic	image/png	607,154	24/3/2024, 1...	Origin
3	/pwa/assets/images/cta-bg.jpg	basic	image/jpeg	13,809	24/3/2024, 1...	Origin
4	/pwa/assets/images/cta-icon.png	basic	image/png	4,965	24/3/2024, 1...	Origin
5	/pwa/assets/images/hero-banner.jpg	basic	image/jpeg	151,984	24/3/2024, 1...	Origin
6	/pwa/assets/images/product-1.jpg	basic	image/jpeg	8,941	24/3/2024, 1...	Origin
7	/pwa/assets/images/product-1_0.jpg	basic	image/jpeg	13,910	24/3/2024, 1...	Origin
8	/pwa/assets/images/product-2.jpg	basic	image/jpeg	20,174	24/3/2024, 1...	Origin
9	/pwa/assets/images/product-2_0.jpg	basic	image/jpeg	18,427	24/3/2024, 1...	Origin
10	/pwa/assets/images/product-3.jpg	basic	image/jpeg	18,191	24/3/2024, 1...	Origin

The table indicates that there are 17 total entries in the cache.



**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.