DATE:

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 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.

- What can we do with service workers?

1. You can dominate Network Traffic
2. You can Cache
3. You can manage Push Notifications
4. You can continue.

- What can't we do with service workers?

1. You can't access the window
2. You can't work it on port 80

DATE:

- Service Worker cycle

1. Registration - To install a service worker, you need to register it in your main JavaScript code.

2. Installation - Once the browsers registers a service worker, installation can be attempted. A service worker installation triggers. an install event on the installing service worker.

3. Activation - once a service worker has sucessfully installed, it transitions into the activation stage. When the new service worker activates, an activate agent revent is triggered in the activating service worker.

Implementation :

- serviceworker.js / sw.js

```
const cacheName = 'Ecommerce';
const staticAssets = [
    './',
    './index.html',
    './about.html',
    './drip.html',
    './electronics.html',
    './furniture.html',
    './general.html',
    './index.html',
    './laptops.html',
    './phones.html',
    './sneakers.html',
    './manifest.json',
    './style.css',
];

self.addEventListener('install',async e =>{
    const cache = await caches.open(cacheName);
    await cache.addAll(staticAssets);
    return self.skipWaiting();
});

self.addEventListener('activate',e =>{
    self.clients.claim();
});

self.addEventListener('fetch', async e => {
    const req = e.request;
    const url = new URL(req.url);
```

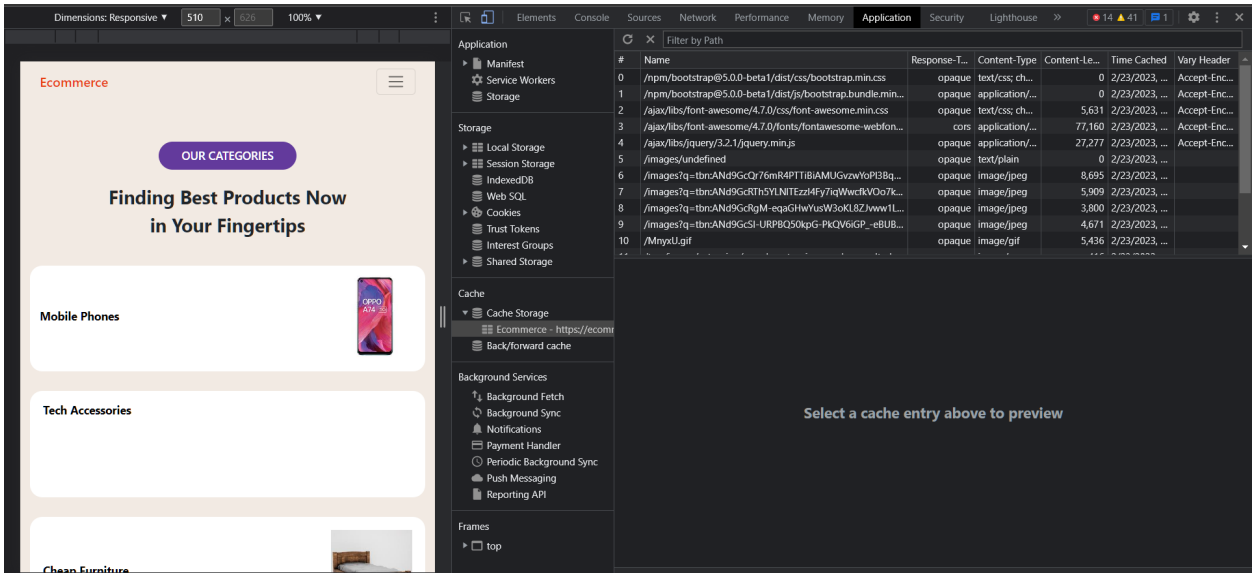
```
  if (url.origin === location.origin) {
    e.respondWith(cacheFirst(req));
  } else {
    e.respondWith(networkAndCache(req));
  }
});

async function cacheFirst(req) {
  const cache = await caches.open(cacheName);
  const cached = await cache.match(req);
  return cached || fetch(req);
}

async function networkAndCache(req) {
  const cache = await caches.open(cacheName);
  try {
    const fresh = await fetch(req);
    await cache.put(req, fresh.clone());
    return fresh;
  } catch (e) {
    const cached = await cache.match(req);
    return cached;
  }
}
```

Output :

Cache files -



Service Worker