Name: Yukta Bhatia

RollNo: 07

Class: D15A

Experiment-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 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.

What can we do with Service Workers?

- You can dominate **Network Traffic**
  You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

- You can **Cache**
  You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.

- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.

What can’t we do with Service Workers?

- You can’t access the **Window**
  You can’t access the window, therefore, You can’t manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.

- You can’t work it on **80 Port**
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

    <meta name="description" content="Web site created using create-react-app"/>
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <!-- Manifest File link -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <meta name="apple-mobile-web-app-status-bar" content="#aa7700">
    <meta name="theme-color" content="black">
</head>
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
     <script>
```

```
    // Add event listener to execute code when page loads
    window.addEventListener('load', () => {
      // Call registerSW function when page loads
      registerSW();
    });

    // Register the Service Worker
    async function registerSW() {
      // Check if browser supports Service Worker
      if ('serviceWorker' in navigator) {
        try {
          // Register the Service Worker named 'serviceworker.js'
          await navigator.serviceworker.register('serviceworker.js');
        }
        catch (e) {
          // Log error message if registration fails
          console.log('SW registration failed');
        }
      }
    }
  </script>
 </body>
</html>
```

index.css

```
* {
 font-family: 'Open Sans, sans-serif';
}

main {
 min-height: 83vmin;
 padding-bottom: 69px;
 width: 1200px;
```

```
  margin: auto;
}

.slick-next:before,
.slick-prev:before {
  color: #37BE91!important;
  font-size: 2rem!important;
}

.slick-dots li button:before{
  color: #37BE91!important;
}

.sectionTitle {
  position: relative;
  align-items: center;
  text-align: center;
}

.sectionTitle::before {
  content: '';
  width: 30px;
  height: 30px;
  margin-right: 10px;
  background: url('./assets/leaf.png') no-repeat;
  background-size: cover;
  display: inline-block;
}
.sectionTitle::after {
  content: '';
  width: 30px;
  height: 30px;
```

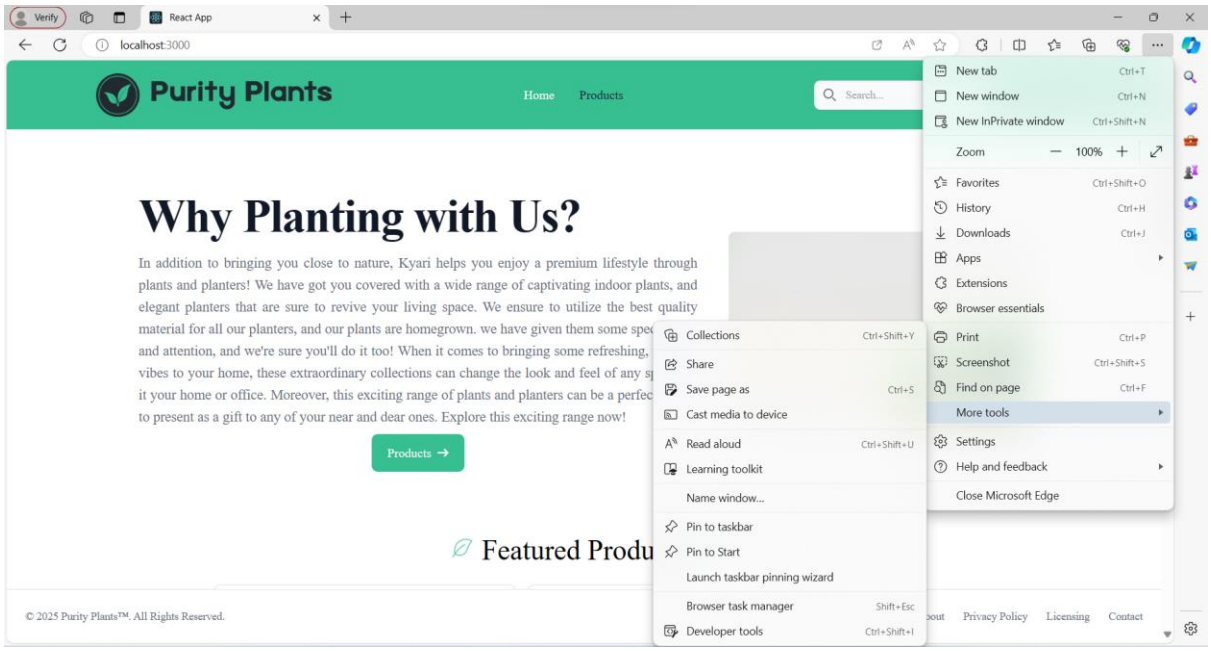
```
  margin-left: 10px;
  background: url('./assets/leaf.png') no-repeat;
  background-size: cover;
  display: inline-block;
}
```

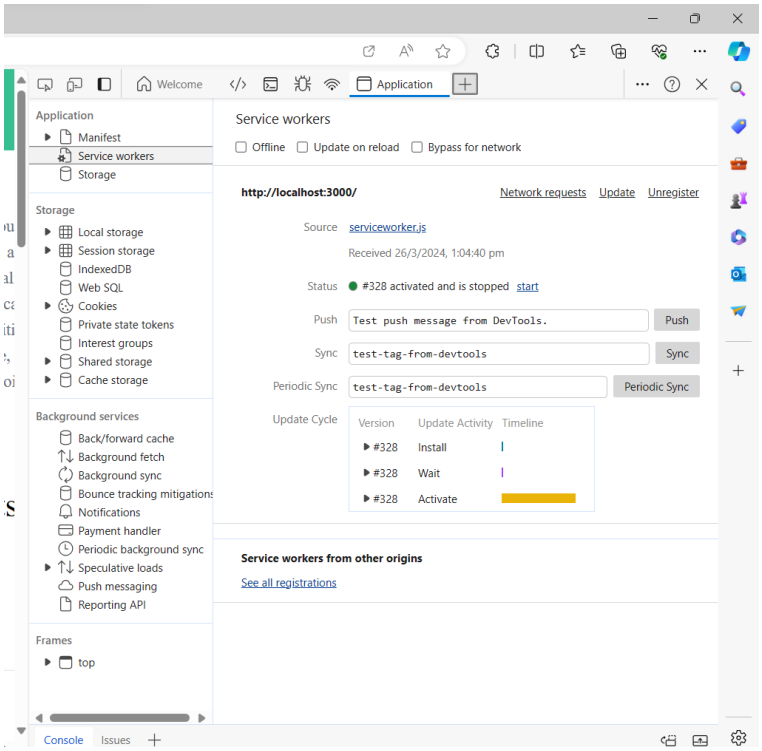
serviceworker.js

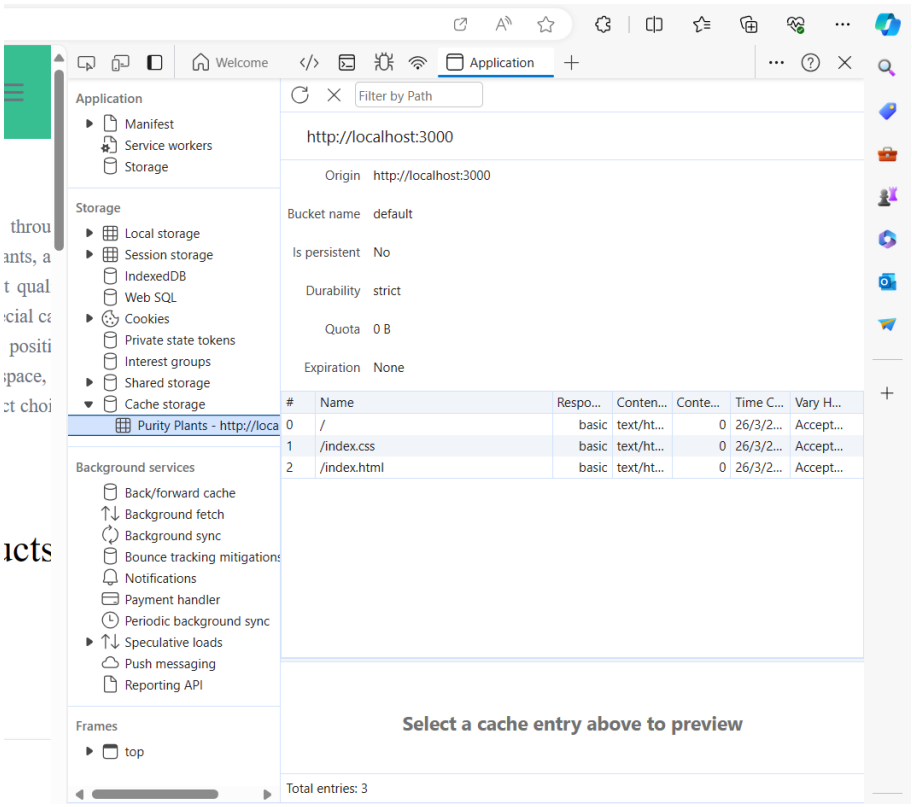
```
const cacheName = 'Purity Plants';
const assetsToCache = [
        '/',
        '/index.html',
        '/index.css',
]
this.addEventListener('install', event => {
        event.waitUntil(
                caches.open(cacheName)
                        .then(cache => {
                        return cache.addAll(assetsToCache);
                })
        );
});
this.addEventListener('activate', event => {
        event.waitUntil(
                caches.keys().then(cacheNames => {
                        return Promise.all(
                                cacheNames.filter(name => {
                                return name !== cacheName;
                                }).map(name => {
                                return caches.delete(name);
                                })
                        );
                })
        );
```

});

Output:


Purity Plants
Home
Products
Why Planting with Us?
In addition to bringing you close to nature, Kyari helps you enjoy a premium lifestyle through plants and planters! We have got you covered with a wide range of captivating indoor plants, and elegant planters that are sure to revive your living space. We ensure to utilize the best quality material for all our planters, and our plants are homegrown. we have given them some spe and attention, and we're sure you'll do it too! When it comes to bringing some refreshing, vibes to your home, these extraordinary collections can change the look and feel of any s it your home or office. Moreover, this exciting range of plants and planters can be a perfec to present as a gift to any of your near and dear ones. Explore this exciting range now!
Products →
Featured Produ
© 2025 Purity Plants™. All Rights Reserved.
Privacy Policy
Licensing
Contact
New tab Ctrl+T
New window Ctrl+N
New InPrivate window Ctrl+Shift+N
Zoom 100%
Favorites Ctrl+Shift+O
History Ctrl+H
Downloads Ctrl+J
Apps
Extensions
Browser essentials
Collections Ctrl+Shift+Y
Share
Save page as Ctrl+S
Cast media to device
Read aloud Ctrl+Shift+U
Learning toolkit
Name window...
Pin to taskbar
Pin to Start
Launch taskbar pinning wizard
Browser task manager Shift+Esc
Developer tools Ctrl+Shift+I
Print Ctrl+P
Screenshot Ctrl+Shift+S
Find on page Ctrl+F
More tools
Settings
Help and feedback
Close Microsoft Edge


Welcome
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigations
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Service workers
Offline
Update on reload
Bypass for network
http://localhost:3000/
Network requests
Update
Unregister
Source
serviceworker.js
Received 26/3/2024, 1:04:40 pm
Status
#328 activated and is stopped
start
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
#328
Install
#328
Wait
#328
Activate
Service workers from other origins
See all registrations
Console
Issues



Conclusion: Registered a service worker, and completed the installation and activation process for a new service worker for the PWA.