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:

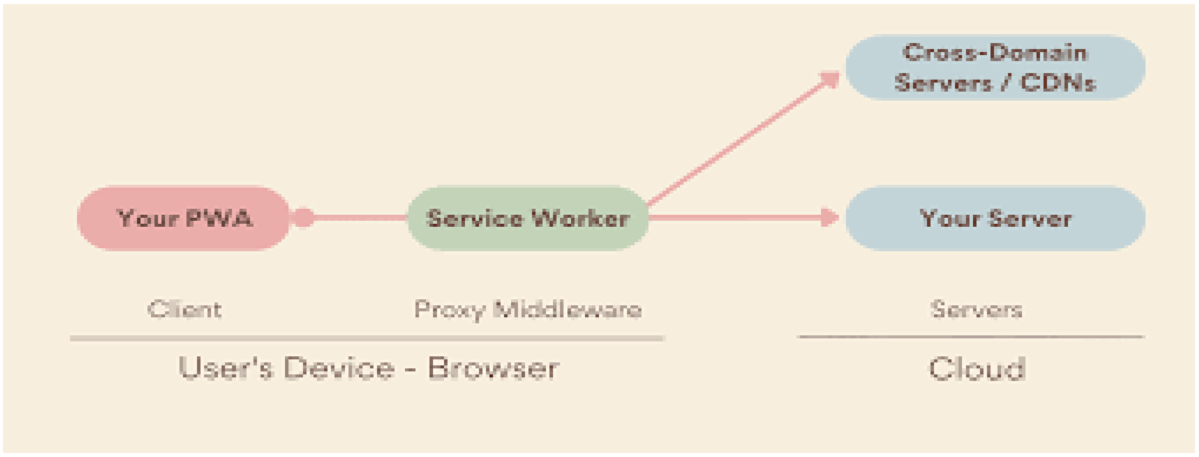
A service worker is a type of web worker that runs in the background of a web application, separate from the main JavaScript thread. It enables developers to implement features like offline caching, push notifications, background synchronization, and more, making web applications behave more like native apps. Service workers play a crucial role in creating Progressive Web Apps (PWAs) because they allow web apps to offer a reliable, fast, and engaging user experience, even in challenging network conditions.

Here are some key concepts and capabilities of service workers in PWAs:

1. **Background Processing:** Service workers run independently in the background, allowing them to perform tasks without interfering with the user interface. This enables features like background synchronization and push notifications.
2. **Offline Caching:** One of the primary benefits of service workers is their ability to cache resources for offline use. Developers can define caching strategies to store assets like HTML, CSS, JavaScript, images, and API responses locally, allowing the web app to continue functioning even when the device is offline. Improved.
3. **Performance:** By caching assets locally, service workers can significantly improve the performance of web apps, especially on repeat visits. Cached resources can be served quickly from the device's storage, reducing latency and enhancing the overall user experience.
4. **Push Notifications:** Service workers enable web apps to receive push notifications, similar to native mobile apps. This capability enhances user engagement by allowing apps to send timely updates and alerts to users, even when the app is not actively open in the browser.
5. **Background Sync:** Service workers can perform background synchronization tasks, such as syncing data with a server when the network connection is available. This ensures that the app stays up-to-date with the latest information and provides a seamless experience for users.

6. **Installable Web Apps**: Service workers, combined with a web app manifest (JSON file), enable web apps to be installed on a user's device like a native app. This installation creates an icon on the device's home screen and provides an app-like experience, including offline access and full-screen mode.
7. **Security and Scope:** Service workers operate within a specific scope (e.g., the root directory of the app), and they adhere to security policies such as the same-origin policy. This ensures that service workers cannot access resources from other origins unless explicitly allowed.

Overall, service workers are a powerful tool for web developers to create high-performance, reliable, and engaging web applications that offer native-like experiences across different devices and platforms. However, it's important to note that service workers require careful implementation and testing to ensure proper functionality and compatibility with various browsers and devices.



Code:

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

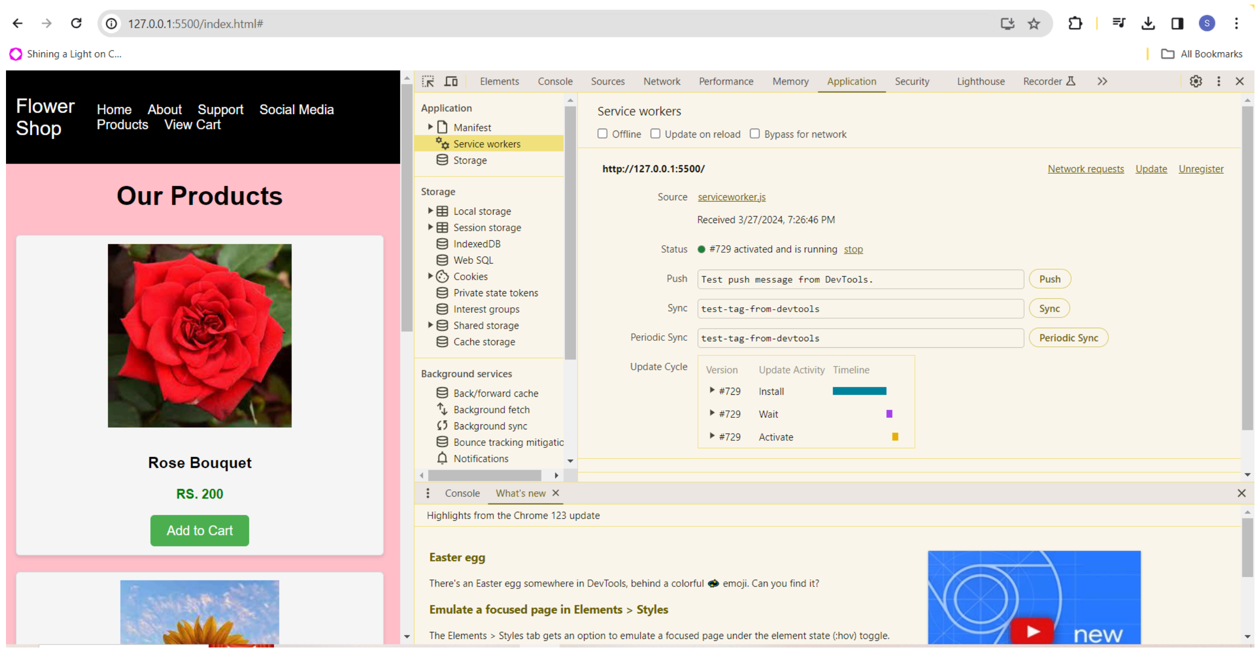
1. Create the Service Worker File (service-worker.js)

```
// Service Worker Installation
self.addEventListener('install', event => {
    console.log('Service Worker installed');
});

// Service Worker Activation
self.addEventListener('activate', event => {
    console.log('Service Worker activated');
});

// Fetch Event - Cache and Network Strategy
self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request)
            .then(response => {
                // Cache hit - return response
                if (response) {
                    return response;
                }

                // Clone the request
                let fetchRequest = event.request.clone();

                // Make a network request
                return fetch(fetchRequest)
                    .then(networkResponse => {
                        // Check if we received a valid response
                        if (!networkResponse || networkResponse.status !==
200 || networkResponse.type !== 'basic') {
                            return networkResponse;
                        }

                        // Clone the network response
                        let responseToCache = networkResponse.clone();

                        // Cache the response
                        caches.open('your-cache-name')
                            .then(cache => {
                                cache.put(event.request, responseToCache);
                            });

                        return networkResponse;
                    });
            })
    );
});
```

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

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

Output :


127.0.0.1:5500/index.html#

Flower Shop — Home About Support Social Media Products View Cart

Our Products

Rose Bouquet
RS. 200
Add to Cart

Application > Service workers

Service workers
☐ Offline ☐ Update on reload ☐ Bypass for network

http://127.0.0.1:5500/ — Network requests Update Unregister

Source serviceworker.js
Received 3/27/2024, 7:26:46 PM
Status ● #729 activated and is running stop
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
#729	Install	
#729	Wait	
#729	Activate	

Highlights from the Chrome 123 update

Easter egg
There's an Easter egg somewhere in DevTools, behind a colorful emoji. Can you find it?
Emulate a focused page in Elements > Styles
The Elements > Styles tab gets an option to emulate a focused page under the element state (:hov) toggle.

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