Experiment No: - 9

Aim:- To implement Service worker events like fetch, sync and push for E-commerce PWA.

Theory:-

A service worker is a script that your browser runs in the background, separate from a web page, and can be used to handle tasks such as push notifications, background synchronization, and caching resources to improve performance and offline functionality.

Fetch Event:

The fetch event is triggered whenever the browser makes a network request, such as fetching resources like HTML, CSS, JavaScript files, images, or API calls.
Service workers intercept these fetch requests, allowing them to modify the request or response, or even serve cached responses from the cache storage.
This event is commonly used to implement caching strategies, including serving cached resources when the network is slow or offline, or fetching fresh content from the network when available.

Sync Event:

The sync event is a special event available in service workers that allows you to perform background synchronization tasks.
It enables your PWA to synchronize data with the server even when the app is not actively in use or when the device is offline.
Sync events are useful for tasks like sending queued data to the server, updating content in the background, or refreshing cached resources periodically.

Push Event:

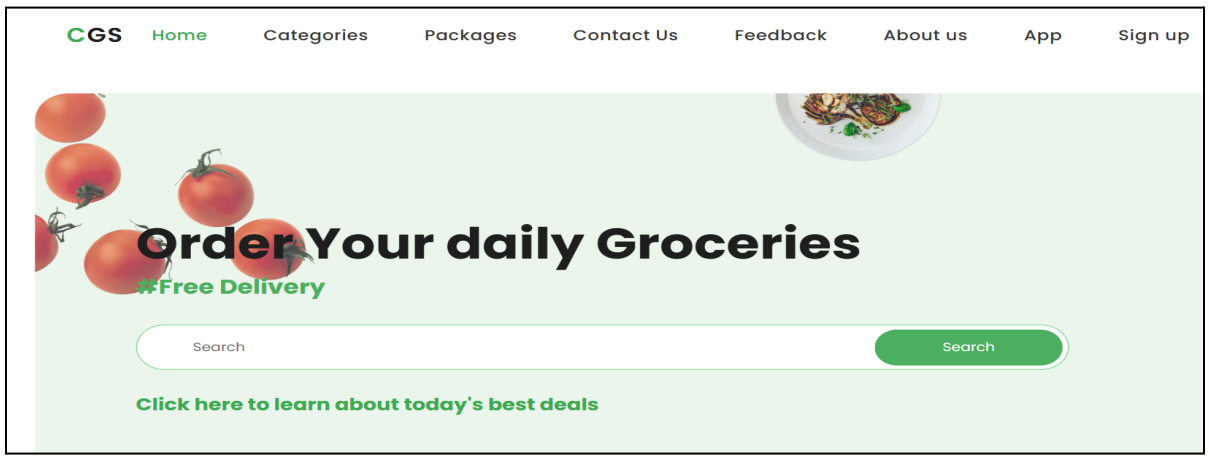
The push event occurs when the service worker receives a push notification from a server, typically initiated by a backend server or another client application.
This event allows service workers to display notifications to users, even when the web app is not currently open or active.
Push notifications are effective for engaging users with timely updates, notifications.

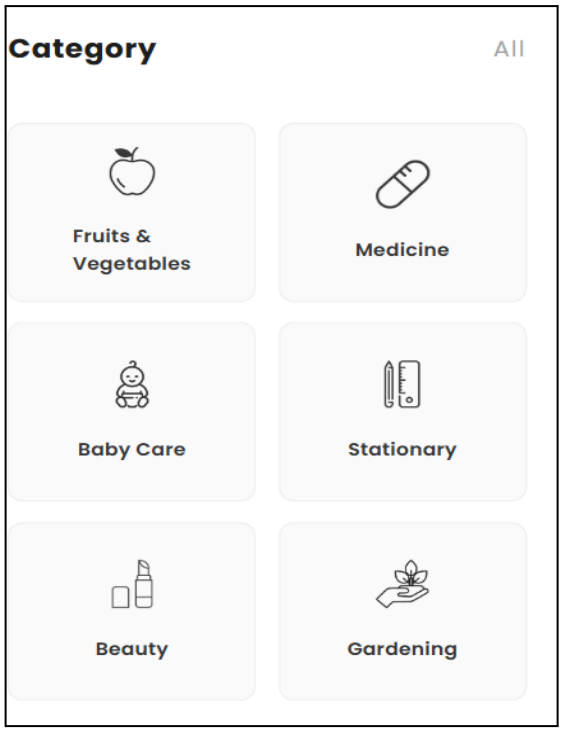
ServiceWorker.js:

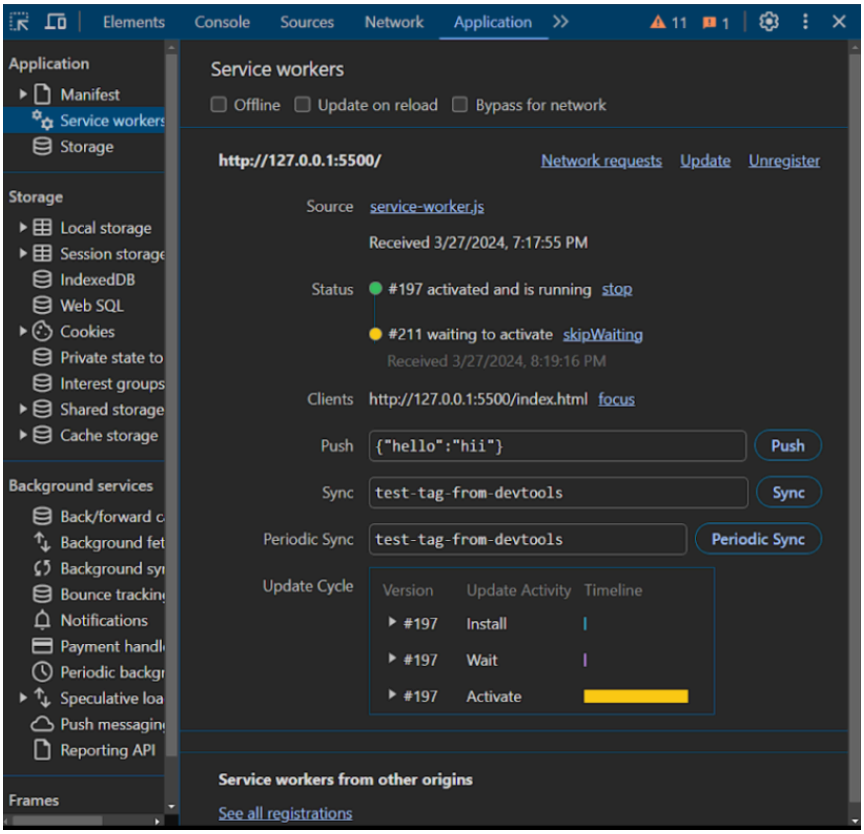
```
self.addEventListener('fetch', event => event.respondwith(
caches.match(event.request)
.then(response => { if (response) {
return response;
}
return fetch(event.request);
);
})
console.log("Fetch Successful");
});

self.addEventListener('sync', event => {
console.log("Sync Successful");
});

self.addEventListener('push', event => {
if (self.Notification.permission
===
'granted') {
const payload= event.data? event.data.text(): 'Default message'; event.waitUntil(
self.registration. showNotification('Title', {
body: payload,
})
```

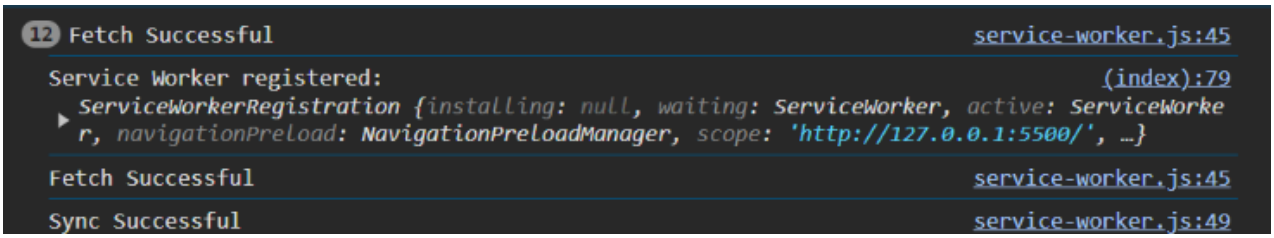
```
);
console.log("Push Successful");
} else {
}
});
console.warn('Notification permission has not been granted.');
```

Output:









Conclusion:

Hence, successfully implemented a service worker for the E-commerce Progressive Web App (PWA) involves writing the service worker code to handle caching and network requests, registering the service worker in the main JavaScript file of the app, and completing the install and activation process.