Experiment No: 9

Name:- Vishal Gori **Batch**: A
Roll No.: 18 **Division**:- D15B

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

THEORY:-

1. **Fetch Event:**

- The fetch event is triggered whenever a resource (like an HTML page, CSS file, JavaScript file, image, etc.) is fetched from the network.
- In the service worker, you can intercept these fetch requests and provide a response from the cache if the requested resource is available there. This is useful for offline browsing and improving performance by serving cached assets.
- The respondWith method allows you to intercept the fetch request and respond with a cached version if available, or fetch the resource from the network and cache it for future use if not.
- In the provided code, the service worker intercepts fetch requests, checks if the requested resource is available in the cache, and responds with the cached version if found. If not, it fetches the resource from the network, caches it, and then returns the response.

2. **Sync Event:**

- The sync event is triggered when the browser is online and the service worker has registered a sync event.
- It allows you to perform background synchronization tasks, such as sending data to the server, even when the user is offline or the network connection is unreliable.
- In the provided code, the service worker listens for a sync event with the tag name 'helloSync'. When this event is triggered, it logs a message and provides a placeholder for handling the synchronization task, such as sending data to the server.

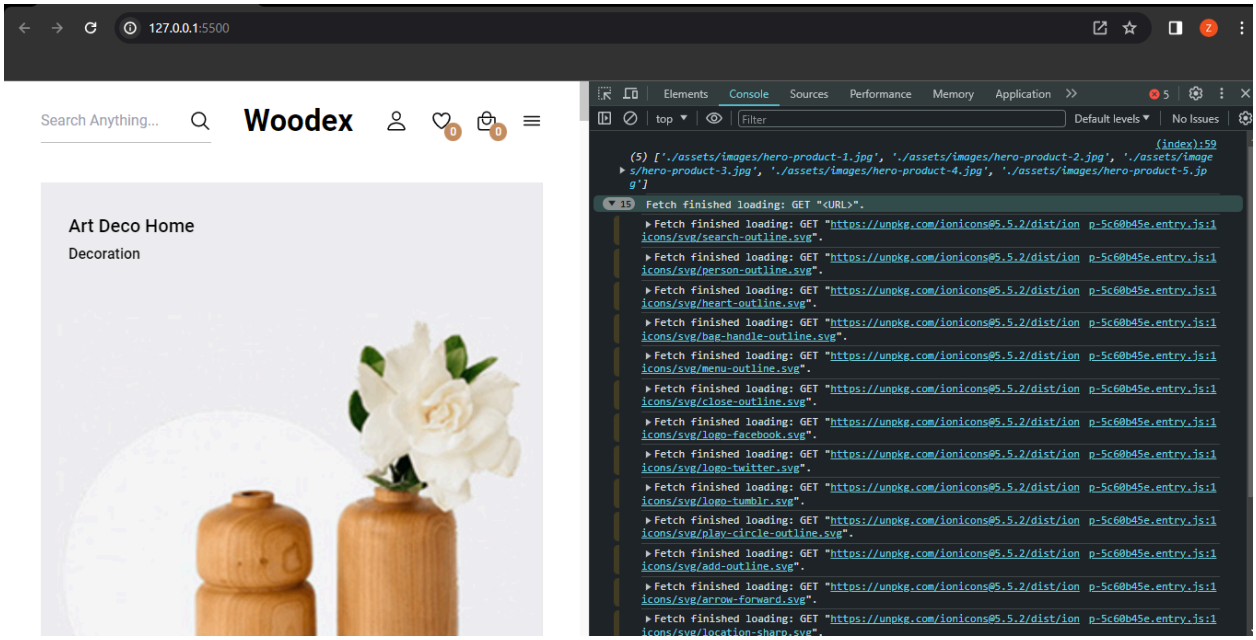
3. **Push Event:**

- The push event is triggered when a push notification is received from a push service, such as Firebase Cloud Messaging (FCM).
- It allows you to display notifications to users even when they are not actively using your website or web app.
- In the provided code, the service worker listens for a push event. When a push notification is received and it contains data with a specific method (in this case, 'pushMessage'), it constructs a notification with the provided message and icon and displays it to the user.

'Fetch' Event to Cache Data:

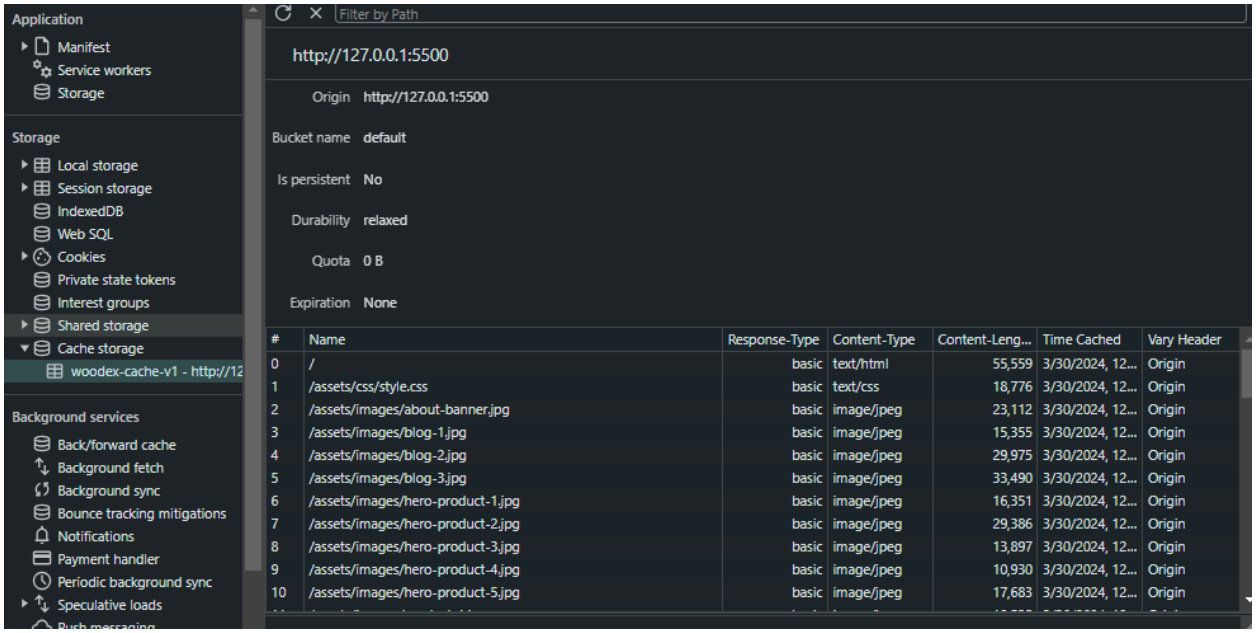
- On clicking the profile Icon  All the files and folders that I have set will be Cached

```
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        if (response) {
          return response;
        }
        return fetch(event.request)
          .then(response => {
            let responseClone = response.clone();
            caches.open(CACHE_NAME)
              .then(cache => {
                cache.put(event.request, responseClone);
              });
            return response;
          })
          .catch(error => {
            console.error('Fetch failed:', error);
          });
      })
  );
});
```



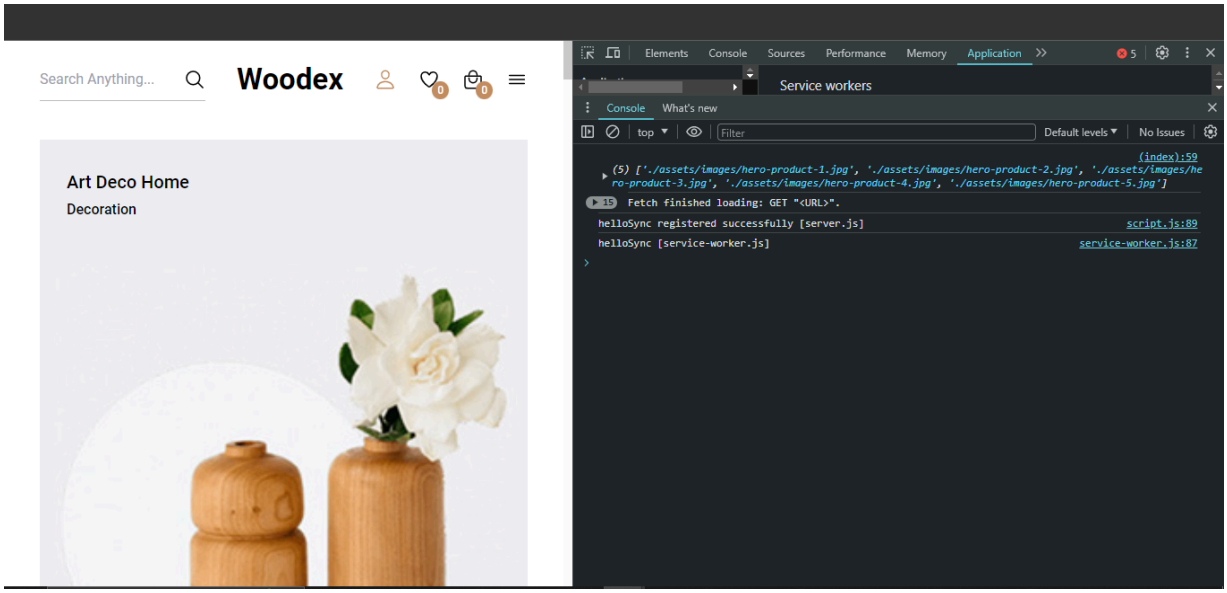
(index):59
(5) ['./assets/images/hero-product-1.jpg', './assets/images/hero-product-2.jpg', './assets/images/hero-product-3.jpg', './assets/images/hero-product-4.jpg', './assets/images/hero-product-5.jpg']
15 Fetch finished loading: GET "<URL>".
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/logo-facebook.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/add-outline.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/heart-outline.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/bag-handle-outline.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/logo-twitter.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/search-outline.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/logo-tumblr.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/play-circle-outline.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/person-outline.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/call-sharp.svg". p-5c60b45e.entry.js:1
Fetch finished loading: GET "https://unpkg.com/ionicons@5.5.2/dist/ionicons/svg/... p-5c60b45e.entry.js:1

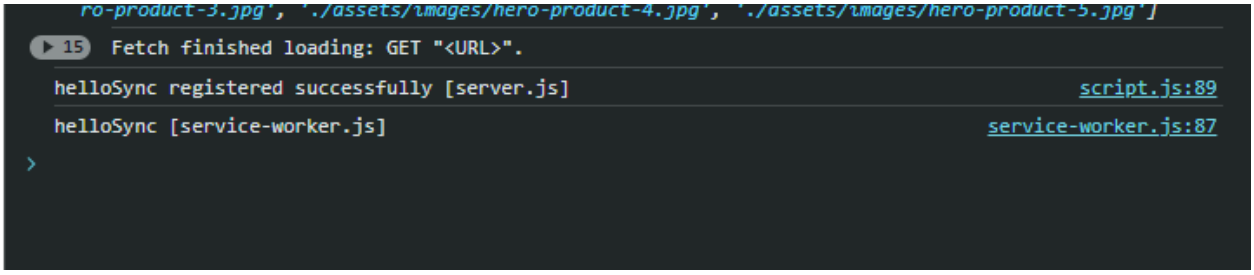
Cached Data



‘Sync’ Event to Sync Data:

```
self.addEventListener('sync', event => {
  if (event.tag === 'helloSync') {
    console.log("helloSync [service-worker.js]");
    // Handle the sync event here
    // You can perform actions such as sending data to the server
  }
});
```

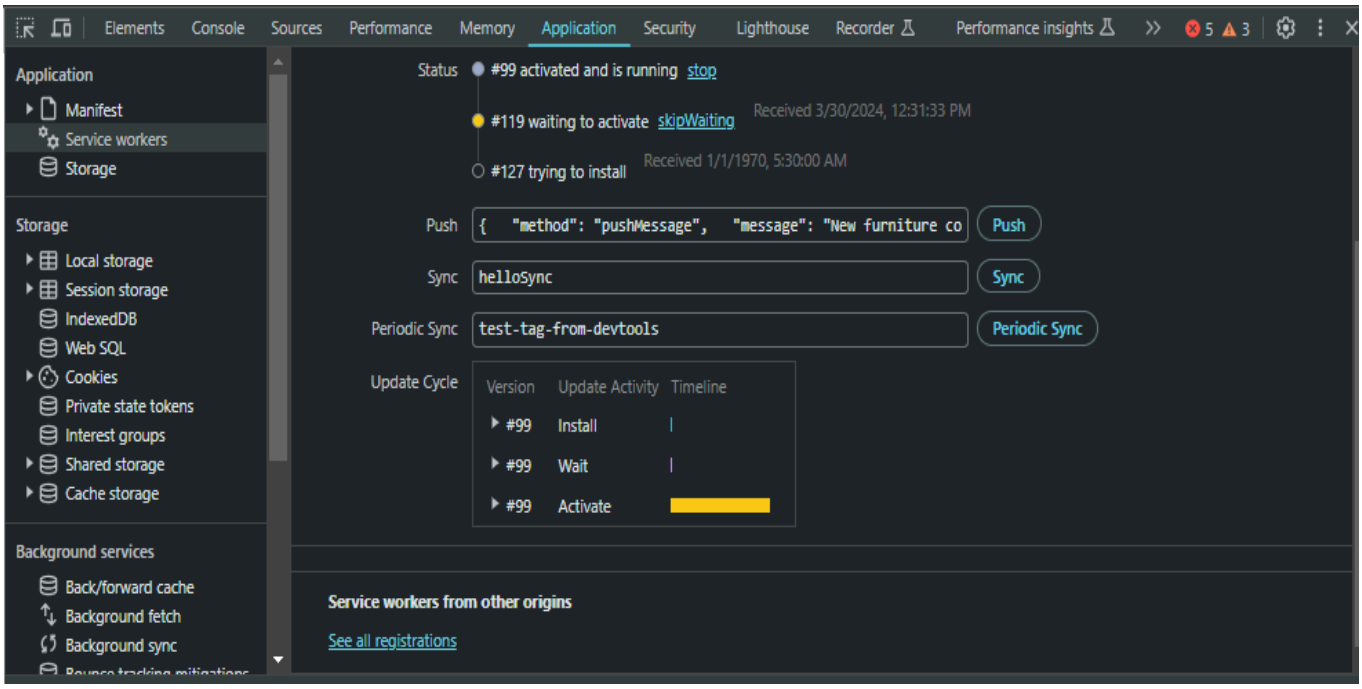


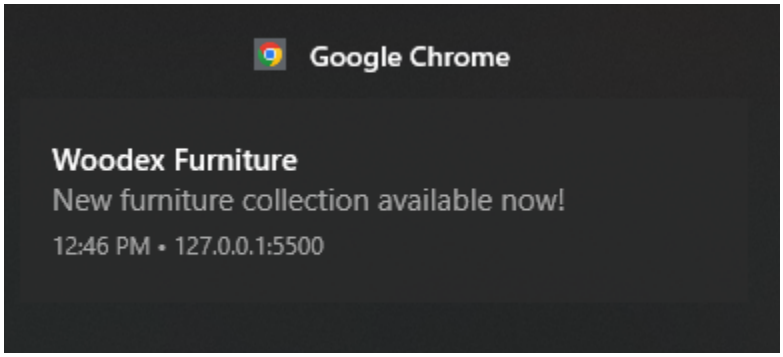


'Push' Event to give Notification:

```
self.addEventListener('push', function(event) {
  if (event && event.data) {
    const data = event.data.json();
    if (data.method === "pushMessage") {
      const title = 'Woodex Furniture';
      const options = {
        body: data.message,
        icon: 'logo.png' // Path to your Woodex logo image
      };

      event.waitUntil(
        self.registration.showNotification(title, options)
      );
    }
  }
});
```





Conclusion:-

In conclusion, through the integration and registration of a service worker in my E-commerce Progressive Web App (PWA), I've significantly enhanced its functionality. With the implementation of push notifications, offline caching through fetch events, and data synchronization using sync events, my PWA now offers improved user experience and accessibility, even in low or no connectivity scenarios.