Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Experiment 8

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the ToDo PWA.

Theory:

Service Workers in Progressive Web Apps (PWAs)

A Service Worker is a background JavaScript script that operates separately from the main browser thread. It enhances web apps by enabling features such as:

- Offline caching – Stores assets locally for offline use.
- Push notifications – Sends updates even when the app isn't open.
- Background sync – Queues data and syncs when the user reconnects.
- Network request interception – Serves cached content when offline.

Importance of Service Workers in PWAs

Service Workers form the backbone of PWA functionality by enabling:

- Offline Support: Ensures access to critical resources without internet.
- Faster Performance: Cached files load instantly, reducing wait times.
- Reliability: Keeps the app functional under weak or no network.
- Background Capabilities: Handles background tasks like syncing and notifications.

Service Worker Lifecycle

A Service Worker follows a three-phase lifecycle:

1. Registration

Initiated from the main JavaScript file:

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js')
    .then(reg => console.log('Service Worker registered:', reg.scope))
    .catch(err => console.log('Registration failed:', err));
}
```

- The browser fetches and parses sw.js.

2. Installation

Occurs on first registration or when the file changes. It's used to cache essential files:

```
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('v1').then(cache => {
      return cache.addAll([
        '/',
        '/index.html',
        '/styles.css',
        '/app.js',
        '/logo.png'
      ]);
    })
  );
});
```

3. Activation
Follows installation. Used to remove outdated caches:

```
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(names => {
      return Promise.all(
        names.filter(name => name !== 'v1')
             .map(name => caches.delete(name))
      );
    })
  );
});
```

Handling Fetch Requests
Service Workers intercept fetch calls and respond with cached resources or fall back to the network:

```
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(resp => resp || fetch(event.request))
  );
});
```

Browser Support & Security

- Supported in Chrome, Firefox, Edge, and partially in Safari.
- Requires HTTPS for security (except on localhost during development).

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Code:

1. **Service worker registration**

```
<!-- Service Worker Registration Script -->
<script>
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/Flutter_PWA/taskverse/service-worker.js')
      .then((registration) => {
        console.log('Service Worker registered with scope:', registration.scope);
      })
      .catch((error) => {
        console.log('Service Worker registration failed:', error);
      });
  }
</script>
```

2. **Service worker lifecycle**

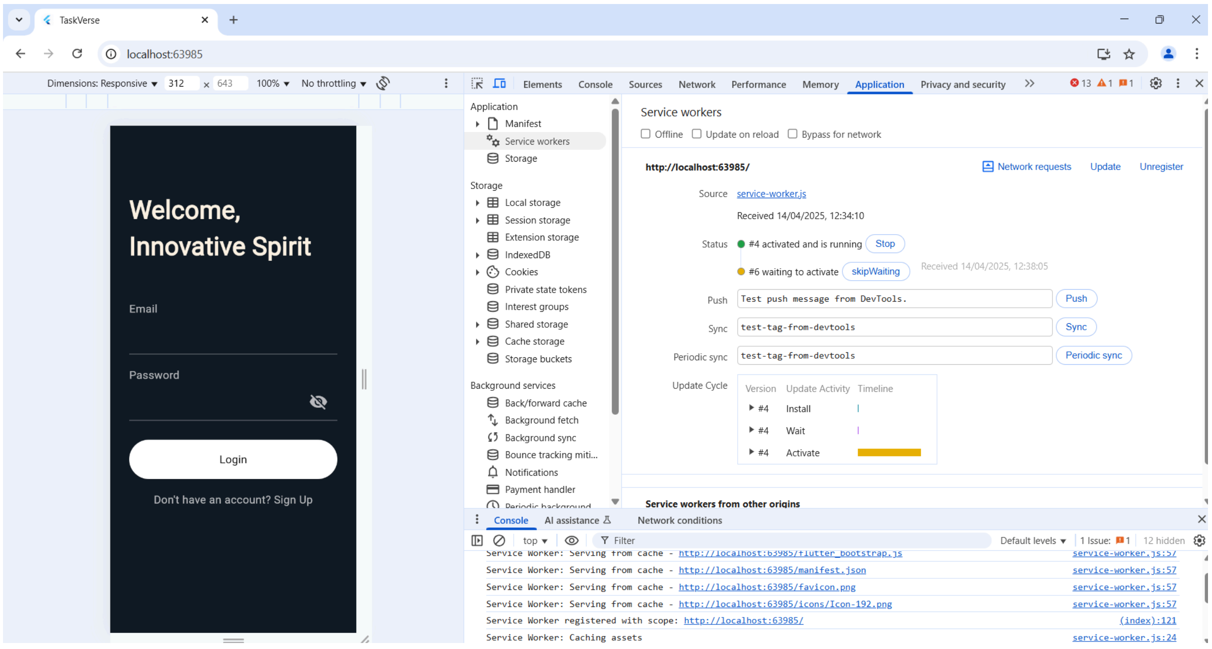
```
const CACHE_NAME = 'taskverse-cache-v1';
const urlsToCache = [
  '/Flutter_PWA/taskverse/',
  '/Flutter_PWA/taskverse/index.html',
  '/Flutter_PWA/taskverse/manifest.json',
  '/Flutter_PWA/taskverse/favicon.png',
  '/Flutter_PWA/taskverse/flutter_bootstrap.js',
  '/Flutter_PWA/taskverse/splash/img/light-1x.png',
  '/Flutter_PWA/taskverse/splash/img/light-2x.png',
  '/Flutter_PWA/taskverse/splash/img/dark-1x.png',
  '/Flutter_PWA/taskverse/splash/img/dark-2x.png',
  '/Flutter_PWA/taskverse/icons/Icon-192.png',
  '/Flutter_PWA/taskverse/icons/Icon-512.png',
  '/Flutter_PWA/taskverse/icons/Icon-maskable-192.png',
  '/Flutter_PWA/taskverse/icons/Icon-maskable-512.png',
  // Add other resources that are critical to load when offline
];
// Install event: Cache important assets
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('Service Worker: Caching assets');
        return cache.addAll(urlsToCache);
      })
      .catch((error) => {
        console.error('Service Worker: Failed to cache', error);
    })
  );
  self.skipWaiting(); // Activate new SW immediately
});
// Activate event: Clean up old caches
self.addEventListener('activate', (event) => {
  const cacheWhitelist = [CACHE_NAME];
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (!cacheWhitelist.includes(cacheName)) {
            console.log(`Service Worker: Deleting old cache ${cacheName}`);
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
  self.clients.claim(); // Take control immediately
});
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request).then((cachedResponse) => {
      if (cachedResponse) {
        console.log(`Service Worker: Serving from cache - ${event.request.url}`);
        return cachedResponse;
      }
      return fetch(event.request)
        .then((networkResponse) => {
          // Only cache valid GET responses
          if (
            networkResponse &&
            networkResponse.status === 200 &&
            event.request.method === 'GET'
          ) {
            const responseClone = networkResponse.clone(); // 👈 Clone before reading
            caches.open(CACHE_NAME).then((cache) => {
              cache.put(event.request, responseClone);
            });
          }
          return networkResponse;
        })
```

```
            .catch((error) => {
              console.error(`Service Worker: Fetch failed - ${event.request.url}`, error);
              return caches.match('/Flutter_PWA/taskverse/offline.html');
            });
          })
        );
      });
```

Console:

```
Live reload enabled.                                          (index):110
ServiceWorker registration successful                          (index):73
```

Service worker registered:



Service workers

Offline | Update on reload | Bypass for network

http://localhost:63985/

Source service-worker.js

Received 14/04/2025, 12:34:10

Status #4 activated and is running

#6 waiting to activate Received 14/04/2025, 12:38:05

Push: Test push message from DevTools.

Sync: test-tag-from-devtools

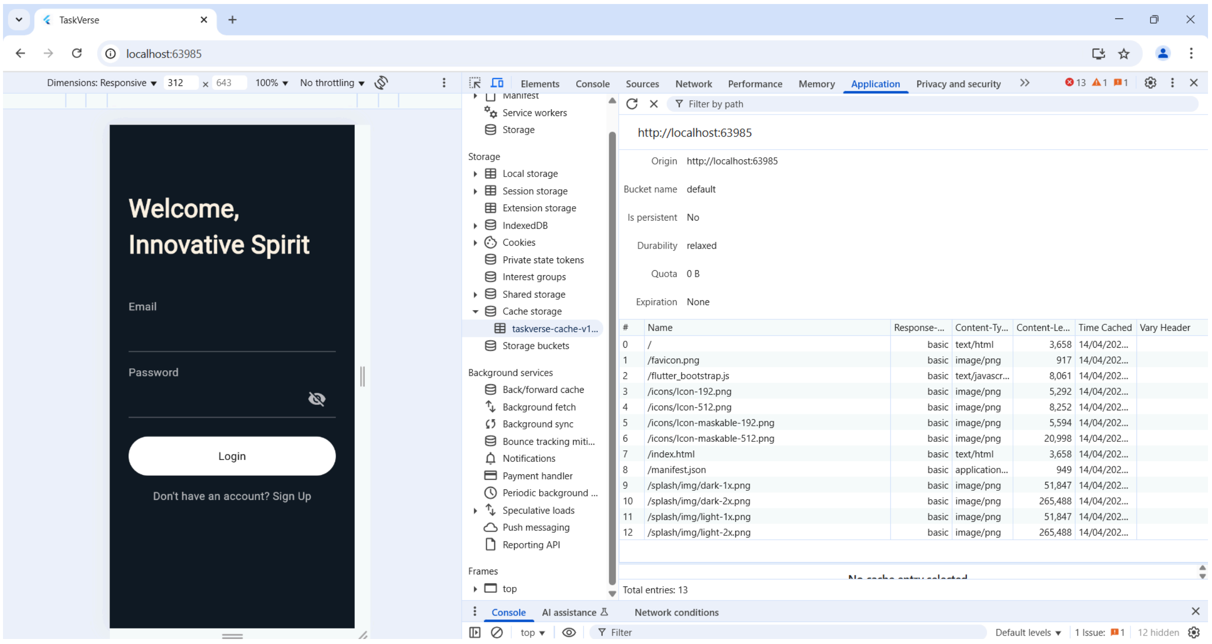
Periodic sync: test-tag-from-devtools

Version	Update Activity	Timeline
#4	Install	
#4	Wait	
#4	Activate	

Service workers from other origins

```
Service Worker: Serving from cache - http://localhost:63985/flutter_bootstrap.js   service-worker.js:57
Service Worker: Serving from cache - http://localhost:63985/manifest.json          service-worker.js:57
Service Worker: Serving from cache - http://localhost:63985/favicon.png            service-worker.js:57
Service Worker: Serving from cache - http://localhost:63985/icons/Icon-192.png     service-worker.js:57
Service Worker registered with scope: http://localhost:63985/                             (index):121
Service Worker: Caching assets                                                     service-worker.js:24
```

Cache:



http://localhost:63985

Origin http://localhost:63985

Bucket name default

Is persistent No

Durability relaxed

Quota 0 B

Expiration None

#	Name	Response-...	Content-Ty...	Content-Le...	Time Cached	Vary Header
0	/	basic	text/html	3,658	14/04/202...	
1	/favicon.png	basic	image/png	917	14/04/202...	
2	/flutter_bootstrap.js	basic	text/javascr...	8,061	14/04/202...	
3	/icons/Icon-192.png	basic	image/png	5,292	14/04/202...	
4	/icons/Icon-512.png	basic	image/png	8,252	14/04/202...	
5	/icons/Icon-maskable-192.png	basic	image/png	5,594	14/04/202...	
6	/icons/Icon-maskable-512.png	basic	image/png	20,998	14/04/202...	
7	/index.html	basic	text/html	3,658	14/04/202...	
8	/manifest.json	basic	application...	949	14/04/202...	
9	/splash/img/dark-1x.png	basic	image/png	51,847	14/04/202...	
10	/splash/img/dark-2x.png	basic	image/png	265,488	14/04/202...	
11	/splash/img/light-1x.png	basic	image/png	51,847	14/04/202...	
12	/splash/img/light-2x.png	basic	image/png	265,488	14/04/202...	

Total entries: 13

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

GitHub Link: https://github.com/BKCODE2003/Flutter_PWA

Conclusion:

By successfully coding and registering a service worker, and completing its installation and activation process, we have enabled core Progressive Web App functionalities such as offline support and background processing. This significantly enhances the reliability and performance of the eCommerce PWA, allowing it to function smoothly even in low or no network conditions. Implementing a service worker is a crucial step toward delivering a fast, resilient, and app-like experience to users across different platforms.