

### Experiment 08

**Aim:** To code and register a service worker, and complete the install and activation process for a new service worker for the Weather app PWA (Progressive Web App).

**Theory:**

A **Service Worker** is a background script that acts as a proxy between a web application, the browser, and the network. It enables advanced features like **offline access**, **caching**, **push notifications**, and **background sync**, which are essential for building **Progressive Web Apps (PWAs)**.

Key features of a PWA include:

- Offline support
- Improved performance through caching
- App-like behavior
- Ability to work independently of network conditions

Service workers follow a **lifecycle**, consisting of the following main events:

1. **Install**
2. **Activate**
3. **Fetch**

**Implementation:**

1. `navigator.serviceWorker.register()`: This method registers the service worker when the page loads. It checks if service workers are supported in the browser and then registers `sw.js` located at the root level.

2. `self.addEventListener('install')`: Triggered when the service worker is first installed. This is where assets are typically cached for offline use. In this case, we simply log the installation and immediately activate using `self.skipWaiting()`.

3. **`self.skipWaiting()`**: Forces the newly installed service worker to activate immediately rather than waiting for the old one to be terminated.

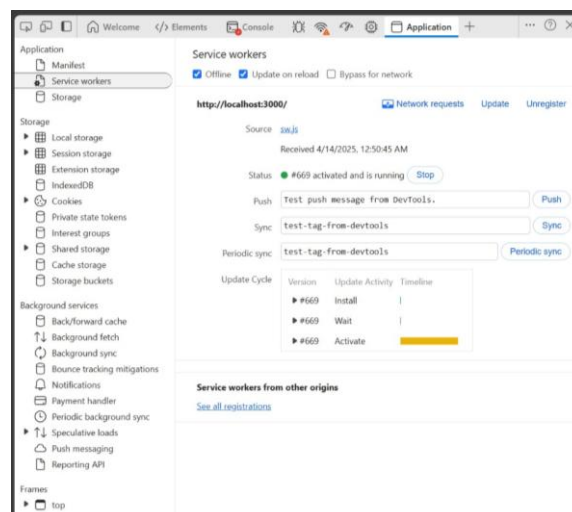
4. **`self.addEventListener('activate')`**: Triggered after the service worker is installed. This event is used to clean up old caches or perform updates. In our code, it's used to log activation status.

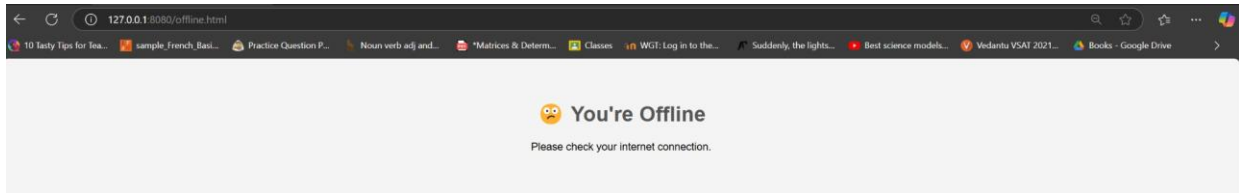
5. **`self.addEventListener('fetch')`**: Intercepts every network request. This is where caching or fallback responses are generally served. In our implementation, it simply logs each fetch request.

### Output:

We tested the application using:

- **Chrome DevTools > Application Tab** to inspect service worker status
- **Simulated offline mode** in DevTools to check behavior when network is turned off
- Verified whether assets were served from the cache or network



**Conclusion:**

Through this implementation, we successfully:

- Registered and activated a service worker
- Understood and used lifecycle events: install, activate, fetch
- Observed basic PWA behavior through browser DevTools
- Laid the foundation for caching strategies and offline support in PWAs

This setup ensures that our PWA behaves more like a native app and provides a better user experience, especially in low or no connectivity environments.