Name: Ameya M. Angne Class: D15B
Roll no.: 01

PWA Experiment No. 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:

Service Worker
Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.

What can we do with Service Workers?

- You can dominate Network Traffic
  You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

- You can Cache
  You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.

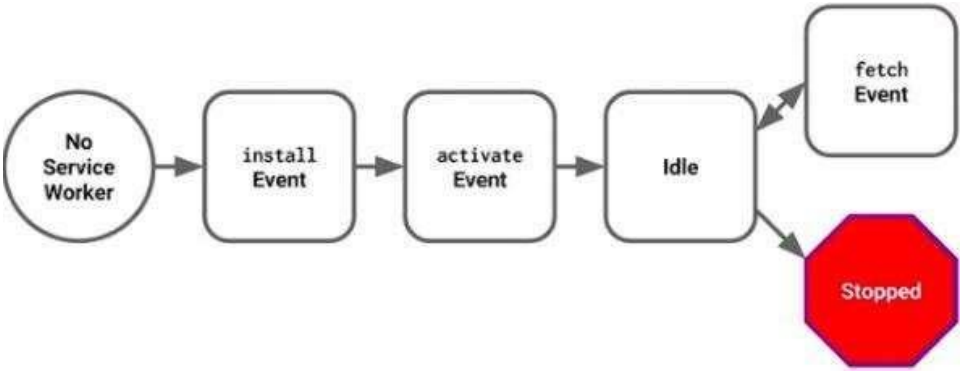
- You can manage Push Notifications
  You can manage push notifications with Service Worker and show any information message to the user.
- You can Continue
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can’t we do with Service Workers?

- You can’t access the Window
  You can’t access the window, therefore, You can’t manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.

- You can’t work it on 80 Port
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Service Worker Cycle:



A service worker goes through three steps in its life cycle:

- Registration

```
<script>

   if ('serviceWorker' in navigator) {
navigator.serviceWorker.register('/serviceworker.js')
      .then(function (registration) {
        console.log('Registration successful, scope is:', registration.scope);
      })
      .catch(function (error) {          console.log('Service worker
registration failed, error:', error);
      });
   }
 </script>
```

- Installation self.addEventListener("install",
  function (event) { event.waitUntil(preLoad());
  });

```
var filesToCache = ['/',
        '/index.html',
];
```

- Activation self.addEventListener('activate',
  function (event) {
  ```
        event.waitUntil(
                // Perform cleanup tasks or cache management
  here          // For example, deleting outdated caches
              caches.keys().then(function (cacheNames) {
                    return Promise.all(
   cacheNames.filter(function (cacheName) {
                                        version
                                }).map(function (cacheName) {
                                // Delete the outdated cache
                                        return caches.delete(cacheName);
                                })
                        );
                })
        );
  });
  ```

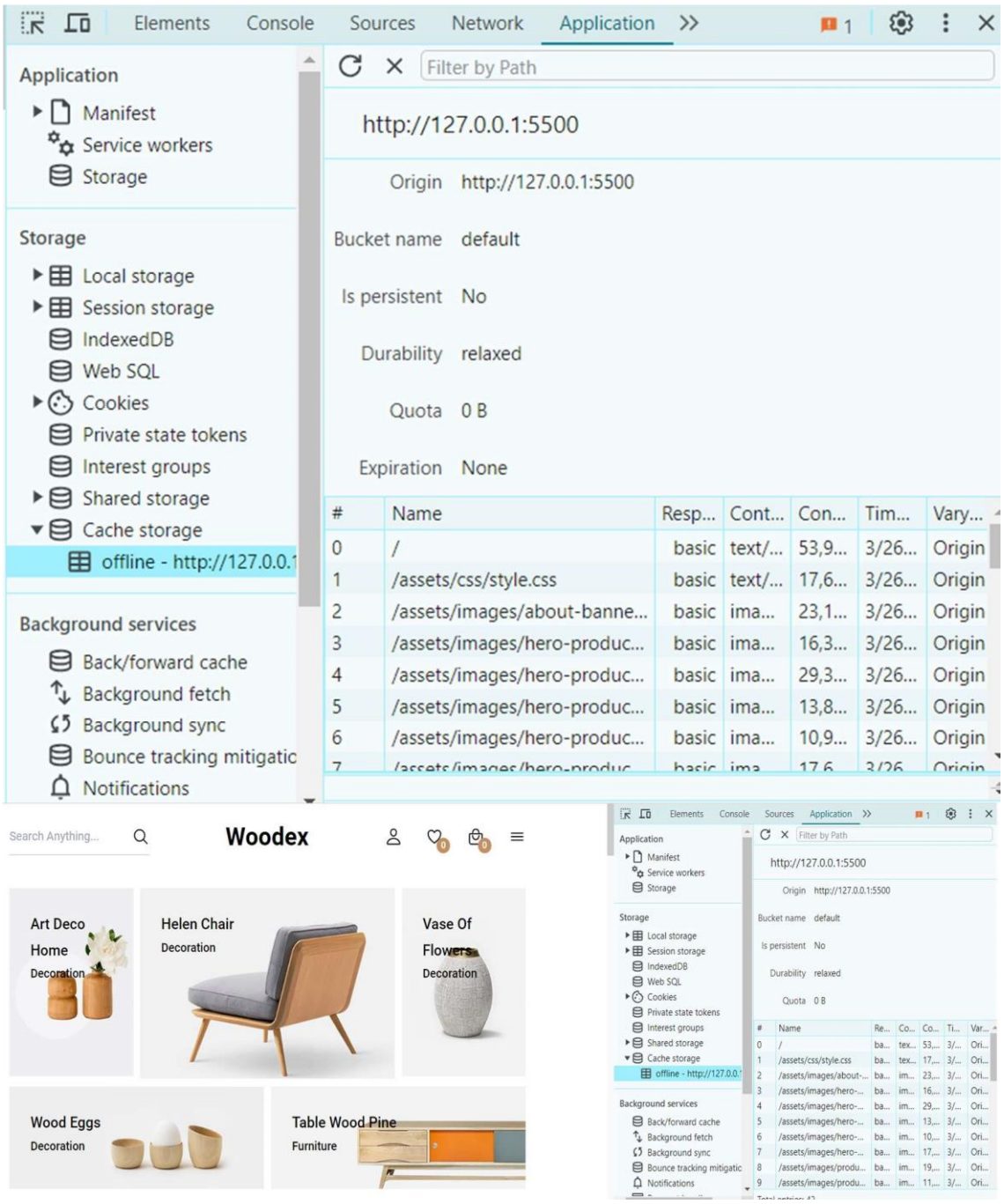
Implementation:

```
serviceworker.js:
self.addEventListener("install", function (event) {
event.waitUntil(preLoad());
});

var filesToCache = ['/',
        '/index.html',
];

var preLoad = function () {      return
caches.open("offline").then(function (cache) {
// caching index and important routes
                return cache.addAll(filesToCache);
        });
};

self.addEventListener("fetch", function (event) {
event.respondWith(checkResponse(event.request).catch(function () {
                return returnFromCache(event.request);
```

```
        }));
        event.waitUntil(addToCache(event.request));
});

var checkResponse = function (request) {
        return new Promise(function (fulfill, reject) {
                fetch(request).then(function (response) {
                if (response.status !== 404) {
fulfill(response);
                        } else {
                        reject();
                        }
                }, reject);
        });
};

var addToCache = function (request) {           return
caches.open("offline").then(function (cache) {
                return fetch(request).then(function (response) {
                        return cache.put(request, response);
                });
        });
};

var returnFromCache = function (request) {     return
caches.open("offline").then(function (cache) {                  return
cache.match(request).then(function (matching) {
if (!matching || matching.status == 404) {
                                return cache.match("offline.html");
                        } else {
                return matching;
                        }
                });
        });
};


self.addEventListener('activate', function (event) {
        event.waitUntil(
                // Perform cleanup tasks or cache management
here            // For example, deleting outdated caches
                caches.keys().then(function (cacheNames) {
                        return Promise.all(
                                cacheNames.filter(function (cacheName) {
```

```
                        version
                    }).map(function (cacheName) {
                    // Delete the outdated cache
                        return caches.delete(cacheName);
                })
            );
        })
    );
});
```

Output:



Conclusion: From this experiment, we have understood the concept of service worker in PWA, their applications, how to use it and how caches are loaded for our web app. Also, we have implemented the lifecycle of service worker including registration, installation and activation.