NAME: HARSHITA DUBEY ROLL NO.:14
SUBJECT: MAD LAB
CLASS:D15A/BATCH A

EXP 8

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

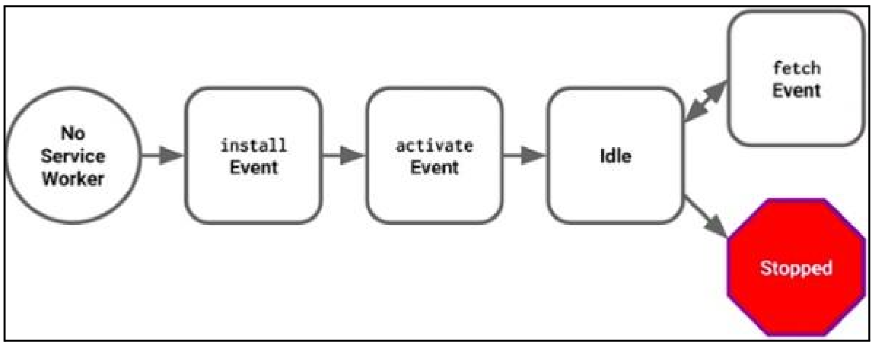
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

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



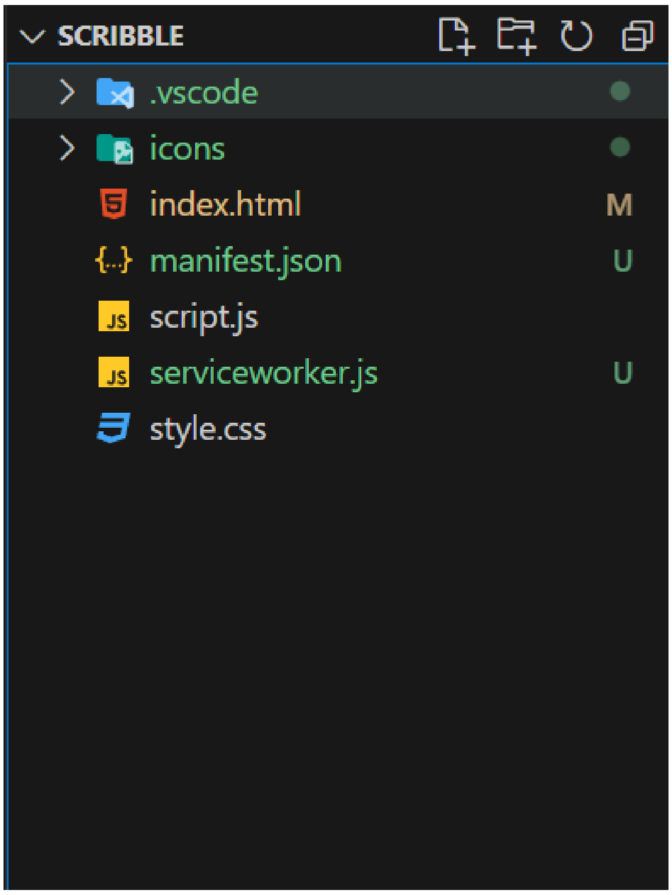
A service worker goes through three steps in its life cycle:

- Registration
- Installation
- Activation

Registration
To install a service worker, you need to register it in your main JavaScript code.

Registration tells the browser where your service worker is located, and to start installing it in the background.

My Folder Structure



```
SCRIBBLE
  > .vscode                ●
  > icons                  ●
    index.html             M
    manifest.json          U
    script.js
    serviceworker.js       U
    style.css
```

CODE:

index.html

```
<!DOCTYPE html>

<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Scribble</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="script.js" defer></script>
    <!-- Manifest File link -->
<link rel="manifest"
href="manifest.json">
```

```
</head>
<body>
  <div class="container">
    <section class="tools-board">
      <div class="row">
        <label class="title">Shapes</label>
        <ul class="options">
          <li class="option tool" id="rectangle">
            <img src="icons/rectangle.svg" alt="">
            <span>Rectangle</span>
          </li>
          <li class="option tool" id="circle">
            <img src="icons/circle.svg" alt="">
            <span>Circle</span>
          </li>
          <li class="option tool" id="triangle">
            <img src="icons/triangle.svg" alt="">
            <span>Triangle</span>
          </li>
          <li class="option">
            <input type="checkbox" id="fill-color">
            <label for="fill-color">Fill color</label>
          </li>
        </ul>
      </div>
      <div class="row">
        <label class="title">Options</label>
        <ul class="options">
          <li class="option active tool" id="brush">
            <img src="icons/brush.svg" alt="">
            <span>Brush</span>
          </li>
          <li class="option tool" id="eraser">
            <img src="icons/eraser.svg" alt="">
            <span>Eraser</span>
          </li>
          <li class="option">
            <input type="range" id="size-slider" min="1" max="30" value="5">
          </li>
        </ul>
      </div>
      <div class="row colors">
        <label class="title">Colors</label>
        <ul class="options">
```

```
          <li class="option"></li>
          <li class="option selected"></li>
          <li class="option"></li>
          <li class="option"></li>
          <li class="option">
            <input type="color" id="color-picker" value="#4A98F7">
          </li>
        </ul>
      </div>
      <div class="row buttons">
        <button class="clear-canvas">Clear Canvas</button>
        <button class="save-img">Save As Image</button>
      </div>
    </section>
    <section class="drawing-board">
      <canvas></canvas>
    </section>
  </div>
  <script>
    // Add event listener to execute code when page loads
    window.addEventListener('load', () => {
      // Call registerSW function when page loads
      registerSW();
    });

    // Register the Service Worker
    async function registerSW() {
      // Check if browser supports Service Worker
      if ('serviceWorker' in navigator) {
        try {
          // Register the Service Worker named 'serviceworker.js'
          await navigator.serviceworker.register('serviceworker.js');
        }
        catch (e) {
          // Log error message if registration fails
          console.log('SW registration failed');
        }
      }
    }
  </script>

 </body>
</html>
```

Installation

Once the browser registers a service worker, installation can be attempted. This occurs if the service worker is considered to be new by the browser, either because the site currently doesn't have a registered service worker, or because there is a byte difference between the new service worker and the previously installed one.

A service worker installation triggers an install event in the installing service worker. We can include an install event listener in the service worker to perform some task when the service worker installs. For instance, during the install, service workers can precache parts of a web app so that it loads instantly the next time a user opens it (see caching the application shell). So, after that first load, you're going to benefit from instant repeat loads and your time to interactivity is going to be even better in those cases. An example of an installation event listener looks like this:

```
var staticCacheName = "pwa";
self.addEventListener("install", function (e) {
e.waitUntil(
caches.open(staticCacheName).then(function (cache) {
return cache.addAll(["/"]);
})
);
});
```

-Step 1 of service worker installation.

Activation

Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches (see the Offline Cookbook for an example).

```
self.addEventListener('activate', event => {
event.waitUntil(
caches.keys().then(cacheNames => {
return Promise.all(
cacheNames.filter(name => {
return name !== staticCacheName;
}).map(name => {
return caches.delete(name);
})
);
})
);
});
```
-Step 2 of service worker activation.

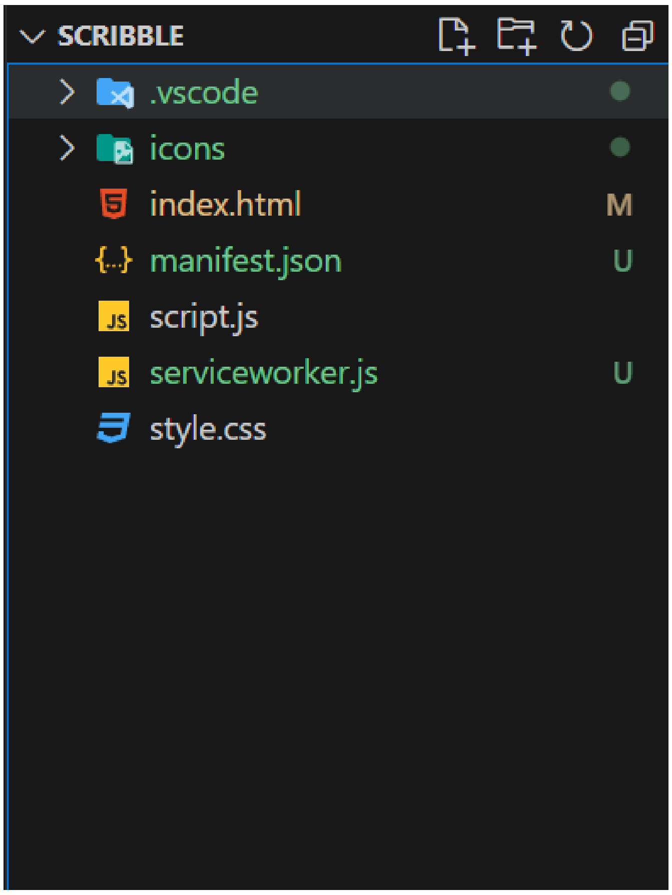
Thus the **serviceworker.js** will look like this

```
var staticCacheName = "pwa";
self.addEventListener("install", function (e) {
e.waitUntil(
caches.open(staticCacheName).then(function (cache) {
return cache.addAll(["/"]);
})
);
});
self.addEventListener('activate', event => {
event.waitUntil(
caches.keys().then(cacheNames => {
return Promise.all(
cacheNames.filter(name => {
return name !== staticCacheName;
}).map(name => {
return caches.delete(name);
})
);
})
);
});
self.addEventListener("fetch", function (event) {
console.log(event.request.url);
event.respondWith(
caches.match(event.request).then(function (response) {
return response || fetch(event.request);
})
);
});
```

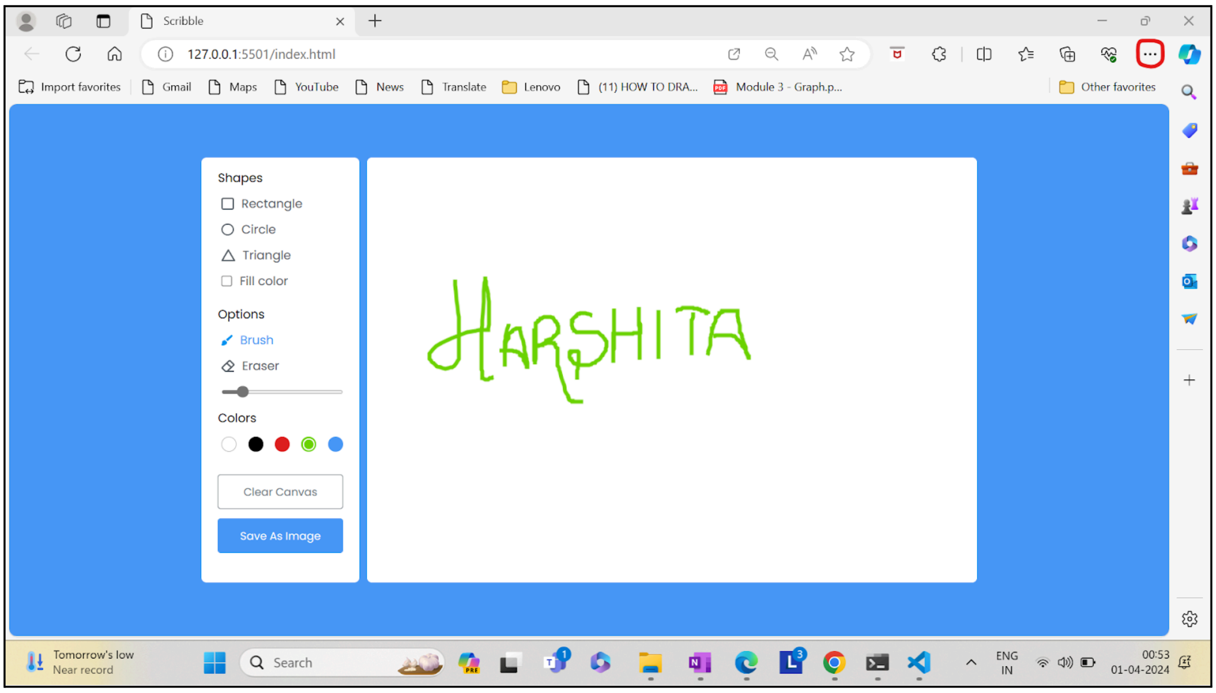
<u>IMPLEMENTATION</u>

STEP 1: Open any web application through any Text Editor eg:VSCODE.

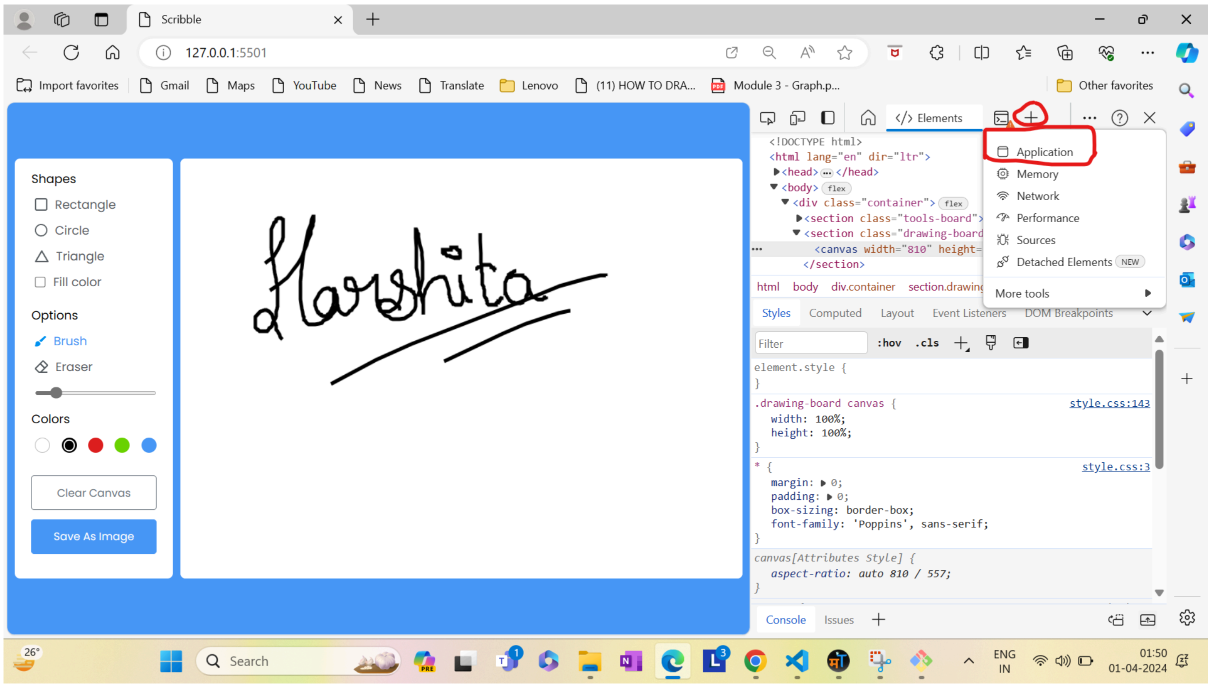
STEP 2: Add few files to this. Folder structure of my application is:



STEP 4 : Open Live Server and http://127.0.0.1:5501/index.html will be opened

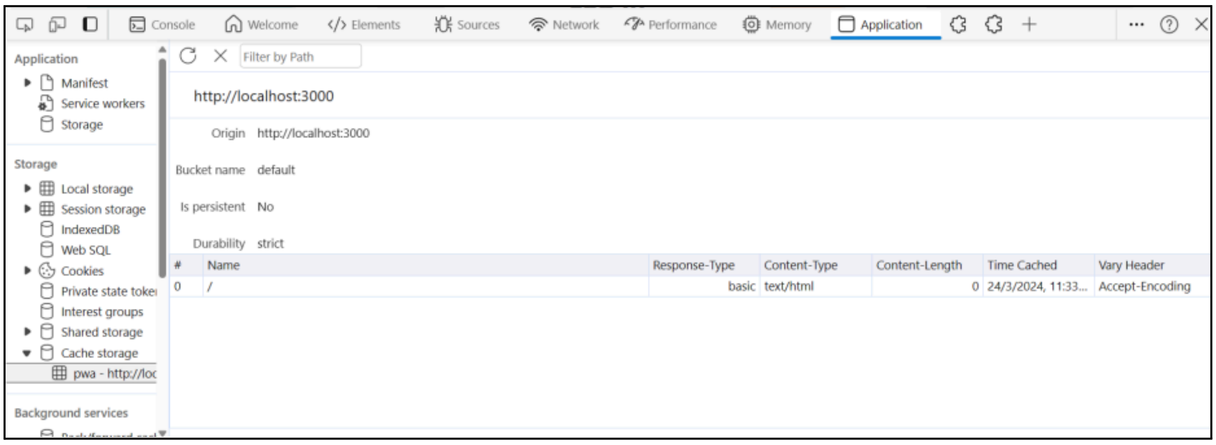


STEP 4: Right Click on the browser and click on inspect —-> Plus icon —-> Application



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
 ▶<head> … </head>
 ▼<body> flex
   ▼<div class="container"> flex
     ▶<section class="tools-board">
     ▼<section class="drawing-board
         <canvas width="810" height=
       </section>
```

```
element.style {
}
.drawing-board canvas {                    style.css:143
   width: 100%;
   height: 100%;
}
* {                                          style.css:3
   margin: ▶ 0;
   padding: ▶ 0;
   box-sizing: border-box;
   font-family: 'Poppins', sans-serif;
}
canvas[Attributes Style] {
   aspect-ratio: auto 810 / 557;
}
```



http://localhost:3000

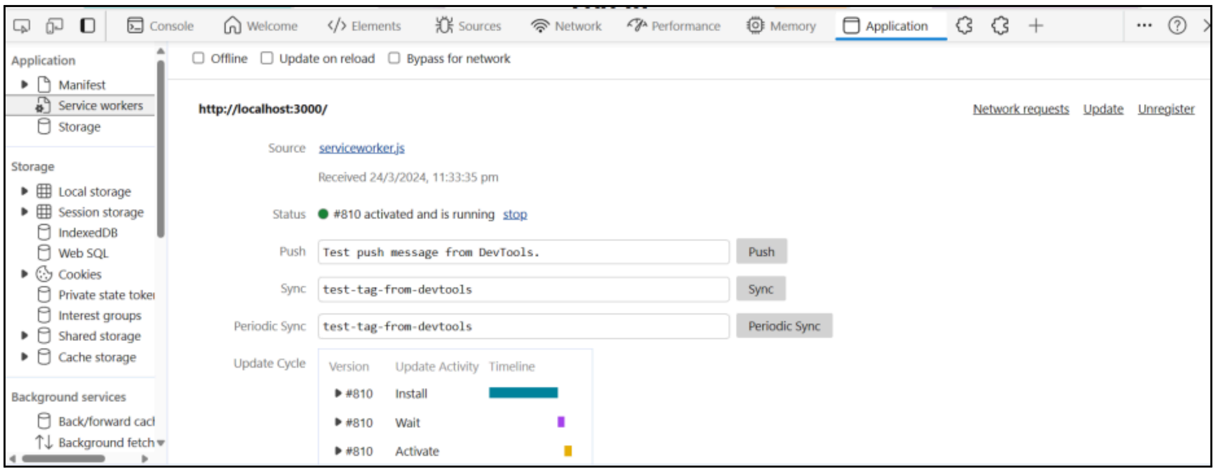
Origin http://localhost:3000

Bucket name default

Is persistent No

Durability strict

#	Name	Response-Type	Content-Type	Content-Length	Time Cached	Vary Header
0	/	basic	text/html	0	24/3/2024, 11:33...	Accept-Encoding



☐ Offline ☐ Update on reload ☐ Bypass for network

http://localhost:3000/

Source serviceworker.js
Received 24/3/2024, 11:33:35 pm

Status ● #810 activated and is running stop

Push `Test push message from DevTools.`

Sync `test-tag-from-devtools`

Periodic Sync `test-tag-from-devtools`

Update Cycle

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

Conclusion:

To enhance the user experience and resilience of a Progressive Web Application (PWA), a service worker (serviceworker.js) was implemented and registered. This service worker intercepts network requests, caches critical resources, and facilitates offline functionality and improved performance. Upon completion of the installation and activation process, the PWA gains features such as offline access and accelerated page loads. This strategic integration of a service worker not only elevates user experience but also fortifies the application against network disruptions, ensuring seamless functionality even in challenging connectivity conditions.