Name: Sujal S. Tekwani
Class: DISB
Roll no: 59

# PWA Assignment - 2

04/05

Q1) Define Progressive Web App (PWA) and explain its significance in modern web development. Discuss the key characteristics that differentiate PWAs from traditional mobile apps.

Ans.1 A progressive web app (PWA) is a web application that combines the best features of both web and mobile apps to deliver a seamless, reliable, and fast user experience. PWAs work offline, load quickly, and provide an app like experience on web browsers.

Significance in Modern Web development:

→ Platform Independance - Runs on any device with a web browser

→ Improved Performance - Faster load times due to caching and service workers.

→ Offline Functionality - Works without an internet connection.

→ No App store Dependencies - Users can install PWAs directly from the browser.

→ Engaging User Experience - Provides push notification and background syncing.

- Key characteristics of PWA v/s Traditional Mobile Apps:-

① Installation:

→ PWAs are installed from a browser.

→ Traditional mobile apps are downloaded from app store.

(2) Platform Dependency:
→ PWAs works across platforms with one codebase.
→ ~~Traditional~~ Traditional mobile apps requires seperate development for ios and Android.

(3) Offline Support:
→ PWAs use Service Service workers for offline access.
→ Traditional mobile apps usually requires native implementation.

(4) Updates:
→ PWAs Update automatically via the web.
→ Traditional mobile apps requires app store updates.

(5) Performance:
→ PWAs are faster due to caching and lightweight assets.
→ Traditional mobile apps can be

Q2) Define responsive web design and explain its importance in context of Progressive Web apps. Compare and contrast responsive, fluid and adaptive web design approaches.

Ans. 2 Responsive web design in an approach that ensures web pages adapt to different screen sizes and orientations using flexible grids, media queries, and scalable images.

Importance of responsive web design in PWAs:-
→ Ensures a consistent user experience across different devices.
→ Eliminates need for multiple codebases for different devices.
→ Enhances usability by making content readable on various screen.

- Comparison of Responsive, Fluid and Adaptive Web design.

| Feature | Responsive | Fluid | Adaptive |
|---|---|---|---|
| Difinition | Uses CSS media queries to adjust layout dynamically | Uses percentage based units for elements to scale. | Uses predefined layout for different screen sizes. |
| Flexibility | Highly Flexible | Completely flexible | Fixed at specific breakpoints. |
| Performance | Efficient but requires more CSS adjustments. | Smooth Scaling. | May cause layout shifts. |
| Best use Case | Websites and PWAs for all screen. | Apps requiring seamless scaling. | Websites with predefined layouts. |

Q3) Describe the lifecycle of service workers, including registration, installation, and ~~installation~~, and activation phases.

Ans. 3 A service worker is a background script that runs independently from the main browser thread, enabling features like offline caching and push notification.

Lifecycle Phases :-

① Registration :-
The service worker is registered in Java script using navigator service workers register().
Eg.
if ('service Worker' in navigator) {
navigator · service Worker . register ('/sw.js')

```
then(() => console.log('Service Worker Registered'));
}
```

② Installation:.

→ Occures when the service worker is first downloaded.

→ Typically used for caching assets.

Eg.

```
self.addEventListner('install', event => {
    event.waitUntil(
        caches.open('v1').then(cache => {
            return cache.addAll(['/index.html', '/style.css'])
        })
    );
});
```

③ Activation:.

→ Runs after installation ~~of~~ and ensures old caches are cleared if necessary.

Eg.

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

④ Fetching and Updates :-

→ The Service workers intercepts network requests and serves cached content.

Eg.

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

Q4) Explain the use of Indexed DB in the Service Worker for data storage.

Ans:4 Indexed DB is a low-level NoSQL database in the browser that allows web apps to store and ~~return~~ retrive large amount of structured data efficiently.

Use of IndexedDB in Service Workers :-

① Offline Storage - Saves user data when offline and syncs it when back online.

② Persistent Data - Unlike local storage, Indexed DB is ~~asynchronous or~~ asynchronous and handles large amount of data.

③ Background Sync - Service Workers can use Indexed DB to store data and sync it later.

Eg. Eg.

```
Const dbRequest = indexedDB.open('myDatabase', 1);
dbRequest.onupgradeneeded = event => {
  Const db = event.target;
db.createObjectStore('messages', { KeyPath = 'id' });
};
dbRequest.onsuccess = event =>
{ Const db = event.target.result;
Const transaction = db.transaction('messages', 'readwrite');
Const store = transaction.objectStore('messages');
store.put({ id: 1, text = 'Hello from IndexedDB' });
};
```