Name :- Vivek. S. Gupta
Div :- D15B
Roll no :- 19

(A)

MPL assignment 2.

1. define Progressive web app (PWA) and explain its significance in modern web development. Discuss the key characteristics that diffrentiate PWAs from traditional mobile apps.

→. A progressive Web App (PWA) is a type of web application that works like a mobile app but runs in a browser. It can be installed on a device, works offline and provide a fast and smoth user experience.

Significance of PWA in Modern Web Development.

1. Cross-Platform Compatibility - Works on both mobile and desktop with a single codebase.
2. Offline support - Can function without the internet using cached data.
3. Fast performance - loads quickly, even on slow networks.
4. No App Store Required - Users can install is directly from the browser.
5. lower development Cost - One PWA can replace seperate Android and IOS apps.

Key difference between PWA and Traditional Mobile Apps -

Feature	PWA	Traddion App
Installation	Direct from browser	Download from App store.
Internet Required.	Works offline with caching	Usuall requires intenet.

Performance	Fast with service workers	~~Usually~~ Faster but needs installation.
Updates	Automatic, no app store approval.	Manual updates needed.
Development cost	lower (one codebase for all)	Higher (seperate apps for each platform)

PWAs combine the best of web and mobile apps, making them efficient and user friendly.

Q. 2. Define responsive web design and explain the importance.

→ Definition of Responsive Web design

Responsive Web design is a technique that makes web pages adjust automatically to different screen sizes and devices. It ensures a good user experience on mobiles, to tablets and desktops without needing seperate versions.

Importance of Responsive design in PWAs :-

1. Better User experience - PWAs work smoothly on any device
2. Faster load time - Optimized design improves speed.
3. SEO Benefits - Google ranks responsive sites higher.
4. Cost - Effective - No need to build multiple versions.

Comparison of Web design Approaches :-

Approach	How it works	Pros	Cons
Responsive	Uses flexible grids and CSS media queries.	Works on all devices	Can be complex to design
Fluid	Uses percent based widths instead of fixed pixels so elements resize.	Works well on different screen size.	Less control over layout on large screen

3]. Describe the lifecycle of Service workers, including registration, installation and activation phases.

→. Lifecycle of Service workers :-
A service worker is a script that runs in the background and helps a web app work offline

i]. Registration process :-
→ The browser registers the Service worker using JS.

eg :-

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

2. Intallation of phase.

- The service worker downloads necessary files and stores them in cache.
- If succesfull, it moves to the activation phase.

eg.

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

3) Activation phase.
- The old service worker is replace with the new one.

eg.

```
self.addEventListener('activate', event => {
event.waitUntil(
    caches.keys().then(keys => {
    return Promise.all(keys.map(key => {
     if (key !== "app-cache'){
       return caches.delete(key);
      }
     }));
    })
   );
  });
```

9.4 Explain the use of IndexedDB in the Service Worker for data storage.

→ Use of IndexedDB in Service Worker for Data Storage.
IndexedDB is a browser database that stores large amounts of structured data like JSON objects. It helps PWAS work offline by savings.

Why Use IndexedDB in service workers.
1. Offline support - stores data when offline.
2. Efficient storage - saves structured data.
3. faster access - Retrieves data quickly

How Service Workers use IndexedDB?

Opening the database.

```
let db;
 let request = indexedDB.open ('MyDatabase', 1);
 request.onsuccess = function (event) {
     db = event.target.result.
 }.
```

fetching data in Service Worker.

```
 let transaction = db.transaction ('Users', 'readonly');
 let store = transaction.objectStore ('Users');
```

```
let getUser = store.get();

getUser.onsuccess = function() {
  console.log(getUser.result);
};
```

FOR EDUCATIONAL USE