05/08

Name: Niddhi Rijhwani
Class: D15B Roll No: 47
Subject: MPL - Assignment 2

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

A] A progressive Web App (PWA) is a web application that delivers a native app-like experience using modern web technologies. It works accross devices, supports offline access, and can be installed without an app store.

- Significance in Modern Web Development

1. Cross-platform: Runs on any device with a web browser.
2. Online Support: Uses Service workers for caching.
3. fast Performance: Loads quickly, even on slow networks.
4. Engagement: Supports push notifications & installation
5. Lower Costs: A single codebase reduces development effort.

	PWA	Traditional Mobile App
1.	Installation: Via browser	Installation: App store
2.	Office support: Via caching	Via explicit design.
3.	Push Notification: Yes	No push notification.
4.	Platform dependency: works on all browsers.	platform specific dependencies.
5.	Updates: Automatic	Requires user updates

Q2. Define responsive web design and explain its importance in the context of PWA. Compare and contrast responsive, fluid and adaptive web design approaches.

Responsive web Design ensures a website adapts dynamically to different screen sizes using flexible layouts and media queries.

→ Importance in PWAs:-

1. Consistent User Experience: Optimizes for all screen sizes.

2. Better Accessibility: Enhances usability across devices.
3. Single Codebase: Reduces development effort.
4. SEO Benefits: Google favors mobile friendly design.

- Comparison of web Design Approaches.

feature	Responsive	Fluid	Adaptive
dfinition	Adjusts layout dynamically	Uses percentage based width	Uses fixed breakpoints
flexibility	High	High	Limited
development effort	Moduate	Low	High
Best use case	PWAs, Modern sites	Simple layouts	Specific devic layouts

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

→ Lifecycle of services workers :-
A services worker is a script that runs in the background and helps a web app work offline, load faster and send push notifications, its lifecycle has three main phases :-

1. Registration phase :- The browser register the service worker using javascript.

Eg:-

```
if('serviceworker' in navigator){
    navigator.serviceworker.register("/sw.js")
    .then() => console.log('service Registered");

    .catch(error => console.log("Registered
        failed:", error));
}
```

2. Installation phase :- ① The service worker downloads necessary files (HTML, CSS, JS) and stores them in cache.

② If succesful, it moves to the activation phase.

code eg:-

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

3. Activation Phase :- ① The old services worker is replaced with new one.

② Unused cache files from the previous version are deleted.

final step: fetch & sync.

- Once activated, the services worker intercept network request, serves cached files and syncs data.

Q4. Explain the Ure of indexed DB in the services worker for data storage.

→

Ure of IndexDB in service worker for Data Storage:-
Indexed DB is a browser database that stores large amount of structured data like JSON objects. It helps pop work offline by saving and retrieving data efficiency.

why ure IndexedDB in service workers?

1. Offline support - Store data when offline and sync its later.
2. Efficient storage - Saves structure data like user setting, cart items or form inputs.
3. faster Access - Retrieves data quickly without needing a network request.
4. Persistent data - Data remains saved even after the browser is closed.

→P.T.O

How service workers ure IndexedDB?

opening the Database:

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

Creating a store & adding Data

```
request.onupgradeneeded = function (event) {
    let db = event.target.result;
    let store = db.createObjectStore('users',
      {keyPath: 'id'});
    store.add ({id: 1, name: 'John Doe', age: 25});
};
```

fetching Data in service worker.

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

let getUser = store.get(1);
getUser.onsuccess = function () {
    console.log (getUserResult);
};
```