Vivekanand Education Society's
Institute of Technology
(An Autonomous Institute Affiliated to University of Mumbai)
Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	Assignment-2
Title.	
Roll No.	48
Name	Swayam Rath
Class	D15 B
Subject	MAD & PWA
Grade:	

Rault

POS: 13/3/25

MAD Assignment - II

DATE:

Define a PWA and explain its significance in modern web app development. Discuss key characteristics that differentiate PWAs from mobile app

A progressive web app is a web application capabilities to deliver an app-like experience to users. It represents a fundamental shift in how we approach web development by combining the best of web and native application. Key characteristics of PWA is Traditional mobile app-

Installation & distribution:
PWAs: can be access easily through browser
Native: Require downloads from App stores

Updates:
PWAS: Update automatically when new content is available.
Traditional App: Require manual updates through app stores

Cross-platform Compatibility:
PWAs: Work across platforms through web browsers.
Native: Work on platform specific OS
PWAS: Work across platforms through web browsers.

(d) Key features of PWAs:

(i) Progressive Enhancement: Works for every user regardless of browser choice.

(ii) Responsive: Adapts to various screen sizes and orientations.

(iii) Connectivity Independent: Functions offline or with poor network.

(iv) App-like interface

(v) Secure: Served via HTTPS.

(2) Responsive Web Design in PWAs:

Ans:- Responsive Web design (RWD) is an approach that makes webpages render well on a variety of different devices & window/screen sizes.

Importance

- Ensures consistent user experience across all devices.
- Improves accessibility and usability
- Reduces maintenance overhead
- Better SEO performance
- Essential for PWA adoption.

Comparison of design approaches

a) Responsive design:

```
.container {
    width: 100%;
    max-width: 1200px;
}
```

DATE:

@media (max-width: 768px) {
.container {
padding: 0 20px; } }
Uses fluid grids
Flexible images
Media queries
Single layout that adapts.

Fluid Design:
.container {
width: 90%;
margin: 0 auto; }
.column {
width: 33.33%;
float: left; }

Uses percentage-based width.
Continuously adapts
No breakpoints
Smoother transitions

Adaptive designs:
Uses distinct layouts
Fixed width at breakpoints
Device specific layouts
Less flexible but more controlled.

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

Ans Service worker lifecycle phases consists several phases :-

(a) Registration

```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('sw.js')
    .then(registration => {
        console.log('Service Worker registered');
    });
    .catch(err => console.log, ("", "failed", err));
```

b) Installation phase

```
self.addEventListener('install', event => {
    event.waitUntil(
        caches.open('v1.1').then(cache => {
            return cache.addAll([
                '/',
                '/styles/main.css',
                '/styles/app.js'
                scripts
            ]);
        });
    });
```

(c) Activation phase

```
self.addEventListener('activate', even => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.map(cacheName => {
                    if (cacheName !== 'v1') {
                        return caches.delete(cacheName);
                    }
                }));
        }));
```

DATE:

Explain the use of IndexDB in the service worker for data storage.

IndexedDB is a low-level API for client-side storage of significant amounts of structured data. It's used in service workers in the following manner:

Opening a database
key characteristics
Offline data storage
Large data sets support
Structured data storage
Transactional Database operations
multiple Storage types support.
Asynchronous API
Complex Data queries.

Core components:
Database: Highest level of storage.
Object Store: Similar to table in SQL
Index: For faster searching of records
Transactions: Groups operations together for data integrity.
Cursor: For iterating over multiple records.

Operations:
Basic Use
Opening a database connection
Creating object Stores.

DATE:

- Adding / Updating records.
- Querying data using indexes.
- Deleting records
- Using transactions for data consistency

Common use cases

- Offline data storage
- Caching application data
- Storing user-generated content
- Client-side database for web apps.
- Temporary storage for pending uploads