**WEEK 6**

**Superset ID : 6440577**

**Hands-on 1**

**1 Define SPA and its benefits**

A Single‑Page Application (SPA) is a web app that loads a single HTML page initially and dynamically rewrites the displayed content (via JavaScript) in response to user actions, rather than loading full new pages from the server

**Benefits:**

1. **Faster, smoother experience**
2. **Reduced server load & bandwidth**
3. **App-like UX**
4. **Developer efficiency**

**2 Define React and identify its working**

React (also React.js or ReactJS) is a free, open‑source JavaScript library developed by Meta and community for building user interfaces with components

**How it Works:**

You describe UI as components (function or class) that render based on props and state.

When state or props change, React re-renders the component to produce a new **Virtual DOM** tree.

React then performs a **diffing (reconciliation)** between the new and previous Virtual DOM tree.

Only the changed parts of the real DOM are updated via patch operations

**3 Identify the differences between SPA and MPA**

|  |  |
| --- | --- |
| **SPA** | **MPA** |
| Single page application | Multi-page application |
| One initial load; subsequent updates handled via JavaScript | Each navigation triggers full page request and rendering from the server |
| Seamless, app-like transitions | Traditional full page reloads |
| Requires client-side routing, state management | Simpler architecture, server‑side rendering |
| Rich interactive apps (email clients, dashboards, etc.) | Content-heavy, publicly-indexed sites (blogs, marketing sites, stores) |

**4 Explain Pros & Cons of Single-Page Application**

**Advantages:**

**Fast interactivity and transitions** since only data is exchanged, not whole pages

**Reduced bandwidth usage** after initial load

**Rich, app-like UX**, ideal for platforms requiring lots of client interaction

**Cross-platform friendly**: same codebase for web, mobile, etc.

**Disadvantages:**

**SEO challenges**: Bots may not execute JS fully; requires extra steps like server-side rendering or prerendering

**Longer initial load** due to downloading the entire app upfront.

**Higher browser resource usage**: All logic runs client-side; heavier memory/CPU usage

**Potential security risks**, like XSS, if not properly sanitized

**5 Explain about React**

React is a lightweight, UI‑focused library:

It’s **declarative**—you define what UI should look like in each state, and React ensures it updates accordingly

Built using **components** and **props/state**, encouraging modular and reusable UI parts

Supports both **class-based** and (since v16.8) **function components with Hooks** (useState, useEffect, etc.)

It does **not include routing or data fetching**; instead, it integrates with libraries like React Router, Redux, or MobX for full application architecture

## 6. Define ****Virtual DOM****

**Definition:** A lightweight in-memory representation of the actual DOM used by React to optimize updates

**How it works:**

Render the UI into a Virtual DOM tree (JavaScript object).

On state/props change, React builds a new Virtual DOM tree.

React **diffs** the new tree against the old tree to spot changes.

Only **patches** the real DOM where needed — called reconciliation

**Benefits:**

Minimizes expensive DOM operations: more efficient rendering and smoother UI

Simplifies coding by letting developers think in terms of application state, not manual DOM manipulation

Improves cross-browser consistency, since updates abstract away browser quirks

**Drawbacks:**

Some memory overhead to maintain virtual tree in memory.

Diffing introduces computation cost, which can be inefficient for very simple apps.

Debugging can be a bit harder since you don’t directly manipulate the real DOM

## Explain ****Features of React****

**Declarative Syntax**  
Write what UI should look like for a given state; React handles the rest

**Component-Based Architecture**  
UI is broken into reusable components using props & internal state

**Virtual DOM & Efficient Rendering**  
React updates only the changed parts of the DOM for performance

**Hooks & Functional Components**  
Since v16.8, Hooks like useState, useEffect, useMemo provide state and side-effect handling within function components

**Unidirectional Data Flow**  
Data flows down via props; changes originate from user or store actions (Flux/Redux pattern)

**Ecosystem & Extensibility**  
Though UI-focused, React integrates easily with routing (e.g. React Router), global state libraries (Redux, MobX), server-side rendering frameworks (Next.js)

**Output:**



