**OBJECTIVES**

1. **Define SPA (Single-Page Application)**

* A Single-Page Application (SPA) is a web application that loads a single HTML page and dynamically updates the content as the user interacts with the app.
* Instead of reloading the entire page from the server, only the required data is fetched and displayed using JavaScript (often with frameworks like React, Angular, or Vue).
* **Benefits of SPA**

1. Faster Navigation
   * No need to reload the whole page; only parts of the page are updated.
2. Improved User Experience
   * Feels more like a desktop application with smoother transitions.
3. Reduced Server Load
   * Only data (usually JSON) is exchanged, not entire HTML pages.
4. Efficient Bandwidth Usage
   * Since the initial page loads once, less bandwidth is used after that.
5. Easier to Build Responsive Apps
   * JavaScript handles dynamic updates based on user actions.
6. Offline Capabilities
   * With service workers, SPAs can cache resources and work offline.
7. Code Reusability
   * Components and logic can be reused across the application.
8. **Define React**

* React is an open-source JavaScript library used for building user interfaces, especially for Single-Page Applications (SPAs).
* Developed and maintained by Facebook.
* Allows developers to create reusable UI components.
* React uses JSX, a syntax extension that allows mixing HTML with JavaScript.
* **Identify How React Works;**

1. **Component-Based Architecture**
   * UI is divided into small, reusable components.
   * Each component has its own logic and rendering.
2. **Virtual DOM**
   * React maintains an in-memory copy of the actual DOM.
   * When changes occur, React compares the new virtual DOM with the previous one and updates only the changed parts in the real DOM (called "diffing" and "reconciliation").
3. **Unidirectional Data Flow**
   * Data flows from parent components to child components using **props**.
   * This makes data flow predictable and easier to debug.
4. **State Management**
   * Components can maintain their own **state** using useState.
   * When state changes, React re-renders only the necessary parts of the UI.
5. **Declarative UI**
   * Developers describe *what* the UI should look like for a given state, and React takes care of updating the DOM.
6. **React Hooks**
   * Functions like useState, useEffect allow developers to manage state and lifecycle features in functional components.
7. **Differences Between SPA and MPA**

| **Feature** | **SPA (Single-Page Application)** | **MPA (Multi-Page Application)** |
| --- | --- | --- |
| **Page Structure** | Single HTML page loaded once | Multiple HTML pages, each served separately |
| **Navigation** | Dynamic (uses JavaScript, no full reload) | Full page reload on each navigation |
| **Speed & Performance** | Faster after initial load | Slower due to frequent full-page loads |
| **User Experience (UX)** | Smooth and app-like | Traditional web experience |
| **SEO (Search Engine Opt.)** | Harder to implement (requires SSR or prerendering) | Easier, as all content is available in HTML |
| **Technology Stack** | Often uses frameworks like React, Angular | Can be built with HTML, CSS, JavaScript, PHP etc. |
| **Server Load** | Lighter after first load (less requests) | Heavier, server sends full pages every time |
| **Initial Load Time** | Higher due to JS bundles | Faster initial load (per page) |
| **Examples** | Gmail, Facebook, Twitter | Amazon, LinkedIn (classic), Wikipedia |

1. **Pros of SPA (Single-Page Application)**

* **Fast Navigation**
  + Only the required content is updated; no full page reloads.
* **Improved User Experience**
  + Smooth and seamless interaction, like a desktop or mobile app.
* **Reduced Server Load**
  + After the initial load, only data (like JSON) is exchanged with the server.
* **Efficient Resource Use**
  + HTML, CSS, and JS are loaded once and reused.
* **Offline Support**
  + With service workers, SPAs can cache resources and work offline.
* **Easier Development for Mobile**
  + SPA architecture aligns well with mobile apps and PWA (Progressive Web Apps).

1. **Cons of SPA (Single-Page Application)**

* **SEO Limitations**
  + Content is rendered by JavaScript, making it harder for search engines to crawl (needs SSR or prerendering).
* **Longer Initial Load Time**
  + The first load can be slow due to large JavaScript bundles.
* **JavaScript Dependency**
  + SPA won't function properly if JavaScript is disabled in the browser.
* **Complex State Management**
  + Requires advanced handling of app state (e.g., using Redux or Context API).
* **Security Risks**
  + Exposes more client-side logic, increasing the risk of attacks like XSS.
* **Browser History & Back Button Handling**
  + Requires manual configuration to support browser navigation properly (using React Router or similar).

1. **React – Explanations and features**
2. **What is React?**
   * React is an open-source JavaScript library used for building user interfaces (UIs), especially for Single-Page Applications (SPAs).
3. **Who Created It?**
   * Developed and maintained by Facebook (now Meta).
4. **Key Purpose**
   * Helps build fast, interactive, and reusable UI components.
5. **Component-Based Architecture**
   * UI is broken into small, reusable pieces called components (functional or class-based).
6. **JSX (JavaScript XML)**
   * React uses JSX, a syntax that allows writing HTML-like code inside JavaScript.
7. **Virtual DOM**
   * React maintains a virtual DOM to improve performance by reducing actual DOM updates.
8. **One-Way Data Flow**
   * Data flows from parent to child components using props, making the code more predictable.
9. **Hooks**
   * Modern React uses hooks (e.g., useState, useEffect) to handle state and side effects in functional components.
10. **Ecosystem Support**
    * Works with tools like React Router (routing), Redux or Context API (state management), and Next.js (for SSR).
11. **Popular Use Cases**
    * Dashboards, SPAs, Progressive Web Apps (PWAs), dynamic form handling, etc.
12. **Define Virtual DOM (VDOM)**
13. **What is Virtual DOM?**
    * The Virtual DOM is a lightweight, in-memory copy of the real DOM used by libraries like React to optimize UI rendering.
14. **Purpose**
    * To improve performance by minimizing direct manipulation of the real DOM, which is slow.
15. **Benefits**

* Faster UI updates.
* Efficient rendering.
* Better performance for complex UIs.