**LAB 1**

Define SPA and its benefits.

**SPA (Single-Page Application)** is a web application that dynamically updates a single HTML page without reloading the entire page during navigation.

**Benefits:**

* Faster user experience
* Less bandwidth usage
* Smooth transitions (no full page reload)
* Reusable components

Define React and Identify Its Working.

**React** is a JavaScript library developed by Facebook for building fast and interactive user interfaces.

**How React Works:**

* Uses **components** to build UI
* Uses **Virtual DOM** to optimize updates
* Renders changes efficiently with **reconciliation**

Differences Between SPA and MPA.

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |
| Page Load | Loads once, dynamic content updates | Loads new pages for each request |
| Speed | Faster after first load | Slower with full page reloads |
| SEO | Harder (requires SSR for best SEO) | Easier (each page is separate) |
| Development | Mostly client-side | Server + client-side |

**Pros & Cons of SPA**

**Pros:**

* Fast navigation
* Responsive UI
* Reduced server load

**Cons:**

* Poor SEO (unless SSR used)
* First load can be large
* More complex front-end logic

**Explain About React**

React is:

* Component-based
* Declarative (you describe **what** UI should look like)
* Maintained by Meta (Facebook)

**Define Virtual DOM**

Virtual DOM is:

* A lightweight copy of the real DOM
* React uses it to detect changes and update only the changed parts of the DOM efficiently.

**Features of React**

* Component-Based Architecture
* JSX (JavaScript + HTML)
* Virtual DOM
* Unidirectional Data Flow
* React Hooks (for state and effects).

A screenshot of a computer

AI-generated content may be incorrect.

**App.js**

import React from 'react';

function App() {

return (

<div>

<h1>Welcome to the first session of React</h1>

</div>

);

}

export default App;

**LAB 2**

**1. Explain React Components**

**A React component is a self-contained, reusable building block of a UI (User Interface). Think of it like a Lego block — each block represents part of the UI, and when combined, they make the full application.**

**There are two main types of components:**

* **Class Components (older, but still used)**
* **Function Components (modern, preferred way)**

**2. Identify the Differences Between Components and JavaScript Functions**

| **Feature** | **React Components** | **JavaScript Functions** |
| --- | --- | --- |
| **Purpose** | **Build UI and manage rendering logic** | **Perform operations or return values** |
| **Syntax** | **Use JSX (JavaScript + HTML-like syntax)** | **Use plain JavaScript** |
| **Return Type** | **Returns JSX (UI elements)** | **Returns primitive values or objects** |
| **Lifecycle Methods** | **Supported (in class components)** | **Not applicable** |
| **State Management** | **Can manage state and hooks** | **Cannot manage React state by default** |

**3. Identify the Types of Components**

**React has two primary types of components:**

1. **Class Components**
   * **Created using class keyword**
   * **Must have a render() method**
   * **Can use lifecycle methods like componentDidMount()**
2. **Function Components**
   * **Created using plain JavaScript functions**
   * **Use React Hooks like useState, useEffect**
   * **More concise and preferred in modern React**

**4. Explain Class Component**

**A Class Component is a traditional way of writing React components using ES6 class syntax. It looks like this:**

**jsx**

**CopyEdit**

**class MyComponent extends React.Component {**

**render() {**

**return <h1>Hello from class component</h1>;**

**}**

**}**

**Features:**

* **Uses render() method to return JSX**
* **Can have a constructor for initializing state**
* **Supports lifecycle methods**

**5. Explain Function Component**

**A Function Component is a simpler way to create components using a function:**

**jsx**

**CopyEdit**

**function MyComponent() {**

**return <h1>Hello from function component</h1>;**

**}**

**Or with arrow function:**

**jsx**

**CopyEdit**

**const MyComponent = () => <h1>Hello</h1>;**

**With state using Hooks:**

**jsx**

**CopyEdit**

**const Counter = () => {**

**const [count, setCount] = useState(0);**

**return <button onClick={() => setCount(count + 1)}>Click {count}</button>;**

**};**

**6. Define Component Constructor**

**The constructor in a class component is a special function that runs when the component is created. It’s often used to:**

* **Initialize state**
* **Bind event handlers**

**Example:**

**jsx**

**CopyEdit**

**class MyComponent extends React.Component {**

**constructor(props) {**

**super(props);**

**this.state = { name: 'React' };**

**}**

**}**

**Note: super(props) must be called to use this.props.**

**7. Define render() Function**

**The render() function is required in class components and is used to describe what the UI should look like. It returns JSX.**

**Example:**

**jsx**

**CopyEdit**

**render() {**

**return (**

**<div>**

**<h1>Hello from render()</h1>**

**</div>**

**);**

**}**

**About.js**

import React from 'react';

class About extends React.Component {

  render() {

    return (

      <div>

        <h2>Welcome to the About page of the Student Management Portal</h2>

      </div>

    );

  }

} **export default Contact;**

**Contact.js**

import React from 'react';

class Contact extends React.Component {

  render() {

    return (

      <div>

        <h2>Welcome to the Contact page of the Student Management Portal</h2>

      </div>

    );

  }

}

export default Contact;

**home.js**

import React from 'react';

class Home extends React.Component {

  render() {

    return (

      <div>

        <h2>Welcome to the Home page of Student Management Portal</h2>

      </div>

    );

  }

}

export default Home;

**App.js**

import React from 'react';

import './App.css';

import Home from './components/home';

import About from './components/About';

import Contact from './components/contact';

function App() {

  return (

    <div className="App">

      <Home />

      <About />

      <Contact />

    </div>

  );

}

export default App;

**OUTPUT**

**A screenshot of a computer

AI-generated content may be incorrect.**

**LAB 3**

App.js

import React from 'react';

import { CalculateScore } from './Components/CalculateScore';

**function** App() {

  return (

    <div>

      <CalculateScore

        Name="Steeve"

        School="DNV Public School"

        total={284}

        goal={3}

      />

    </div>

  );

}

export default App;

CalculateScore.js

import '../Stylesheets/mystyle.css';

**const** percentToDecimal = (decimal) **=>** {

  return (decimal ).toFixed(2) + '%';

};

**const** calcScore = (total, goal) **=>** {

  return percentToDecimal(total / goal);

};

export **const** CalculateScore = ({ Name, School, total, goal }) **=>** (

  <div className="formatstyle">

    <h1 className="heading">Student Details:</h1>

    <div className="Name">

      <b><span> Name: </span></b>

      <span>{Name}</span>

    </div>

    <div className="School">

      <b><span> School: </span></b>

      <span>{School}</span>

    </div>

    <div className="Total">

      <b><span>Total:</span></b>

      <span>{total}</span>

      <span> Marks</span>

    </div>

    <div className="Score">

      <b>Score:</b>

      <span> {calcScore(total, goal)}</span>

    </div>

  </div>

);

myStyle.css

.Name {

  font-weight: 300;

  color: blue;

}

.School {

  color: crimson;

}

.Total {

  color: darkmagenta;

}

.formatstyle {

  text-align: center;

  font-size: large;

}

.Score {

  color: forestgreen;

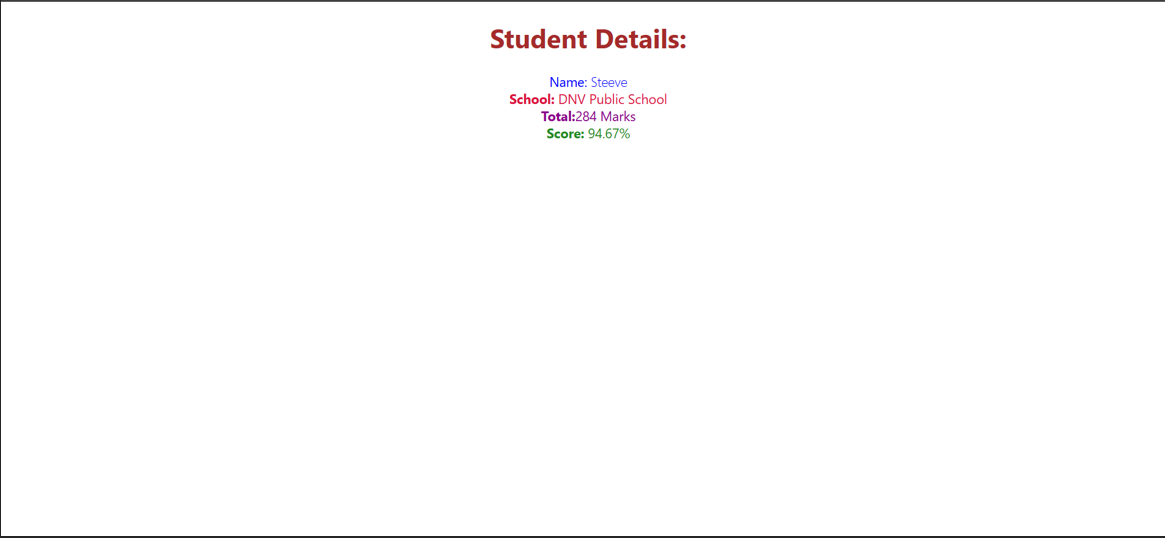
}

.heading {

  color: brown;

}

OUTPUT

****

**LAB 4**

**1. Explain the Need and Benefits of Component Lifecycle**

**What is the Component Lifecycle?**

**In React, a component lifecycle refers to the series of methods that are called at different stages of a component’s existence — from creation, updating, to removal (unmounting).**

**Why is it Needed?**

* **To control what happens before, during, and after rendering**
* **To fetch data, set timers, or manipulate the DOM at the right moment**
* **To optimize performance and avoid memory leaks**
* **To respond to prop or state changes**

**Benefits**

| **Benefit** | **Explanation** |
| --- | --- |
| **Control over behavior** | **You can decide what should happen when the component mounts, updates, or unmounts.** |
| **Data fetching timing** | **Fetch data after the component is ready (componentDidMount or useEffect).** |
| **Cleanup actions** | **Remove timers or subscriptions before a component unmounts (componentWillUnmount).** |
| **Optimization** | **Prevent unnecessary re-renders with methods like shouldComponentUpdate.** |

**2. Identify Various Lifecycle Hook Methods**

**Lifecycle methods exist mainly in Class Components (though Function Components use Hooks instead).**

**Class Component Lifecycle Methods:**

| **Phase** | **Method Name** | **Description** |
| --- | --- | --- |
| **Mounting** | **constructor()** | **Initializes state and bindings** |
|  | **static getDerivedStateFromProps()** | **Syncs state with props** |
|  | **render()** | **Returns JSX for rendering** |
|  | **componentDidMount()** | **Called once after first render (used for API calls, DOM setup)** |
| **Updating** | **shouldComponentUpdate()** | **Controls whether re-rendering happens** |
|  | **getSnapshotBeforeUpdate()** | **Captures values (like scroll position) before update** |
|  | **componentDidUpdate()** | **Called after updates** |
| **Unmounting** | **componentWillUnmount()** | **Cleanup tasks (timers, listeners) before component is destroyed** |
| **Error Handling** | **componentDidCatch()** | **Catches rendering errors** |

**Function Component Equivalent (using Hooks):**

| **Hook** | **Equivalent Lifecycle** |
| --- | --- |
| **useEffect(() => {}, [])** | **componentDidMount()** |
| **useEffect(() => {... return () => {...} }, [])** | **componentDidMount() + componentWillUnmount()** |
| **useEffect(() => {}, [dependencies])** | **componentDidUpdate()** |
| **useLayoutEffect** | **Similar to getSnapshotBeforeUpdate** |

**3. List the Sequence of Steps in Rendering a Component**

**Let’s look at the full lifecycle flow in class components:**

**Mounting (Component is being created)**

1. **constructor()**
2. **static getDerivedStateFromProps()**
3. **render()**
4. **componentDidMount()**

**Updating (Props or state changes)**

1. **static getDerivedStateFromProps()**
2. **shouldComponentUpdate()**
3. **render()**
4. **getSnapshotBeforeUpdate()**
5. **componentDidUpdate()**

**Unmounting (Component is removed from DOM)**

**componentWillUnmount()**

**CODE**

**Post.js**

1. *// src/Post.js*
2. **class** Post {
3. **constructor**(id, title, body) {
4. this.id = id;
5. this.title = title;
6. this.body = body;
7. }
8. }
9. export default Post;

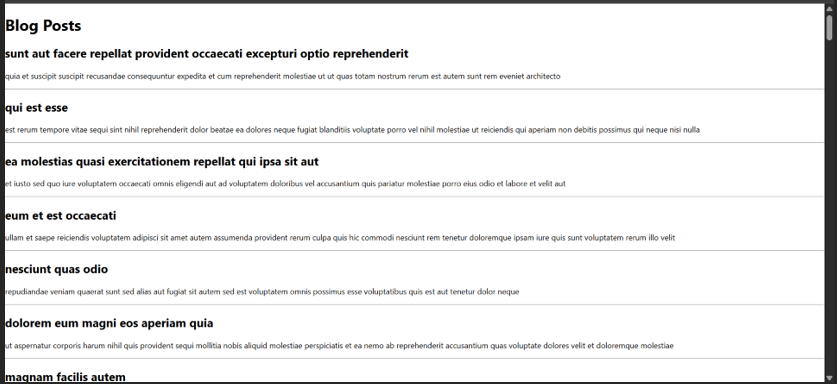
**Posts.js**

1. *// src/Posts.js*
2. import React from "react";
3. import Post from "./Post";
4. **class** Posts **extends** React.Component {
5. **constructor**(props) {
6. super(props);
7. this.state = {
8. posts: [],
9. };
10. }
11. loadPosts() {
12. fetch("https://jsonplaceholder.typicode.com/posts")
13. .then((response) **=>** response.json())
14. .then((data) **=>** {
15. **const** postObjects = data.map(
16. (item) **=>** new Post(item.id, item.title, item.body)
17. );
18. this.setState({ posts: postObjects });
19. })
20. .catch((error) **=>** {
21. console.error("Error loading posts:", error);
22. alert("Error loading posts");
23. });
24. }
25. componentDidMount() {
26. this.loadPosts();
27. }
28. componentDidCatch(error, info) {
29. alert("An error occurred in Posts component.");
30. console.error("Caught error:", error, info);
31. }
32. render() {
33. return (
34. <div>
35. <h1>Blog Posts</h1>
36. {this.state.posts.map((post) **=>** (
37. <div key={post.id}>
38. <h2>{post.title}</h2>
39. <p>{post.body}</p>
40. <hr />
41. </div>
42. ))}
43. </div>
44. );
45. }
46. }
47. export default Posts;

**App.js**

1. *// src/App.js*
2. import React from "react";
4. import Posts from "./Components/Posts";
5. **function** App() {
6. return (
7. <div className="App">
8. <Posts />
9. </div>
10. );
11. }
12. export default App;

**Output**

****

**LAB 5**

**CODE:**

**App.css**

**.App {**

**text-align: center;**

**}**

**.App-logo {**

**height: 40vmin;**

**pointer-events: none;**

**}**

**@media (prefers-reduced-motion: no-preference) {**

**.App-logo {**

**animation: App-logo-spin infinite 20s linear;**

**}**

**}**

**.App-header {**

**background-color: #282c34;**

**min-height: 100vh;**

**display: flex;**

**flex-direction: column;**

**align-items: center;**

**justify-content: center;**

**font-size: calc(10px + 2vmin);**

**color: white;**

**}**

**.App-link {**

**color: #61dafb;**

**}**

**@keyframes App-logo-spin {**

**from {**

**transform: rotate(0deg);**

**}**

**to {**

**transform: rotate(360deg);**

**}**

**}**

**Cohort.js**

**class Cohort {**

**constructor(cohortCode,**

**startDate,**

**technology,**

**trainerName,**

**coachName,**

**currentStatus) {**

**this.cohortCode = cohortCode;**

**this.coachName = coachName;**

**this.trainerName = trainerName;**

**this.technology = technology;**

**this.startDate = startDate;**

**this.currentStatus = currentStatus;**

**}**

**}**

**const CohortsData =[**

**new Cohort('INTADMDF10','22-Feb-2022', '.NET FSD', 'Jojo Jose','Aathma', 'Scheduled'),**

**new Cohort('ADM21JF014','10-Sep-2021', 'Java FSD', 'Elisa Smith','Apoorv', 'Ongoing'),**

**new Cohort('CDBJF21025','24-Dec-2021', 'Java FSD', 'John Doe','Aathma', 'Ongoing'),**

**new Cohort('INTADMJF12','22-Feb-2022', 'Java FSD', 'To Be Assigned','Ibrahim', 'Scheduled'),**

**new Cohort('CDE22JF011','24-Dec-2021', 'Java FSD', 'Emma Swan','Apoorv', 'Ongoing'),**

**new Cohort('INTADMDF09','22-Feb-2022', 'Dataware Housing', 'Babjee Rao','Aathma', 'Scheduled'),**

**new Cohort('ADM22DF001','10-Sep-2021', '.NET FSD', 'Marie Curie','Ibrahim', 'Ongoing'),**

**];**

**export {Cohort, CohortsData};**

**CohortDetails.js**

***// src/CohortDetails.js***

**import React from 'react';**

**import styles from './CohortDetails.module.css'; *// ✅ Import CSS Module***

**function CohortDetails(props) {**

**const { cohort } = props;**

***// Conditional color logic for h3***

**const titleStyle = {**

**color: cohort.currentStatus === 'Ongoing' ? 'green' : 'blue'**

**};**

**return (**

**<div className={styles.box}> {*/\* ✅ Apply box style \*/*}**

**<h3 style={titleStyle}>**

**{cohort.cohortCode} - <span>{cohort.technology}</span>**

**</h3>**

**<dl>**

**<dt>Started On</dt>**

**<dd>{cohort.startDate}</dd>**

**<dt>Current Status</dt>**

**<dd>{cohort.currentStatus}</dd>**

**<dt>Coach</dt>**

**<dd>{cohort.coachName}</dd>**

**<dt>Trainer</dt>**

**<dd>{cohort.trainerName}</dd>**

**</dl>**

**</div>**

**);**

**}**

**export default CohortDetails;**

**CohortDetails.module.css**

***/\* src/CohortDetails.module.css \*/***

**.box {**

**width: 300px;**

**display: inline-block;**

**margin: 10px;**

**padding: 10px 20px;**

**border: 1px solid black;**

**border-radius: 10px;**

**}**

**dt {**

**font-weight: 500;**

**}**

**App.js**

***// src/App.js***

**import React from 'react';**

**import './App.css';**

**import { CohortsData } from './Cohort';**

**import CohortDetails from './CohortDetails';**

**function App() {**

**return (**

**<div className="App">**

**<h1>Cohorts Details</h1>**

**<div style={{ display: 'flex', flexWrap: 'wrap', justifyContent: 'center' }}>**

**{CohortsData.map((cohort, index) => (**

**<CohortDetails key={index} cohort={cohort} />**

**))}**

**</div>**

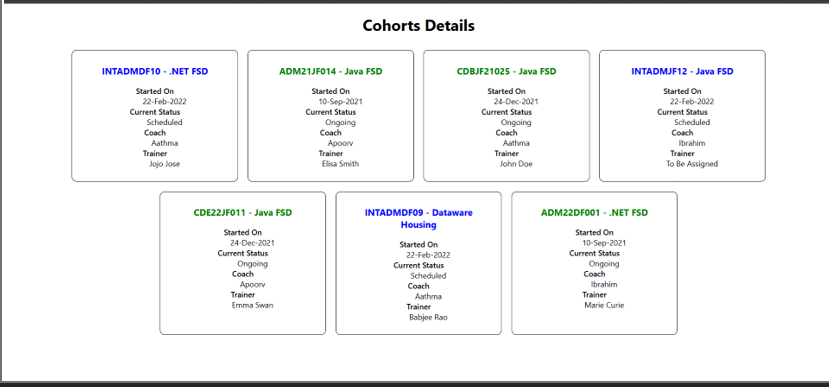
**</div>**

**);**

**}**

**export default App;**

**Output**

****