# **REACT JS**

**HANDS-ON 1**

**1. Theoretical Objectives:**

**Define SPA and its benefits:**

SPA (Single Page Application) is a web app that loads a single HTML page and dynamically updates the content without reloading the whole page.  
Benefits:

* Faster page transitions
* Better user experience
* Reduced server load
* Efficient caching

**Define React and identify its working:**

React is a JavaScript library for building user interfaces, maintained by Meta (Facebook).  
Working:

* Uses components to build UI
* Uses Virtual DOM to efficiently update the browser DOM
* Updates only the necessary parts of the UI when data changes

**Differences between SPA and MPA:**

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |
| Page Loads | Single page load | Multiple full page loads |
| Speed | Faster after initial load | Slower navigation |
| UX | Smoother, dynamic experience | Traditional, full reload |
| SEO | Harder (but can be optimized) | Better out of the box |

**Pros & Cons of SPA:**

**Pros:**

* Fast navigation
* Responsive UI
* Reusable components

**Cons:**

* SEO challenges
* Initial load might be heavy
* Complex state management in large apps

**Explain about React:**

React is a component-based, declarative JavaScript library used to build fast and interactive UIs. It helps manage the view layer of web applications and works well with other libraries or frameworks.

**Define Virtual DOM:**

Virtual DOM is a lightweight copy of the real DOM. React uses it to track changes and only updates the parts of the actual DOM that changed — making it efficient and fast.

**Features of React:**

* JSX (JavaScript + HTML)
* Component-based architecture
* Unidirectional data flow
* Virtual DOM
* Reusable components
* Rich ecosystem

**Practical Lab Steps**

**Tools Used:**

* Node.js
* npm
* Visual Studio Code

**Steps Followed:**

1. Installed Node.js and npm from <https://nodejs.org/en/download>
2. Installed create-react-app using:

npm install -g create-react-app

1. Created app using:

npx create-react-app myfirstreact

1. Navigated to app directory:

cd myfirstreact

1. Opened folder in VS Code:

code .

1. Edited App.js inside src folder with the following code:

import React from 'react';

function App() {

return (

<div>

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

</div>

);

}

export default App;

1. Started the app using:

npm start

1. Browser opened http://localhost:3000 showing:

**Welcome to the first session of React**

**OUTPUT SCREENSHOTS:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**The React app ran successfully and displayed the message: Welcome to the first session of React**

**HANDS-0N 2**

**Objectives**

**1. React Components**

Components are reusable pieces of UI in React. Each component returns JSX which gets rendered to the DOM.

**2. Components vs JavaScript Functions**

| **React Component** | **JavaScript Function** |
| --- | --- |
| Returns JSX (UI) | Returns data or value |
| Can maintain state | Cannot maintain state |
| Has lifecycle methods | No lifecycle |
| Renders part of the UI | Used for logic or computation |

**3. Types of Components**

* Class Components – ES6 classes, support state & lifecycle
* Function Components – Simple functions (can use hooks)

**4. Class Component**

A React component defined using a class that extends React.Component. It must have a render() method.

**5. Function Component**

A simple JavaScript function that returns JSX. Preferred for most cases, especially with Hooks.

**6. Component Constructor**

In a class component, the constructor is used to initialize state and bind methods. It runs once when the component is created.

**7. render() Function**

Every class component must have a render() method that returns the JSX to render on screen.

**Hands-on Lab – Practical Steps**

**Tools Required:**

* Node.js
* npm
* Visual Studio Code

**Step-by-Step Implementation:**

**Step 1: Create a React Project**

Open Command Prompt and run:

npx create-react-app StudentApp

**Step 2: Create Components Folder**

Go to your project folder (StudentApp/src) and:

1. Create a folder named: Components
2. Inside Components, create three files:
   * Home.js
   * About.js
   * Contact.js

#### **Step 3: Code for Home.js**

**Inside src/Components/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;

**Step 4: Code for About.js**

**Inside src/Components/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 About;

**Step 5: Code for Contact.js**

**Inside src/Components/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;

**Step 6: Modify App.js to Call All 3 Components**

**Open src/App.js and replace everything with:**

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

return (

<div>

<h1>Student Management Portal</h1>

<Home />

<About />

<Contact />

</div>

);

}

export default App;

**Step 7: Run the App**

**In terminal (inside StudentApp folder):**

npm start

**Step 8: View the Output**

**Open your browser and go to:**

[**http://localhost:3000**](http://localhost:3000)

**You should see:**

Student Management Portal

Welcome to the Home page of Student Management Portal

Welcome to the About page of the Student Management Portal

Welcome to the Contact page of the Student Management Portal

**OUTPUT SCREENSHOTS:**

A screenshot of a computer

AI-generated content may be incorrect.

**HANDS-ON- 3**

**Objectives Section (Theory)**

**1. React Components**

Components are the building blocks of React applications. They allow you to break the UI into reusable and isolated parts.

**2. Components vs JavaScript Functions**

| React Component | JavaScript Function |
| --- | --- |
| Returns JSX/UI | Returns values/data |
| Can have state & props | No concept of state |
| Meant for UI | Meant for logic |

**3. Types of Components**

* Class Component – Uses ES6 classes, can have lifecycle methods
* Function Component – Uses functions, supports Hooks for state

**4. Class Component**

Defined using class, extends React.Component, and must have render().

**5. Function Component**

Simpler component defined as a JavaScript function, returning JSX directly.

**6. Component Constructor**

Only used in class components to initialize state or bind methods.

**7. render() Function**

Only present in class components. Used to return JSX that defines the UI.

**Hands-on Lab – Step-by-Step**

**Tools Required**

* Node.js
* npm
* Visual Studio Code

### **Step 1: Create the React App**

In **VS Code terminal** run:

npx create-react-app scorecalculatorapp

**Step 2: Create Components Folder**

Inside the project directory (scorecalculatorapp/src):

* Create a new folder named:

Components

Inside that, create a new file:

CalculateScore.js

**Step 3: Code for CalculateScore.js**

import React from 'react';

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

function CalculateScore(props) {

const average = props.total / props.goal;

return (

<div className="score-box">

<h2>Student Score Summary</h2>

<p><strong>Name:</strong> {props.name}</p>

<p><strong>School:</strong> {props.school}</p>

<p><strong>Total Score:</strong> {props.total}</p>

<p><strong>Goal:</strong> {props.goal}</p>

<p><strong>Average Score:</strong> {average.toFixed(2)}</p>

</div>

);

}

export default CalculateScore;

**Step 4: Add Styling (CSS)**

Create a folder inside src named:

Stylesheets

Inside that, create a file:

mystyle.css

.score-box {

border: 2px solid #007acc;

padding: 20px;

margin: 20px;

border-radius: 10px;

background-color: #f0f8ff;

font-family: Arial, sans-serif;

color: #333;

}

**Step 5: Modify App.js to Use the Component**

Open src/App.js and replace all code with:

import React from 'react';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div>

<h1>Student Management Portal</h1>

<CalculateScore name="Bhavya Injam" school="KL University" total={480} goal={6} />

</div>

);

}

export default App;

**Step 6: Run the App**

In terminal:

cd scorecalculatorapp

npm start

**Step 7: View the Output**

**Open browser and go to:**

**http://localhost:3000**

**You should see:**

Student Management Portal

Student Score Summary

Name: Bhavya Injam

School: KL University

Total Score: 480

Goal: 6

Average Score: 80.00

**OUTPUT SCREENSHOT:**

A screenshot of a computer

AI-generated content may be incorrect.

**HANDS-ON 4**

**Steps to Follow for "blogapp" Hands-on:**

**1. Create the React App**

npx create-react-app blogapp

**2. Open the project in VS Code**

cd blogapp

code .

**3. Inside src/, create a file named Post.js**

import React from 'react';

class Post extends React.Component {

render() {

return (

<div>

<h2>{this.props.title}</h2>

<p>{this.props.body}</p>

</div>

);

}

}

export default Post;

**4. Create Posts.js file (class-based component)**

import React from 'react';

import Post from './Post';

class Posts extends React.Component {

constructor(props) {

super(props);

this.state = {

posts: [],

hasError: false

};

}

loadPosts = () => {

fetch('https://jsonplaceholder.typicode.com/posts')

.then((response) => response.json())

.then((data) => this.setState({ posts: data }))

.catch((error) => {

console.error("Fetch Error:", error);

this.setState({ hasError: true });

});

};

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

alert("An error occurred: " + error);

}

render() {

if (this.state.hasError) {

return <h2>Something went wrong.</h2>;

}

return (

<div>

<h1>Blog Posts</h1>

{this.state.posts.slice(0, 10).map((post) => (

<Post key={post.id} title={post.title} body={post.body} />

))}

</div>

);

}

}

export default Posts;

**5. Modify App.js to use Posts**

import React from 'react';

import Posts from './Posts';

function App() {

return (

<div className="App">

<Posts />

</div>

);

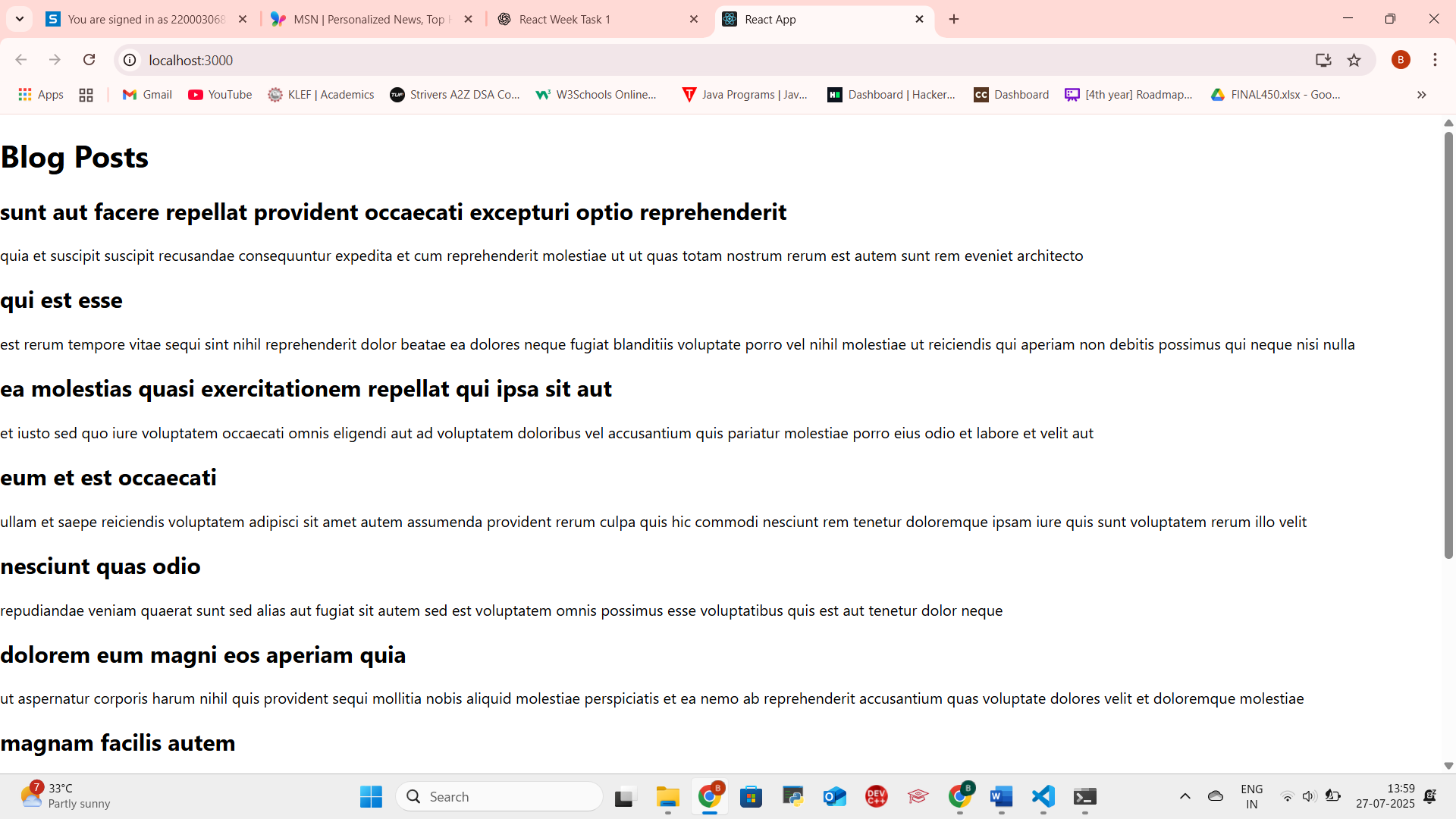
}

export default App;

**6. Run the app**

npm start

**OUTPUT SCREENSHOTS:**



**HANDS-ON 5**

**STEP-BY-STEP**

**STEP-BY-STEP GUIDE**

**🔹 Step 1: Create a React App**

Open terminal (PowerShell or Command Prompt):

npx create-react-app cohort-dashboard

After it finishes:

cd cohort-dashboard

code .

**Step 2: Set Up Folder Structure**

Inside src/, create a new folder components.

Inside components/, create:

* CohortDetails.js – for the component
* CohortDetails.module.css – for the CSS Module

**Step 3: Add Data and Component Code**

**CohortDetails.js**

import React from 'react';

import styles from './CohortDetails.module.css';

const CohortDetails = () => {

const cohorts = [

{ name: 'React Bootcamp', status: 'ongoing', trainer: 'Suresh' },

{ name: 'Spring Boot Fundamentals', status: 'completed', trainer: 'Anjali' },

{ name: 'Cloud Foundations', status: 'ongoing', trainer: 'Vikram' },

];

return (

<div>

<h2>Cohort Dashboard</h2>

{cohorts.map((cohort, index) => (

<div key={index} className={styles.box}>

<dl>

<dt>Cohort Name:</dt>

<dd>{cohort.name}</dd>

<dt>Status:</dt>

<dd>

<h3 style={{ color: cohort.status === 'ongoing' ? 'green' : 'blue' }}>

{cohort.status}

</h3>

</dd>

<dt>Trainer:</dt>

<dd>{cohort.trainer}</dd>

</dl>

</div>

))}

</div>

);

};

export default CohortDetails;

**Step 4: Add CSS Styles**

**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;

}

**Step 5: Use the Component in App.js**

Open App.js and replace everything with:

import React from 'react';

import CohortDetails from './components/CohortDetails';

function App() {

return (

<div className="App">

<CohortDetails />

</div>

);

}

export default App;

**Step 6: Run the App**

In terminal:

npm start

Visit http://localhost:3000 in your browser.

**OUTPUT SCREENSHOTS:**

