**Week6\_hands\_on**

**Handson 1:**

**1. Define SPA and its Benefits**

* **SPA (Single Page Application):** A web application that dynamically rewrites the current page instead of loading new pages from the server.
* **Benefits:**
  + Faster navigation (no full-page reloads)
  + Better user experience (like mobile apps)
  + Efficient use of API (fetches only required data)

**2. Define React and Identify its Working**

* **React:** A JavaScript library developed by Facebook for building user interfaces using components.
* **How React works:**
  + Uses **Virtual DOM** to efficiently update the UI.
  + Breaks UI into reusable **components**.
  + Uses **JSX** (JavaScript XML) syntax to write HTML in JS.

**3. Difference Between SPA and MPA**

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| Pages | Single page dynamically updates | Multiple pages (reloads each time) |
| Speed | Faster, after initial load | Slower (loads new pages) |
| Examples | Gmail, Facebook | Amazon, News Sites |

**4. Pros & Cons of SPA**

| **Pros** | **Cons** |
| --- | --- |
| Fast and Responsive UI | Initial load time can be high |
| Easier for Mobile-like feel | SEO can be challenging |
| Efficient API data fetching | Can be complex to manage |

**5. Explain about React**

* **React** builds modern UIs using:
  + **Components** – small reusable blocks.
  + **JSX** – write HTML + JS together.
  + **Virtual DOM** – faster updates.
  + **React Hooks** – manage state and lifecycle without classes.

**6. Define Virtual DOM**

* **Virtual DOM** is a lightweight copy of the actual DOM in memory.
* React compares old vs new Virtual DOM using **diffing algorithm** and updates only the changed elements, making React apps fast.

**7. Features of React**

* Component-based architecture
* Virtual DOM for faster performance
* JSX syntax
* React Hooks
* One-way Data Binding
* High Performance UI

Step 1:

npm install -g create-react-app

step2:

npx create-react-app myfirstreact

step3:

cd myfirstreact

step4:

import React from 'react';

function App() {

return (

<div>

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

</div>

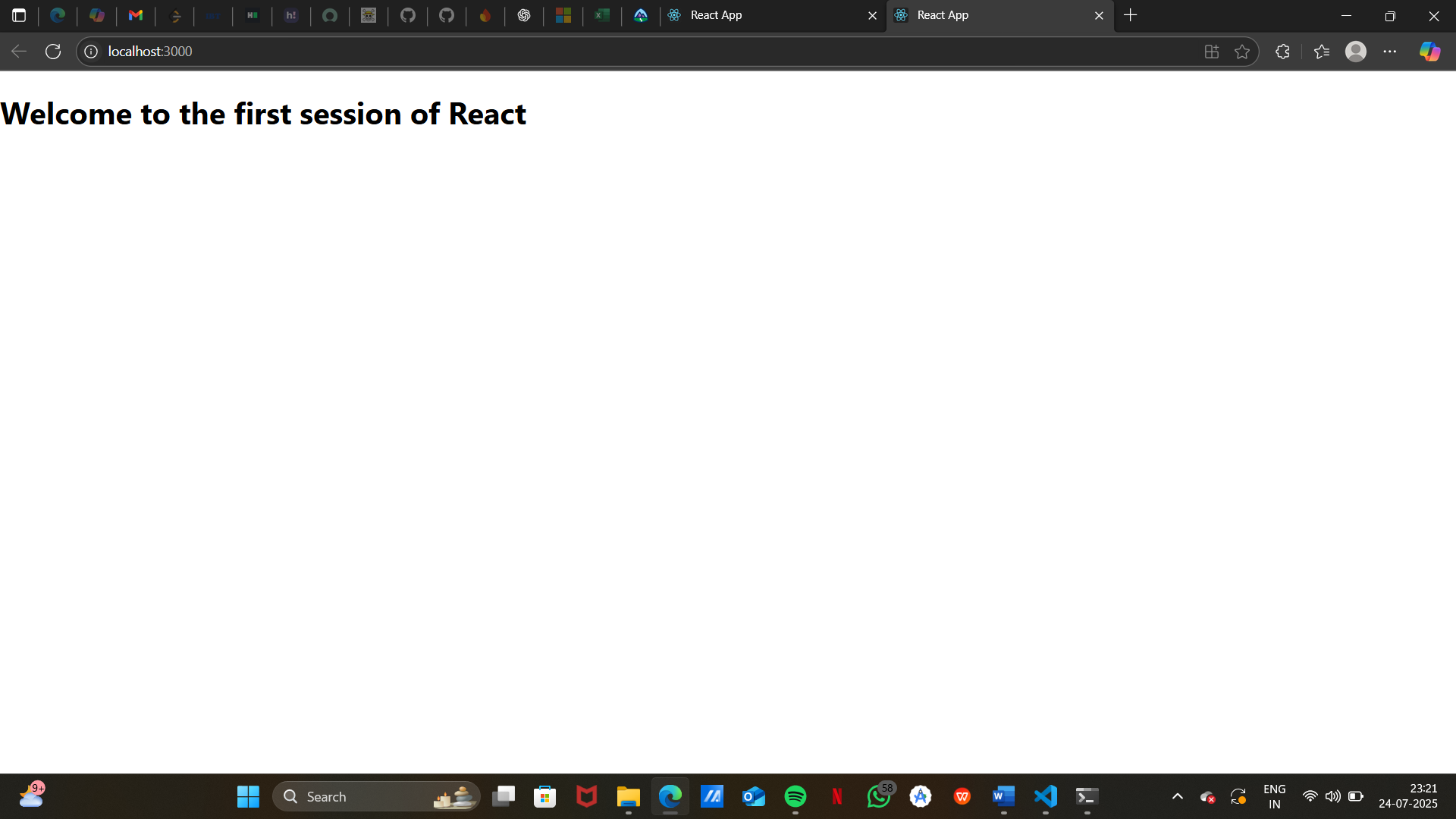
);

}

export default App;

step 5:

npm start

Output:

Handson2:

**1. Explain React Components**

* **React components** are the **building blocks** of a React application.
* They return **JSX** (HTML-like syntax) to render UI.

**2. Difference: React Components vs JS Functions**

| **JavaScript Function** | **React Component** |
| --- | --- |
| Returns a value | Returns JSX (UI) |
| Pure logic or operations | Describes UI elements |
| No lifecycle methods | Can have lifecycle methods |

**3. Types of Components**

* **Functional Components**
* **Class Components**

**4. Explain Class Component**

* A class component uses ES6 class syntax.
* Must extend React.Component.
* Has a render() method to return JSX.

import React, { Component } from 'react';

class Example extends Component {

render() {

return <h1>This is a class component</h1>;

}

}

**5. Explain Function Component**

* Simple JavaScript function that returns JSX.

function Example() {

return <h1>This is a function component</h1>;

}

**6. Define Component Constructor**

* **Constructor** initializes the component’s state.

constructor(props) {

super(props);

this.state = { message: "Hello" };

}

**7. Define render() Function**

* The render() method returns what should be displayed on the UI.

**Step 1:**

npx create-react-app StudentApp

cd StudentApp

**Step 2:**

**Path:** src/Components/Home.js

import React, { Component } from 'react';

class Home extends Component {

render() {

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

}

}

export default Home;

**Step 3:**

**Path:** src/Components/About.js

import React, { Component } from 'react';

class About extends Component {

render() {

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

}

}

export default About;

**Step 4:**

**Path:** src/Components/Contact.js

import React, { Component } from 'react';

class Contact extends Component {

render() {

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

}

}

export default Contact;

**Step 5:**

**Path:** src/App.js

Replace the code with:

import React from 'react';

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>

);

}

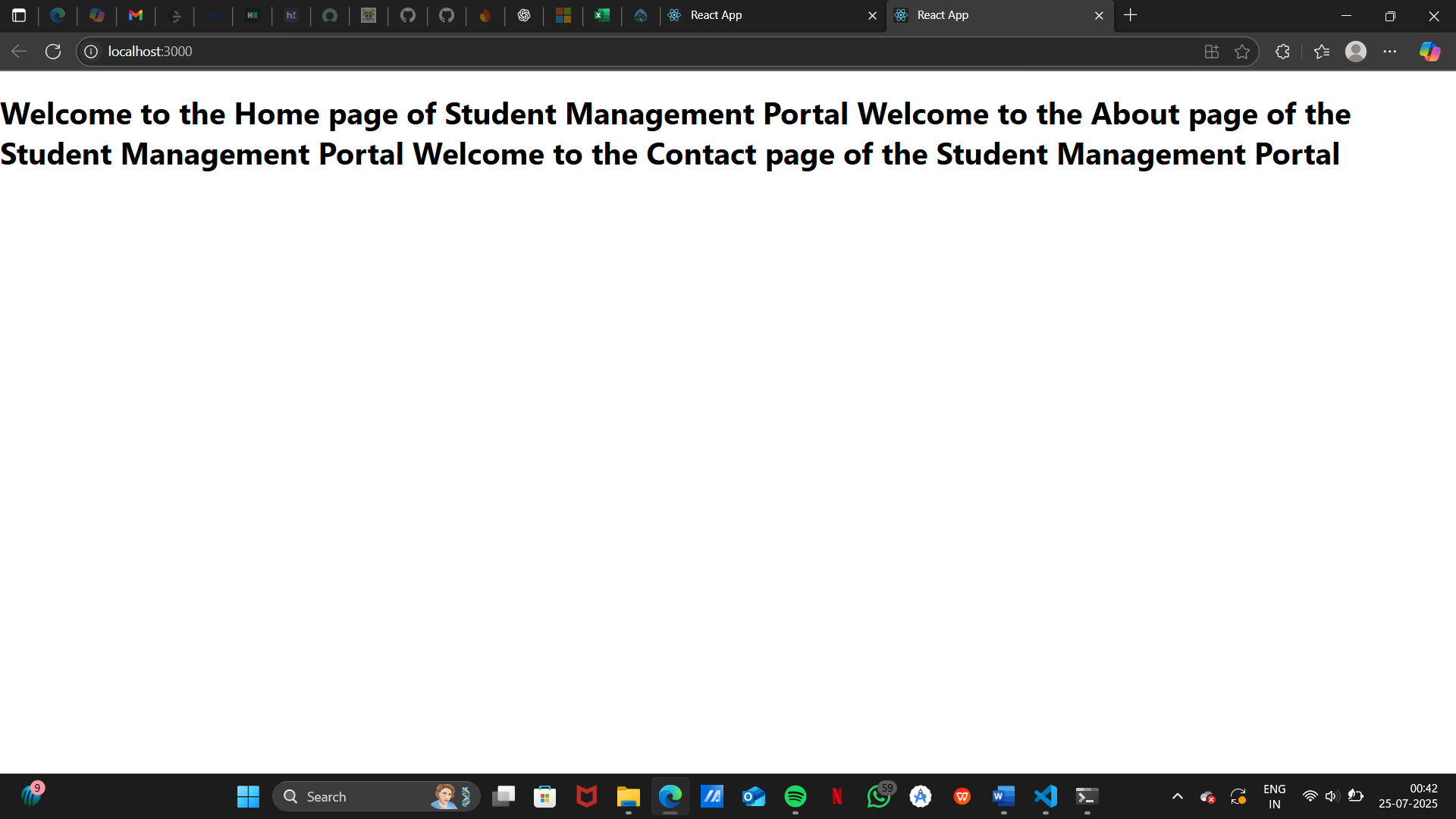
export default App;

**Step 6:**

In the terminal:

npm start

**outpt:**



Handson 3:

**React components**: Reusable UI pieces that return JSX.

* **Components vs JS functions**:
  + JS function → any value.
  + React component → JSX + lifecycle/state (via hooks or class APIs).
* **Types of components**: **Function components** (modern, hooks) & **Class components** (older, render()).
* **Class component**: ES6 class extending React.Component, usually has constructor, state, render().
* **Function component**: Plain function returning JSX; uses **hooks** (useState, useEffect, …).
* **Component constructor**: In class components, constructor(props){ super(props); ... } to init state/bindings.
* **render()**: Class component method that returns JSX.

**step1:**

npx create-react-app scorecalculatorapp

cd scorecalculatorapp

code .

**step2:**

* Components/CalculateScore.js
* Stylesheets/mystyle.css

**step3:**

import React from 'react';

import PropTypes from 'prop-types';

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

const CalculateScore = ({ name, school, total, goal }) => {

// Interpret “goal” as number of subjects (so average per subject)

const average = goal ? (total / goal).toFixed(2) : 'N/A';

return (

<div className="card">

<h1 className="title">Student Score Calculator</h1>

<div className="row">

<span className="label">Name:</span>

<span>{name}</span>

</div>

<div className="row">

<span className="label">School:</span>

<span>{school}</span>

</div>

<div className="row">

<span className="label">Total:</span>

<span>{total}</span>

</div>

<div className="row">

<span className="label">Goal (subjects):</span>

<span>{goal}</span>

</div>

<h2 className="average">Average: {average}</h2>

</div>

);

};

CalculateScore.propTypes = {

name: PropTypes.string.isRequired,

school: PropTypes.string.isRequired,

total: PropTypes.number.isRequired,

goal: PropTypes.number.isRequired,

};

export default CalculateScore;

**step4:**

body {

font-family: system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,

Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue", sans-serif;

background: #f7f7fb;

}

.card {

max-width: 520px;

margin: 2rem auto;

background: #ffffff;

padding: 1.75rem 2rem;

border-radius: 12px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.08);

}

.title {

margin: 0 0 1.25rem 0;

text-align: center;

}

.row {

display: flex;

justify-content: space-between;

margin: 0.4rem 0;

}

.label {

font-weight: 600;

color: #555;

}

.average {

margin-top: 1.5rem;

text-align: center;

color: #0d6efd;

}

**step5:**

import React from 'react';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div>

<CalculateScore

name="Sanjana Shetty"

school="GSSSIETW Mysore"

total={455}

goal={6}

/>

</div>

);

}

export default App;

**step6:**

npm start

Output:



Handson 4:

**Need & Benefits of Component Lifecycle**

React component lifecycle allows you to:

* Run **code at specific points** (e.g., when the component mounts, updates, or unmounts)
* Handle **API calls**, **cleanups**, and **error boundaries**
* Improve performance and maintainability

**Lifecycle Hook Methods**

1. **Mounting Phase**
   * constructor()
   * static getDerivedStateFromProps()
   * render()
   * componentDidMount()
2. **Updating Phase**
   * shouldComponentUpdate()
   * componentDidUpdate()
3. **Unmounting Phase**
   * componentWillUnmount()
4. **Error Handling**
   * componentDidCatch()

**Rendering Sequence**

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

step1:

npx create-react-app blogapp

cd blogapp

code .

step 2:

class Post {

constructor(userId, id, title, body) {

this.userId = userId;

this.id = id;

this.title = title;

this.body = body;

}

}

export default Post;

step 3:

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

constructor(props) {

super(props);

this.state = {

posts: [],

error: null

};

}

// Fetch API data

loadPosts = async () => {

try {

const response = await fetch('https://jsonplaceholder.typicode.com/posts');

const data = await response.json();

const postList = data.map(

(item) => new Post(item.userId, item.id, item.title, item.body)

);

this.setState({ posts: postList });

} catch (err) {

this.setState({ error: err.message });

}

};

// Lifecycle: Called after render

componentDidMount() {

this.loadPosts();

}

// Lifecycle: Catch errors in rendering

componentDidCatch(error, info) {

alert(`Error occurred: ${error}`);

}

render() {

const { posts } = this.state;

return (

<div style={{ padding: '2rem' }}>

<h1>Blog Posts</h1>

{posts.length === 0 ? (

<p>Loading posts...</p>

) : (

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

<div key={post.id} style={{ marginBottom: '1.5rem' }}>

<h2>{post.title}</h2>

<p>{post.body}</p>

<hr />

</div>

))

)}

</div>

);

}

}

export default Posts;

Step 4:

Replace src/App.js with:

import React from 'react';

import Posts from './Posts';

function App() {

return (

<div className="App">

<Posts />

</div>

);

}

export default App;

step5:

npm start

Handson 5:

CSS Modules  
 Conditional inline styling  
 React component structure best practices

**Why Style React Components?**

* Improves user interface (UI)
* Separates concerns (styles vs logic)
* Supports component reusability and modularization

**Styling Techniques in React**

1. **CSS Modules** – Scoped styles using .module.css
2. **Inline styles** – Dynamic styling with objects

**Step 1:**

Run:

npm install

**Step 2:**

.box {

width: 300px;

display: inline-block;

margin: 10px;

padding: 10px 20px;

border: 1px solid black;

border-radius: 10px;

}

dt {

font-weight: 500;

}

**Step 3:**

import React from 'react';

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

function CohortDetails({ name, batch, status }) {

const statusStyle = {

color: status.toLowerCase() === 'ongoing' ? 'green' : 'blue'

};

return (

<div className={styles.box}>

<dl>

<dt>Cohort Name:</dt>

<dd>{name}</dd>

<dt>Batch:</dt>

<dd>{batch}</dd>

<dt>Status:</dt>

<dd>

<h3 style={statusStyle}>{status}</h3>

</dd>

</dl>

</div>

);

}

export default CohortDetails;

**Step 4:**

import React from 'react';

import CohortDetails from './CohortDetails';

function App() {

return (

<div>

<h1>My Academy Dashboard</h1>

<CohortDetails name="React Basics" batch="Jan 2025" status="Ongoing" />

<CohortDetails name="Advanced Java" batch="Feb 2025" status="Completed" />

</div>

);

}

export default App;

**Step 5:**

npm start

output:

