Week-6 React

# **Hands-on 1**

## Scenario:

Create a new React Application with the name 'myfirstreact', run the application to print 'Welcome to the first session of React' as heading of that page.

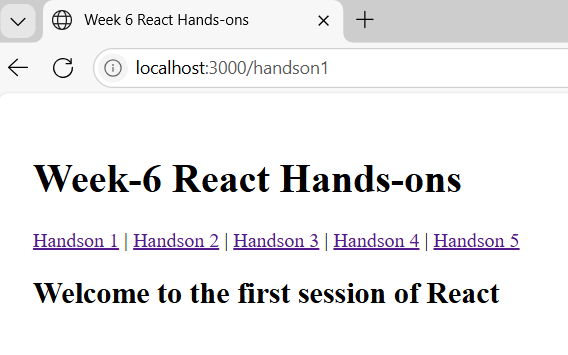
## Objectives:

Understand how to set up a React environment using create-react-app and display basic content.

## Code:

**File: myfirstreact/Handson1.js**import React from 'react';  
export default function Handson1() {  
 return <h2>Welcome to the first session of React</h2>;  
}

Output:



# Hands-on 2

## Scenario:

Create Student Management Portal with Home, About, Contact components each displaying a welcome message.

## Objectives:

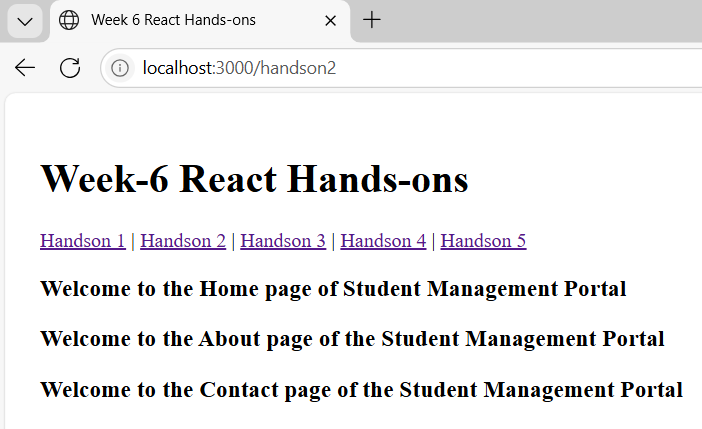
Learn about creating multiple React components and rendering them together.

## Code:

**File:StudentApp/Handson2.js**import React from 'react';  
function Home() { return <h3>Welcome to the Home page of Student Management Portal</h3>; }  
function About() { return <h3>Welcome to the About page of the Student Management Portal</h3>; }  
function Contact() { return <h3>Welcome to the Contact page of the Student Management Portal</h3>; }  
  
export default function Handson2() {  
 return (  
 <div>  
 <Home />  
 <About />  
 <Contact />

</div> );}

**Output:**



# Hands-on 3

## Scenario:

Create a Score Calculator function component to calculate average score of a student using Name, School, Total, and Goal inputs.

## Objectives:

Learn about functional components, useState, and styling in React.

## Code:

**File: scorecalculatorapp/Handson3.js**import React, { useState } from 'react';  
import React, { useState } from 'react';

import './Handson3.css';

export default function Handson3() {

  const [name, setName] = useState('');

  const [school, setSchool] = useState('');

  const [total, setTotal] = useState('');

  const [goal, setGoal] = useState('');

  const [percentage, setPercentage] = useState(null);

  const calculate = () => {

    if (parseFloat(goal) > 0) {

      const percent = ((parseFloat(total) / parseFloat(goal)) \* 100).toFixed(2);

      setPercentage(percent);

    } else {

      setPercentage(null);

      alert('Maximum marks must be greater than 0');

    }

  };

  return (

    <div className="container">

      <h2>Student Details</h2>

      <input placeholder="Name" value={name} onChange={e => setName(e.target.value)} />

      <input placeholder="School" value={school} onChange={e => setSchool(e.target.value)} />

      <input placeholder="Marks Obtained" value={total} onChange={e => setTotal(e.target.value)} />

      <input placeholder="Maximum Marks" value={goal} onChange={e => setGoal(e.target.value)} />

      <button onClick={calculate}>Calculate Percentage</button>

      {percentage && <p>{name} from {school} has scored {percentage}%</p>}

    </div>

  );

}

**/\* Handson3.css \*/**input {

  display: block;

  margin: 5px;

  padding: 5px;

}

button {

  margin-top: 10px;

  background-color: green;

  color: white;

  padding: 8px 16px;

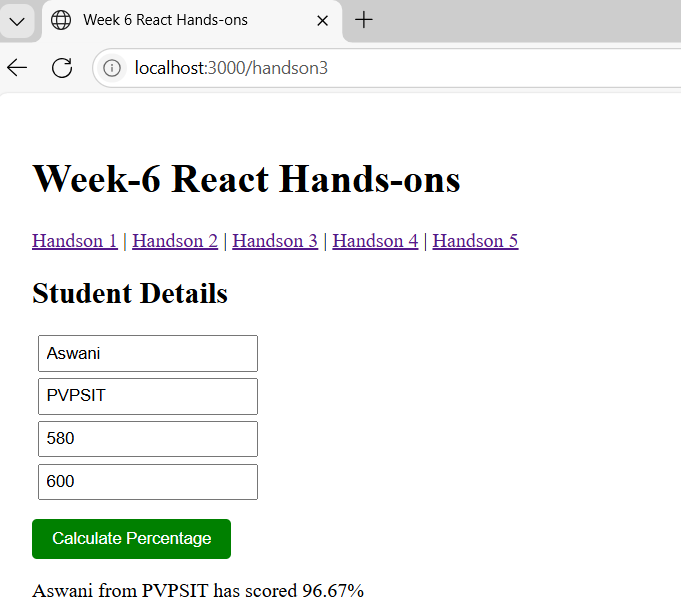
  border: none;

  border-radius: 4px;

  cursor: pointer;

}

**Output:**



# Hands-on 4

## Scenario:

Create a Blog App demonstrating componentDidMount to fetch posts and componentDidCatch to handle errors.

## Objectives:

Understand React class components, lifecycle methods, and error handling.

## Code:

**File:blogapp/Handson4.js**  
import React, { Component } from 'react';

export default class Handson4 extends Component {

  constructor() {

    super();

    this.state = { posts: [] };

  }

  componentDidMount() {

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

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

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

      .catch(error => alert('Error loading posts: ' + error));

  }

  render() { return (

      <div>

        <h2>Blog Posts</h2>

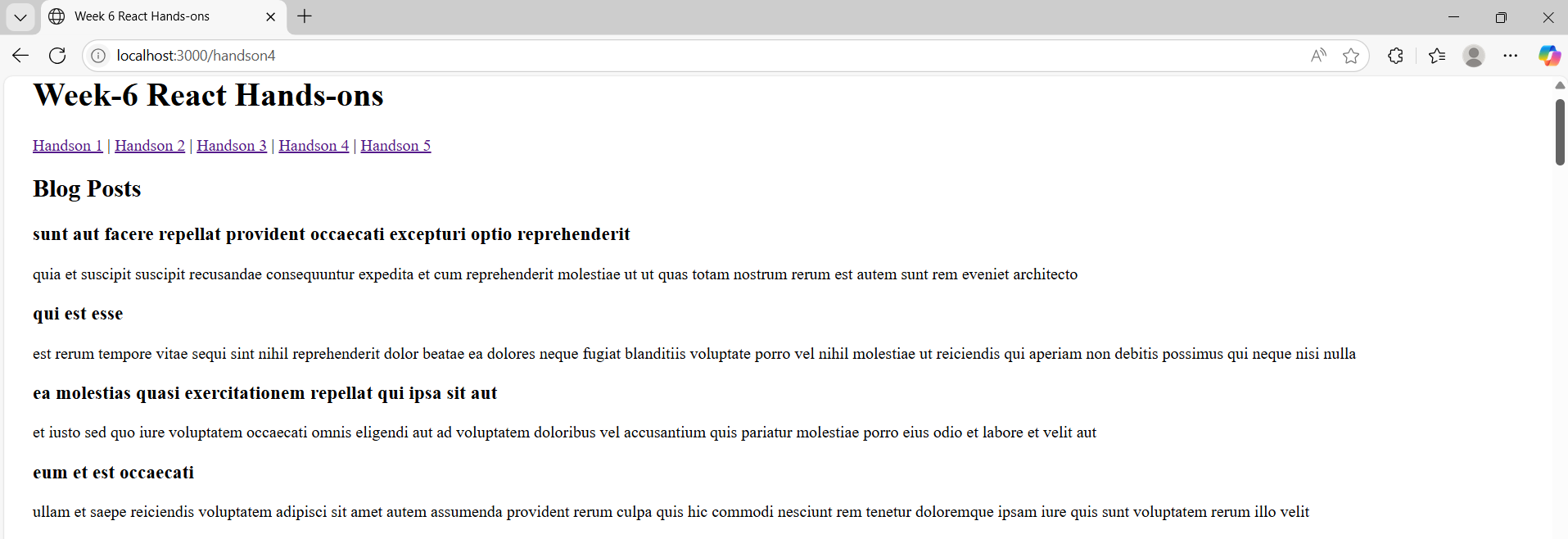
        {this.state.posts.map(post => (

          <div key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.body}</p>   </div>   ))} </div> );  }}

**Output:**

****

# Hands-on 5

## Scenario:

Create Cohort Details dashboard styled using CSS Modules and inline styles, with different colors for ongoing and completed statuses.

## Objectives:

Learn how to style React components using CSS Modules and conditional inline styles.

## Code: **File:cohorttracker/Handson5.js** import React from 'react'; import styles from './CohortDetails.module.css'; export default function Handson5() { const cohorts = [ { id: 1, name: 'React Basics', status: 'ongoing' }, { id: 2, name: 'Advanced React', status: 'completed' } ]; return ( <div> <h2>Cohort Details</h2> {cohorts.map(c => ( <div key={c.id} className={styles.box}> <h3 style={{ color: c.status === 'ongoing' ? 'green' : 'blue' }}>{c.name}</h3> <dt>Status: {c.status}</dt> </div> ))} </div> ); } **/\* 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; }

**Output:**

