**EXERCISE 2**

**OBJECTIVES:**

1. **Explain React components**

React components are reusable, independent pieces of UI in a React application. They return JSX, which describes what should appear on the screen. Components can manage their own state and accept inputs called props. They help break the UI into smaller, manageable pieces. There are two main types: class components and function components.

1. **Identify the differences between components and JavaScript functions**

React components return JSX, while regular JavaScript functions return primitive values or objects. Components can manage state and lifecycle methods; JS functions cannot. React components are used to build UI; JS functions are used for logic and computations. React components must follow specific naming conventions (start with capital letters). Components can receive props; normal functions don’t have a built-in props mechanism.

1. **Identify the types of components**

There are two main types of React components:

* **Class Components** – Use ES6 class syntax and can hold state and lifecycle methods.
* **Function Components** – Use plain functions and React Hooks for state and effects. Class components were used more in older versions of React. Function components are now preferred for most use cases due to simplicity and hooks.

1. **Explain class component**

A class component is a React component defined using the ES6 class keyword.  
It extends React. Component and must include a **render()** method. Class components can hold internal state using **this.state**. They can use lifecycle methods like **componentDidMount()** and **componentDidUpdate().** Although powerful, they are now less commonly used in favor of function components with hooks.

1. **Explain function component**

A function component is a plain JavaScript function that returns JSX. It receives props as an argument and is simpler than a class component. With React Hooks, function components can manage state and side effects. They are easy to read, write, and test, making them the preferred choice today. They don’t require lifecycle methods; instead, use hooks like **useEffect()**.

1. **Define component constructor**

The **constructor()** is a special method used in class components. It’s called when the component is created and is used to initialize state. You must call **super(props)** before using **this** in the constructor. It helps bind methods and set initial values for the component. It runs only once when the component is first instantiated.

1. **Define render() function**

The **render()** function is required in every class component.  
It returns the JSX (UI) that should be displayed on the screen.  
Whenever state or props change, the render method is called again.  
It must return a single parent element, often a **<div>**.  
The content inside **render()** describes the visual output of the component.

**HANDS ON PRACTICE**

1. **Create a React project named “scorecalculatorapp” type the following command in terminal of Visual studio**

npx create-react-app scorecalculatorapp

1. **Create a new folder under src folder with the name “Components”. Add a new file named “CalculateScore.js**

**calculateScore.js**

import '../styles/calculateScore.css';

const percentToDecimal = (decimal) => {

  return (decimal.toFixed(2) + '%');

}

const calcScore = (total, goal) => {

  return percentToDecimal(total/goal);

}

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

  return (

    <div className = "container">

      <h1><font color='brown'>Student Details</font></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 Marks:</span></b>

        <span>{total}</span>

      </div>

      <div classname='score'>

        <b><span>Percentage:</span></b>

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

      </div>

    </div>

  );

}

export default CalculateScore;

1. **Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components**

**calculateScore.css**

.container{

  text-align: center;

  font-size: large;

}

.name{

  color: blue;

}

.school{

  color: crimson;

}

.total{

  color: blueviolet

}

.score{

  color:green

}

1. **Edit the App.js to invoke the CalculateScore functional component**

**App.js**

import logo from './logo.svg';

import './App.css';

import CalculateScore from './components/calculateScore';

function App() {

  return (

    <div>

      <CalculateScore

      name={"Sanjeev V A"}

      school={"Sri Seshaas International Public School"}

      total={365}

      goal={5}/>

    </div>

  );

}

export default App;

1. **Output**

**A screenshot of a computer

AI-generated content may be incorrect.**