**Explain React components**

React components are the building blocks of a React application. They are reusable, self-contained pieces of code that define how a part of the user interface (UI) should appear and behave. Each component can manage its own state, receive data through props, and render UI elements. Components can be composed together to build complex interfaces from smaller, manageable pieces. They help maintain a clean, modular structure and promote code reusability and separation of concerns.

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

While both React components and regular JavaScript functions can contain logic, they serve different purposes. JavaScript functions are general-purpose blocks of code used to perform actions or return values. In contrast, React components are specialized functions or classes that return React elements (UI) and are used to build user interfaces. React components follow specific rules, such as using JSX for rendering and handling props and state, whereas regular JavaScript functions do not. Additionally, React components can trigger lifecycle methods and rendering behavior in the virtual DOM, while normal functions cannot.

**Identify the types of components**

React components are primarily categorized into two types: **class components** and **function components**. Class components are ES6 classes that extend React.Component and include methods like render() and optional lifecycle methods. Function components are simpler and are written as JavaScript functions that return JSX. Modern React development favors function components with hooks, as they are less verbose and easier to read while still offering powerful features like state and side effects through hooks such as useState and useEffect.

**Explain class component**

A class component in React is a component defined using an ES6 class that extends React.Component. It must include a render() method, which returns the JSX to be displayed in the UI. Class components can hold and manage their own state using this.state and can respond to lifecycle events using methods like componentDidMount, componentDidUpdate, and componentWillUnmount. While class components were the standard in early versions of React, they are now less commonly used in favor of function components with hooks.

**Explain function component**

Function components are simple JavaScript functions that accept props as input and return JSX to render UI elements. Initially, function components were stateless, but with the introduction of hooks, they can now manage state, handle side effects, and mimic lifecycle behavior. They are concise, easy to read, and currently the preferred way of writing components in modern React applications. Function components promote a cleaner and more functional programming style in React development.

**Define component constructor**

The constructor is a special method used in class components to initialize the component's state and bind event handlers. It is called automatically when a component is created. The constructor takes props as an argument and typically includes a call to super(props) to ensure the component inherits properly from React.Component. It is commonly used to set the initial value of this.state and perform any setup before the component is rendered for the first time.

**Define render() function**

The render() function is a required method in class components that describes what should be displayed in the UI. It returns JSX, which is then translated into actual DOM elements by React. Whenever a component's state or props change, the render() method is called again to update the view. This function must be pure, meaning it should only return the same output for the same input, and should not modify the state or interact with external systems directly.

**Create a react app for Student Management Portal named scorecalculatorapp and create a function component named “CalculateScore” which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.**

**Index.js :-**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

reportWebVitals();

**App.js :-**

import React from 'react';

import './App.css';

import CalculateScore from './Components/CalculateScore';

function App() {

  return (

    <div className="App">

      <CalculateScore

        name="John Doe"

        school="ABC High School"

        total={450}

        goal={5}

      />

    </div>

  );

}

export default App;

**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);

  }

}

**CalculateScore.js :-**

import React from 'react';

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

function CalculateScore(props) {

  const { name, school, total, goal } = props;

  const average = total / goal;

  return (

    <div className="score-card">

      <h2>Student Score Calculator</h2>

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

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

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

      <p><strong>Number of Goals:</strong> {goal}</p>

      <p className="highlight"><strong>Average Score:</strong> {average}</p>

    </div>

  );

}

export default CalculateScore;

**mystyle.css :-**

.score-card {

  background-color: #f0f8ff;

  padding: 20px;

  margin: 30px auto;

  width: 50%;

  border-radius: 10px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.score-card h2 {

  text-align: center;

  color: #333;

}

.score-card p {

  font-size: 18px;

  line-height: 1.5;

  color: #555;

}

.highlight {

  color: green;

  font-weight: bold;

}

Output :-



