# 1. Explain React Components

A React component is a reusable and independent piece of the user interface. Components let developers split the UI into isolated parts that can be developed, tested, and maintained separately. Each component is like a small JavaScript function or class that returns JSX, which tells React what should be displayed on the screen. Components can receive input through props (properties) and can maintain their own state to manage dynamic behavior. By combining multiple components, developers can build large, complex applications in a modular and scalable way.

# 2. Identify the Differences Between Components and JavaScript Functions

While both React components and JavaScript functions are written using the function syntax, they serve different purposes:  
  
- React Components are used to create visual parts of a web application. They return JSX, can handle state using hooks (in function components) or lifecycle methods (in class components), and are managed by the React DOM.  
  
- JavaScript Functions, on the other hand, are general-purpose blocks of reusable code that return primitive values or objects and are not related to the UI.  
  
Key differences:  
- Components start with a capital letter, functions don’t have that requirement.  
- Components return JSX, while functions return values like numbers, strings, or objects.  
- React components can interact with the React lifecycle, regular functions cannot.

# 3. Identify the Types of Components

There are two main types of React components:  
  
1. Class Components:  
- Defined using the ES6 class syntax.  
- Extend React.Component.  
- Include a render() method.  
- Can use state and lifecycle methods.  
  
Example:  
class Hello extends React.Component {  
 render() {  
 return <h1>Hello World</h1>;  
 }  
}  
  
2. Function Components:  
- Defined using standard JavaScript functions.  
- Use useState, useEffect, and other React Hooks for state and lifecycle behavior.  
- Simpler and easier to write and test.  
  
Example:  
function Hello() {  
 return <h1>Hello World</h1>;  
}

# 4. Explain Class Component

A class component in React is a JavaScript class that extends from React.Component. It must include a render() method that returns JSX. Class components are capable of managing their own state and can use lifecycle methods such as componentDidMount, componentDidUpdate, and componentWillUnmount. These features make class components ideal for handling complex logic and interactions. Though still supported, class components are now less commonly used due to the simplicity and flexibility of function components with hooks.  
  
Example:  
class Welcome extends React.Component {  
 render() {  
 return <h1>Hello, {this.props.name}</h1>;  
 }  
}

# 5. Explain Function Component

A function component is a simpler form of a React component that is written as a JavaScript function. It accepts props as an argument and returns JSX to be rendered in the browser. With the introduction of React Hooks (such as useState, useEffect), function components can now handle state and side effects, making them just as powerful as class components but easier to write and maintain. Function components are now the preferred way to write components in modern React development.  
  
Example:  
function Welcome(props) {  
 return <h1>Hello, {props.name}</h1>;  
}

# 6. Define Component Constructor

In class components, the constructor is a special method used to initialize the component's internal state and bind methods. It is called automatically when a component is created and is the first method invoked in the component lifecycle. The constructor typically includes a call to super(props) to ensure the component has access to this.props. You can also initialize this.state and bind event handlers in the constructor.  
  
Example:  
class Counter extends React.Component {  
 constructor(props) {  
 super(props);  
 this.state = { count: 0 };  
 this.increment = this.increment.bind(this);  
 }  
  
 increment() {  
 this.setState({ count: this.state.count + 1 });  
 }  
  
 render() {  
 return <button onClick={this.increment}>{this.state.count}</button>;  
 }  
}

# 7. Define render() Function

The render() function is a required method in class components. It tells React what the component should display in the browser. This method returns JSX and is invoked every time the component is rendered or updated. The render() function must be pure, meaning it should not alter the component state or interact with external systems. Its job is only to return the UI layout based on the component’s current state and props.  
  
Example:  
class Greeting extends React.Component {  
 render() {  
 return <h1>Welcome, {this.props.name}</h1>;  
 }  
}

**src/Components/Home.js**

import React from 'react';

function Home() {

return (

<div>

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

</div>

);

}

export default Home;

**src/About.js**

import React from 'react';

function About() {

return (

<div>

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

</div>

);

}

export default About;

**src/Contact.js**

import React from 'react';

function Contact() {

return (

<div>

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

</div>

);

}

export default Contact;

**src/App.js**

import React from 'react';

import Home from './Components/Home';

import About from './About';

import Contact from './Contact';

function App() {

return (

<div>

<Home />

<About />

<Contact />

</div>

);

}

export default App;

**Output:**



