**Question 1: How do you handle forms in React? Explain the concept of controlled components.**

**✍️ Handling Forms in React:**

In React, form elements like <input>, <textarea>, and <select> are usually handled through controlled components. This means React controls the form input values via component state.

**Example:**

import React, { useState } from "react";

function MyForm() {

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

const handleSubmit = (e) => {

e.preventDefault(); // Prevent page reload

alert(`Submitted name: ${name}`);

};

return (

<form onSubmit={handleSubmit}>

<label>Name:

<input type="text" value={3}onChange={(e) => setName(e.target.value)} />

</label>

<button type="submit">Submit</button>

</form>

);

}

**🔑 What are controlled components?**

A **controlled component** is a form element where:

* The **value** is controlled by React state (useState()).
* Any change in input triggers an **onChange event**, which updates the state.
* The **UI always reflects the state**, making form data predictable and easier to validate.

**Why use controlled components?**

* Better control over input values
* Easy form validation and conditional rendering
* React has a **single source of truth** (state).

**Question 2: What is the difference between controlled and uncontrolled components in React?**

React supports two ways to handle form inputs: controlled and uncontrolled components. Here’s a simple comparison to make it super clear:

**🧠 Controlled Components**

* **React controls the input** via state (useState)
* Value is set with a value prop
* Updated with onChange handler

**Example:**

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

<input type="text" value={name}

onChange={(e) => setName(e.target.value)}

/>

**🔓 Uncontrolled Components**

* The **DOM** itself controls the input
* Use a **ref** to access the value
* No need for useState

**Example:**

const nameRef = useRef();

<input type="text" ref={nameRef} />