ReactJS Part-1 - Lab Assignment

1. Setting Up the React Environment

✅ **Concepts Covered**: Installing Node.js, Creating a React App, Running a React Project
🔷 **Task**:

- Install **Node.js** and check the version using `node -v` and `npm -v`.
- Create a new React project using:

```
npx create-react-app my-app
```

- Run the project using:

```
npm start
```

- Modify the `App.js` file to display a **custom message**.

---

2. Creating and Rendering a React Component

✅ **Concepts Covered**: Functional Components, JSX
🔷 **Task**:

- Create a simple **functional component** in `App.js`.
- Render an **H1 heading** with a welcome message.
- Example:

```
function Welcome() {
    return <h1>Welcome to React!</h1>;
}
export default Welcome;
```

---

3. Using JSX to Display Variables and Expressions

✅ **Concepts Covered**: JSX, JavaScript Expressions
🔷 **Task**:

- Define a variable in React and display it using JSX.
- Example:

```
const name = "Alice";
return <h1>Hello, {name}!</h1>;
```

- Modify it to display **dynamic greetings** based on the current time.

---

4. Creating Multiple Components and Rendering Them

✅ **Concepts Covered**: Component Composition
🔷 **Task**:

- Create **two functional components**: `Header` and `Footer`.
- Render both components inside `App.js`.
- Example:

```
function Header() {
    return <h1>Welcome to My Website</h1>;
}

function Footer() {
    return <p>© 2025 My Website</p>;
}
```

- Import and use these components in `App.js`.

---

5. Using JSX to Display Lists

✅ **Concepts Covered**: Lists in JSX, `map()` function
🔷 **Task**:

- Define an **array of items** and display them as an unordered list.
- Example:

```
const fruits = ["Apple", "Banana", "Cherry"];

return (
    <ul>
        {fruits.map((fruit, index) => (
            <li key={index}>{fruit}</li>
        ))}
    </ul>
);
```

---

6. Conditional Rendering in JSX

✅ **Concepts Covered**: `if` Conditions, Ternary Operators in JSX
🔷 **Task**:

- Show a **greeting message** based on whether the user is logged in or not.
- Example:

```
const isLoggedIn = false;

return (
    <h1>{isLoggedIn ? "Welcome Back!" : "Please Log In"}</h1>
);
```

---

7. Using Inline and Internal CSS in JSX

✅ **Concepts Covered**: Styling in JSX
🔷 **Task**:

- Apply inline styles and internal CSS inside React components.
- Example:

```
const myStyle = { color: "blue", fontSize: "20px" };

return <h1 style={myStyle}>Styled Heading</h1>;
```

---

8. Creating a Button with an OnClick Event

✅ **Concepts Covered**: JSX, Event Handling
🔷 **Task**:

- Create a React component that displays a button.
- When clicked, it should display an alert message.
- Example:

```
function ClickMe() {
    function handleClick() {
        alert("Button clicked!");
    }

    return <button onClick={handleClick}>Click Me</button>;
}
```

---

9. Displaying the Current Date and Time

✅ **Concepts Covered**: JSX, JavaScript Date Object
🔷 **Task**:

- Create a React component that displays the **current date and time**.
- The date should update **automatically every second**.
- Example:

```
function CurrentTime() {
    return <h2>Current Time: {new Date().toLocaleTimeString()}</h2>;
}
```

- Use `setInterval` to update the time dynamically.

---

10. Rendering an Image in JSX

✅ **Concepts Covered**: JSX, Image Rendering
🔷 **Task**:

- Create a React component that displays an image.
- Example:

```
function ProfilePicture() {
    return <img src="https://via.placeholder.com/150" alt="Profile"
/>;
}
```

---

11. Rendering a List of Users from an Array

✅ **Concepts Covered**: JSX, `map()` function
🔷 **Task**:

- Create an **array of user names** and render them as a list.
- Example:

```
const users = ["Alice", "Bob", "Charlie"];

function UserList() {
    return (
        <ul>
            {users.map((user, index) => (
                <li key={index}>{user}</li>
            ))}
        </ul>
    );
}
```

---

12. Creating a Greeting Component with Props

✅ **Concepts Covered**: Props, JSX
🔷 **Task**:

- Create a `Greeting` component that takes a **name** as a prop and displays a personalized message.
- Example:

```
function Greeting(props) {
    return <h1>Hello, {props.name}!</h1>;
}

function App() {
    return <Greeting name="Alice" />;
}
```

---

13. Using React Fragments to Return Multiple Elements

✅ **Concepts Covered**: React Fragments
🔷 **Task**:

- Use `<React.Fragment>` to return **multiple elements** without using a `<div>`.
- Example:

```
function Info() {
    return (
        <>
            <h1>Title</h1>
            <p>This is a description.</p>
        </>
    );
}
```

---

14. Creating a Simple Counter App

✅ **Concepts Covered**: useState Hook, JSX
🔷 **Task**:

- Create a counter with **Increment** and **Decrement** buttons.
- Example:

```
import { useState } from "react";

function Counter() {
    const [count, setCount] = useState(0);

    return (
        <div>
            <h1>Count: {count}</h1>
            <button onClick={() => setCount(count +
1)}>Increment</button>
            <button onClick={() => setCount(count -
1)}>Decrement</button>
        </div>
    );
}
```

---

15. Displaying Different Messages Based on User Input

✅ **Concepts Covered**: JSX, Conditional Rendering
🔷 **Task**:

- Create an input box where users type their **age**.
- If the age is **18 or above**, show **"You are an adult"**, otherwise show **"You are a minor"**.
- Example:

```
import { useState } from "react";

function AgeChecker() {
    const [age, setAge] = useState("");

    return (
        <div>
            <input type="number" onChange={(e) =>
setAge(e.target.value)} />
            <p>{age >= 18 ? "You are an adult" : "You are a
minor"}</p>
        </div>
    );
}
```