Programme	:	**B.Tech.**	Semester	:	**Winter24-25**
Course	:	**BCSE203E: Web Programming Lab**	Slot	:	**TE1/TE2**
Faculty	:	**Dr. LM Jenila Livingston**	Marks	:	**10**

Date: 26/03/2025

Exercise 15: JSX- Part III

1. You are developing a React application that consists of multiple functional components (Header, Content, and Footer). The main App component organizes these components and displays them on the screen.

   (i) Your task is to define and export an **App** component that contains multiple components:

   a. A Header component that receives a title as a prop.
   b. A Content component that displays a random joke when a button is clicked.
   c. A Footer component that displays a static footer message.

   (ii) Import and render the App component in **index.js** using ReactDOM.render(). Ensure the **index.html file has a root element** where React will mount the application.

2. **Styling in React – Inline CSS:**

- Create a StyledButton component that applies inline CSS for background color, padding, and font size.

3. **Styling in React – Internal CSS:**

- Modify the StyledButton component to include an internal <style> tag within the component for styling.

4. **Styling in React – External CSS:**

- Create a separate styles.css file and apply external styling to the StyledButton component by importing the CSS file.

5. Develop a LifecycleDemo **class** component that logs messages at each stage of its lifecycle
    - Lifecycle (constructor, componentDidMount, componentDidUpdate, and componentWillUnmount).
    - Implement a button to update the state and trigger componentDidUpdate().
    - Unmount the component dynamically to observe the effect of componentWillUnmount()

6. **React Props:**

- Design a Parent component that sends a message prop to a Child component.
- Ensure the Child component properly receives and displays the message.

7. **React Props Validation:**

- Modify the Child component to validate the message prop using prop-types.
- Ensure that the prop is required and of type string.

8. **State Hooks: (useState and useReducer)**

- Create a React component called Counter using the **useState()** hook. The component should display a count with two buttons: **Increase** and **Decrease**.
- Modify the component to use the **useReducer()** hook instead of useState(), handling increment and decrement actions efficiently.

9. **Effect Hooks (useEffect)**:

- Develop a React component that fetches and displays a random joke from an API when the component mounts.
- Add functionality to refresh the joke when a button is clicked.

10. **Ref Hooks (useRef)**:

- Build a simple form with an input field and a button.
- When the button is clicked, the input field should automatically get focused using the useRef() hook.

11. **Context Hooks (useContext)**:

- Create a React application where the theme (dark or light mode) is shared across multiple components using useContext().
- Implement a button to toggle between dark and light themes.

12. Passing Values from a Form Using useState and useRef

(i) Create a form with fields for **Name** and **Email**. Use useState to manage input values and display them dynamically.

- Create a new React component.
- Use useState to track form values.
- Display the values dynamically as the user types.
- Submit the form and prevent default page reload.

(ii) Create the same form but use useRef to retrieve values on form submission without managing state updates.

- Create a new React component.
- Use useRef to get form values.
- Display values only when the form is submitted.