Date

ASSIGNMENT NO :- 3

Aim :- Design an interactive front-end application using React by implementing templating using components States and Props, Class, Events. It must be responsive to scale across different platforms.

Objectives :- To develop a responsive, interactive front-end application using React.js that effectively demonstrates the fundamental concepts of component-based architecture, State management and event handling. The application will serve as a practical exercise in building a scalable uses interface by implementing templates with components managing dynamic data with states and props and handling uses interactions with events, ensuring a seamless uses experience across various devices and screen sizes.

Theory :-

Q1] Explain the role of State and Props in React. How do they differs and what is the Primary purpose of each in managing data flow within a component-based application.

→ • State :-
A component's own data [mutable].

Page No.
Date

2] Managed inside the components using hooks [useState] or inside a class [this.state].
3] When state changes, the component re-renders automatically.
4] Example :- A counter value that changes when the user clicks a button.

- Props

1] External data passed into a component from its parent
2] Read-only
3] Used to configure or customize child components.
4] Example :- Passing a button label (<Button text="Click Me" />)

Key difference:

- State → internal & mutable
- Props → external & immutable

Q2) What is a React component? Differentiate between a class component and a functional component, and discuss the advantages of using a functional component with hooks like useState and useEffect over a class component.

→ A React component is a reusable, independent piece of UI

- Class Component
  Use class syntax
  State managed with this.state.

Lifecycle methods.

```
Class Welcome extends React.Component {
 Constructor () {
 Super ();
 this.State = {count: 0};
 }
 rendes () {
   return <h1> Hello {this.props.name}, count:
{this.State.count} </h1>;
 }
}
```

- Functional components

Uses plain Javascript functions.
State and lifecycle via hooks

```
function Welcome ({name}) {
 const [count, setcount] =
React.useState(0);
 return <h1> Hello {name}, count:
 {count} </h1>;
}
```

Q3] Describe the concept of templating using components" in React. Why is this approach considered superior to traditional web development methods that rely on monolithic HTML files?

→ Instead of coriting a single monolithic HTML file, React encourages building small, reusable components. Example :- Instead of coriting <header>, <nav>, <footer> manually in each file, create reusable <Header/> , <Nav/> <Footer/> components.

Advantages:

- Reusability → less code duplication.
- Maintainability → Easier to update one component than multiple HTML pages.
- Composability → Components can be nested like Lego blocks

Q4] How do you handle uses events in React (eg. a button click)? Provide a simple code snippet to demonstrate how an event handler is defined in a component and how it can be used to update the component's state.

→ React handles event like onClick, onChange etc.
Example Button click to update state.

```
import React, { useState } from "react";
function Counter() {
  const [count, setCount] = useState(0);
const handleClick = () => {
  setCount(count + 1);
};
```

```
return (
 <div>
  <p> Count : {count} </p>
 <button onClick = {handleClick}
 Increase </button>
  </div>
 );
}
export default Counter;
```

When the → button is clicked the event handler updates state, causing React to re-render.

Q5) What is responsive web design and why is it crucial for modern applications? Describe how you would implement a responsive design in a React application using CSS media queries or a CSS in JS library.

→ Responsive design ensures the UI adapts to different screen sizes (mobile, tablet, desktop).
Crucial because users access apps from various devices.

1) CSS Media queries.

```
.Containes {
 width : 100%.
  Padding : 20px;
}
```

```
@media (max-width : 600px) {
  .Container {
  background-color : lightblue;
   font-size : 14px;
  }
}
```

2) CSS in js libraries.

```
import styled from
"Styled-components";
Const Box = Styled.div
  width : 100%;
  padding : 20px;
@media (max-width : 600px) {
  background-color : lightblue;
  font-size : 14px;
  };
```

UI Libraries

Provide responsive grid systems and backpoints out of the box