T.Y.B.Tech CSE 2023-24

Name : Parimal Shrikant Kolhe
Roll No : PB_01
Sub : FSD

Sem : 5
Panel : B
Batch : B1

Lab Assignment 5

Aim: Design and develop an interactive user interface using React.

Objectives:
1. Articulate what React is and why it is useful.
2. Explore the basic architecture of a React application.
3. Use React components to build interactive interfaces

Theory :

1. What is React? Steps to run a React app using create-react-app.

React is an open-source JavaScript library for building user interfaces. It's maintained by Facebook and a community of individual developers and companies. React allows you to create interactive and dynamic user interfaces for web applications. It uses a component-based architecture, where you break down your UI into reusable components, making it easier to manage and maintain your code.To create a React app using create-react-app, follow these steps:

1. Install Node.js if not already done.
2. Create a new React app by running: npx create-react-app my-react-app.
3. Navigate to the app directory: cd my-react-app.
4. Start the development server: npm start.

2. Passing data through props (Small example):

In React, you can pass data from a parent component to a child component using props. Props are like parameters for your React components. Here's a small example to illustrate how to pass data through props:

```
// ParentComponent.js
import React From 'react';
Function ParentComponent() {
const dataToPass = "Hello From Parent!";
return (
<ChildComponent message={dataToPass} />
);
}
export deFault ParentComponent;
// ChildComponent.js
import React From 'react';
Function ChildComponent(props) {
return (
<div>
<p>Received data From parent: {props.message}</p>
</div>
);
}
export deFault ChildComponent;
```

FAQS :

1. What are React states and hooks?

React states and hooks are fundamental concepts in React for managing the dynamic behavior of components and sharing logic between different components. They were introduced in React 16.8 to help manage component-level state and side effects in functional components.

• **React States:**

A state in React is an object that represents how a component's data should behave over time. It allows components to have dynamic behavior and re-render when the state changes. You can declare and initialize state using the useState hook in functional components. Here's an example:

```
import React, { useState } From 'react';
Function Counter() {
const [count, setCount] = useState(0);
return (
<div>
<p>Count: {count}</p>
<button onClick={() => setCount(count + 1)}>Increment</button>
</div>
);
}
```

- **React Hooks:**

React hooks are functions that allow you to "hook into" React features like state, lifecycle, and context in functional components. Apart from useState, React provides several other built-in hooks like useEffect for managing side effects, useContext for accessing context, and more.
Here's an example using the useEffect hook to fetch data :

```
import React, { useState, useEFFect } From 'react';
Function DataFetching() {
const [data, setData] = useState(null);
useEFFect(() => {
Fetch('https://api.example.com/data')
.then(response => response.json())
.then(data => setData(data))
.catch(error => console.error(error));
}, []);
return (
<div>
{data ? <p>Data: {data}</p> : <p>Loading data...</p>}
</div>
);
}
```

Output Images :