**Module 10 – Single Page Application framework – React**

**14.ReactJS-HOL**

**1. Explain the need and benefits of React Context API**

In React, when we want to share data between components, we usually pass it through props. But in a large application, this becomes very difficult — especially when the data has to be passed through many layers of components. This is known as **prop drilling**, and it makes the code messy and hard to manage.

To solve this, React provides the Context API, which lets us directly share values like themes, language, or user data with deeply nested components without passing them through every level manually.

**Benefits of using Context API:**

* It avoids prop drilling and keeps the code clean.
* We can easily manage global data (like themes, login status, etc.).
* It improves maintainability and scalability of the app.
* Components get updated automatically when the context value changes.

**2. Working with createContext()**

To use the Context API, the first step is to create a context using the createContext() function from React.

Here’s how it works in simple steps:

**Create the context:**

import { createContext } from 'react';

const ThemeContext = createContext('light'); // 'light' is the default value

This creates a context named ThemeContext with the default value 'light'.

**Provide the context value:**  
In a parent component (usually App.js), we wrap the part of our app with the ThemeContext.Provider and give it a value.

<ThemeContext.Provider value={'dark'}>

<MyComponent />

</ThemeContext.Provider>

**Consume the context:**  
In any child component, we can use the useContext() hook to get the value.

import { useContext } from 'react';

const theme = useContext(ThemeContext);

Now, this component will have access to the theme value directly without needing to receive it via props.

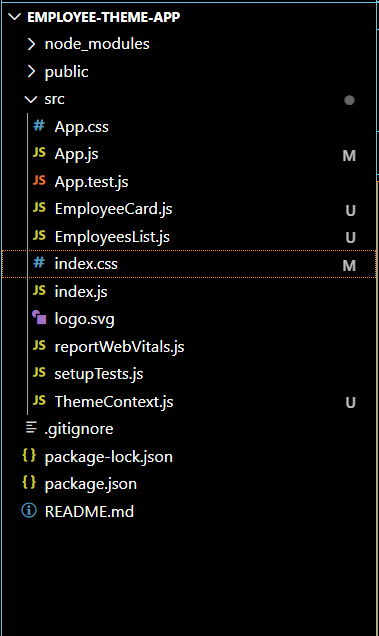
**3. List the types of Router components**

In React, when we want to create a multi-page experience without reloading the page, we use **React Router**. It provides different components to manage routing easily.

Here are the main router components:

* **BrowserRouter** – This is the base router that uses the browser’s history to keep track of the current page. It wraps the entire app.
* **Routes** – It is a wrapper for all the different routes defined in the app.
* **Route** – This defines a single route. It maps a URL path to a specific component.
* **Link** – It is used to create navigation links (like anchor tags), but without refreshing the page.
* **Navigate** – It is used for programmatic navigation or redirection.
* **Outlet** – It is used in nested routing to tell React Router where to render the child routes.
* **useNavigate()** – A hook to change routes from inside a function (e.g., after a form submission).
* **useParams()** – A hook to get URL parameters (like /user/:id).

**Structure:**



**EmployeeCard.js:**

import React, { useContext } from 'react';

import ThemeContext from './ThemeContext';

const EmployeeCard = ({ name }) => {

  const theme = useContext(ThemeContext);

  return (

    <div className={`card ${theme}`} style={{

      padding: '10px',

      margin: '10px',

      border: '1px solid gray',

      borderRadius: '5px'

    }}>

      <h3>{name}</h3>

      <button className={`btn ${theme}`}>

        View Profile

      </button>

    </div>

  );

};

export default EmployeeCard;

**EmployeesList.js:**

import React from 'react';

import EmployeeCard from './EmployeeCard';

const EmployeesList = () => {

  const employees = [

    { id: 1, name: 'Vedaprakash' },

    { id: 2, name: 'Karthik' },

    { id: 3, name: 'Sneha' }

  ];

  return (

    <div>

      {employees.map(emp => (

        <EmployeeCard key={emp.id} name={emp.name} />

      ))}

    </div>

  );

};

**ThemeContext.js:**

export default EmployeesList;

import { createContext } from 'react';

const ThemeContext = createContext('light'); // default theme

export default ThemeContext;

**App.js:**

import React, { useState } from 'react';

import ThemeContext from './ThemeContext';

import EmployeesList from './EmployeesList';

function App() {

  const [theme, setTheme] = useState('light');

  const toggleTheme = () => {

    setTheme(prev => (prev === 'light' ? 'dark' : 'light'));

  };

  return (

    <ThemeContext.Provider value={theme}>

      <div className={`App ${theme}`}>

        <h1>Employee Management App</h1>

        <button onClick={toggleTheme}>

          Toggle Theme

        </button>

        <EmployeesList />

      </div>

    </ThemeContext.Provider>

  );

}

export default App;

**index.css:**

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

}

.light {

  background-color: white;

  color: black;

}

.dark {

  background-color: black;

  color: white;

}

button {

  margin-top: 10px;

  padding: 5px 10px;

}

**Output:**

