Higher order component(HOC)

* A function that takes a component and return a component
* Why do we use it?
  + It takes component as i/p and enhance the component and return it back
* HOC are pure functions i.e it will not change or modifiy the parameter passed to it i.e component passed to it .
* It just add some features on top of existing component and return it

export const withExcellentRating = (RestroCards) => {

  //return a component with exhanced version

  return (props) => {

    return (

      <div>

        <div className="absolute bg-black text-white rounded-sm m-2">

          Suggested

        </div>

        <RestroCards {...props} />

      </div>

    );

  };

};

Above is the HOC component that takes restroCards component as i/p and return component with enhanced version

  const RestroWithRating = withExcellentRating(RestroCards);

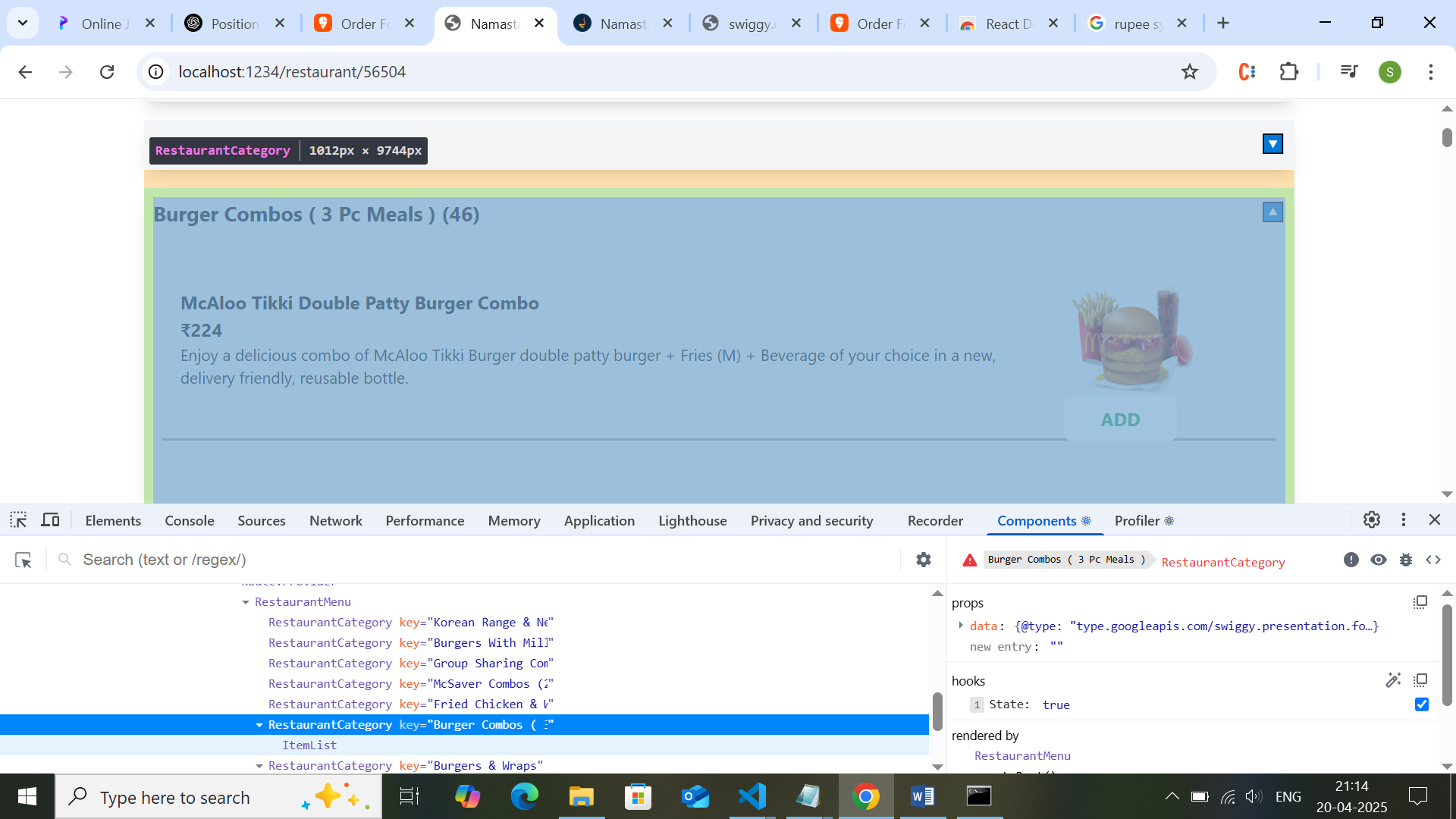
Above is way of calling the HOC inside component

Importance of react data handling:

* Important part of react application is to handle the data
* There are always 2 different layer in react application
  + Ui layer
  + Data layer
* Ui layer is powered by data layer
* The data layer contains state,props etc so if we manage our data then it will make app very fast and flexible

Manage data in app:

Each and every component have their own state managed with it eg: if we render a component inside map .. each time when items iterate through the components are re-rendered and they have separate state for each



It is though to communicate b/t 2 components as they have different state managed and when we need to close remaining accorian when we open one is difficult what to do and how to achive it?

This is achived by lifting the state up from restraurentcateogyr to its parent and we can manage the state from there so parent will handle the state

**Lifting state up**

Sometimes, you want the state of two components to always change together. To do it, remove state from both of them, move it to their closest common parent, and then pass it down to them via props. This is known as lifting state up, and it’s one of the most common things you will do writing React code.

Controlled component vs uncontrolled component

Controlled component : the parent has control to its child component

Uncontrolled component : the component is not controlled by parent and it is managed by itself

Great question! Understanding **controlled vs uncontrolled components** is super important in React.

**🎮 Controlled vs 🌀 Uncontrolled Components**

**🔹 Controlled Component:**

* **State is controlled by React** using useState.
* The value of the form element is tied to a **state variable**.
* You get **real-time control** over the input.

**🔹 Uncontrolled Component:**

* **DOM handles the state**.
* You access the value via **refs** (like in plain HTML).
* React doesn't re-render on every keystroke.

**✅ Controlled Component Example**

import { useState } from "react";

export default function ControlledInput() {

const [value, setValue] = useState("");

const handleChange = (e) => setValue(e.target.value);

return (

<div className="p-4">

<input

className="border p-2"

type="text"

value={value}

onChange={handleChange}

/>

<p>Typed: {value}</p>

</div>

);

}

🧠 Here, React manages the input's value with useState.

**✅ Uncontrolled Component Example**

import { useRef } from "react";

export default function UncontrolledInput() {

const inputRef = useRef();

const handleSubmit = () => {

alert(`Input value: ${inputRef.current.value}`);

};

return (

<div className="p-4">

<input className="border p-2" type="text" ref={inputRef} />

<button

className="ml-2 px-3 py-1 bg-blue-500 text-white rounded"

onClick={handleSubmit}

>

Show Value

</button>

</div>

);

}

🧠 Here, the value is accessed only when needed using a ref.

**🚀 Summary:**

| **Feature** | **Controlled** | **Uncontrolled** |
| --- | --- | --- |
| Data source | React state | DOM (ref) |
| Real-time value | Always available | Only when accessed |
| Ideal for | Forms, validation | Simple, one-off inputs |

Let me know if you want to build a small form using both types!

**How to change I modify the state of parent from children is it possible?**

No.. we cannot do it directly instead we can do it by passing a function to child from parent

Sometimes we need to life state up if we need to solve the issue of data handling

Props drilling:

1.data flow in one direction from parent to children … not from children to parent

2.if we need to pass data from top parent to last child we need to pass the data to all the intermediate parents and access in last child …but this is not good way as it is not useful for intermediate parents this is called props drilling

So to avoid props drilling

Situation : There are some scenario whether we need to pass some global data to components and access it somewhere in application to do this there is concept called **React context**

1.we can avoid props drilling and access global data throw react context

**2.react context is like central store and we don’t need to do props drilling**

**Create context :**

Usecases

1.user logged in info 🡪 user info need anywhere In app

2.dark mode which need to all component

How to implement context 🡪 it solve props drilling

**1.createContext 🡪utility function**

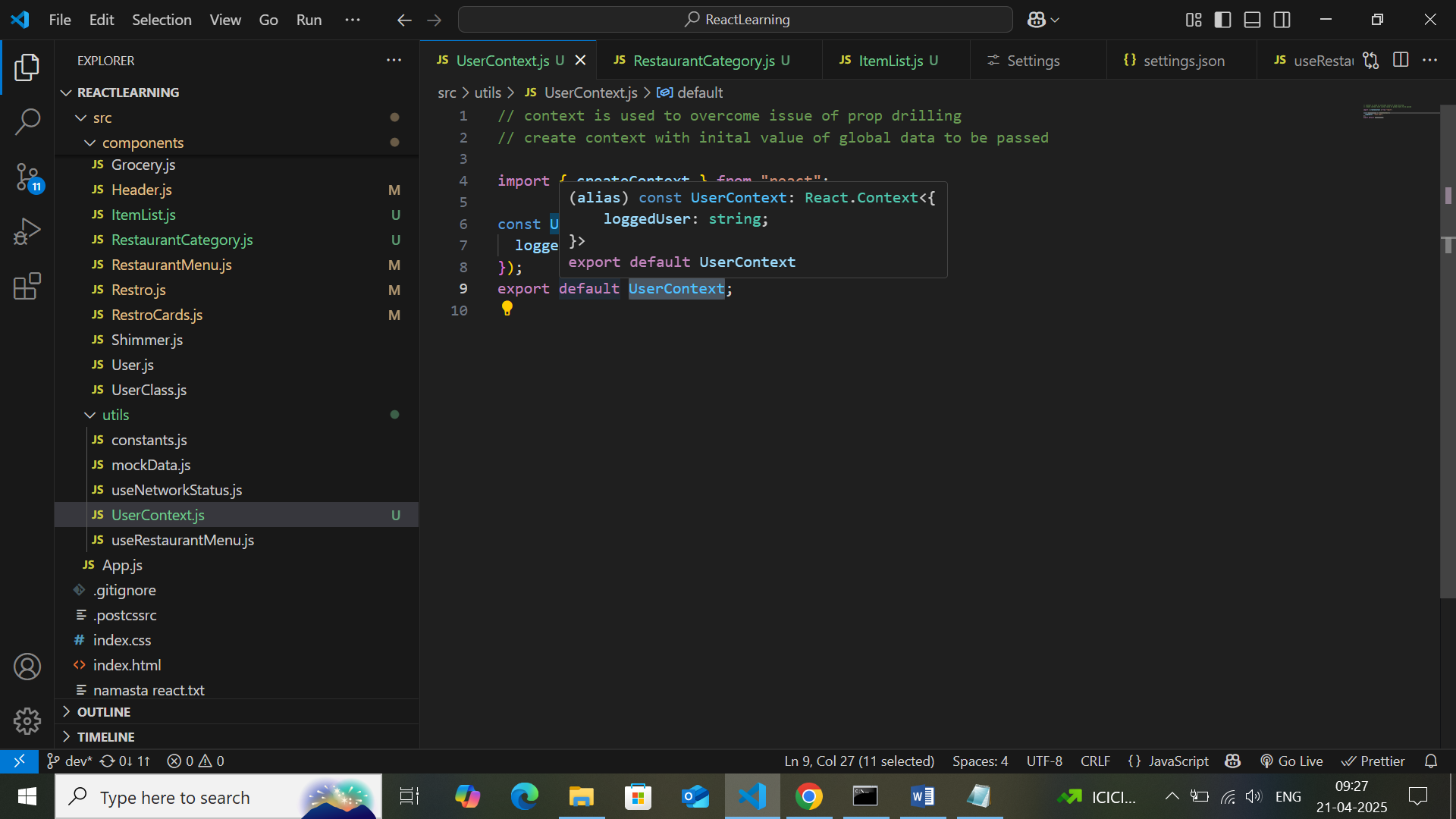
1.it is used to create a context with default value of global data

Const Usercontext = createContext({

User:’default user’ // default value

})

Export default Usercontext



2.**how to access the global data from createContext** .. through **useContext** hook

Const data = useContext(context created with createContext)🡪Usercontext

Why we pass the context … in the react we can create multiple context .. so to identify the context we pass it

  const { loggedUser } = useContext(UserContext);

3.we can access it any where in the component

Can we store all data in context? 🡪no

1.we should not put all data to context so we can avoid props drilling 🡪no.. we can use it for data which need to be used in multiple places

In class based component how to access context?

1.when we create context the react gives access to consumer

<UserContext.consumer>

{(data)=>{

<h1>{data.user}</h1>

}}

</UserContext.consumer>

**3.How to update the context data and how to pass the data to all the component?**

**Through provider we can change the global value change and access it anywhere**

**Context.Provider**

<Context.Provider value={{user:new username}}>

</context.provider>

const AppLayout = () => {

  return (

    // to update value of userContext theere is context provider throgh which we can send changed value

    // we can also update the context value in other components by sending it as props

    <div className="app">

      <UserContext.Provider value={{ loggedUser: "Sangeetha Priya" }}>

        <Header />

        {/\* outlet is used to load the child elements that are children of appLayout based on url called \*/}

        <UserContext.Provider value={{ loggedUser: "User Two" }}>

          <Outlet />

        </UserContext.Provider>

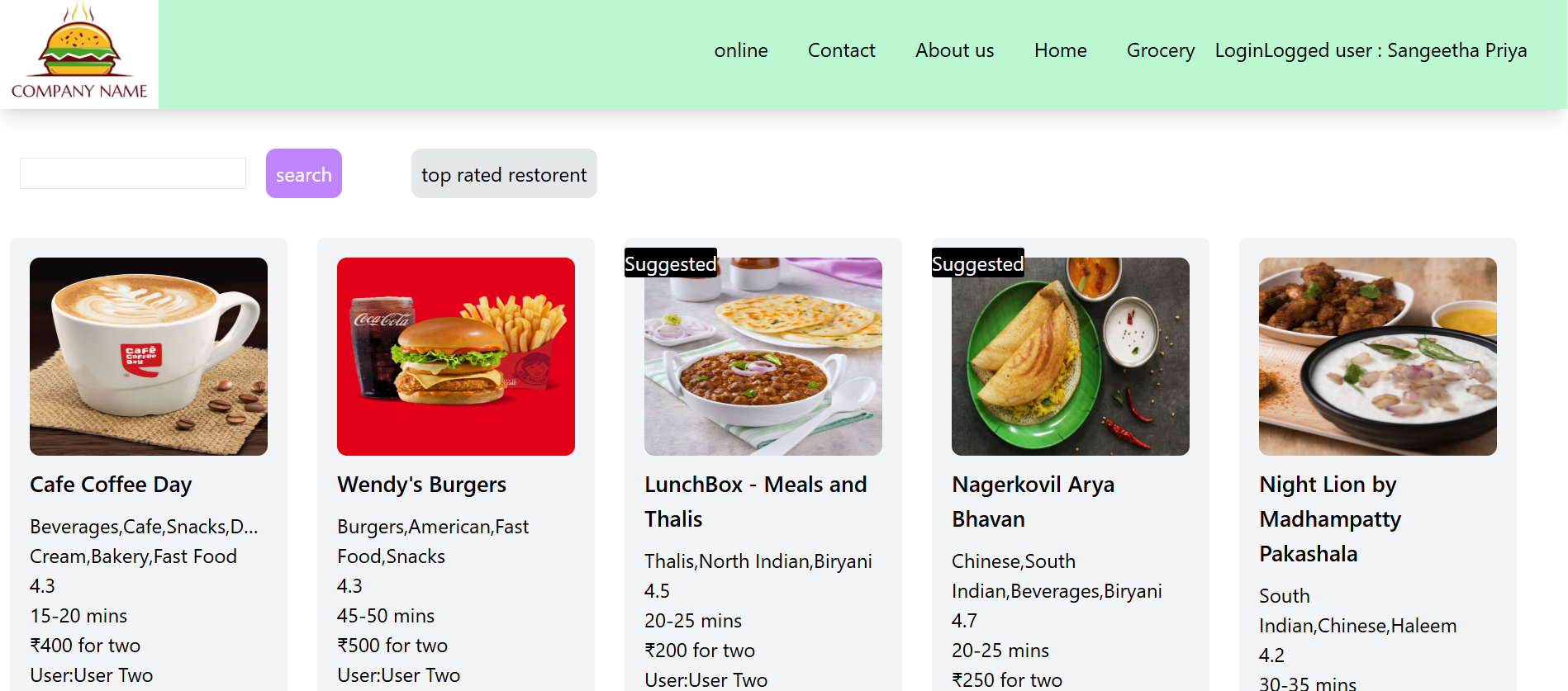
      </UserContext.Provider>

    </div>

  );

};

We can pass nested provider also to the context .. in above code the header component will have value of sangeetha Priya and other component in outlet will have user Two



If we wrap the context.provider to only few component .. then the updated value of context will be access to only those component and in other component where it is used will have default value

**Can I have nested context ..>same user context🡪it is possible🡪yes**

**We can also change the value of global value through passing state to other child component**it also changes the value to all the places even in lazy loading component too

**Even though the lazy loaded component is not accessed during first render still the data in context is accessed or updated in lazy component too**

We can use context for small and mid size components

Redux 🡪external state management library

We can also change the value of global data in another component by passing the set of state to other component

const AppLayout = () => {

  const [user, setUser] = useState();

  useEffect(() => {

    setUser("Sangeetha Priya");

  }, []);

  return (

    // to update value of userContext theere is context provider throgh which we can send changed value

    <div className="app">

      <UserContext.Provider value={{ loggedUser: user }}>

        <Header />

        {/\* outlet is used to load the child elements that are children of appLayout based on url called \*/}

        {/\* // we can also update the context value in other components by sending it as props \*/}

        <UserContext.Provider value={{ loggedUser: user, setUser }}>

          <Outlet />

        </UserContext.Provider>

      </UserContext.Provider>

    </div>

  );

};

In child

  const { loggedUser, setUser } = useContext(UserContext);

 <input

            name="search"

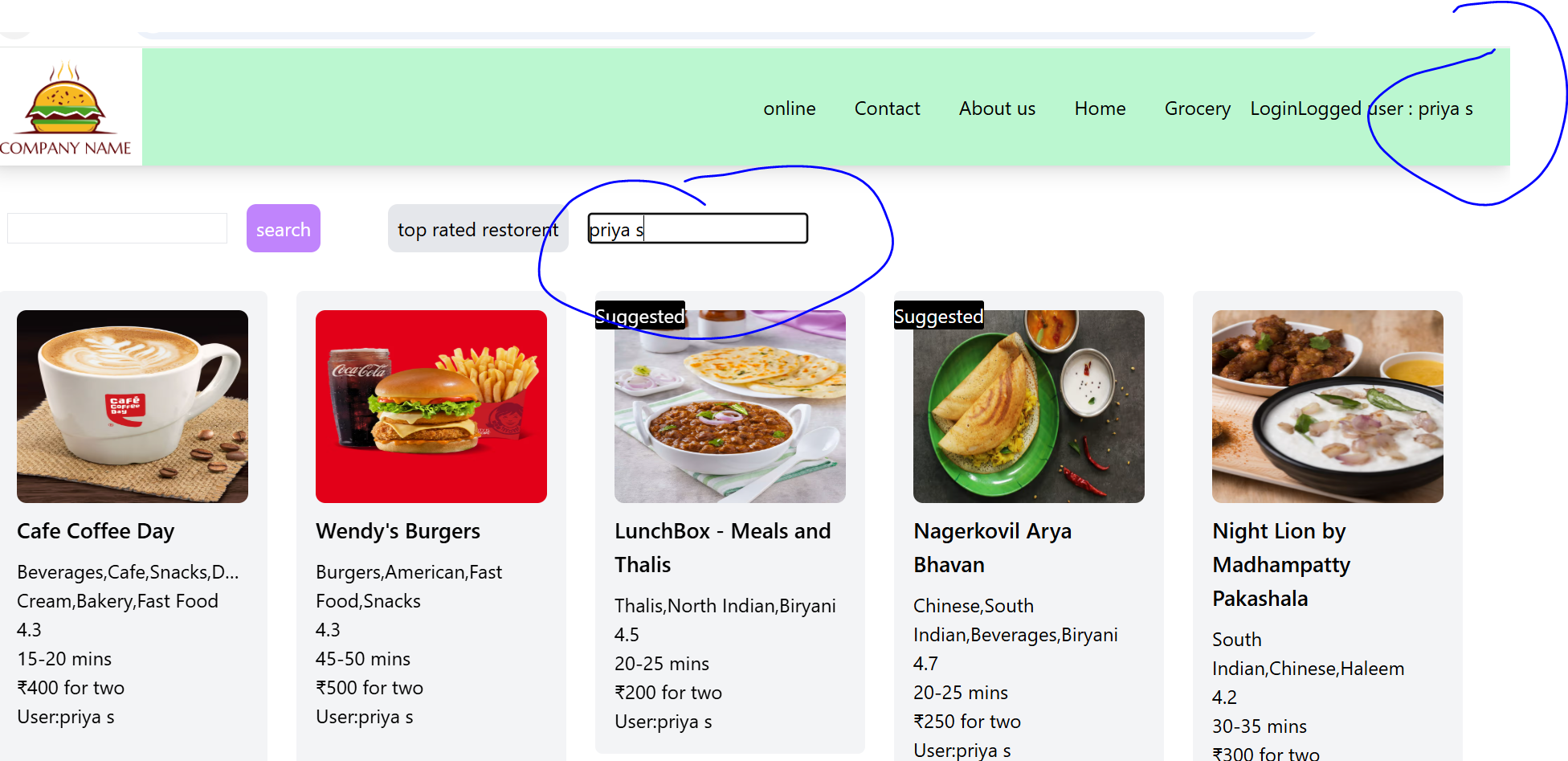
            className="border border-solid "

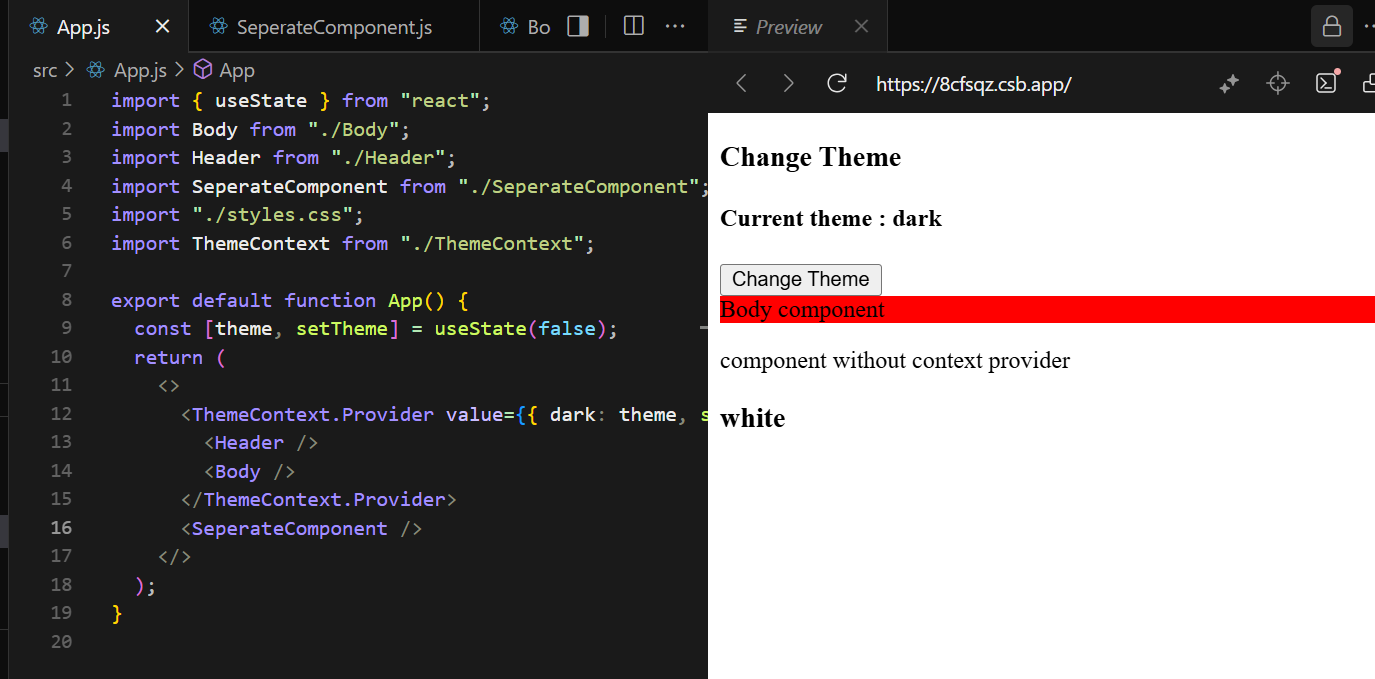
            type="text"

            // value={loggedUser}

            onChange={(e) => setUser(e.target.value)}

          ></input>





The value change of context will be updated to only header and body it wont reflect in seperateComponent