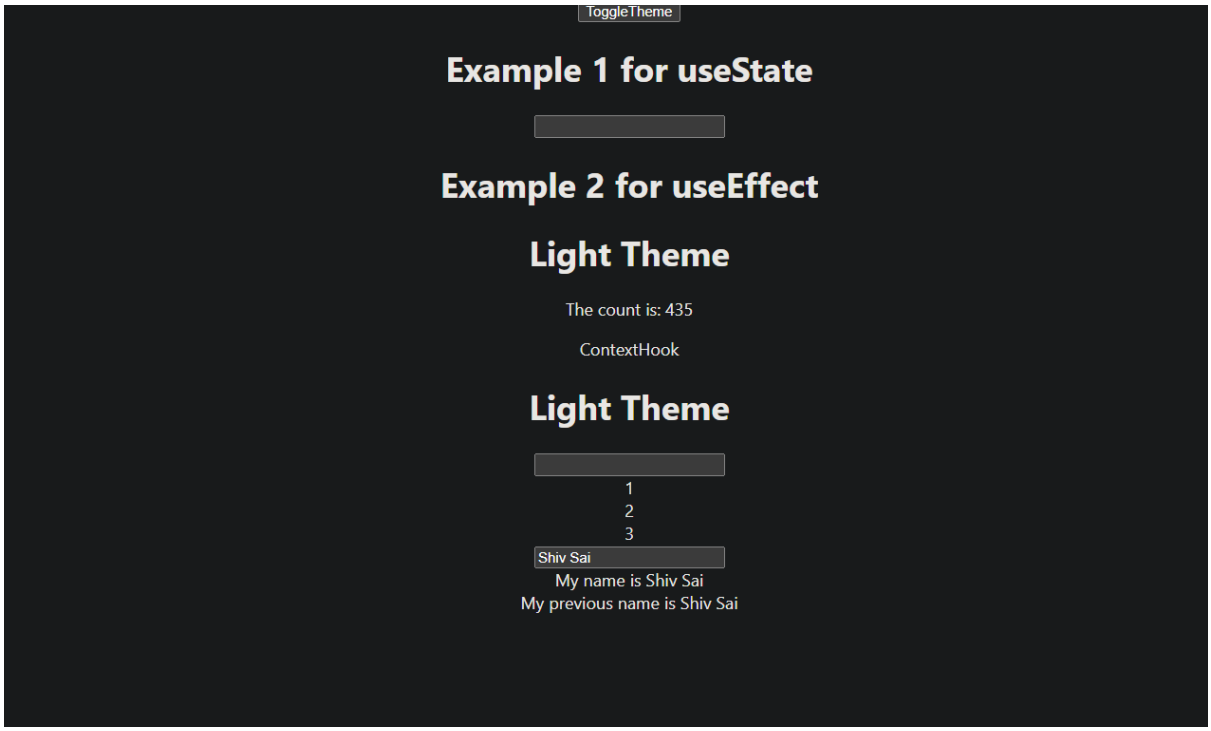
Name: Shiv Sai Indrakanti

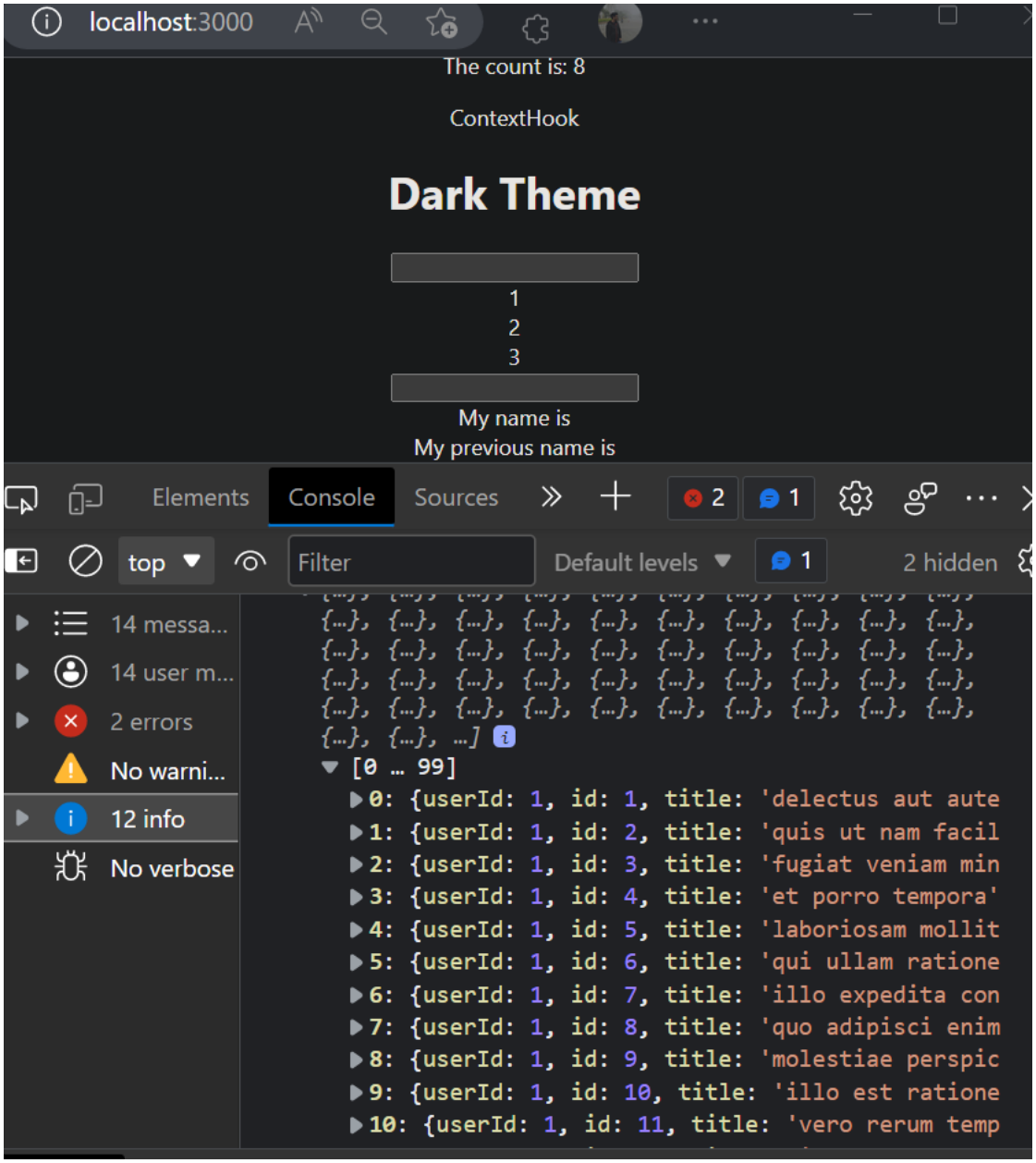
Reg No:19BCT0008

React Assignment 2 (Hooks)

Date of Submission – 28/12/2022

Screenshots





App.js

```
import "./App.css";
import React from "react";
import StateHook from "./Components/27122022/StateHook";
import EffectHook from "./Components/27122022/EffectHook";
import ContextHook from "./Components/27122022/ContextHook";
import CallbackHook from "./Components/27122022/CallbackHook";
import RefHook from "./Components/27122022/RefHook";
```

```
import useApiHook from "./Components/27122022/ApiHook";

import { useState } from "react";
export const ThemeContext = React.createContext();

const App = () => {
  const [number, setNumber] = useState(1);
  const [darkTheme, setDarkTheme] = useState(true);
  const getItems = () => {
    return [number, number + 1, number + 2];
  };
  const toggleTheme = () => {
    setDarkTheme((prevDarkTheme) => !prevDarkTheme);
  };

  const data = useApiHook("https://jsonplaceholder.typicode.com/todos");
  console.log(data);

  return (
    <div class="App">
      <ThemeContext.Provider value={darkTheme}>
        <button onClick={() => toggleTheme()}>ToggleTheme</button>
        <StateHook />
        <EffectHook />
        <ContextHook />
        <input
          type="number"
          onChange={(e) => setNumber(parseInt(e.target.value))}
        />
        <CallbackHook getItems={getItems} />
        <RefHook />
      </ThemeContext.Provider>
    </div>
  );
};

export default App;
```

StateHook.js

```
import { useState } from 'react';

const StateHook = ()=> {

  const [name,setName] = useState('');
  const handleChange= (e)=>{
    setName(e.target.value);
```

```
  }

  return(
    <>
    <h1>Example 1 for useState</h1>
    <input type="text" onChange={(e)=>handleChange(e)}/>
    {name}
    </>
  )
}
export default StateHook;
```

ContextHook.js

```
import React from 'react'
import {useContext} from 'react'
import { ThemeContext } from '../../App'

function ContextHook() {
    const darkTheme = useContext(ThemeContext)
    console.log(darkTheme);

  return (
    <>
    <div>ContextHook</div>
    {darkTheme?<h1>Dark Theme</h1>:<h1>Light Theme</h1>}

    </>
  )
}

export default ContextHook
```

RefHook.js

```
import React,{useState,useEffect,useRef} from 'react'


const RefHook=()=> {
```

```
    const [name,setName] = useState('');
    const prevName = useRef('');
    //useEffect gets called after the render which is why this is possible. -
note
    useEffect(() => {
        prevName.current = name;

    },[name])

  return (
    <>
    <input type="text" onChange={(e)=>setName(e.target.value)}/>
    <div>My name is {name}</div>
    <div>My previous name is {prevName.current}</div>

    </>
  )
}

export default RefHook
```

CallBackHook.js

```
import React from 'react'
import {useEffect,useState} from 'react'
const CallbackHook=({getItems})=> {
    const [items,setItems] = useState([]);
    useEffect(()=>setItems(getItems()),[getItems]);
  return  items.map(item=><div>{item}</div>)

}

export default CallbackHook
```

EffectHook.js

```
import { useState, useEffect } from "react";
import { useContext } from "react";
import { ThemeContext } from "../../App";

const EffectHook = () => {
```

```
  const darkTheme = useContext(ThemeContext);

  const [count, setCount] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setCount((count) => count + 1);
    }, 1000);


  return () => clearInterval(interval);
  }, []);

  return (
    <div>


      <h1>Example 2 for useEffect</h1>
      {darkTheme ? <h1>Dark Theme</h1> : <h1>Light Theme</h1>}
      <p>The count is: {count}</p>
    </div>
  );
};
export default EffectHook;
```

ApiHook.js

```
import { useState, useEffect } from "react";

const useApiHook = (x) => {
    const [data,setData] = useState();
    useEffect(() =>{
        fetch(x).then(data => data.json()).then(a=>setData(a))

    },[])
  return data;
};

export default useApiHook;
```

Explanation:

I made a custom hook which returns data in the form of an array which I displayed in the console.

Here I can also dynamically send the url from whichever component I use the hook in, this ensures code reusability and reduces code duplication.