LAB ASSIGNMENT NO – 6a

Aim: Write a program to implement the concept of props and state.

Create a functional component and pass current date as props and with class component, on button click display both date and time with change font and color.

Theory:

1. Props (Properties):

Props are short for "properties" and are a way to pass data from a parent component to a child component in React.

They are read-only and help make components reusable and composable.

Props are typically used for passing data and functions that child components need to render or interact with.

2. States:

State is used to manage and store data that can change over time within a component.

Unlike props, state is mutable and can be modified by the component itself.

Changes in state trigger component re-renders, updating the user interface.

State is primarily used for managing component-specific data and user interactions.

3. Functional Components:

Functional components are a way to define React components using JavaScript functions.

They are simpler and more concise than class components and do not have their own state or lifecycle methods.

Functional components are commonly used for "presentational" or "dumb" components that focus on rendering data.

Functional components don't have lifecycle methods like componentDidMount, componentDidUpdate, or componentWillUnmount as class components do. Instead, they can use the useEffect hook to handle side effects and lifecycle-like behavior.

Functional components are simpler and easier to read and write compared to class components.

They encourage a more declarative and functional programming style.

Example:

```
import React from 'react';
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
export default Greeting;
```

4. Class Components:

Class components are traditional React components defined as JavaScript classes.

They have their own state, lifecycle methods, and more advanced features.

Class components are often used for "container" or "smart" components that manage state and logic.

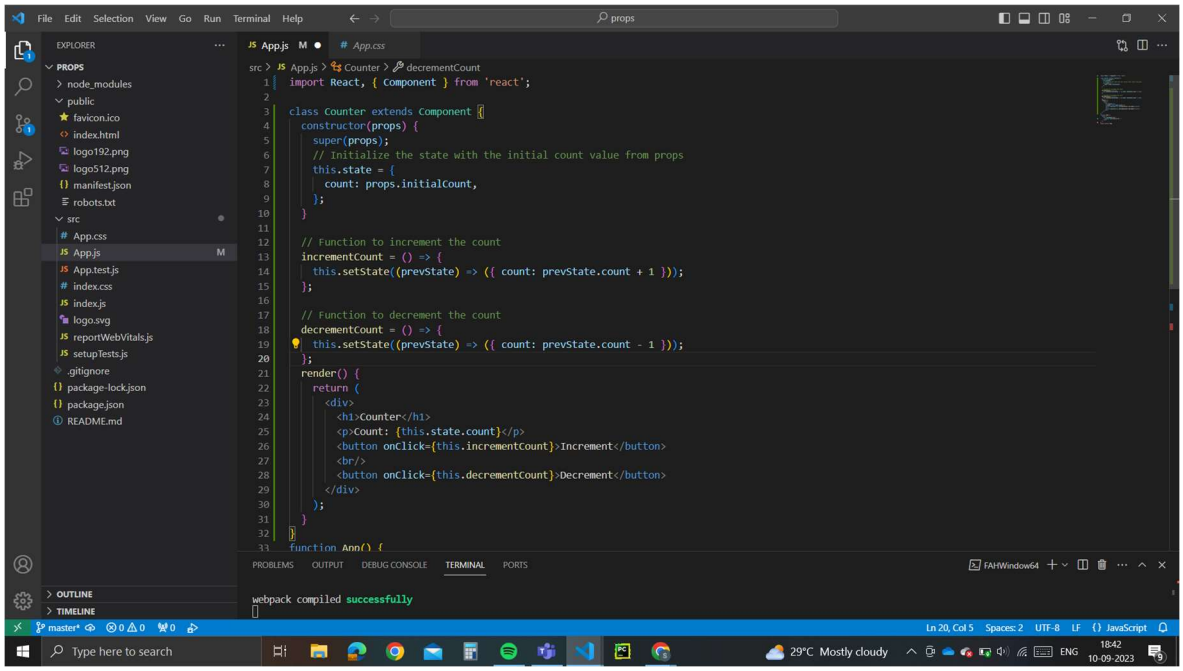
Class components can define lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount.

These methods allow you to perform actions when a component is mounted, updated, or unmounted.

Class components are typically used when you need to manage complex state, perform side effects, or access component lifecycle methods.

Conclusion: This experiment on React states and props, functional components, and class components underscored the essential building blocks of React development. Props facilitated data sharing, enabling component reusability, while states empowered dynamic updates and interactivity. Functional components demonstrated simplicity and readability, while class components showcased complex state management and lifecycle methods. This experiment highlighted the flexibility and versatility of React for crafting responsive and modular user interfaces.

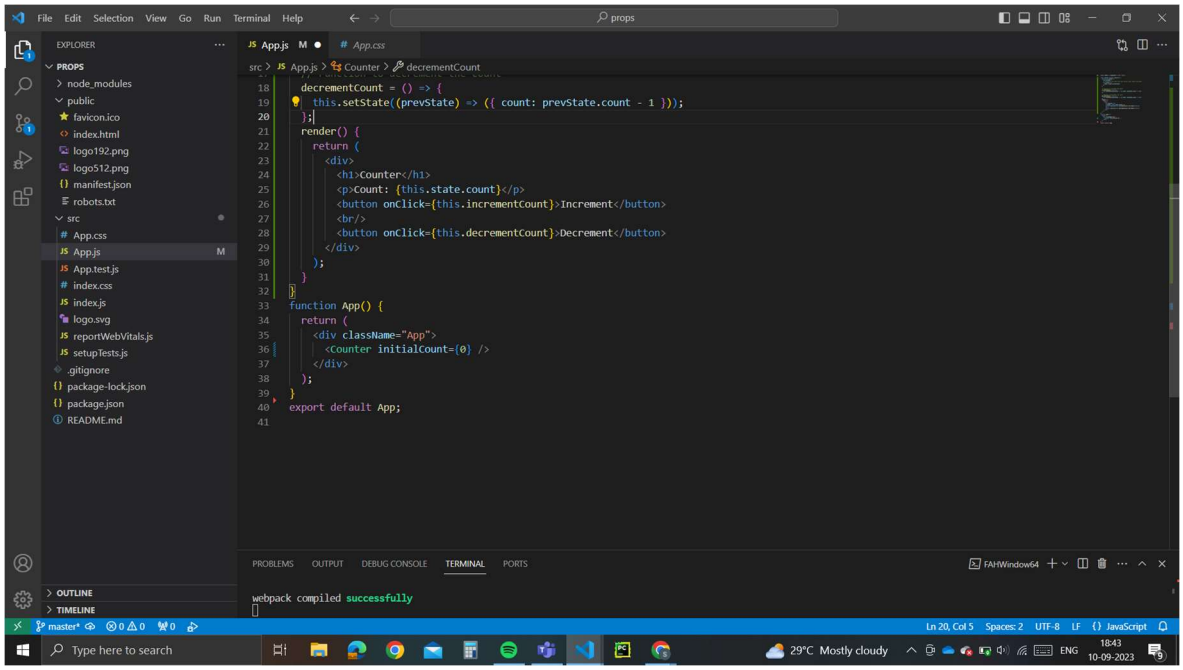
Code:


File Edit Selection View Go Run Terminal Help
EXPLORER
PROPS
node_modules
public
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
src
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
OUTLINE
TIMELINE
App.js
App.css
src > App.js > Counter > decrementCount
import React, { Component } from 'react';
class Counter extends Component {
constructor(props) {
super(props);
// Initialize the state with the initial count value from props
this.state = {
count: props.initialCount,
};
}
// Function to increment the count
incrementCount = () => {
this.setState((prevState) => ({ count: prevState.count + 1 }));
};
// Function to decrement the count
decrementCount = () => {
this.setState((prevState) => ({ count: prevState.count - 1 }));
};
render() {
return (

<h1>Counter</h1>
Count: {this.state.count}
<button onClick={this.incrementCount}>Increment</button>

<button onClick={this.decrementCount}>Decrement</button>

);
}
}
function App() {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
webpack compiled successfully
Ln 20, Col 5 Spaces: 2 UTF-8 LF {} JavaScript
Type here to search
29°C Mostly cloudy
ENG
18:42
10-09-2023


File Edit Selection View Go Run Terminal Help
EXPLORER
PROPS
node_modules
public
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
src
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
OUTLINE
TIMELINE
App.js
App.css
src > App.js > Counter > decrementCount
decrementCount = () => {
this.setState((prevState) => ({ count: prevState.count - 1 }));
};
render() {
return (

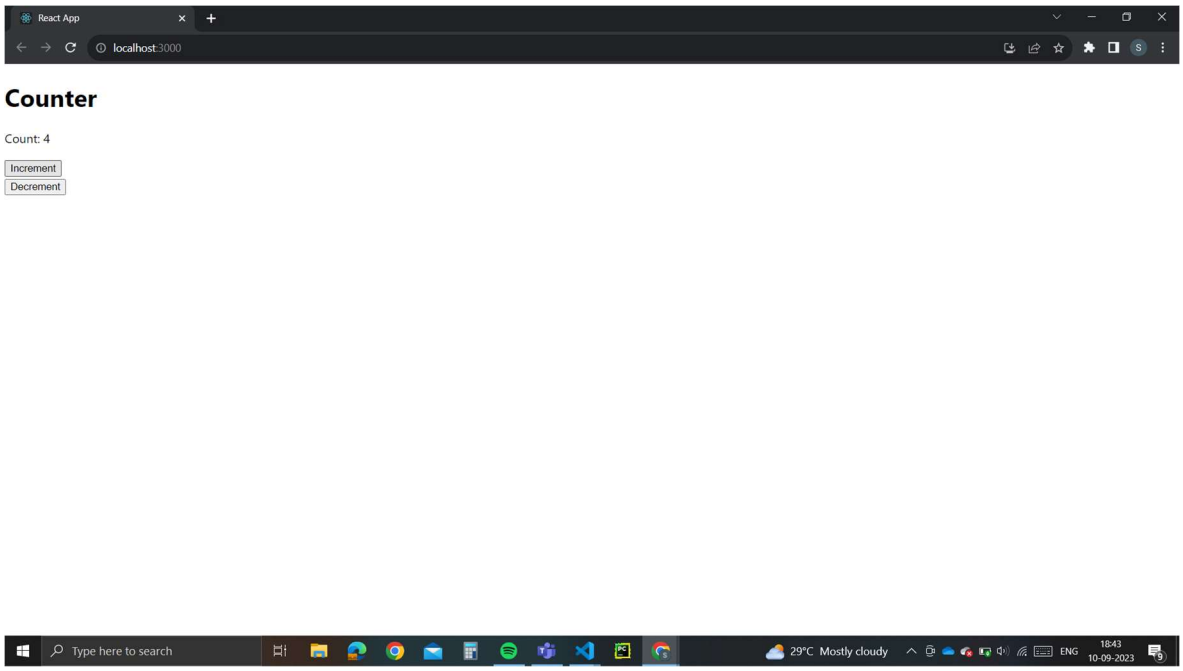
<h1>Counter</h1>
Count: {this.state.count}
<button onClick={this.incrementCount}>Increment</button>

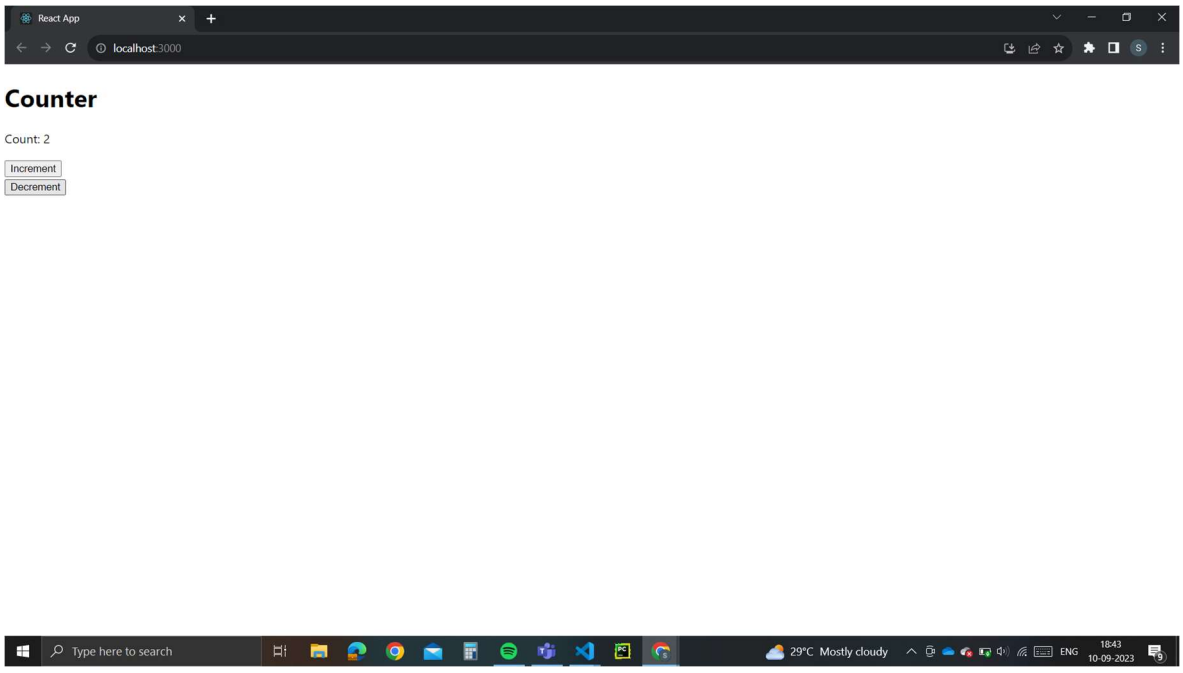
<button onClick={this.decrementCount}>Decrement</button>

);
}
}
function App() {
return (

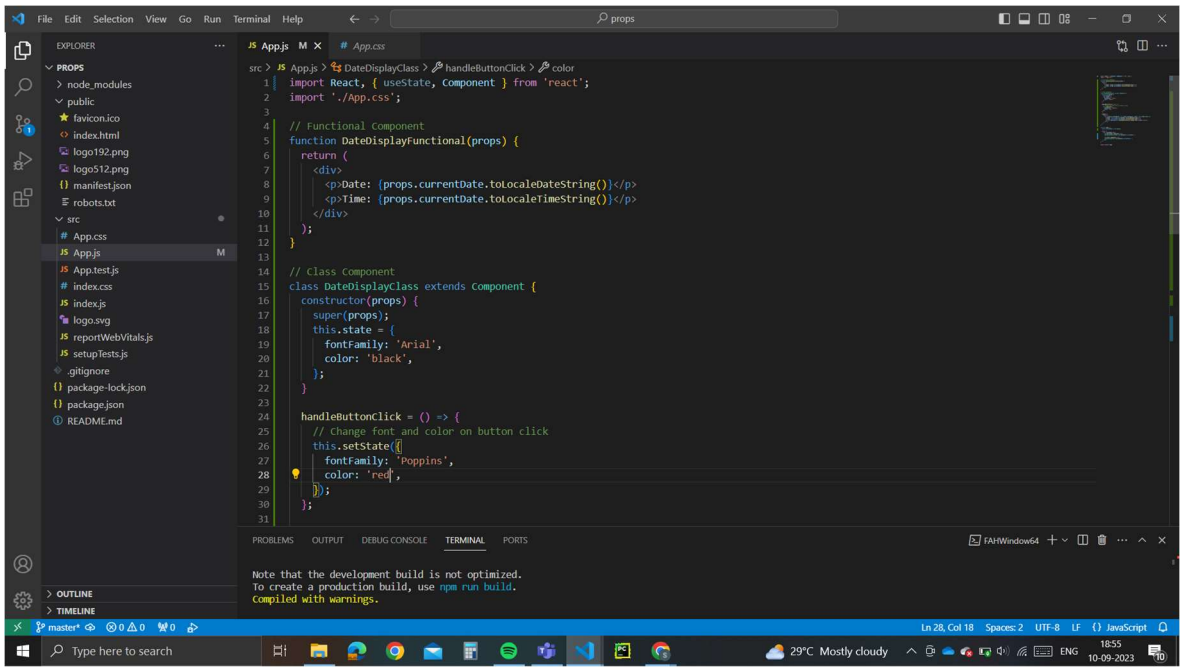
<Counter initialCount={0} />

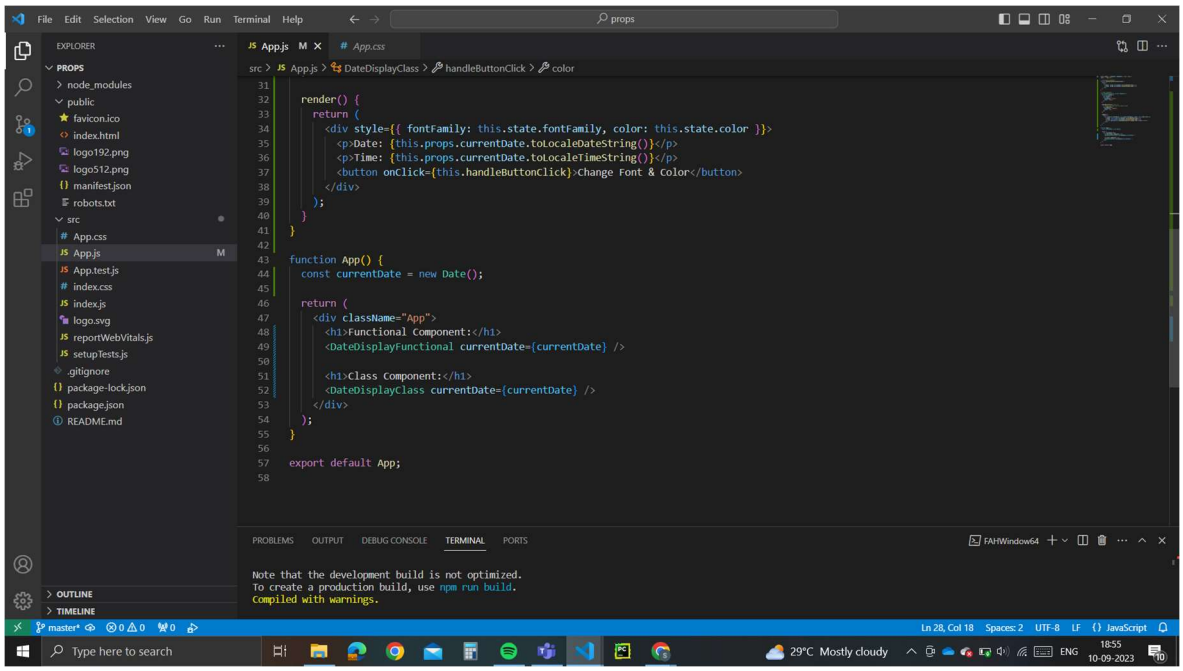
);
}
export default App;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
webpack compiled successfully
Ln 20, Col 5 Spaces: 2 UTF-8 LF {} JavaScript
Type here to search
29°C Mostly cloudy
ENG
18:43
10-09-2023

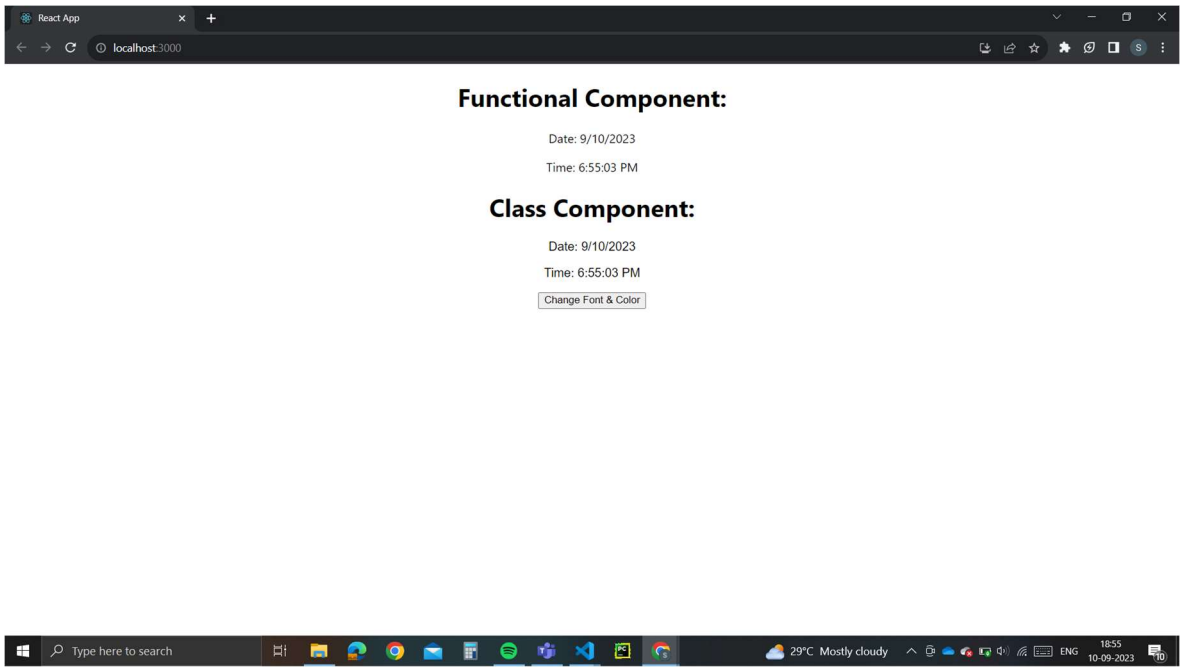

React App
localhost:3000
Counter
Count: 4
Increment
Decrement
Type here to search
29°C Mostly cloudy
ENG
18:43
10-09-2023

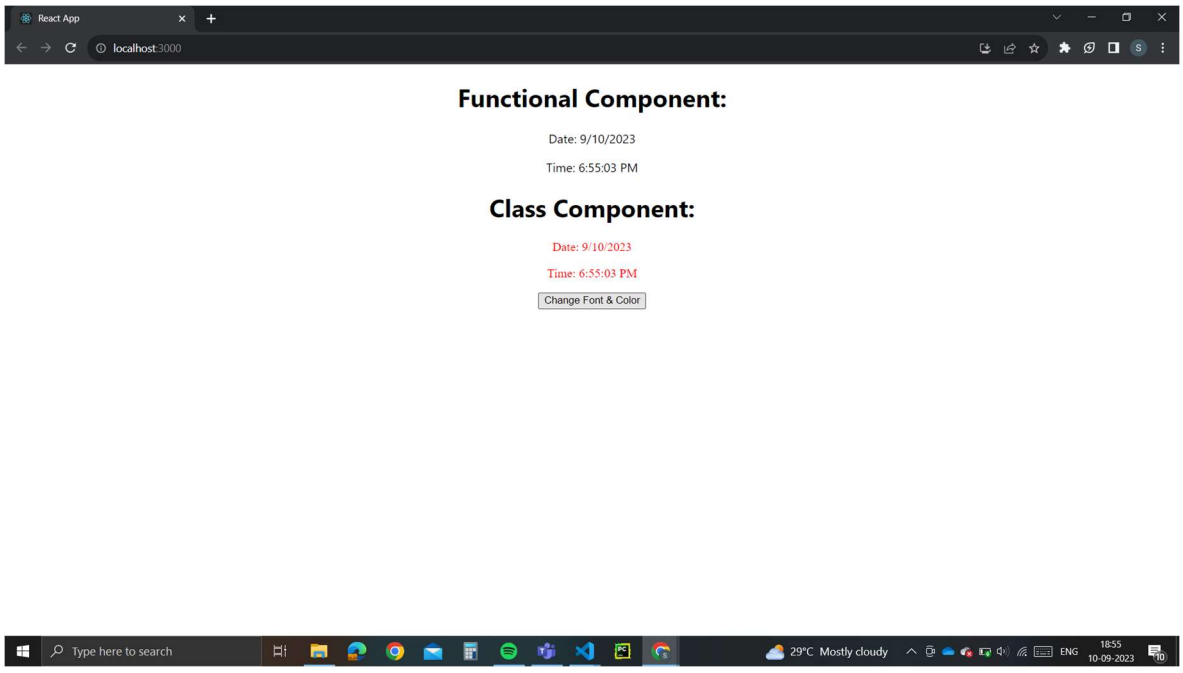

React App
localhost:3000
Counter
Count: 2
Increment
Decrement
Type here to search
29°C Mostly cloudy
ENG
18:43
10-09-2023

Shreya Bagade/ SEM V/ TE.IT/ T11/ Roll No-09/ Assignment no-6a









<u>Lab Outcome :</u> LO5- Construct front end applications using React.