**CN5006 React Lab Report:**

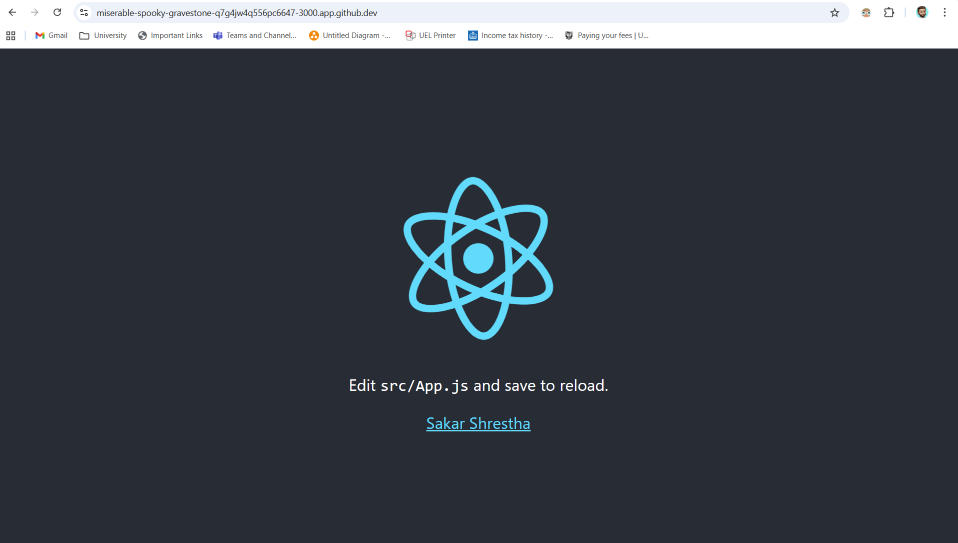
1. Introduction

The development and application of functional components—two fundamental ideas of React programming—were the main focus of this lab exercise. The main learning goals were to use create-react-app to set up a React application, change the text and background color of the application, execute stateless functional components, pass data through properties (props), and integrate event handling for interactive user interfaces.

1. Lab Setup and Initial Application Modification (Tasks 1 & 2)

Using the npx create-react-app command, a new React application was set up to start the lab. The first thing to do was change the default template once the application was operating on the local host:

* 1. Changing Display Text (Task 1)

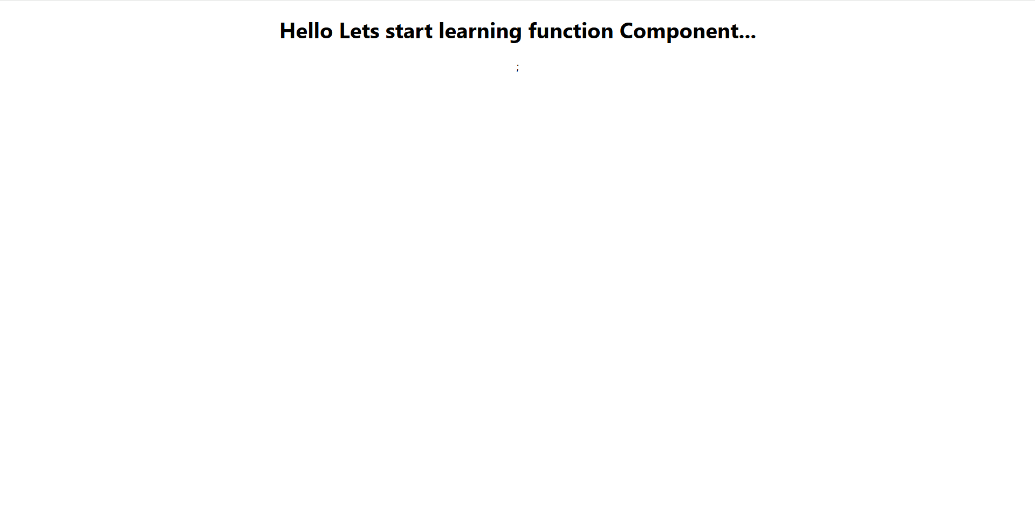


* 1. Changing Background Color (Task 2) 

1. Component Creation and Data Flow

In order to comprehend React's modular structure and data flow through props, the lab's main focus was developing reusable functional components.

* 1. Stateless Functional Component (Task 3)



* 1. Functional Components with Properties (Tasks 4 & 5)



1. Component Interaction and Event Handling (Task 6)
   1. The AppColor ComponentA screenshot of a computer

      AI-generated content may be incorrect.
   2. The AppColor Component

A screenshot of a computer

AI-generated content may be incorrect.

A blue screen with white text

AI-generated content may be incorrect.

A yellow screen with black text

AI-generated content may be incorrect.

1. Conclusion

Several fundamental React ideas were effectively introduced and reinforced in this lab, including the component-based architecture, the distinction between static and prop-driven functional components, and the use of event handling to implement user interactions. The accomplishment of these tasks offers a solid basis for using React to create interactive and modular user interfaces.

1. **Lab Questions and Answers**
2. **What is React?**

React is an open-source JavaScript library, created by Facebook, used for building user interfaces (UIs) or UI components. It is primarily used to develop single-page, fast, and interactive web applications by focusing on component-based architecture and a declarative programming style.

1. **What do you understand by React component and what command do you use to create a React component with or without property?**

A React component is an independent, reusable piece of code that defines how a section of the UI should look and behave. They act like JavaScript functions.

* To create a React component (with or without properties), we define a JavaScript function:

// Component without property (stateless)

function MyComponent() {

return <h1>Hello World</h1>;

}

// Component with property (accepts props as an argument)

function MyComponentWithProp(props) {

return <h1>Hello, {props.name}</h1>;

}

1. **What command will you use to render the newly created component named as MyReact?**

To render a component named MyReact into the HTML DOM, we use the ReactDOM.render() function in your index.js file:

ReactDOM.render(

<MyReact />,

document.getElementById('root')

);

1. **Suppose the MyReact Component has a property heading, write down the code that could be used to render the MyReact Component, and pass the message to the property heading as "this is my first element".**

<MyReact heading="this is my first element"/>

1. **Observe this code and answer the questions below: <AppColor heading="This is first element" lbl="Name : " color="green"/>**
2. What is the name of the React Component?

The name of the React Component is **AppColor**.

1. How many properties this component uses?

This component uses **three** properties: **heading**, **lbl**, and **color**.

1. **Look at the following Code:**

function GreetingElementwithProp(props) {

return (

<div className="App">

<h1>Wellcome, {props.studentname}</h1>;

</div>

);

}

export default ??????

**What will you write to make this export this function correctly?**

We should replace ?????? with the function name, which is GreetingElementwithProp.

**export default GreetingElementwithProp;**

1. **Add a function that takes two properties as numbers, add these numbers on the click event of the button and display the sum.**

Here is the code for a component named AdderComponent that fulfills this requirement:

// Define the functional component that takes two number properties (prop1 and prop2)

function AdderComponent(props) {

// Function to calculate and display the sum on button click

function calculateSum() {

// Ensure props are treated as numbers

const num1 = Number(props.prop1);

const num2 = Number(props.prop2);

// Check for valid numbers and calculate sum

if (!isNaN(num1) && !isNaN(num2)) {

const sum = num1 + num2;

// Display the result using an alert (as suggested by the lab context, though a UI update is preferred)

alert(`The sum of ${num1} and ${num2} is: ${sum}`);

} else {

alert("Please ensure both properties are valid numbers.");

}

}

return (

<div style={{ margin: '20px', padding: '10px', border: '1px solid #ccc' }}>

<h2>Number Adder</h2>

<p>Property 1: {props.prop1}</p>

<p>Property 2: {props.prop2}</p>

<button onClick={calculateSum}>

Calculate Sum

</button>

</div>

);

}

export default AdderComponent;

// Example of how to use it in index.js:

// <AdderComponent prop1="15" prop2="27" />