**Exercise 10**

**Objectives:**

1. **Define JSX**

JSX (JavaScript XML) is a syntax extension for JavaScript used in React. It allows writing HTML-like code inside JavaScript files. JSX makes UI code more readable and easier to write. Under the hood, JSX is transformed into React.createElement() calls. JSX elements must have one parent and follow proper HTML/XML syntax

1. **Explain about ECMA Script**

ECMAScript is the standardized scripting language specification for JavaScript. It defines core features like variables, functions, classes, and syntax rules. ES6 (ECMAScript 2015) introduced major improvements like let, const, classes, and arrow functions. Newer ECMAScript versions (ES7, ES8, etc.) continue to enhance the language. JavaScript engines (like V8 in Chrome) implement ECMAScript specifications.

1. **Explain React.createElement()**

React.createElement() is the core API for creating React elements. JSX is syntactic sugar for React.createElement(). It takes three arguments: the tag/component, props object, and children. Example: React.createElement('h1', null, 'Hello') renders <h1>Hello</h1>. React uses this to build the Virtual DOM before updating the real DOM.

1. **Explain how to create React nodes with JSX**

React nodes are created using JSX by writing elements like <h1>Hello</h1>. You can embed JavaScript expressions inside curly braces: {expression}. Elements can have attributes, children, and conditional rendering. You can also use components as JSX tags like <MyComponent />. Each JSX tag returns a React node used to build the UI structure.

1. **Define how to render JSX to DOM**

JSX can be rendered into the DOM using ReactDOM.render().

Syntax: ReactDOM.render(<App />, document.getElementById('root')); This tells React to render the <App /> component inside the DOM node with ID root. JSX gets converted to JavaScript behind the scenes and added to the DOM. React updates the DOM efficiently using the Virtual DOM diffing algorithm.

1. **Explain how to use JavaScript expressions in JSX**

JavaScript expressions can be inserted into JSX using {} (curly braces). Example: <p>{user.name}</p> will display the user's name. You can use variables, functions, and even ternary operators in JSX. Statements like if or for can’t be used directly in JSX (use ternary or .map()). Expressions make JSX dynamic and interactive based on data or state.

1. **Explain how to use inline CSS in JSX**

Inline CSS in JSX is provided using a style prop with a **JavaScript object**. Example: <h1 style={{ color: 'red', fontSize: '20px' }}>Hello</h1>. The property names are written in **camelCase** instead of hyphenated. Values must be strings or variables (e.g., fontSize: '16px'). Inline styles are useful for dynamic styling or small scoped components.

**Hands On Practice**

1. **Create a React Application named “officespacerentalapp” which uses React JSX to create elements, attributes and renders DOM to display the page.**

npx create-react-app officespacerentalapp

1. **Include:**

* **Create an element to display the heading of the page.**
* **Attribute to display the image of the office space**
* **Create an object of office to display the details like Name, Rent and Address.**
* **Create a list of Object and loop through the office space item to display more data.**
* **To apply Css, Display the color of the Rent in Red if it’s below 60000 and in Green if it’s above 60000.**

**OfficeList.js**

import React from 'react';

function OfficeList() {

  const offices = [

    {

      name: "Tech Park A",

      rent: 45000,

      address: "Koramangala, Bangalore",

      image: "/office1.jpeg"

    },

    {

      name: "Corporate Hub B",

      rent: 75000,

      address: "Gachibowli, Hyderabad",

      image: "/office2.jpg"

    },

    {

      name: "StartUp Zone C",

      rent: 60000,

      address: "Baner, Pune",

      image: "/office3.jpg"

    }

  ];

  const headingStyle = {

    textAlign: "center",

    fontSize: "28px",

    margin: "20px 0",

    color: "#333"

  };

  return (

    <div>

      <h1 style={headingStyle}>Office Space Rentals</h1>

      {offices.map((office, index) => {

        const rentStyle = {

          color: office.rent > 60000 ? 'green' : office.rent < 60000 ? 'red' : 'orange',

          fontWeight: 'bold'

        };

        return (

          <div key={index} style={{ margin: '20px', border: '1px solid #ccc', padding: '15px' }}>

            <img src={office.image} alt={office.name} style={{ width: '300px', height: '150px' }} />

            <h2>{office.name}</h2>

            <p><span style={rentStyle}>Rent: ₹{office.rent}</span></p>

            <p>Address: {office.address}</p>

          </div>

        );

      })}

    </div>

  );

}

export default OfficeList;

**App.js**

import React from 'react';

import OfficeList from './OfficeList';

function App()

{

  return(

    <div>

      <OfficeList/>

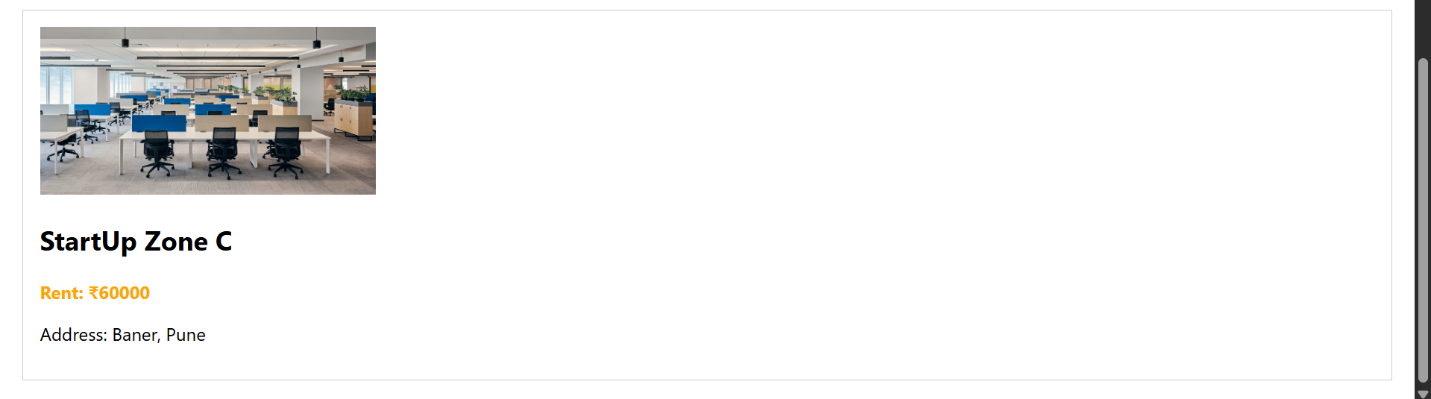
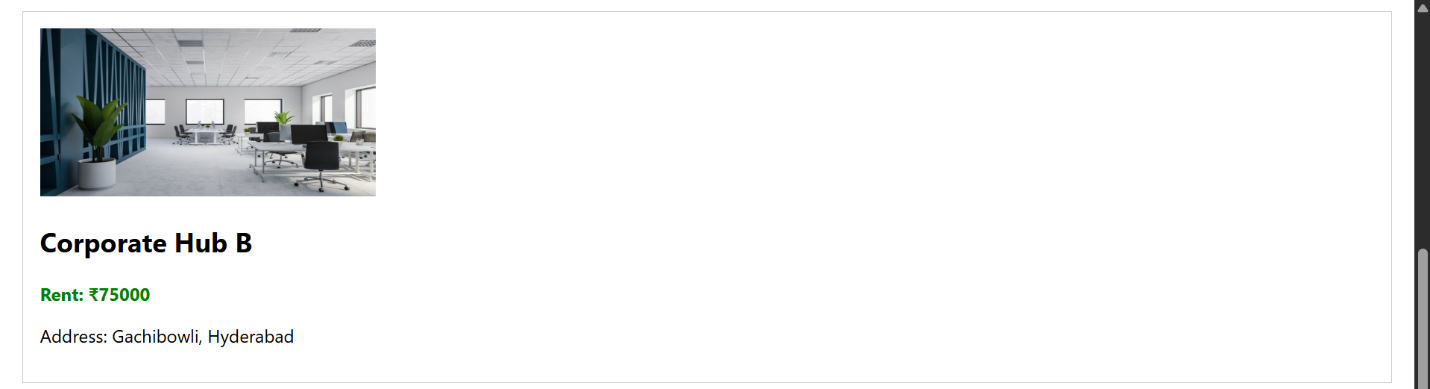
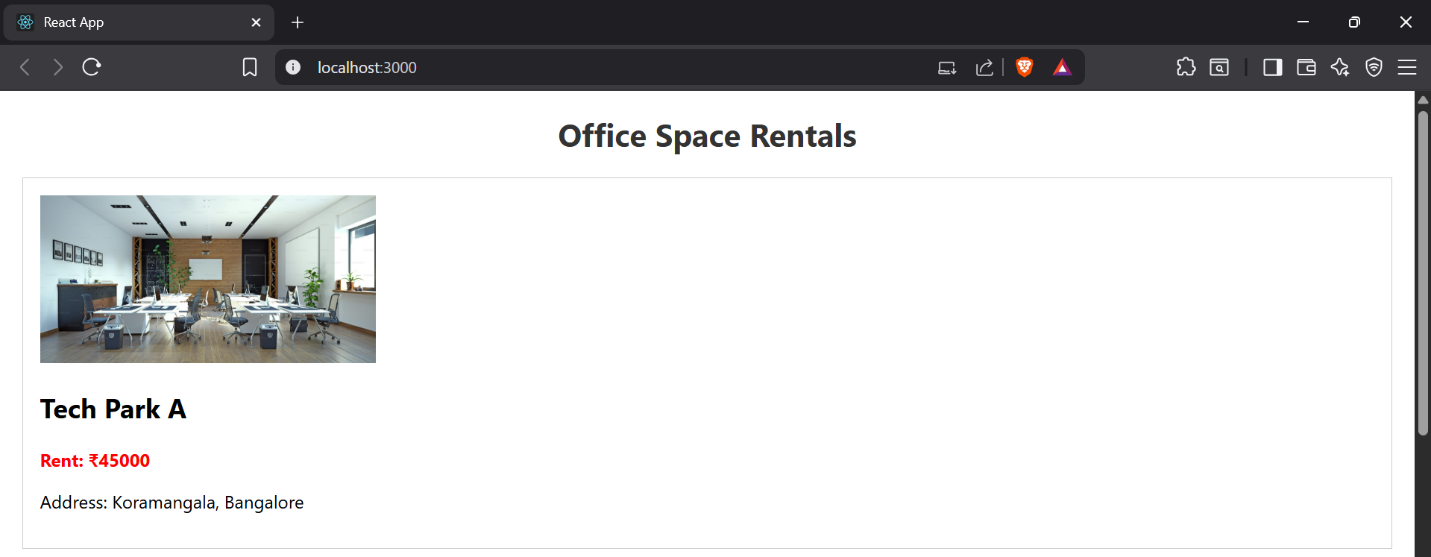
    </div>

  );

}

export default App;

**Output:**

****