**WEEK – 07**

**ReactJS-HOL**

**Superset ID: 6262264**

**EXERCISE 10:**

**Objectives:**

**1. Define JSX.**

JSX (JavaScript XML) is a syntax extension for JavaScript used in React. It allows us to write HTML-like code within JavaScript files. JSX makes it easier to create React elements and understand the component structure visually.

**EXAMPLE:**

const element = <h1>Hello, world!</h1>;

JSX is not HTML, but gets compiled to React.createElement() under the hood.

**2. Explain about ECMA Script**

ECMAScript (ES) is the standardized scripting language that forms the basis of JavaScript. It defines the core features of the language, such as variables, data types, functions, and classes.

* ES6 (ECMAScript 2015) introduced modern features like let, const, arrow functions, classes, modules, destructuring, and more.
* React uses many ES6 features to write clean and efficient component-based code.

**3. Explain React.createElement()**

React.createElement() is a method used by React to create virtual DOM elements.

**Example:**

const element = React.createElement(

'h1',

{ className: 'greeting' },

'Hello, world!'

);

* First argument: Type of element ('h1')
* Second: Props object
* Third: Children (text or nested elements)

JSX is a syntactic sugar for React.createElement().

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

To create React nodes using JSX, we write HTML-like tags inside JavaScript:

**EXAMPLE:**

const node = <p>This is a paragraph</p>;

React nodes can also include components:

**EXAMPLE:**

const MyComponent = () => <div>Hello</div>;

const root = <MyComponent />;

JSX supports nesting, expressions, and custom components.

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

JSX code can be rendered to the browser DOM using ReactDOM.render() in older versions or React 18's createRoot:

**EXAMPLE:**

import React from 'react';

import ReactDOM from 'react-dom/client';

const element = <h1>Hello, world!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(element);

This mounts the virtual DOM JSX element onto a real DOM node.

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

We can embed JavaScript expressions in JSX by wrapping them in {}:

**EXAMPLE:**

const name = "Ruby";

const element = <h1>Hello, {name}</h1>;

Allowed expressions include:

* Variables
* Function calls
* Math operations
* Ternary operators (condition ? a : b)

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

Inline CSS in JSX is done using object syntax inside {}:

**EXAMPLE:**

const style = {

color: 'blue',

fontSize: '18px'

};

const element = <h1 style={style}>Styled Text</h1>;

Or directly:

**EXAMPLE:**

<h1 style={{ color: 'red', textAlign: 'center' }}>Inline Styled Heading</h1>

Note: Property names are written in camelCase (e.g., backgroundColor instead of background-color).

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

***App.js:***

import './App.css';

function App() {

  const heading = <h1>Office Space Rental App</h1>;

  const officeImage = "https://t4.ftcdn.net/jpg/03/84/55/29/360\_F\_384552930\_zPoe9zgmCF7qgt8fqSedcyJ6C6Ye3dFs.jpg";

  const office = {

    name: "Tech Park Tower",

    rent: 55000,

    address: "100 Feet Road, Coimbatore"

  };

  const officeList = [

    { name: "IT Hub", rent: 45000, address: "Peelamedu" },

    { name: "Digital Plaza", rent: 62000, address: "Avinashi Road" },

    { name: "CodeNest", rent: 75000, address: "Gandhipuram" },

    { name: "Startup Hive", rent: 50000, address: "Singanallur" }

  ];

  // JSX Render

  return (

    <div className="App">

      {heading}

      <img src={officeImage} alt="Office" width="400" />

      <h2>Featured Office:</h2>

      <p><strong>Name:</strong> {office.name}</p>

      <p><strong>Rent:</strong> <span style={{ color: office.rent < 60000 ? 'red' : 'green' }}>{office.rent}</span></p>

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

      <hr />

      <h2>Available Office Spaces</h2>

      {officeList.map((item, index) => (

        <div key={index} style={{ border: '1px solid #ccc', padding: '10px', marginBottom: '10px' }}>

          <p><strong>Name:</strong> {item.name}</p>

          <p>

            <strong>Rent:</strong>

            <span style={{ color: item.rent < 60000 ? 'red' : 'green' }}> {item.rent}</span>

          </p>

          <p><strong>Address:</strong> {item.address}</p>

        </div>

      ))}

    </div>

  );

}

export default App;

***Index.js :***

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

***App.css:***

.App {

  font-family: Arial, sans-serif;

  padding: 20px;

  max-width: 800px;

  margin: auto;

}

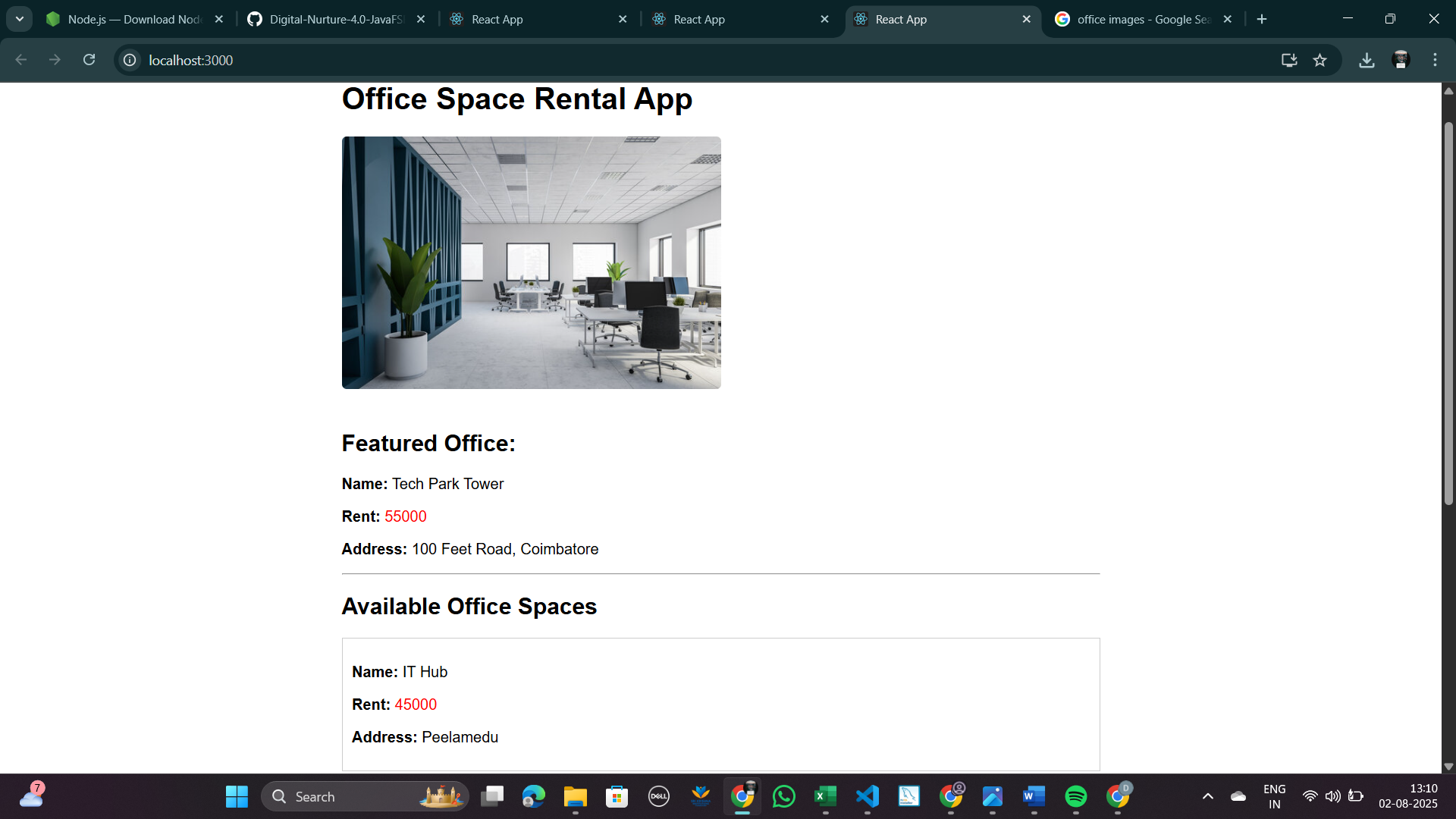
img {

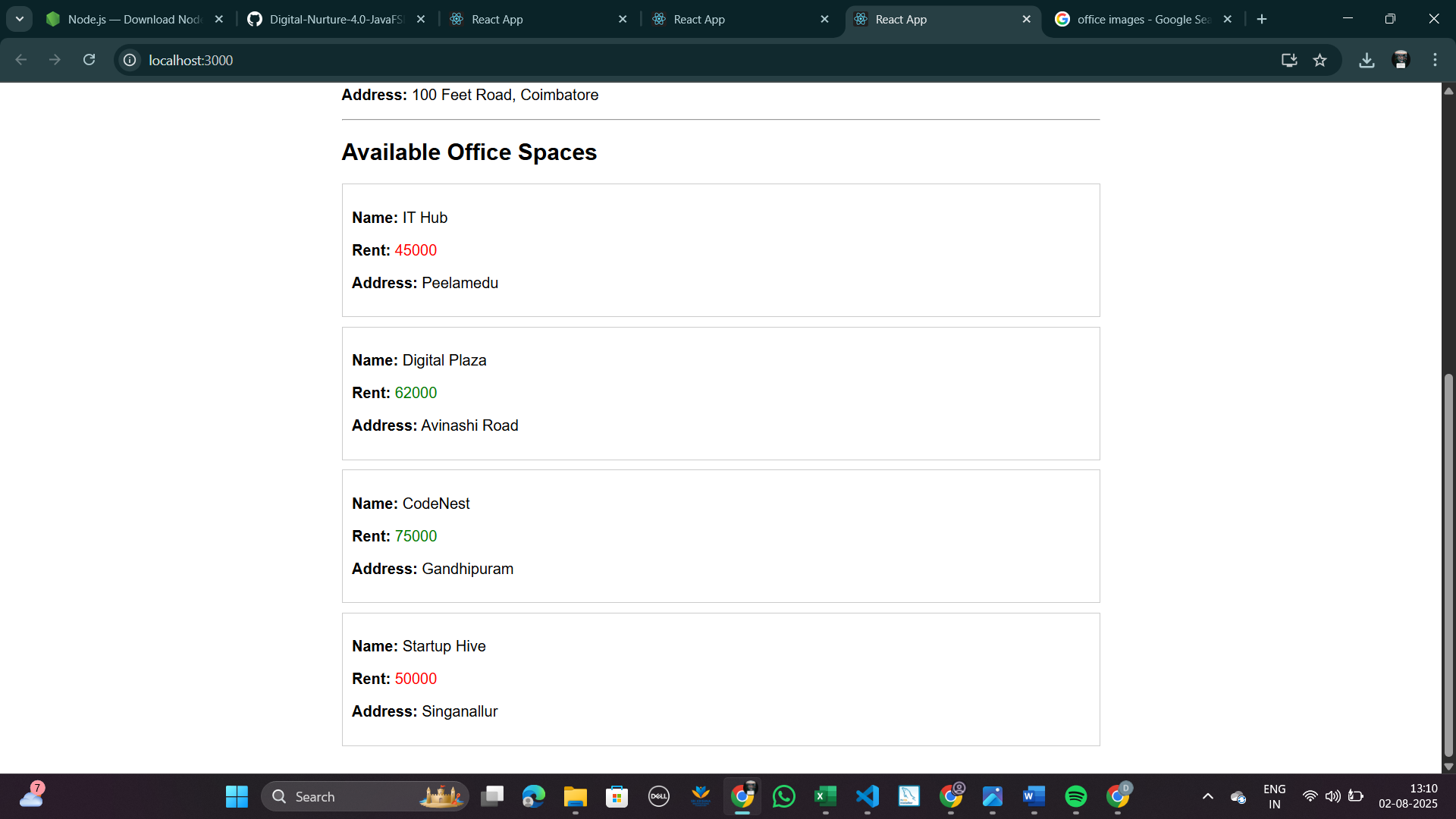
  margin-bottom: 20px;

  border-radius: 5px;

}

**OUTPUT:**





**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**