1. **ReactJS-HOL1:**

Describe SPA and Its Advantages:

One kind of web application is called a Single-Page Application (SPA), which loads a single HTML page and updates its content dynamically while the user interacts with it. SPAs provide a smooth, app-like experience by using AJAX and JavaScript to update specific parts of the page rather than loading whole new pages from the server.

Advantages of SPA:

Faster Loading: After the initial load, only data is transferred rather than entire webpages, which minimizes page reloads and improves user experience.

Smooth User Experience: Like desktop apps, navigation is easy and content changes instantly.

Decreased Server Load: The server only provides backend data, not complete page markup, because the majority of logic is handled on the client side.

* Define React and identify its working

Facebook created the open-source React JavaScript library to help developers create user interfaces, particularly for SPAs with lots of interaction and data changes. With React, developers can create encapsulated components that are capable of managing their own state and then combine them to create intricate user interfaces.

How React Operates:

makes use of a component-based architecture, in which user interfaces are divided into separate, reusable parts known as components.

keeps up a lightweight representation of the real DOM, called a virtual DOM. React efficiently updates only the modified portions of the real DOM after updating the virtual DOM when a component's state changes.

ensures predictable state management by using a unidirectional data flow, in which data flows from parent to child components.

Identify the differences between SPA and MPA:

|  |  |
| --- | --- |
| SPA | MPA |
| loads a single HTML page and makes dynamic updates. | Each request loads a new HTML page from the server |
| Faster navigation, no full page reloads | Slower navigation due to full page reloads. |
| Heavy reliance on client-side JavaScript | Relies more on server-side rendering. |
| SEO can be more challenging (but solvable). | SEO is simpler due to server-rendered content. |
| Example: Gmail, Google Maps. | Example: Amazon.com, Wikipedia. |

* Explain Pros & Cons of Single-Page Application

Pros:

Fast, smooth, and interactive user experience.

Decreased bandwidth usage after initial load.

Mobile-friendly; creates app-like experiences.

Cons:

* SEO can be difficult since content is rendered client-side.
* Initial load time may be longer due to loading all resources.
* JavaScript disabled? App won’t work.
* Can be complex to manage browser history and navigation.

Explain about React:

React is a robust JavaScript library primarily used for building rich, interactive user interfaces in SPAs. It promotes reusable components, encouraging modular code organization and easier maintenance. Its declarative approach means developers describe the UI state, and React takes care of updating the DOM efficiently when data changes.

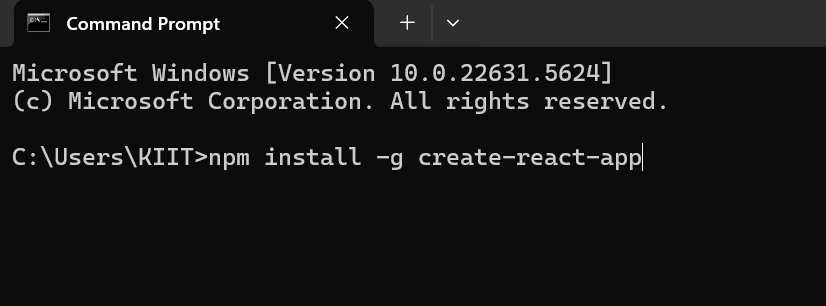
React's ecosystem is extensive, including tools like React Router (for navigation) and state management libraries (Redux, Context API). React can be used for websites, mobile apps (React Native), and even desktop apps, making it versatile for various development needs.

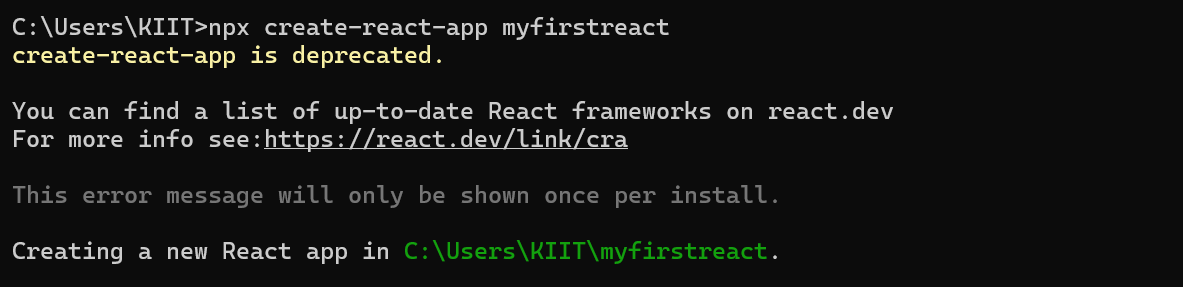
Define virtual DOM:

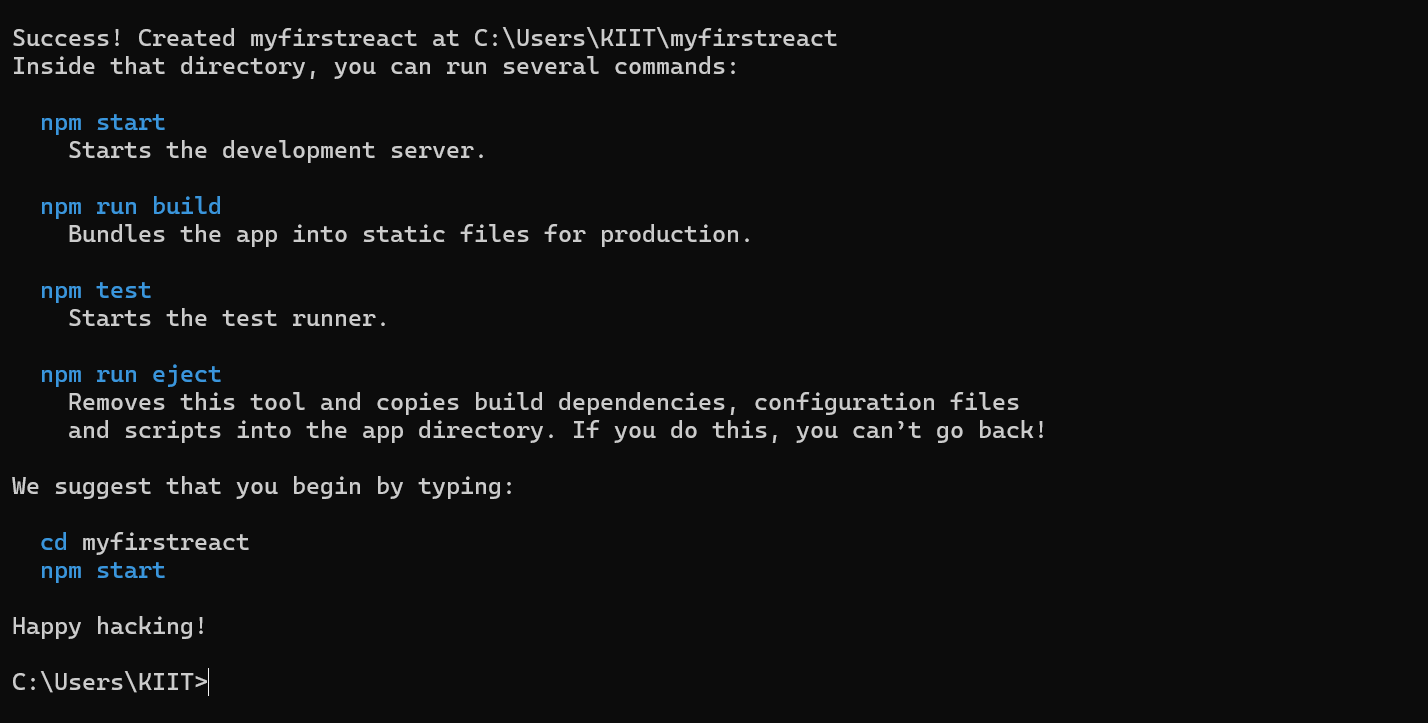
The Virtual DOM is a lightweight JavaScript representation of the actual DOM. React maintains this virtual DOM to optimize UI updates. When data changes, React updates the virtual DOM first, calculates the most efficient way to update the real DOM, and makes those changes in one batch. This minimizes costly direct interactions with the real DOM and improves app performance.

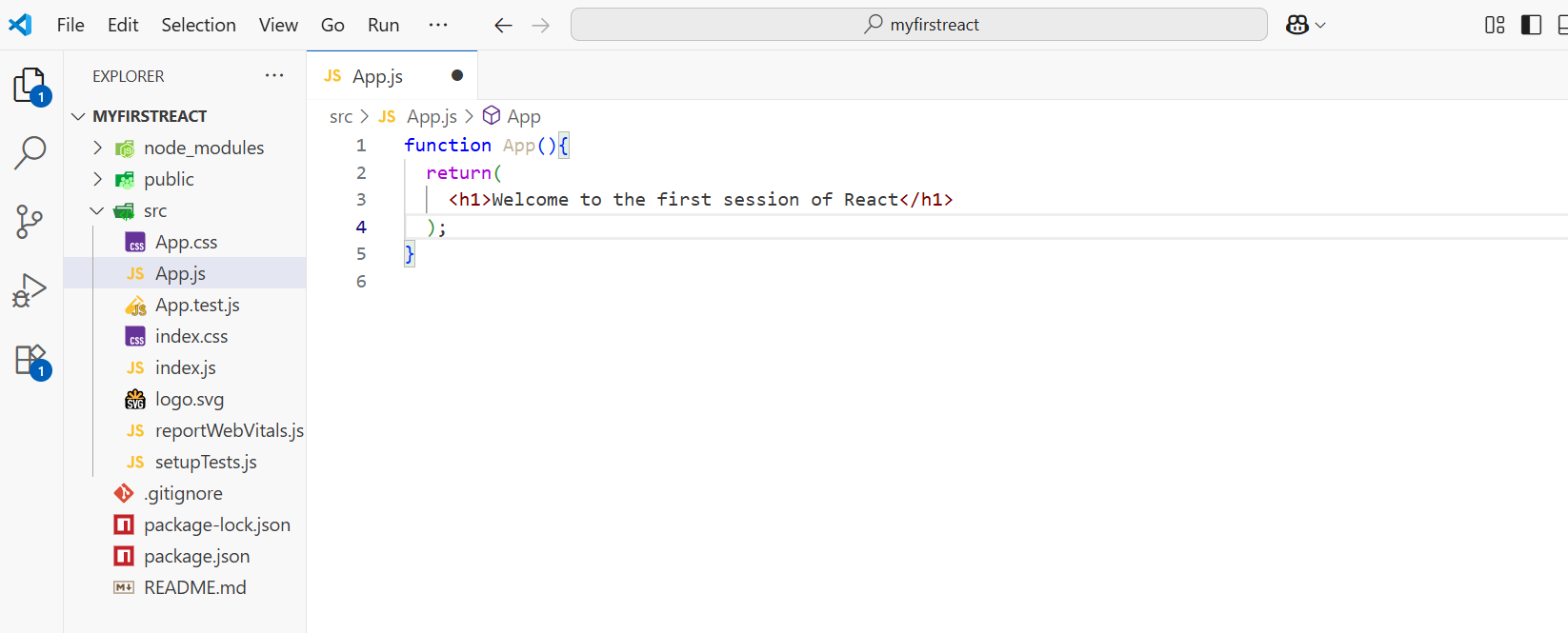
Explain Features of React:

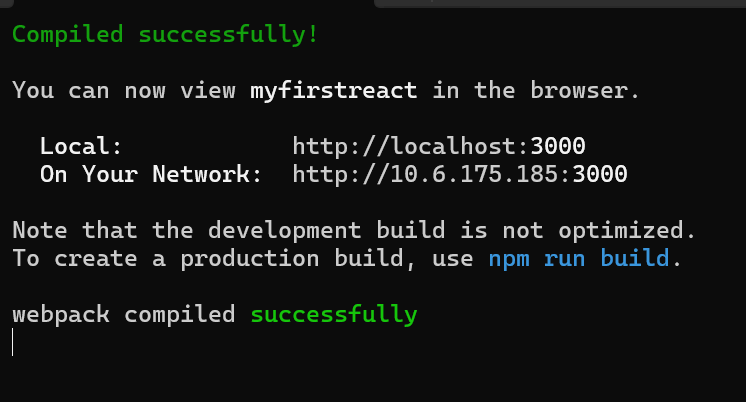
* Component-Based: Builds UIs using self-contained, reusable pieces (components).
* Declarative Syntax: Describe what the UI should look like; React handles updates.
* Virtual DOM: Boosts efficiency by updating only changed UI parts.
* JSX Syntax: Blends HTML-like code with JavaScript for easier development.
* Unidirectional Data Flow: Makes data flow predictable and easier to debug.
* Large Ecosystem: Supported by vast libraries and tools for routing, state management, etc.
* Reusable Components: Code once, use anywhere within the app for consistency.
* Community Support: Large, active community and continuous development.

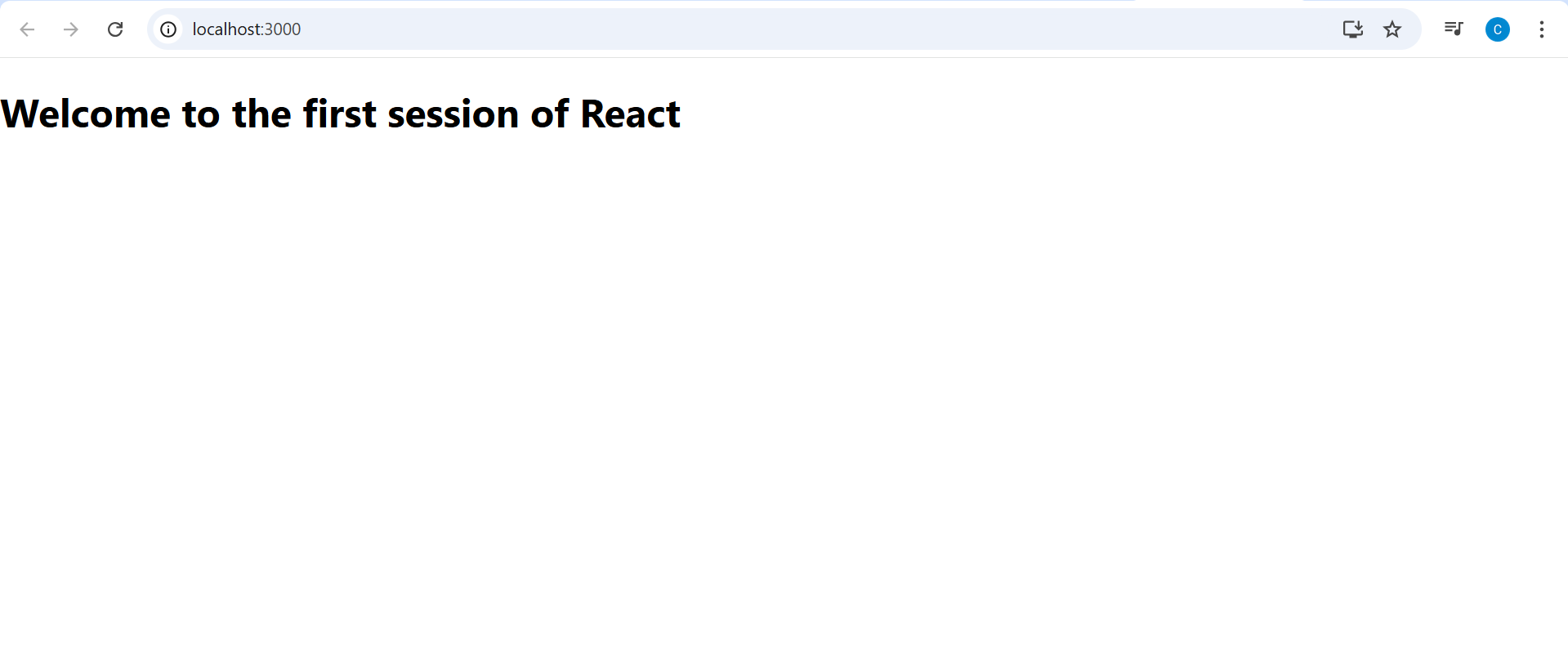












1. **ReactJS-HOL2:**

React Components:

React components are reusable, independent pieces of UI. Each component can have its own logic and state, making apps modular.

Components vs JS Functions

|  |  |
| --- | --- |
| React Component | JavaScript Function |
| Returns JSX (UI) | Returns any value |
| Managed by React | Not managed by React |
| Can have state/props | No built-in state/props |
| Follows lifecycle methods | No lifecycle |

Types of Components:

Function Components: Use functions to define UI, can use Hooks for state.

Class Components: Use ES6 classes, have state, use render() method, support lifecycle methods.

Class Component

A React class component is an ES6 class that extends React.Component, typically has a constructor for state and a render() method to display UI.

Function Component

A function component is just a function that takes props and returns JSX. Hooks let you add state and effects.

Component Constructor

The constructor method in a class component initializes state and binds methods. It’s only used in class components.

render() Function

The render() function (in class components) returns the JSX (UI) to display. It’s required in class components.

App.js:

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

  return (

    <div>

      <h1>Student Management Portal</h1>

      <Home />

      <About />

      <Contact />

    </div>

  );

}

export default App;

Apptest.js

import { render, screen } from '@testing-library/react';

import App from './App';

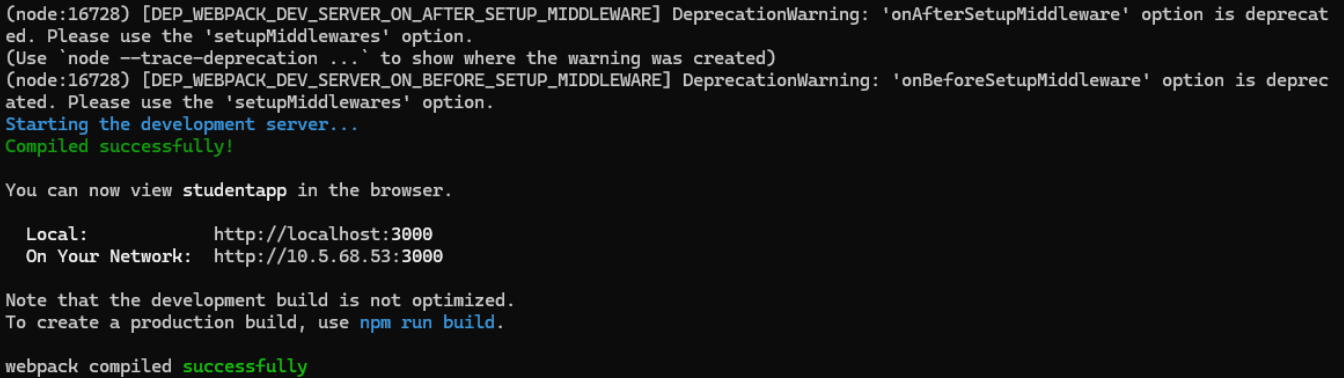
test('renders learn react link', () => {

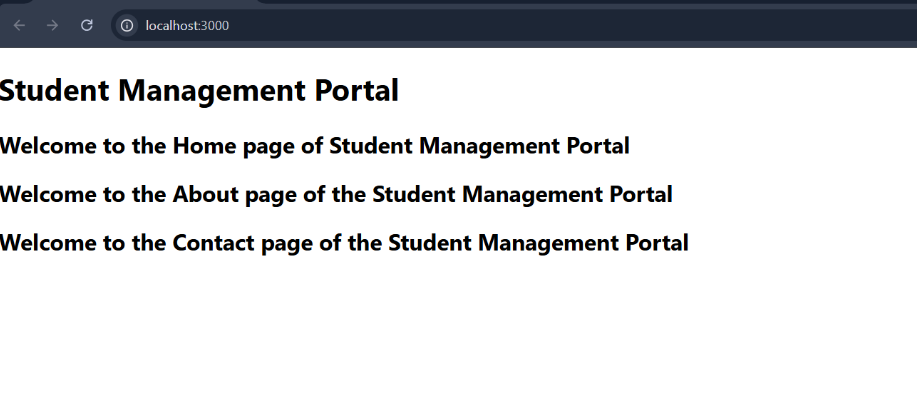
  render(<App />);

  const linkElement = screen.getByText(/learn react/i);

  expect(linkElement).toBeInTheDocument();

});





1. **ReactJs-HOL3:**

App.js:

import React from 'react';

import CalculateScore from './Components/CalculateScore';

function App() {

  return (

    <div>

      <CalculateScore Name="Ishita Chatterjee" School="Agrasain Balika Siksha Sadan" Total={480} Goal={5} />

    </div>

  );

}

export default App;

CalculateScore.js:

import React from 'react';

import '../Stylesheets/mystyle.css';

function CalculateScore(props) {

  const average = props.Total / props.Goal;

  return (

    <div className="center-box">

      <h2>Student Score Calculator</h2>

      <p><strong>Name:</strong> {props.Name}</p>

      <p><strong>School:</strong> {props.School}</p>

      <p><strong>Total Score:</strong> {props.Total}</p>

      <p><strong>Subjects:</strong> {props.Goal}</p>

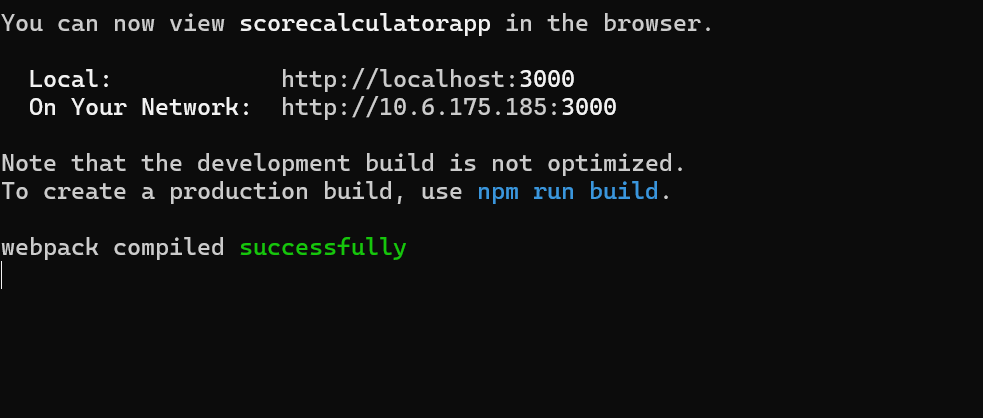
      <p><strong>Average Score:</strong> {average.toFixed(2)}</p>

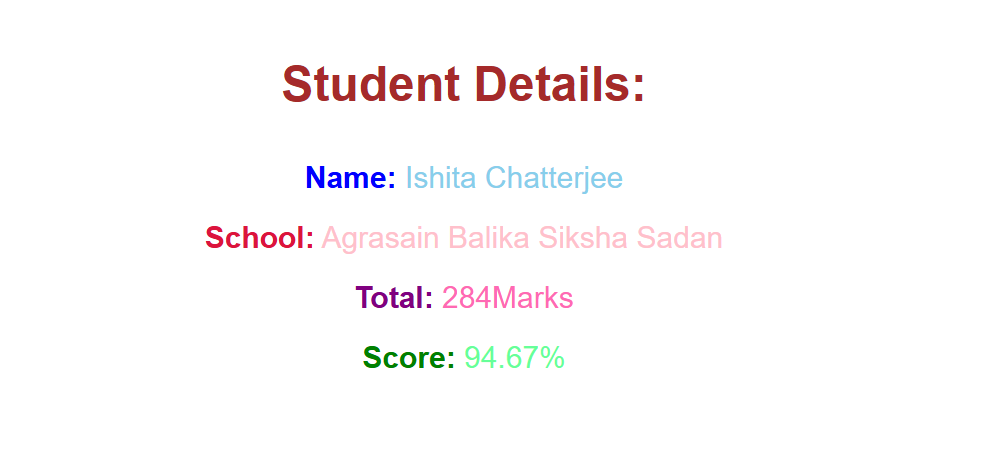
    </div>

  );

}

export default CalculateScore;





**ReactJS-HOL4:**

App.js:

import React from 'react';

import Posts from './Posts';

function App() {

  return (

    <div className="App">

      <Posts />

    </div>

  );

}

export default App;

Post.js:

class Post {

  constructor(userId, id, title, body) {

    this.userId = userId;

    this.id = id;

    this.title = title;

    this.body = body;

  }

}

export default Post;

Posts.js:

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

      error: null

    };

  }

  loadPosts = async () => {

    try {

      const response = await fetch('https://jsonplaceholder.typicode.com/posts');

      const data = await response.json();

      const postObjects = data.map(p => new Post(p.userId, p.id, p.title, p.body));

      this.setState({ posts: postObjects });

    } catch (error) {

      this.setState({ error });

      console.error("Error fetching posts:", error);

    }

  }

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

    alert("An error occurred: " + error);

    console.error("Component Error:", error, info);

  }

  render() {

    const { posts } = this.state;

    return (

      <div>

        <h1>Blog Posts</h1>

        {posts.map(post => (

          <div key={post.id} style={{ marginBottom: '20px' }}>

            <h3>{post.title}</h3>

            <p>{post.body}</p>

          </div>

        ))}

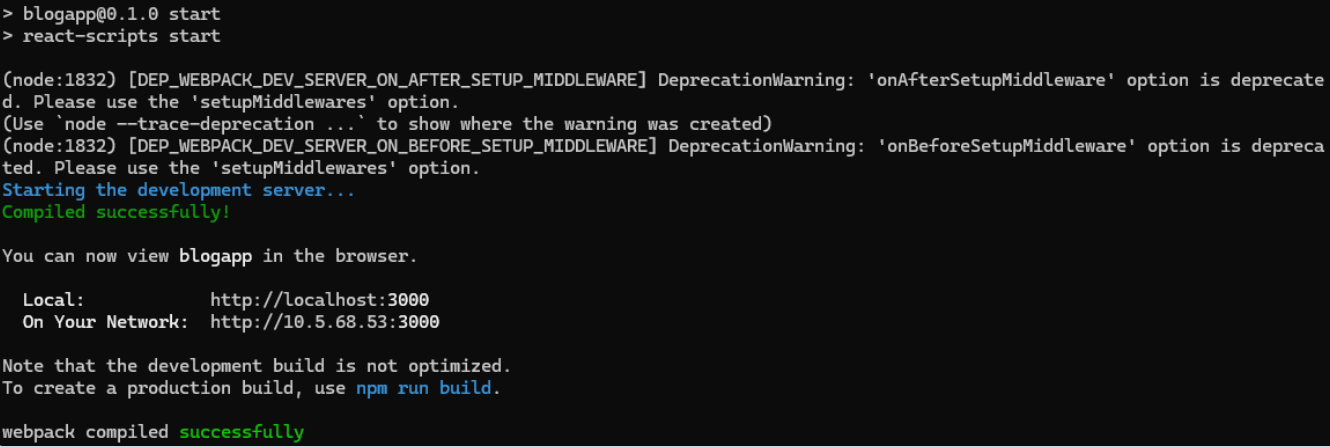
      </div>

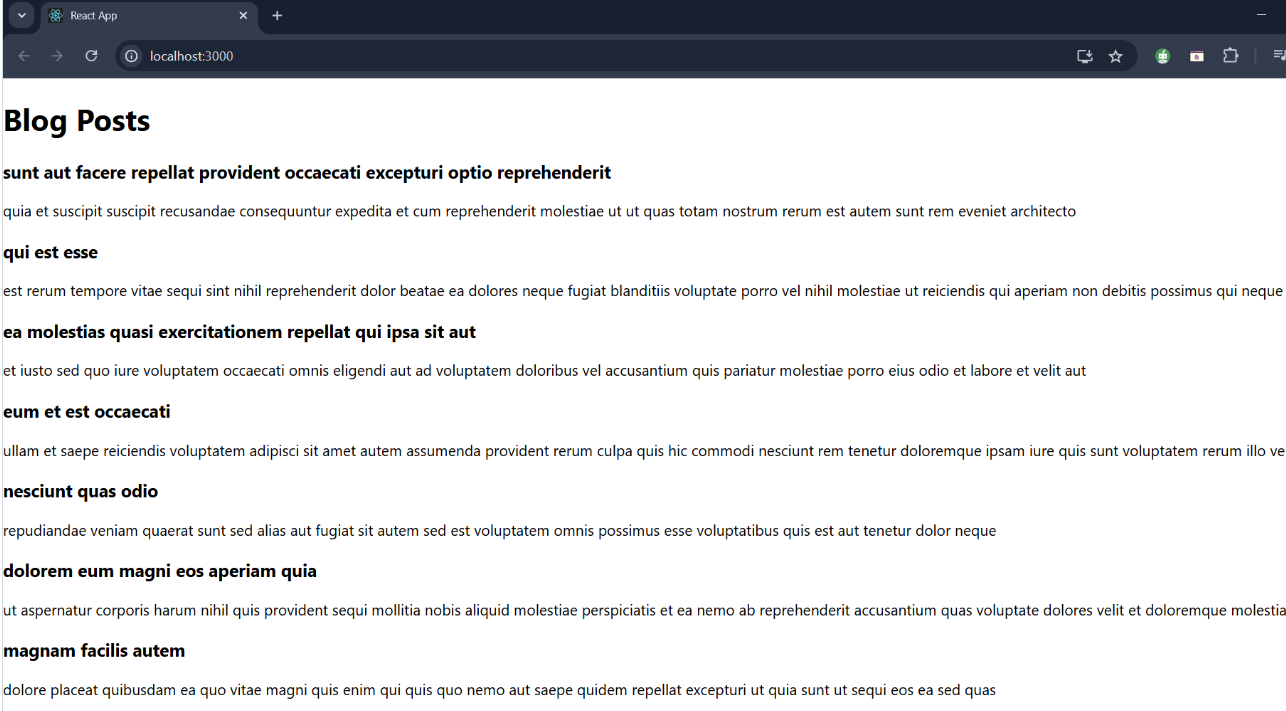
    );

  }

}

export default Posts;





**ReactJs\_HOL5:**

CohortDetails.module.css:

.box {

  width: 300px;

  display: inline-block;

  margin: 10px;

  padding: 10px 20px;

  border: 1px solid black;

  border-radius: 10px;

}

/\* Style for <dt> tag \*/

dt {

  font-weight: 500;

}

CohortDetails.js:

import React from 'react';

import styles from './CohortDetails.module.css';

function CohortDetails(props) {

    const { cohort } = props;

    return (

        <div className={styles.box}>

            <h3 style={{ color: cohort.currentStatus === 'ongoing' ? 'green' : 'blue' }}>

                {cohort.cohortCode} - <span>{cohort.technology}</span>

            </h3>

            <dl>

                <dt>Started On</dt>

                <dd>{cohort.startDate}</dd>

                <dt>Current Status</dt>

                <dd>{cohort.currentStatus}</dd>

                <dt>Coach</dt>

                <dd>{cohort.coachName}</dd>

                <dt>Trainer</dt>

                <dd>{cohort.trainerName}</dd>

            </dl>

        </div>

    );

}

export default CohortDetails;

