**Week-6**

**React**

1.ReactJS-HOL

**Objectives:**

**SPA and its benefits**

A Single Page Application (SPA) is a web app that loads a single HTML page and dynamically updates content without refreshing the page. It offers faster interactions and a smoother user experience.

Benefits:

1.Faster loading and smoother user experience.

2.Reduces server load by handling routing on the client side.

3.Seamless transitions without full page reloads.

**React:**

React is an open-source JavaScript library developed by Facebook for building user interfaces, especially single-page applications.

**Working of React:**  
React uses a component-based architecture and a virtual DOM. When data changes, React updates only the parts of the UI that changed by efficiently comparing the virtual DOM with the real DOM, ensuring high performance.

Difference between SPA and MPA:

SPA loads a single page and updates content dynamically without refreshing, offering a smoother user experience.  
MPA loads a new page from the server for every interaction, making it better for large, content-heavy websites.

Pros and Cons of SPA:

**Pros of SPA:**

1. Fast and responsive user experience.
2. Less server load due to client-side rendering.

**Cons of SPA:**

1. SEO optimization is more difficult.
2. Initial load time may be longer.

**Virtual DOM**:

The Virtual DOM is a lightweight copy of the real DOM used by React.  
When changes occur, React updates the virtual DOM first instead of the real DOM directly.  
It then compares the new virtual DOM with the previous one using a process called diffing.  
Only the changed parts are updated in the real DOM, making updates faster and more efficient.

**Features of React:**

1. **Component-Based** – UI is divided into reusable components.
2. **Virtual DOM** – Efficiently updates and renders only changed elements.
3. **JSX** – Allows writing HTML inside JavaScript for easier UI coding.
4. **Unidirectional Data Flow** – Data flows in one direction, making debugging easier.
5. **Declarative UI** – React updates the UI based on changes in state or props.
6. **Strong Community Support** – Backed by Facebook and a large developer community.

After installation of Node,we create a react app

**Creation of React App**

Install Create-react-app by running the following command in the command prompt:

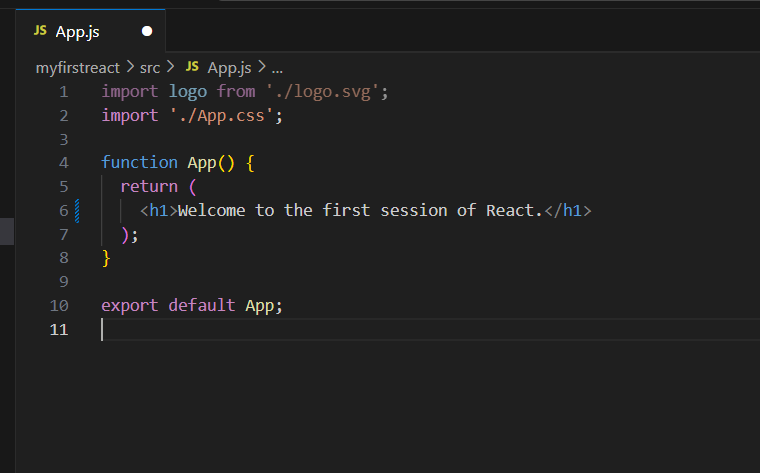
npm install -g create-react-app

To create a react application with the name of myfirstreact, type the following command:

npx create-react-app myfirstreact

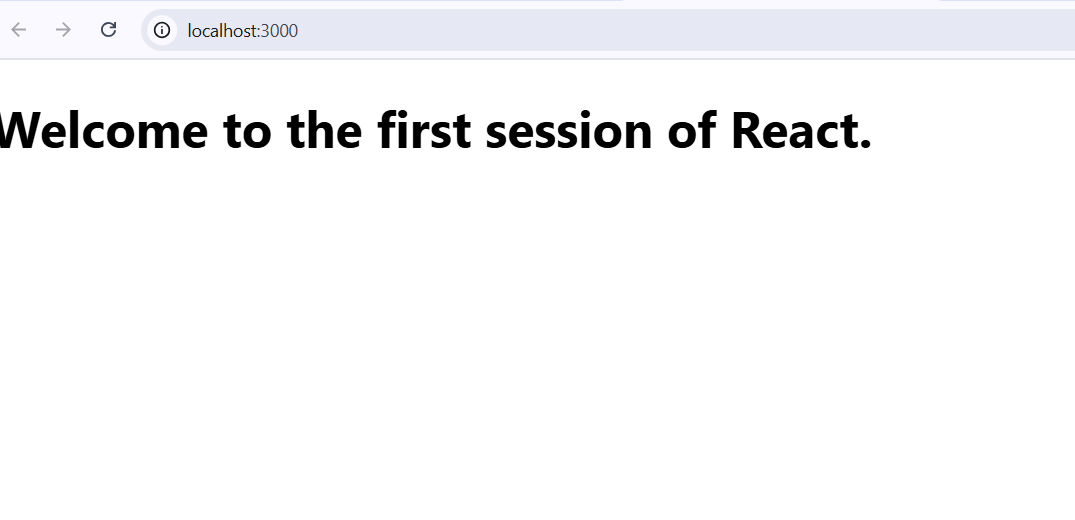
Once the App is created, navigate into the folder of myfirstreact by typing the following command:

cd myfirstreact

In App.js:

To run the following,

npm start

Output:

**2.ReactJS-HOL**

**Explain React components:**  
React components are reusable building blocks of UI that return HTML using JavaScript.  
They help in creating modular and maintainable user interfaces.

**Differences between components and JavaScript functions:**  
React components return JSX and manage UI behaviour with props and state.  
Regular JavaScript functions do not return UI and lack React-specific features.

**Types of components:**  
There are two main types: Class components and Function components.  
Both can render UI, but differ in syntax and how they handle state and lifecycle.

**Explain class component:**  
Class components are ES6 classes that extend React.Component.  
They can hold state and use lifecycle methods like componentDidMount().

**Explain function component:**  
Function components are simple JavaScript functions that return JSX.  
With Hooks, they can also manage state and side effects.

**Define component constructor:**  
The constructor is a special method used in class components to initialize state.  
It runs once when the component is created and receives props as a parameter.

**Define render() function:**  
The render() function is required in class components to return JSX.  
It defines what gets displayed on the UI.

**Handson Lab:**

Created a react App named StudentApp

Created three components namely Home,About,Contact

Called these three components in App.js

Home.js:

In src,created a folder named components:

import React, { Component } from 'react';

class Home extends Component {

render() {

    return (

      <div>

        <h3>Welcome to the Home page of Student Management Portal</h3>

      </div>

    );

  }

}

export default Home;

About.js

import React, { Component } from 'react';

class About extends Component {

render() {

return (

<div>

<h3>Welcome to the About page of Student Management Portal</h3>

</div>

);

}

}

export default About;

Contact.js

import React, { Component } from 'react';

class Contact extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the Contact page of Student Management Portal</h3>

      </div>

    );

  }

}

export default Contact;

App.js

import logo from './logo.svg';

import './App.css';

import Home from './components/Home';

import About from './components/About';

import Contact from './components/Contact';

function App() {

  return (

    <div className='container'>

      <Home/>

      <About/>

      <Contact/>

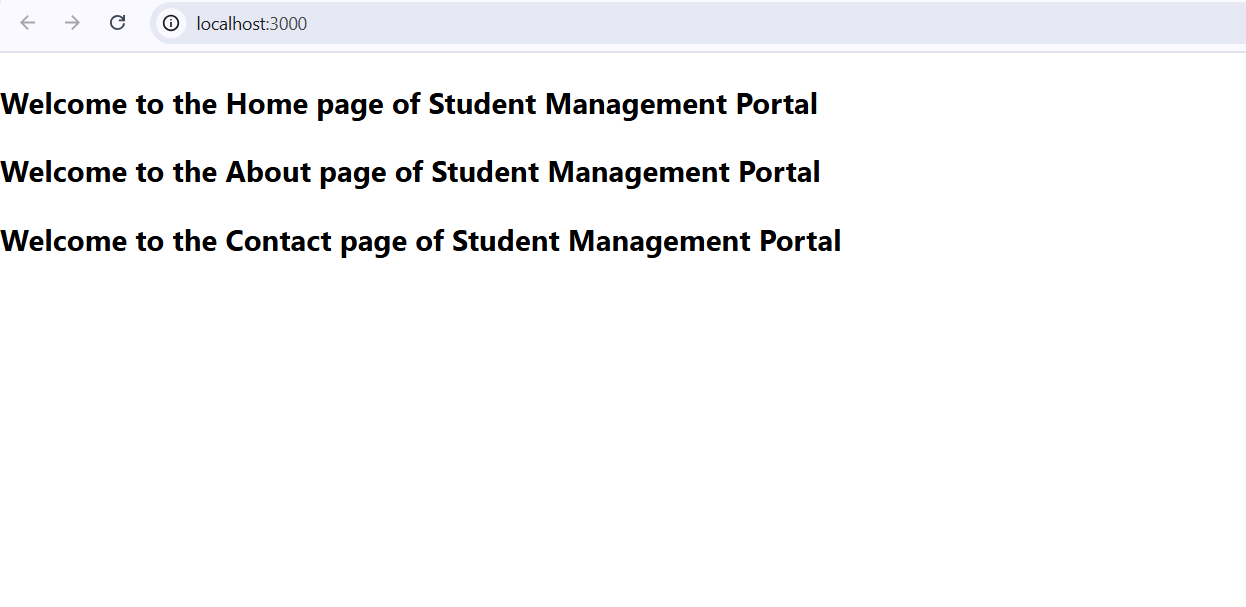
    </div>

  );

}

export default App;

Output:

****

**3.ReactJS-HOL**

Created a react app named scoreCalculatorApp

In src/components/

Created a component named CalculateScore

CalculateScore.js

import React from 'react';

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

function CalculateScore(props) {

  const average = props.total / props.goal;

  return (

    <div className="score-container">

      <h2>Student Score Details</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>Goal:</strong> {props.goal}</p>

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

    </div>

  );

}

export default CalculateScore;

In src/Stylesheets/

mystle.css

.score-container {

  border: 2px solid #333;

  padding: 20px;

  margin: 30px auto;

  max-width: 400px;

  border-radius: 8px;

  background-color: #f5f5f5;

  font-family: Arial, sans-serif;

}

.score-container h2 {

  color: #0066cc;

  margin-bottom: 15px;

}

.score-container p {

  font-size: 16px;

  margin: 5px 0;}

App.js

import React from 'react';

import './App.css';

import CalculateScore from './components/CalculateScore';

function App() {

  return (

    <div className="App">

      <h1>Welcome to Student Management Portal</h1>

      <CalculateScore

        name="John Doe"

        school="ABC High School"

        total={450}

        goal={5}

      />

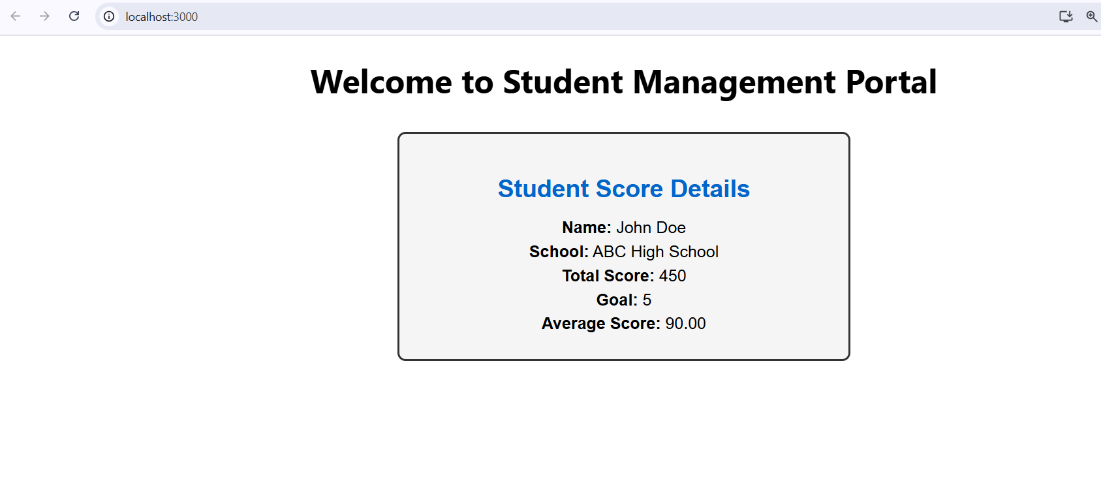
    </div>

  );

}

export default App;

Output: npm start



**4.ReactJS-HOL**

**Explain the need and Benefits of component life cycle:**  
Component lifecycle helps manage how components are created, updated, and removed.  
It allows you to run code at specific points, like fetching data or cleaning up resources.

**Identify various life cycle hook methods:**  
Key lifecycle methods include constructor(), render(), componentDidMount(), shouldComponentUpdate(), and componentWillUnmount().  
These hooks allow control over component behavior during mounting, updating, and unmounting.

**List the sequence of steps in rendering a component:**

1. constructor() is called to initialize state.
2. render() generates the UI, then componentDidMount() runs after the component is added to the DOM.

Created a react application named blogapp

In src/

Post.js

// src/Post.js

class Post {

  constructor(id, title, body) {

    this.id = id;

    this.title = title;

    this.body = body;

  }

}

export default Post;

**Posts.js**

// src/Posts.js

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

    };

  }

  // Step 6: loadPosts() method

  loadPosts() {

    fetch('https://jsonplaceholder.typicode.com/posts')

      .then((response) => response.json())

      .then((data) => {

        const postList = data.map(

          (item) => new Post(item.id, item.title, item.body)

        );

        this.setState({ posts: postList });

      })

      .catch((error) => {

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

      });

  }

  // Step 7: componentDidMount

  componentDidMount() {

    this.loadPosts();

  }// Step 9: Error handling

  componentDidCatch(error, info) {

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

  }

  // Step 8: render()

  render() {

    return (

      <div>

        <h1>Blog Posts</h1>

        {this.state.posts.map((post) => (

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

            <h3>{post.title}</h3>

            <p>{post.body}</p>

          </div>

        ))}

      </div>

    );

  }

}

export default Posts;

**App.js**

import React from 'react';

import './App.css';

import Posts from './Posts';

function App() {

  return (

    <div className="App">

      <Posts />

    </div>

  );

}

export default App;

Output: npm start

**5.ReactJS-HOL**

**The task of styling these react components.**

Navigating to the required folder, and the command:

npm install

To restore the packages required

code .

Created a new CSS Module in a file called “CohortDetails.module.css”.

**CohortDetails.module.css**

.box {

  width: 300px;

  display: inline-block;

  margin: 10px;

  padding: 10px 20px;

  border: 1px solid black;

  border-radius: 10px;

}

dt {

  font-weight: 500;

}

**CohortDetails.js**import styles from './components/CohortDetails.module.css';

function CohortDetails(props) {

    const { cohort } = props;

    // Conditional style for <h3> based on cohort status

    const headingStyle = {

        color: cohort.currentStatus.toLowerCase() === 'ongoing' ? 'green' : 'blue'

    };

    return (

        <div className={styles.box}>

            <h3 style={headingStyle}>

                {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>

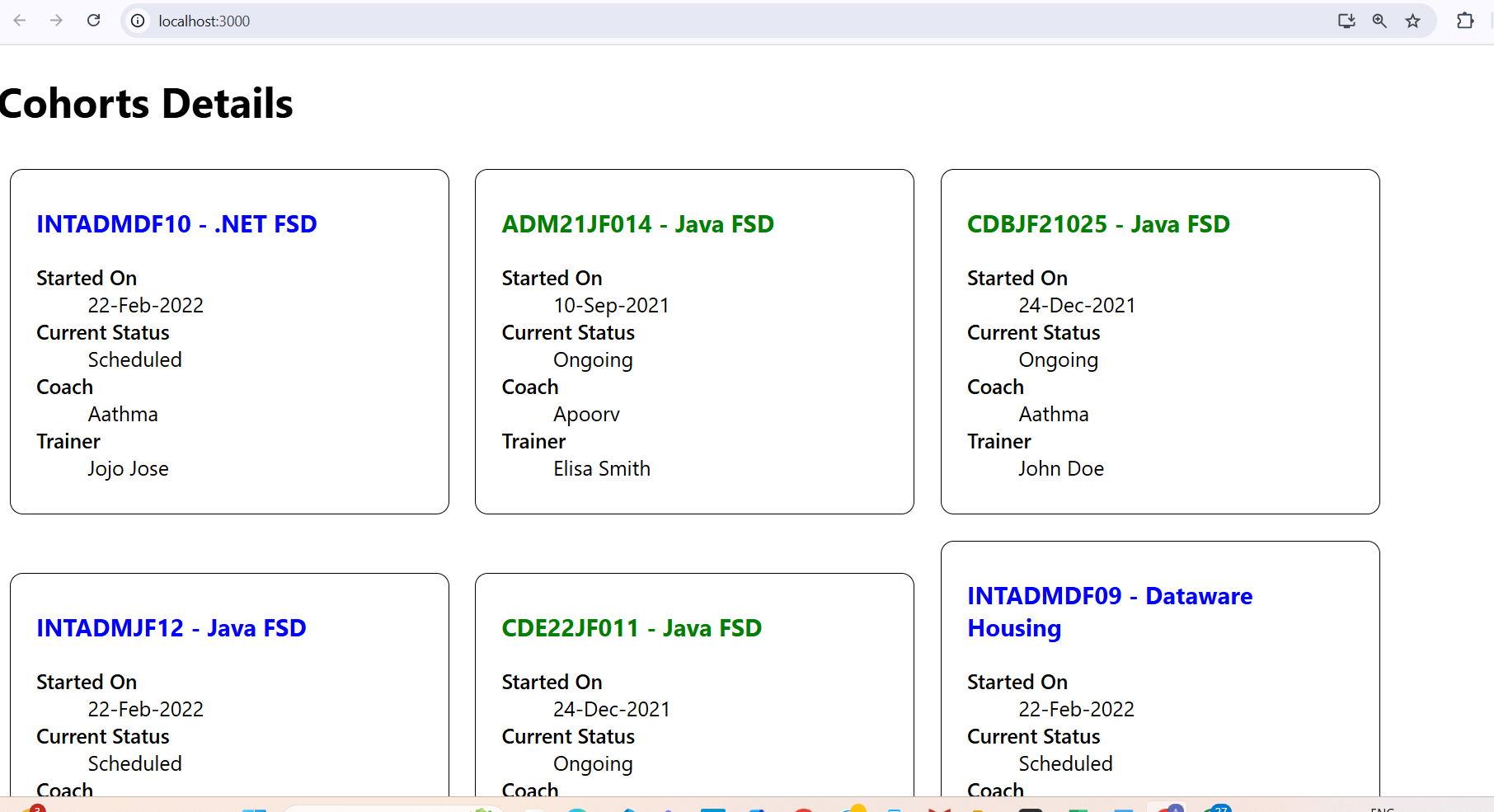
        </div>

    );

}

export default CohortDetails;

In this way, few changes are made to the given unzipped project and the changes are reflected.

Output: