**REACTJS**

**Exercise-1:**

**Objectives**

**• Define SPA and its benefits**

SPA (Single Page Application) is a web application that loads a single HTML page and dynamically updates content without reloading the entire page.  
Benefits:

* Faster user experience
* Reduces server load
* Seamless navigation

• Define React and identify its working

React is a JavaScript library for building user interfaces, developed by Meta.  
Working:

* Uses components to build UIs
* Utilizes virtual DOM for efficient updates
* React updates only changed parts of the UI

**• Identify the differences between SPA and MPA**

|  |  |
| --- | --- |
| **SPA** | **MPA** |
| |  |  | | --- | --- | | Loads a single HTML page |  |  |  |  | | --- | --- | | Faster after initial load |  |  |  |  | | --- | --- | | Client-side routing |  |  |  | | --- | | Better for dynamic content | | |  |  | | --- | --- | |  | Loads multiple HTML pages |  |  |  | | --- | --- | |  | Slower due to page reloads |  |  |  | | --- | --- | |  | Server-side routing |   Better for large apps like e-commerce |

**• Explain Pros & Cons of Single-Page Application**

**Pros Cons**

Fast navigation SEO challenges

Good user experience Initial load can be slow

Efficient data handling via APIs Requires more JavaScript

**• Explain about React**

React is a component-based JavaScript library used to build interactive UIs. It allows reusable UI components, manages UI state efficiently, and supports building SPAs.

**• Define virtual DOM**

Virtual DOM is a lightweight JavaScript representation of the real DOM. React compares the virtual DOM with the previous version (diffing) and updates only the changed parts in the actual DOM for better performance

**• Explain Features of React**

* Component-based architecture
* Virtual DOM for fast rendering
* JSX syntax for combining HTML with JavaScript
* Unidirectional data flow
* Reusable components
* Hooks for managing state and side effects

Create a new React Application with the name “myfirstreact”, Run the application to print “welcome to the first session of React” as heading of that page.

Steps:

1.Initially install the following into your system

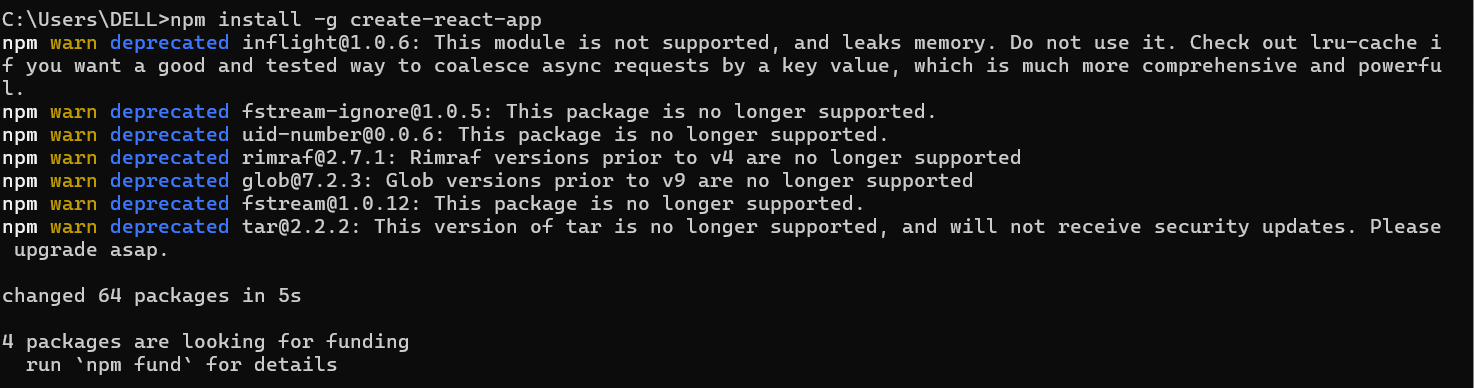
i. Visual Studio

ii. NPM from <https://nodejs.org/en/download/>

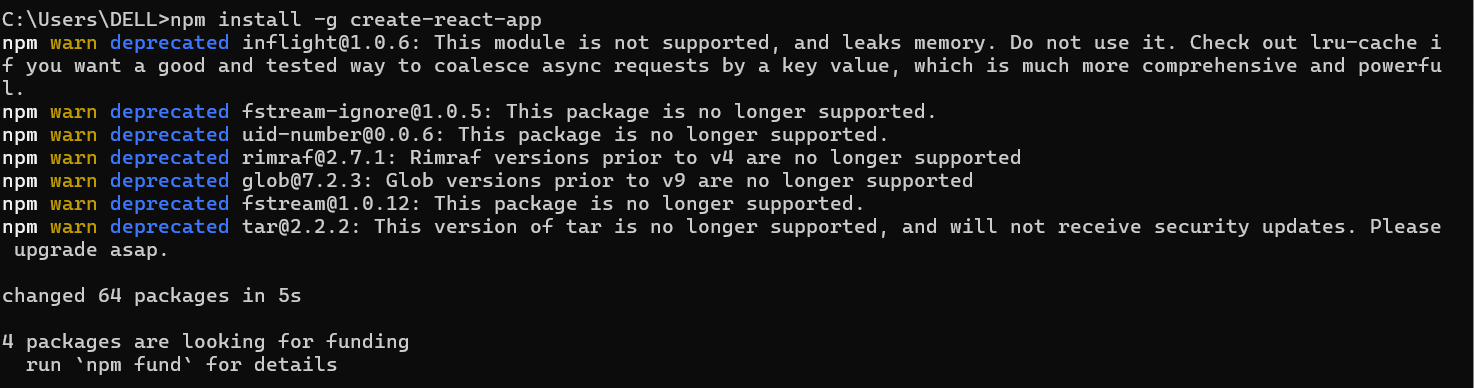
iii. NodeJs from <https://nodejs.org/en/download/>

2. After installing the above components run the following in command prompt

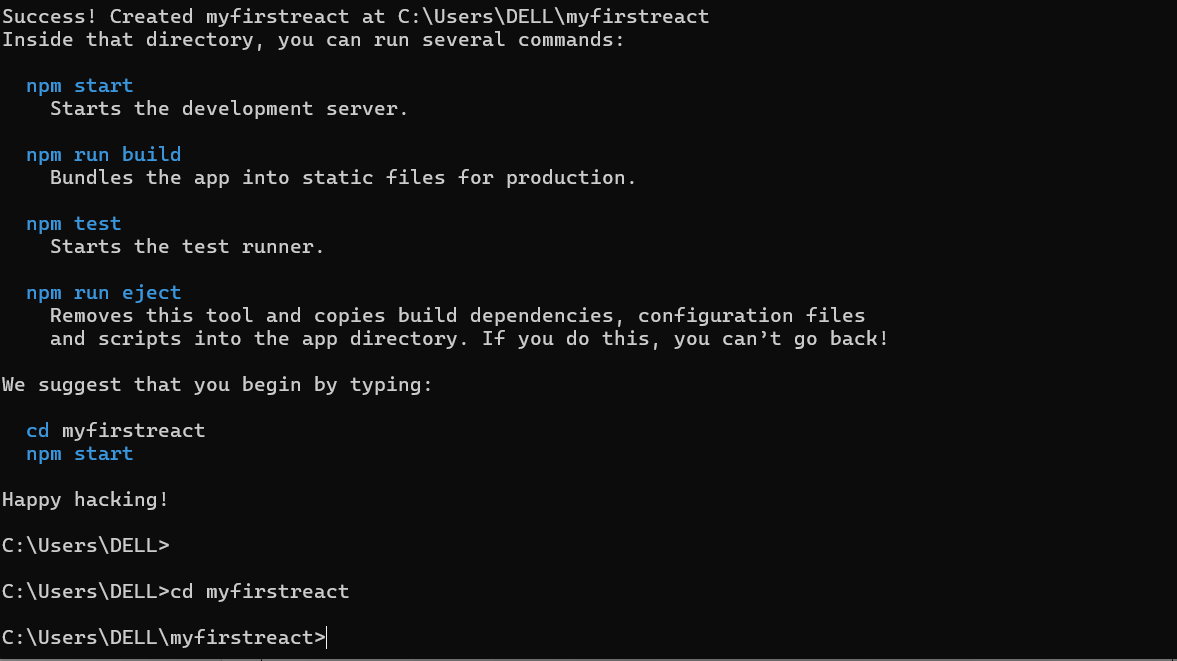
npm install -g create-react-app – installing create-react-app globally



npx create-react-app myfirstreact – creating reactjs named myfirstreact

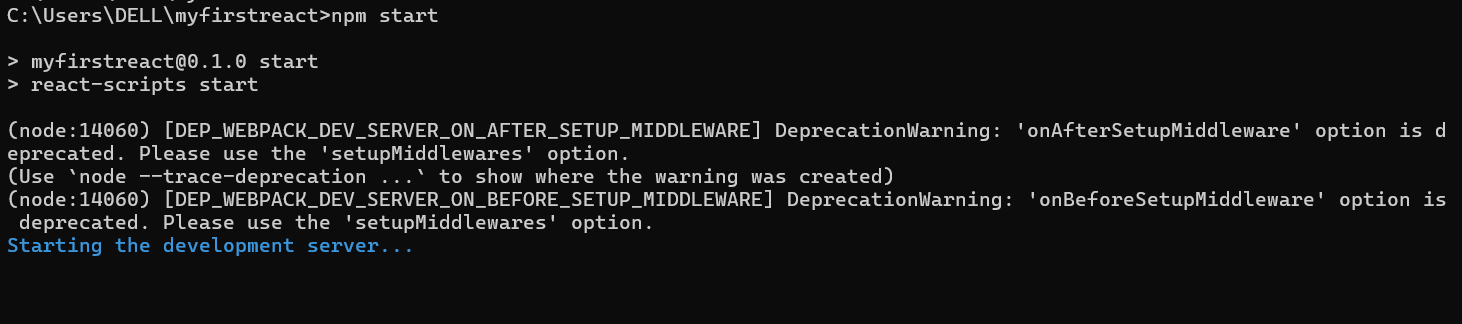


cd myfirstreact – moving to created folder



3. Then open this folder in VS studio and open terminal and run this command

npm start



4.In src/App.js and remove the code in there and write the following code

**App.js :**

import './App.css';

function App() {

  return (

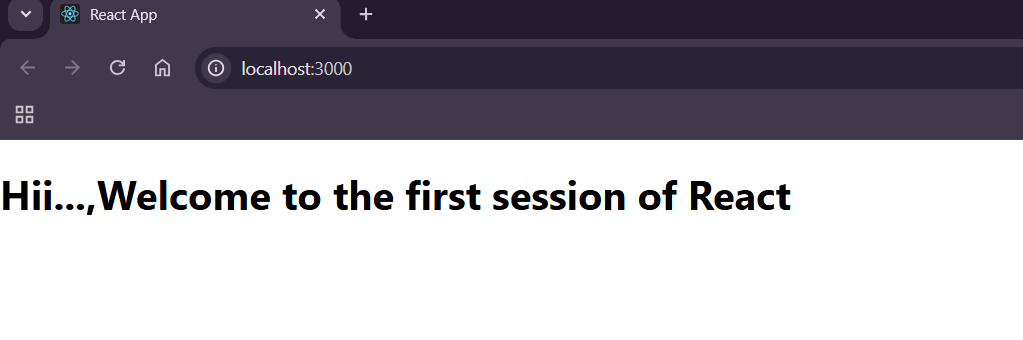
<h1>Hii...,Welcome to the first session of React</h1>

  );

}

export default App;

5.In the browser you will output as below



**Exercise-2:**

**Objectives:**

**• Explain React components**

React components are reusable, independent blocks of UI in a React application. They return JSX that describes how a portion of the UI should appear.

**• Identify the differences between components and JavaScript functions**

| **React Components** | **JavaScript Functions** |
| --- | --- |
| Return JSX (UI elements) | Return values (numbers, strings, etc.) |
| Used to build UIs | Used for general programming logic |
| Can manage state | Cannot manage React state |
| React-specific lifecycle (for class) | No lifecycle methods |

**• Identify the types of components**

1. Class Components
2. Function Components

**• Explain class component**

A classcomponent is a React component defined using ES6 classes. It has access to lifecycle methods and can manage its own state using this.state.

class MyComponent extends React.Component {

render() {

return <h1>Hello from class!</h1>;

}

}

**• Explain function component**

A functioncomponent is a simpler way to write components using functions. With ReactHooks, they can also manage state and side effects.t

function MyComponent() {

return <h1>Hello from function!</h1>;

}

**• Define component constructor**

The constructor is a special method in class components used to initialize state and bind methods. It runs once when the component is created.

constructor(props) {

super(props);

this.state = { count: 0 };

}

**• Define render() function**

The render() function is required in class components. It returns the JSX to be displayed on the screen. It is called whenever the component needs to re-render.

Ask ChatGPT

Create a react app for Student Management Portal named StudentApp and create a component named Home which will display the Message “Welcome to the Home page of Student Management Portal”. Create another component named About and display the Message “Welcome to the About page of the Student Management Portal”. Create a third component named Contact and display the Message “Welcome to the Contact page of the Student Management Portal”. Call all the three components.

Steps:

1.Initially run the following command in command prompt to creat a reasctjs file

npx create-react-app studentapp

2.Open this folder in vs studio

3.In src file create a file as “Components” and create the files as “Home.js”, ”Contact.js”, “About.js”

**About.js:**

import React from 'react'

function About() {

  return (

    <div>

        <h3>Welcome to About class</h3>

    </div>

  )

}

export default About

**Contact.js:**

import React from 'react'

function Contact() {

  return (

    <div>

          <h3>Welcome to Contact class</h3>

    </div>

  )

}

export default Contact;

**Home.js:**

import React from 'react'

function Home() {

  return (

    <div>

          <h3>Welcome to Home class</h3>

    </div>

  )

}

export default Home

**App.js:**

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

import './App.css'

function App() {

  return (

    <div >

      <Home />

      <About />

      <Contact />

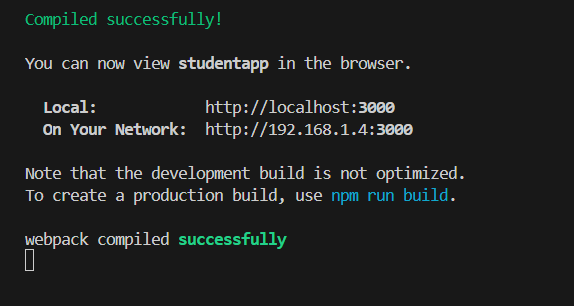
    </div>

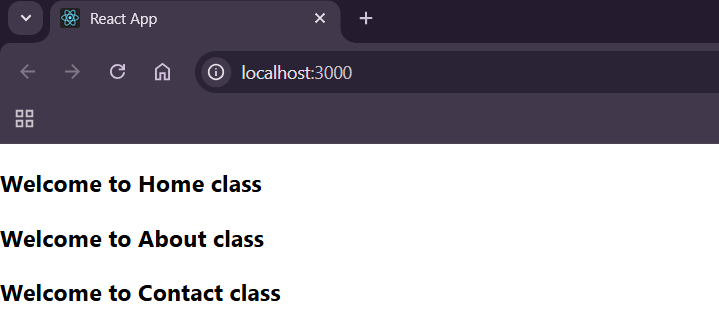
  );

}

export default App;

**Output:**





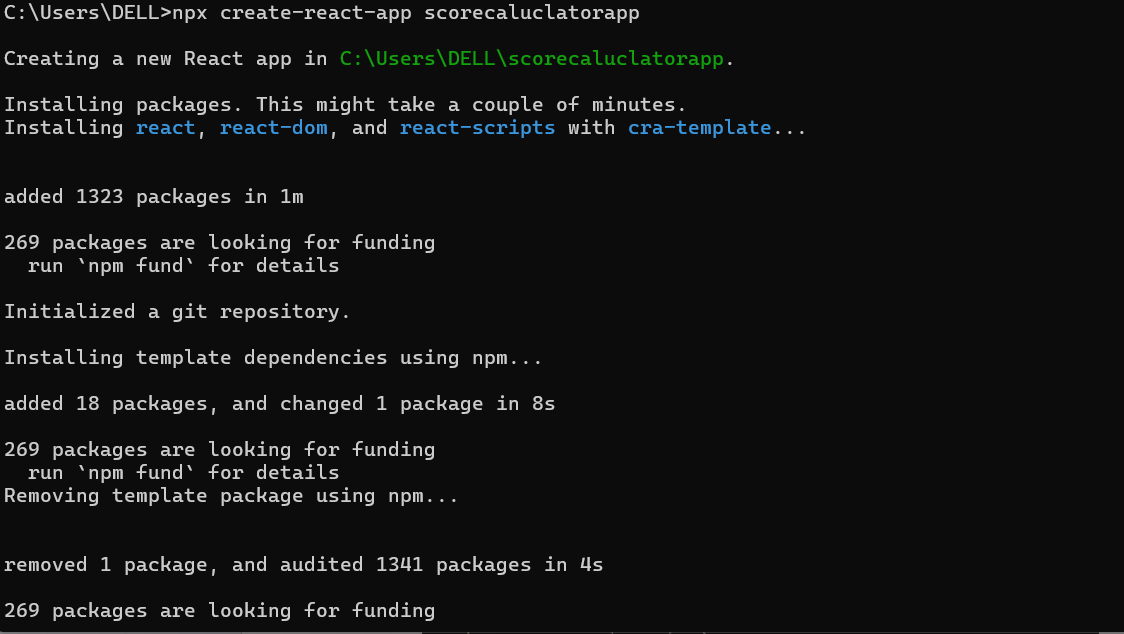
**EXERCISE-3:**

Create a react app for Student Management Portal named scorecalculatorapp and create a function component named “CalculateScore” which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.

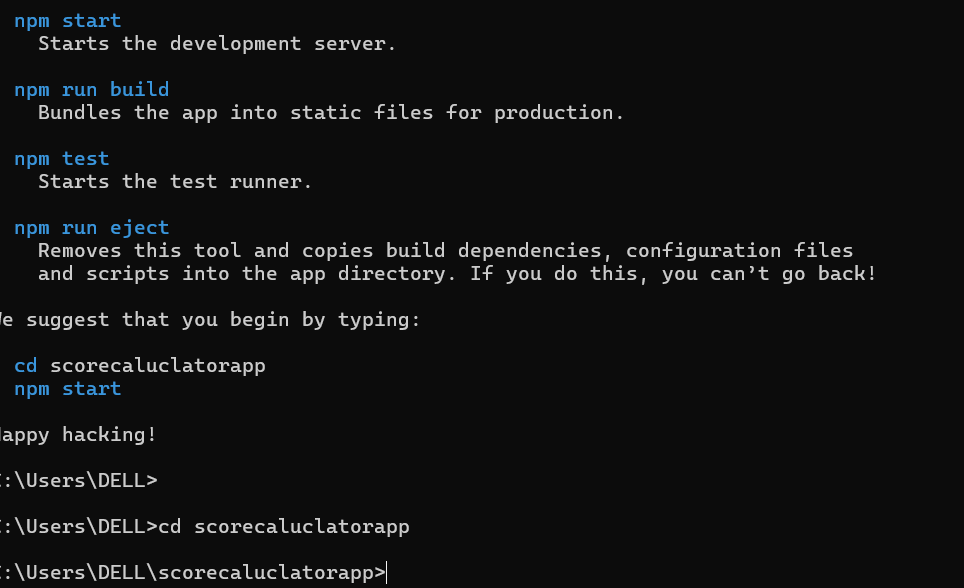
Steps:

1. Run this command in cmd

npx create-react-app scorecaluclatorapp



C:\Users\DELL>cd scorecaluclatorapp



2.Open this folder in vs

3.Create a “Components” file in src and create CaluclateScore.cc, CaluclateScore.js files

**CaluclateScore.js:**

import React from 'react'

import './CaluclateScore.css';

const pToD=(decimal)=>{

    return (decimal.toFixed(2)+'%')

}

const calc=(total,goal)=>{

    return pToD(total/goal)

}

function CaluclateScore({ Name, School, total, goal }) {

  return (

    <div>

        <h1 className='heading'>Student Details</h1>

        <h5 className='name'>Name: <span>{Name}</span> </h5>

        <h5 className='school'>School: <span>{School}</span></h5>

        <h5 className='total'>Total: <span>{total}</span><span>Marks</span></h5>

        <h5 className='score'>Score: <span>{calc(total,goal)}</span></h5>

    </div>

  )

}

export default CaluclateScore

**CaluclateScore.css:**

.heading{

    color:red;

}

.name{

    color: rgb(123, 20, 219);

    font-size: 1.3rem;

}

.score{

    color: rgb(185, 201, 13);

}

.school{

    font-size: 1rem;

    color: rgb(194, 6, 87);

}

.total{

    color: aqua;

}

**App.js:**

import './App.css';

import CaluclateScore from './Components/CaluclateScore';

function App() {

  return (

    <div className="App">

      <CaluclateScore Name={"Devi"}

      School={"Flora E.M School"}

      total={300}

      goal={3}

      />

    </div>

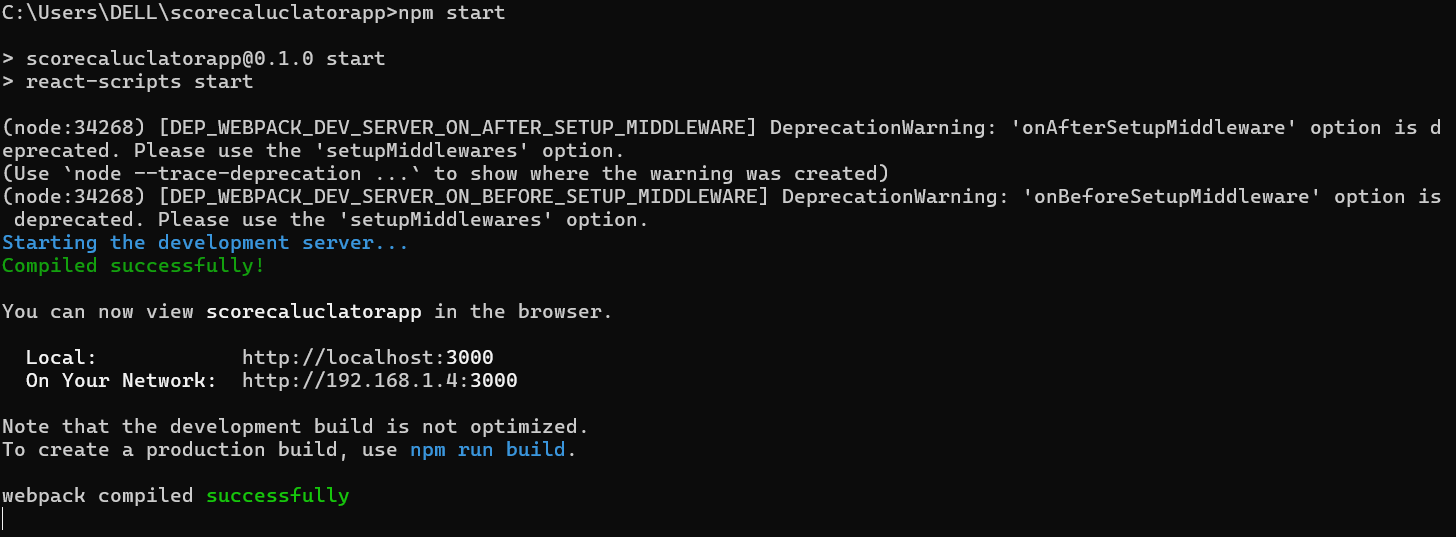
  );

}

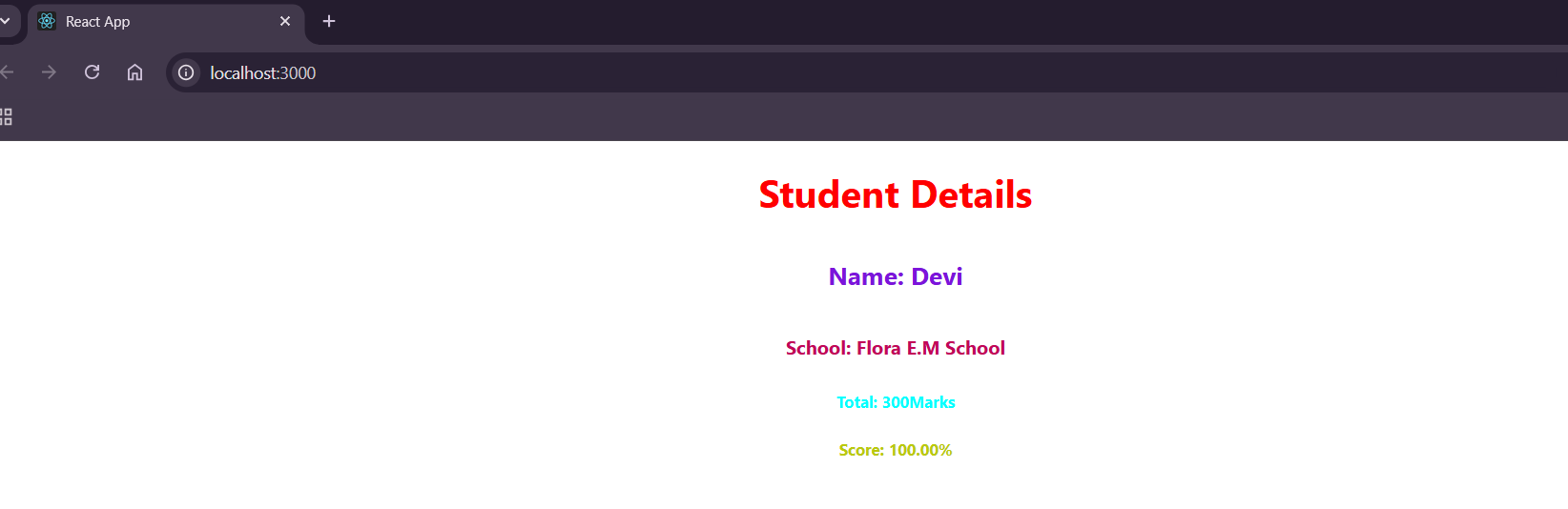
export default App;

4.Run the statement

C:\Users\DELL\scorecaluclatorapp>npm start



**Output:**



**Exercise-4**

**Objectives**

**Need & Benefits of Component Lifecycle**

* Allows you to control what happens when in the component
* Helps in data fetching, event handling, cleanup, and error boundaries.

Ex: render() – Displays UI.

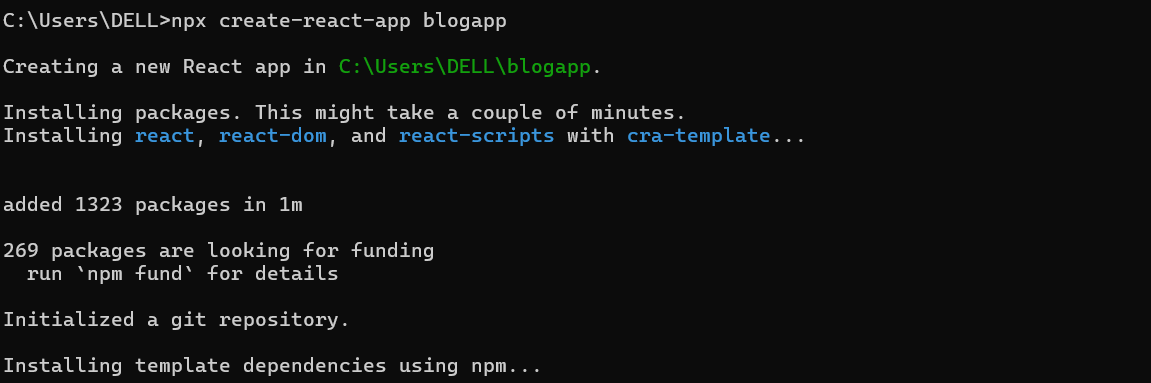
**Identify various life cycle hook methods**

* componentDidMount(): Run logic after component appears.
* componentDidUpdate(): Respond to state/prop changes.
* componentWillUnmount(): Cleanup before component removal.
* componentDidCatch(): Handle errors gracefully.

**Rendering Sequence:**

* Constructor
* Render
* componentDidMount
* Updates (e.g., setState)
* Unmount (if removed from DOM)

1.Create react app named blogapp



2.Open in vs and create the following files

**Post.js:**

import React from 'react';

function Post({ title, body }) {

  return (

    <div className="post">

      <h3>{title}</h3>

      <p>{body}</p>

    </div>

  );

}

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 = () => {

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

      .then((res) => res.json())

      .then((data) => this.setState({ posts: data }))

  };

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

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

  }

  render() {

    const { posts, error } = this.state

    if (error) {

      return <h2>Error loading posts.</h2>;

    }

    return (

      <div>

        <h2>Blog Posts</h2>

        {posts.map((post) => (

          <Post key={post.id} title={post.title} body={post.body} />

        ))}

      </div>

    );

  }

}

export default Posts;

**App.js:**

import React from 'react';

import Posts from './Posts';

function App() {

  return (

    <div className="App">

      <h1>My Blog App</h1>

      <Posts />

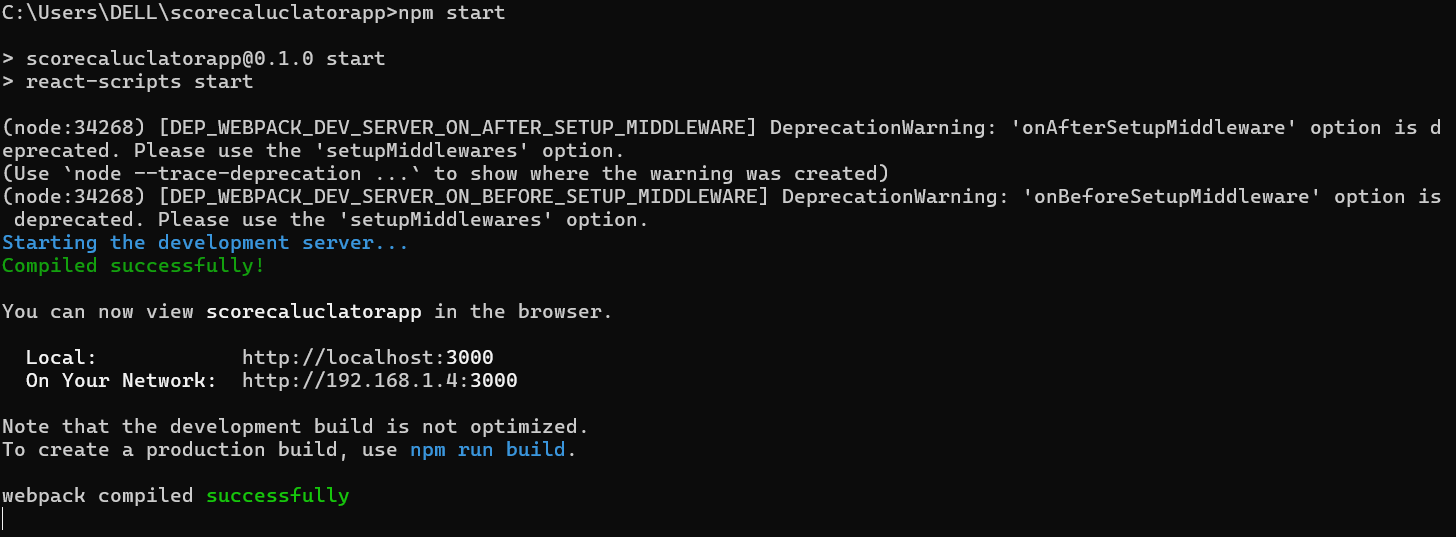
    </div>

  );

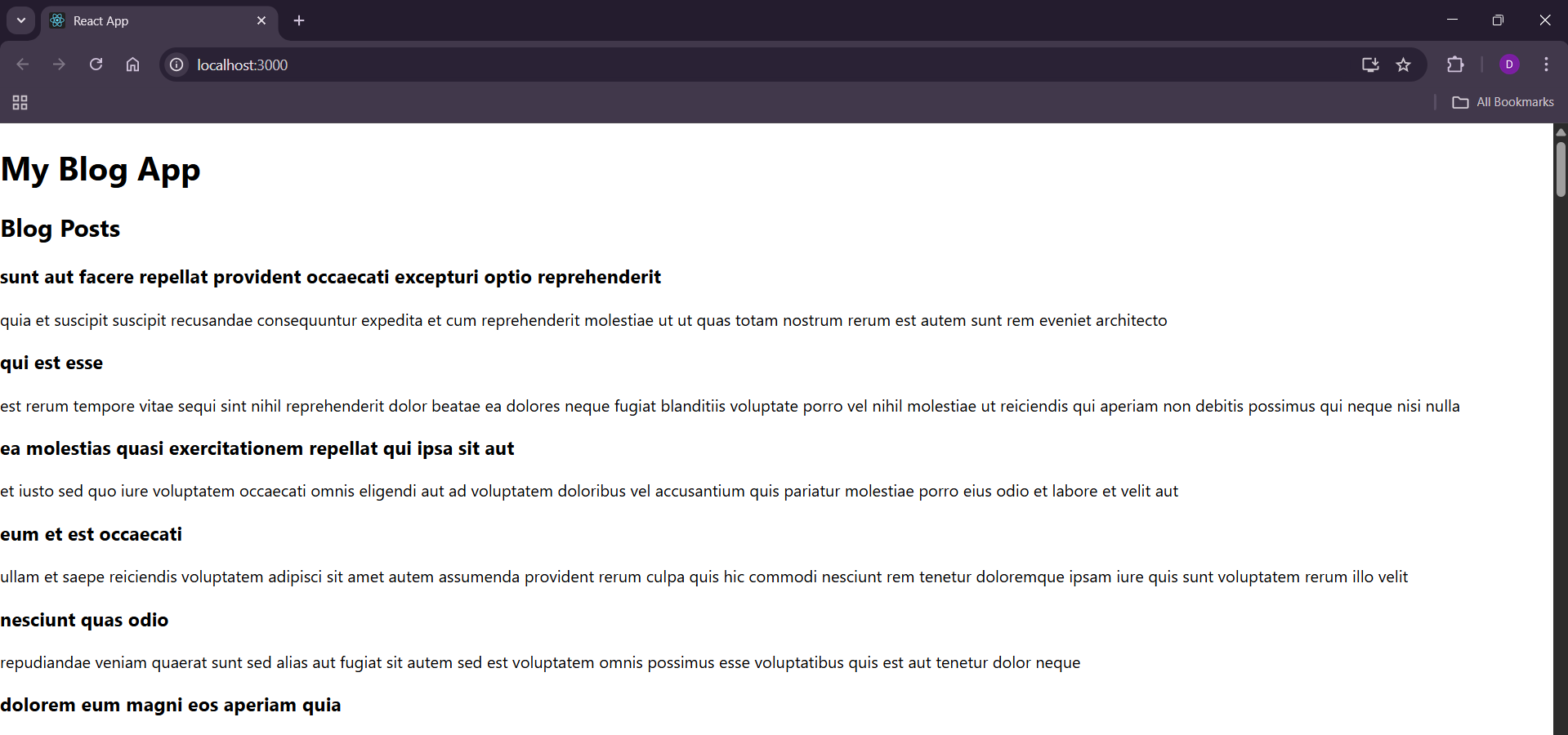
}

export default App;

4.Run the command **npm start**



Output:



**Exercise-5:**

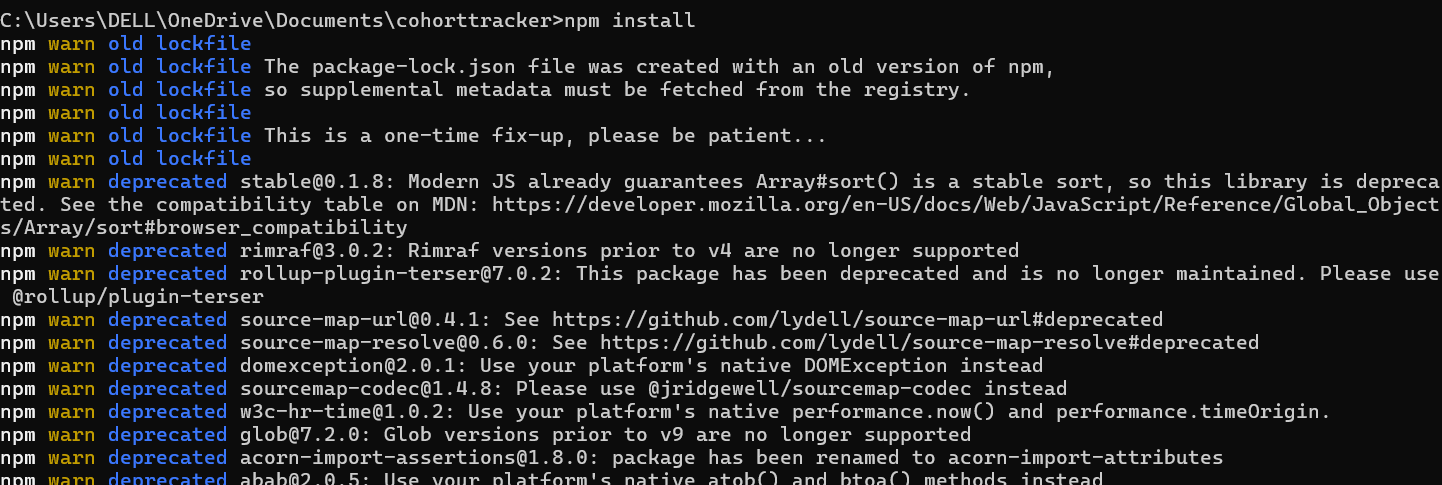
My Academy team at Cognizant want to create a dashboard containing the details of ongoing and completed cohorts. A react application is created which displays the detail of the cohorts using react component. You are assigned the task of styling these react components.

Download and build the attached react application.



1.Double click this file and extract all content

2.Open this file in command prompt and run **npm install**



3.Open this file in vs

4.Create a file as CohortDetails.modules.css

**CohortDetails.modules.css:**

.box {

  width: 300px;

  display: inline-block;

  margin: 10px;

  padding: 10px 20px;

  border: 1px solid black;

  border-radius: 10px;

}

dt {

  font-weight: 500;

}

5.Import this file into CohortDetails.js file as **import './CohartDetails.modules.css';**

6. Add box className to div and modify h3 element as follws

**CohartDetails.js**

import './CohartDetails.modules.css';

function CohortDetails(props) {

    return (

        <div className="box">

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

                {props.cohort.cohortCode} -

                <span>{props.cohort.technology}</span>

            </h3>

            <dl>

                <dt>Started On</dt>

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

                <dt>Current Status</dt>

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

                <dt>Coach</dt>

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

                <dt>Trainer</dt>

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

            </dl>

        </div>

    );

}

export default CohortDetails;

Run npm start command

Output:

