-: 6363213\_WEEK 6 :-

## **-: ReactJS:-**

## **(ReactJS-1) :-**

**Objectives:-**

1. **Define SPA and Its Benefits .**

Single-Page Application (SPA) is a type of web application that loads a single HTML page and dynamically updates content as the user interacts with the app. It does not require reloading the entire page from the server during navigation; instead, JavaScript handles routing and content rendering on the client side.

**Benefits of SPA:-**

**Faster User Experience:-** Only data is exchanged, not full pages, reducing load times.

**Reduced Server Load:-** Fewer HTTP requests for full page reloads.

**Seamless Navigation:-** Transitions between views are smoother without refreshing the page.

**Better Caching:-** Resources are loaded once and cached for future use.

1. **Define React and Identify Its Working.**

React is a JavaScript library developed by Facebook for building user interfaces, especially for SPAs. It allows developers to create reusable UI components that manage their own state.

**Working of React:-**

React builds a component-based architecture, where each UI piece is a component.

It maintains a virtual DOM, an in-memory representation of the real DOM.

When state or props change, React updates the virtual DOM and performs a diffing algorithm to find minimal changes.

Only the necessary parts of the real DOM are updated, ensuring efficient performance.

1. **Identify the Differences Between SPA and MPA .**

| **Aspect** | SPA (Single-Page Application). | MPA (Multi-Page Application) |
| --- | --- | --- |
| **Page Reload** | No full page reload. | Full reload on each navigation. |
| **Speed** | Faster after initial load. | Slower due to repeated page requests. |
| **Server** Interaction | Data fetched via APIs (AJAX/fetch). | Full HTML pages fetched from server. |
| Complexity | More complex client-side logic. | Simpler routing on server-side. |
| SEO Support | Limited (requires extra configuration). | Better by default. |
| Examples | Gmail, Facebook, Twitter. | Amazon, LinkedIn, Government websites. |

1. **Explain Pros & Cons of Single-Page Application.**

**Pros:-**

1. Enhanced user experience with smooth navigation.

II. Efficient data loading via APIs.

III. Reusable UI components.

IV. Better performance due to fewer server requests.

**Cons:-**

I. Poor SEO by default (can be mitigated with SSR).

II. Initial load time may be higher.

III. Requires more client-side logic and development effort.

IV. Browser history and navigation handling can be complex.

1. **Explain About React.**

React is a declarative, component-based JavaScript library primarily used for building modern, interactive web interfaces. It emphasizes building UIs using small, reusable components and managing application state effectively. React allows developers to build scalable front-end applications with improved performance using the virtual DOM and unidirectional data flow.

1. **Define Virtual DOM.**

The Virtual DOM (Document Object Model) is a lightweight JavaScript representation of the real DOM. React maintains this virtual DOM to optimize UI rendering. When changes occur, React compares the new virtual DOM with the previous one (diffing), calculates the minimal updates required, and applies those changes to the real DOM efficiently.

1. **Explain Features of React in Formal Way.**

**Key Features of React:-**

**Component-Based Architecture:-** Encourages modularity by dividing the UI into independent, reusable components.

**JSX Syntax:-** Uses JavaScript XML (JSX), which allows writing HTML-like syntax in JavaScript.

**Virtual DOM:-** Enhances performance by reducing direct manipulation of the real DOM.

**Unidirectional Data Flow:-** Data flows in one direction, making the code easier to debug and manage.

**State Management:-** Components manage their internal state, making them dynamic and interactive.

**Declarative UI:-** Developers describe what the UI should look like; React takes care of rendering it.

**Ecosystem and Tooling:-** Includes tools like React Developer Tools, Create React App, and supports libraries for routing and state management (e.g., React Router, Redux).

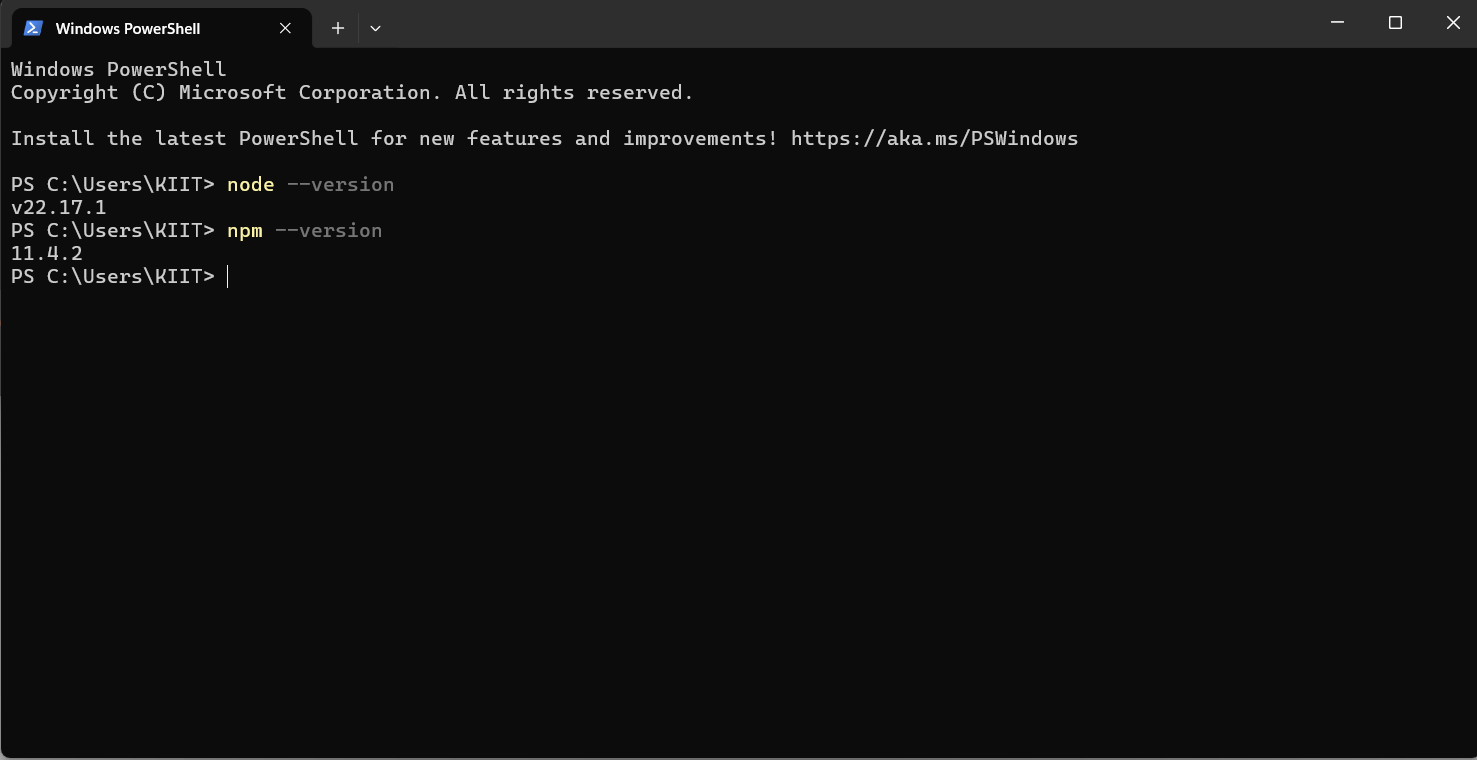
**Create a new React Application with the name “myfirstreact”, Run the application to print “welcome to the first session of React” as heading of that page.**

**-: STEPS :-**

1. **To create a new React app, Install Nodejs and Npm from the following link:-**

<https://nodejs.org/en/download/>

**OUTPUT :-**



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

PS C:\Users\KIIT\OneDrive\Desktop\React Js> npm install -g create-react-app

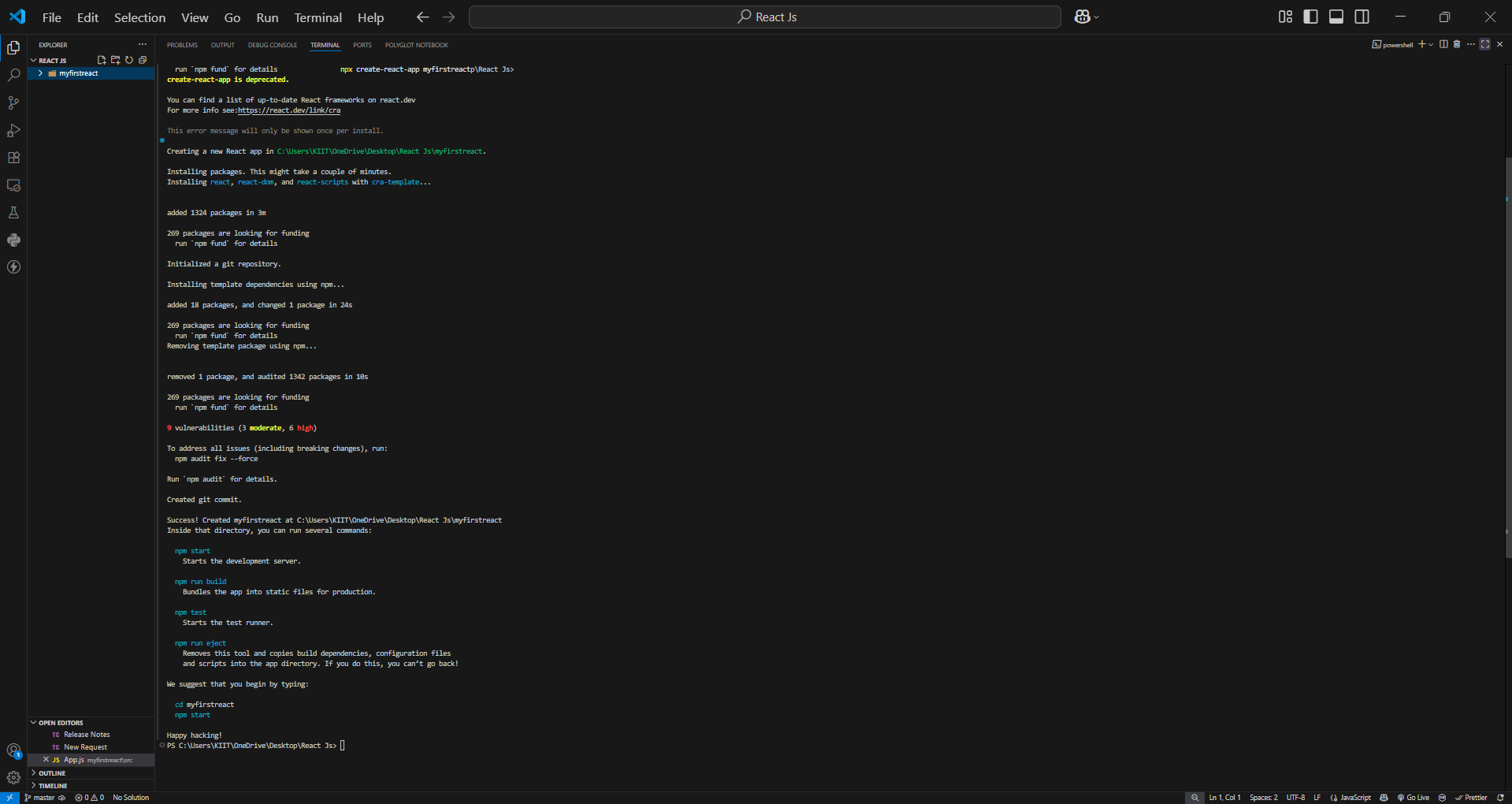
**OUTPUT:-**

****

1. **To create a React Application with the name of “myfirstreact”, type the following command:-**

PS C:\Users\KIIT\OneDrive\Desktop\React Js> npx create-react-app myfirstreact

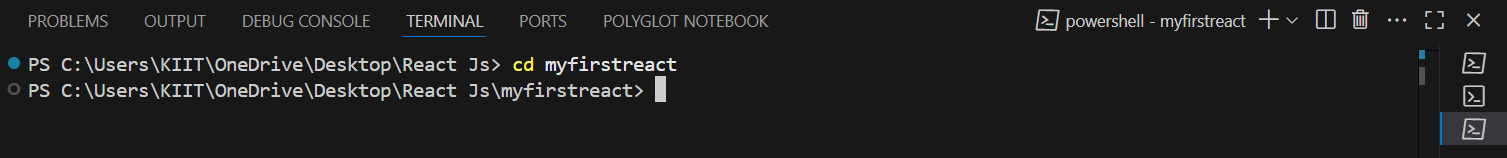
**OUTPUT:-**

****

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

PS C:\Users\KIIT\OneDrive\Desktop\React Js> cd myfirstreact

**OUTPUT:-**



1. **Open the folder of myfirstreact in Visual Studio Code.**
2. **Open the App.js file in Src Folder of myfirstreact.**
3. **Remove the current content of “App.js”.**
4. **Replace it with the following:-**

function App() {

  return (

    <div>

      <h1>Welcome to the first session of React</h1>

    </div>

  );

}

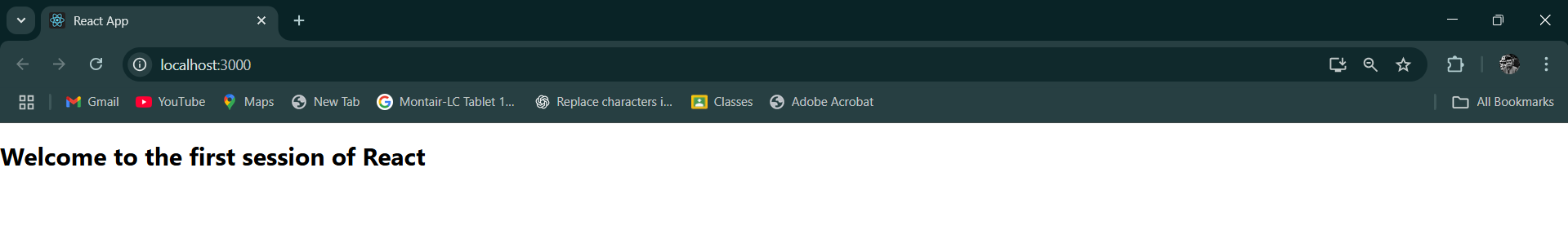
export default App;

1. **Run the following command to execute the React application:-**

PS C:\Users\KIIT\OneDrive\Desktop\React Js\myfirstreact> npm start

1. **Open a new browser window and type “localhost:3000” in the address bar.**

**OUTPUT:-**



## **(ReactJS-2):-**

**Objectives.**

### ****1. Explain React Components.****

React components are the fundamental building blocks of a React application. They are reusable, independent pieces of code that return elements to be rendered on the user interface. Each component encapsulates its own structure, behavior, and styling, enabling modular and maintainable code. Components can accept inputs, called **props**, and manage internal data using **state**.

### ****2. Identify the Differences Between Components and JavaScript Functions.****

While both React components and JavaScript functions may appear syntactically similar, there are key distinctions:

| **Aspect** | **React Components** | **JavaScript Functions** |
| --- | --- | --- |
| Purpose | Designed to return UI elements | Designed to perform generic operations |
| Return Value | JSX (which is rendered to the DOM) | Any data type (string, number, object, etc.) |
| Lifecycle Methods | Available (only in class components) | Not applicable |
| React Hooks | Can use hooks (in function components) | Not applicable |
| Integration with React | Integrates tightly with the React library | Generic; not specific to React |

### ****3. Identify the Types of Components.****

React components are primarily categorized into two types:-

**Class Components** – Components defined using ES6 classes.

**Function Components** – Components defined using JavaScript functions (including arrow functions).

### ****4. Explain Class Component.****

A **class component** in React is a component defined using the class keyword. It must extend React.Component and include a render() method that returns JSX. Class components support lifecycle methods and state management.

**Example:-**

jsx

CopyEdit

class Welcome extends React.Component {

  constructor(props) {

    super(props);

    this.state = { message: "Hello" };

  }

  render() {

    return <h1>{this.state.message}, {this.props.name}</h1>;

  }

}

### ****5. Explain Function Component.****

A **function component** is a simpler way to write components using JavaScript functions. Introduced initially as stateless components, they can now manage state and lifecycle using **React Hooks** (e.g., useState, useEffect).

**Example:-**

jsx

CopyEdit

function Welcome(props) {

  return <h1>Hello, {props.name}</h1>;

}

**Or using hooks:-**

jsx

CopyEdit

import { useState } from 'react';

function Welcome(props) {

  const [message, setMessage] = useState("Hello");

  return <h1>{message}, {props.name}</h1>;

}

### ****6. Define Component Constructor.****

In class components, the **constructor** is a special method used for initializing the component's **state** and binding event handlers. It is called before the component is mounted and must include a call to super(props) to access this.props.

**Syntax:-**

jsx

CopyEdit

constructor(props) {

  super(props);

  this.state = { count: 0 };

}

### ****7. Define render() Function.****

The render() function is a required method in every class component. It determines what the component should display. It must return JSX or null. The render() method is called during the mounting and updating phases of the component lifecycle.

**Syntax:-**

jsx

CopyEdit

render() {

  return (

    <div>

      <h1>Hello, world!</h1>

    </div>

  );

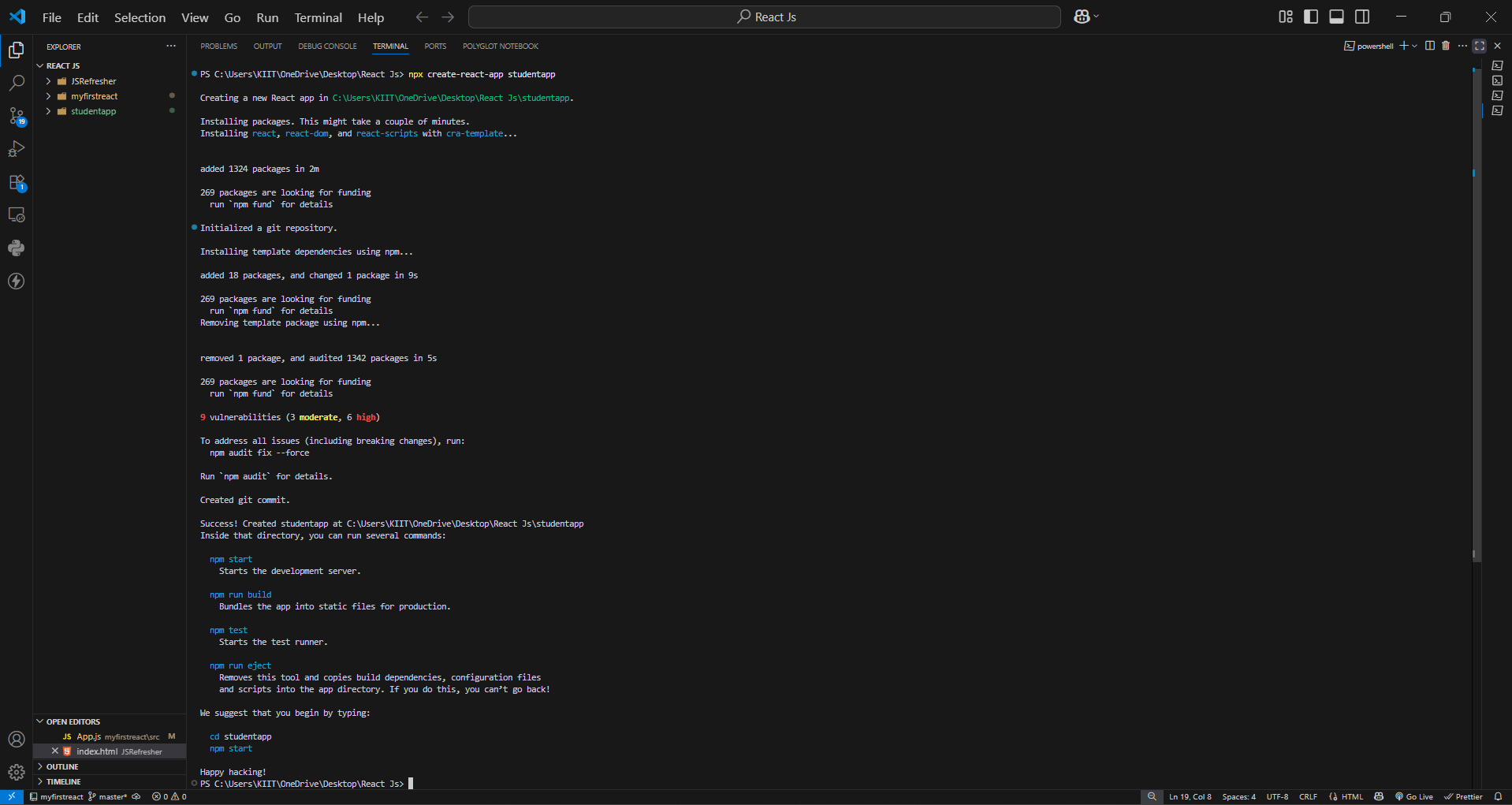
}

**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.**

1. **Create a React project named “StudentApp” type the following command in terminal of Visual studio:-**

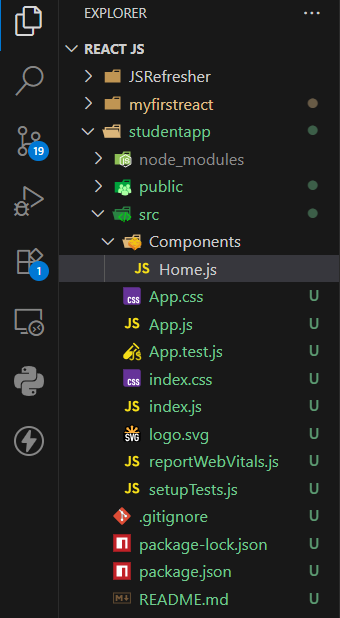
PS C:\Users\KIIT\OneDrive\Desktop\React Js> npx create-react-app studentapp

**OUTPUT:-**



1. **Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js”.**

**OUTPUT:-**



1. **Type the following code in Home.js.**

**CODE:-**

import React from 'react';

function Home() {

  return (

    <div>

      <h2>Welcome to the Home page of Student Management Portal</h2>

    </div>

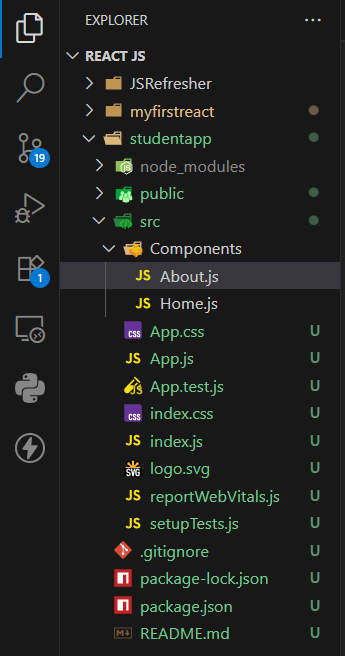
  );

}

export default Home;

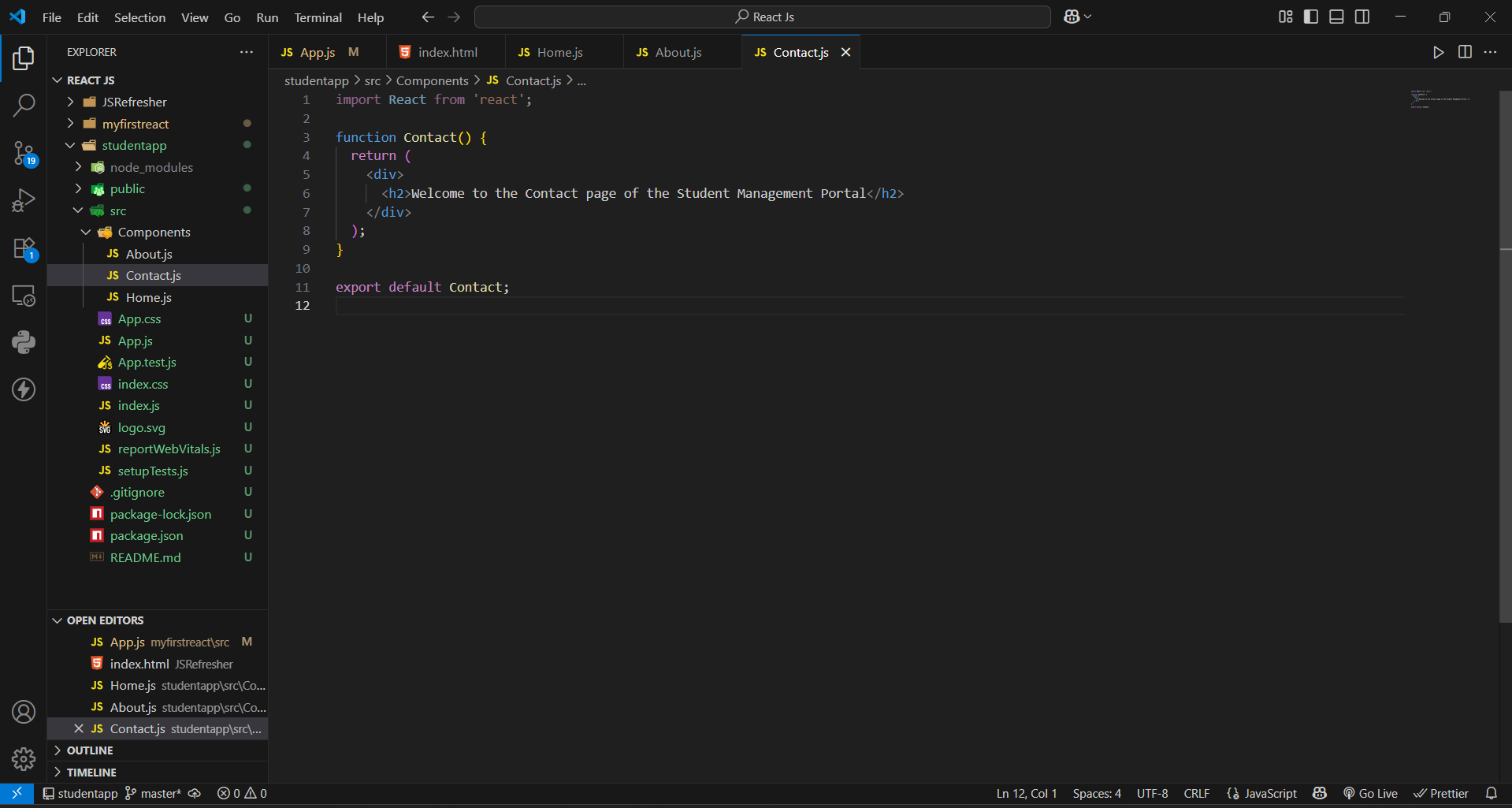
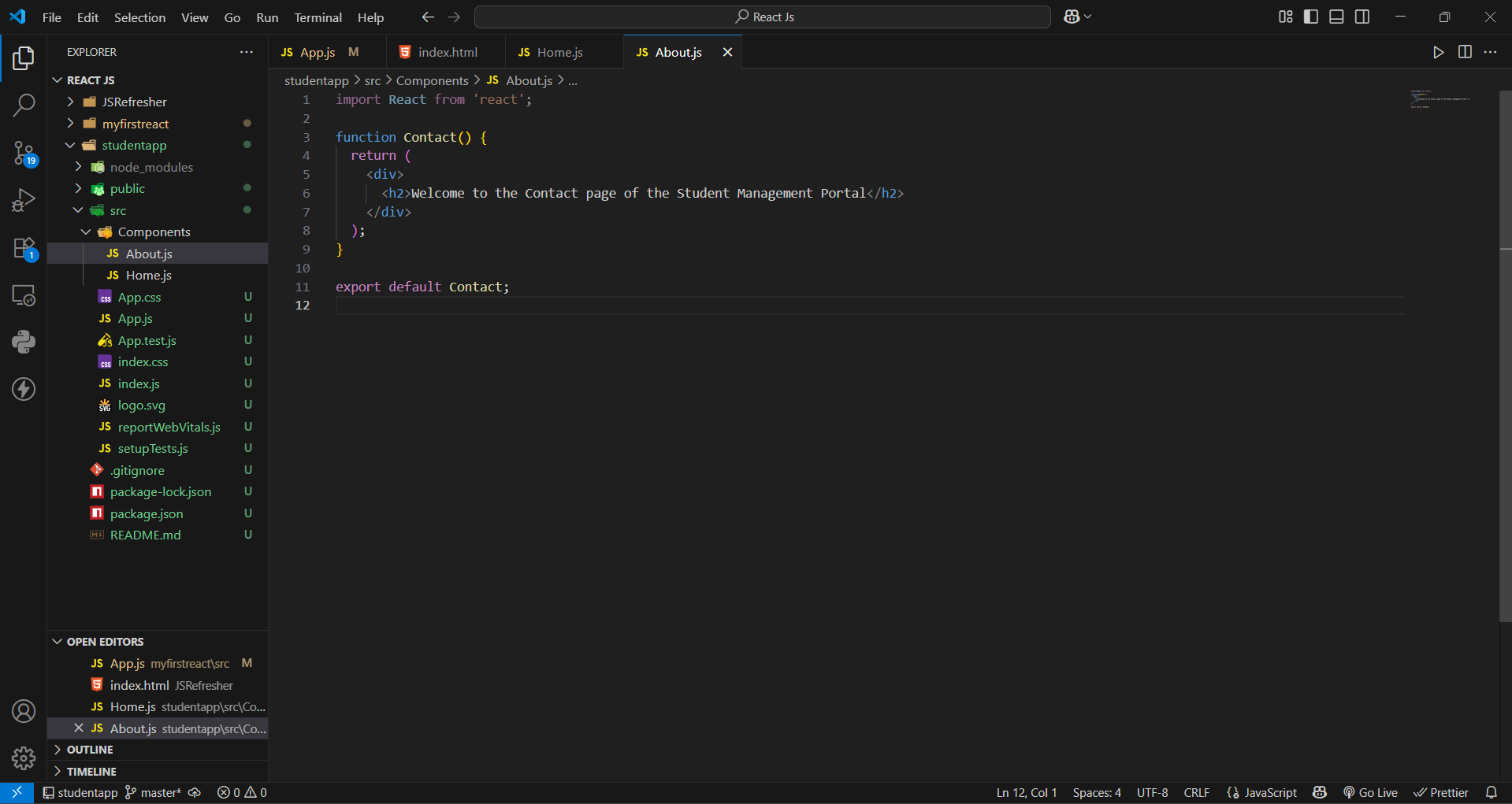
1. **Under Src folder add another file named “About.js”.**

**OUTPUT:-**



1. **Repeat the same steps for Creating “About” and “Contact” component by adding a new file as ”About.js”, “Contact.js” under “Src” folder and edit the code as mentioned for “Home” Component.**

**OUTPUT:-**



1. **Edit the App.js to invoke the Home, About and Contact component as follows:-**

**CODE:-**

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

  return (

    <div className="App">

      <Home />

      <About />

      <Contact />

    </div>

  );

}

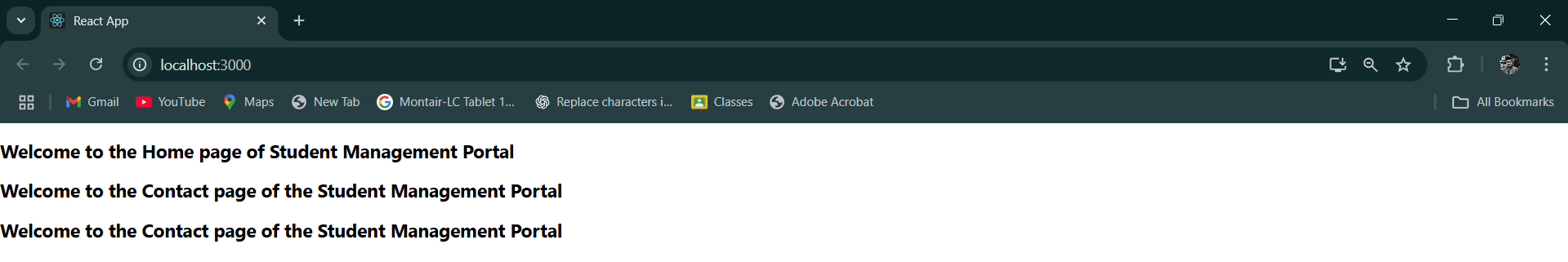
export default App;

1. **In command Prompt, navigate into StudentApp and execute the code by typing the following command:-**

PS C:\Users\KIIT\OneDrive\Desktop\React Js\studentapp> npm start

1. **Open browser and type “localhost:3000” in the address bar:-**

**OUTPUT:-**



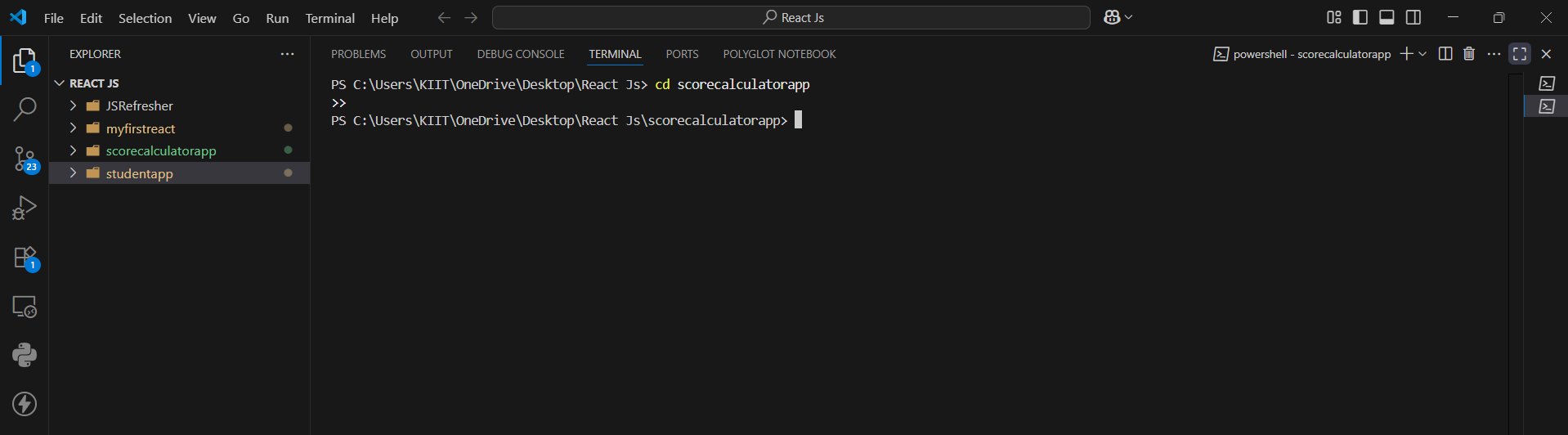
**(ReactJS - 3) :-**

**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:-**

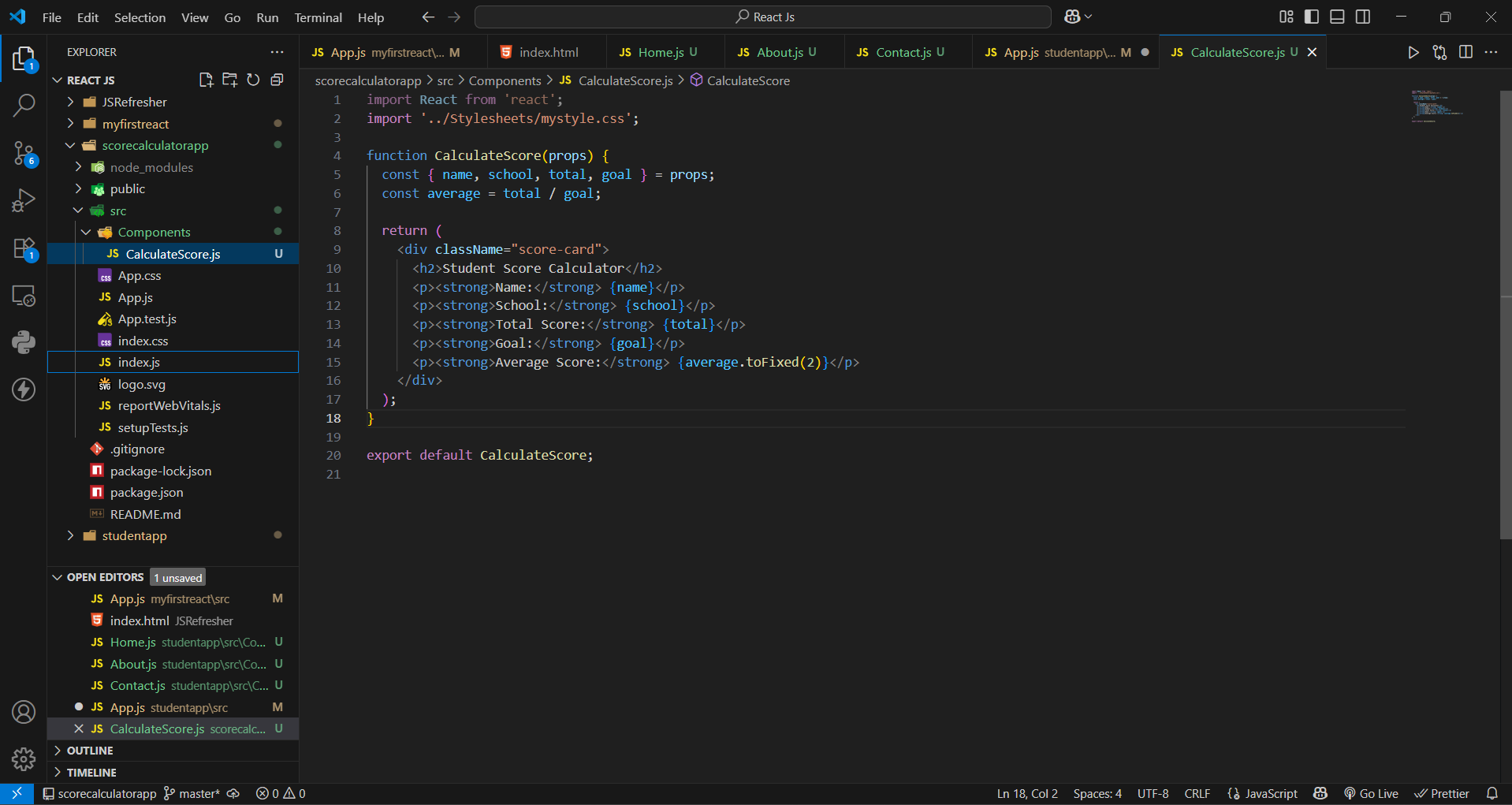
1. **Create a React project named “scorecalculatorapp” type the following command in terminal of Visual studio:-**

**OUTPUT:-**



1. **Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”.**

**OUTPUT:-**



1. **Type the following code in CalculateScore.js.**

**CODE :-**

import React from 'react';

import './Stylesheets/mystyle.css';

const percentToDecimal = (decimal) => {

  return (decimal.toFixed(2) + '%');

}

const calcScore = (total, goal) => {

  return percentToDecimal(total / goal);

}

export const CalculateScore = ({ Name, School, total, goal }) => (

  <div className="formatstyle">

    <h1><font color="Brown">Student Details:</font></h1>

    <div className="Name">

      <b><span> Name: </span></b>

      <span>{Name}</span>

    </div>

    <div className="School">

      <b><span> School: </span></b>

      <span>{School}</span>

    </div>

    <div className="Total">

      <b><span>Total:</span></b>

      <span>{total}</span>

      <span> Marks</span>

    </div>

    <div className="Score">

      <b>Score:</b>

      <span>

        {calcScore(

          total,

          goal

        )}

      </span>

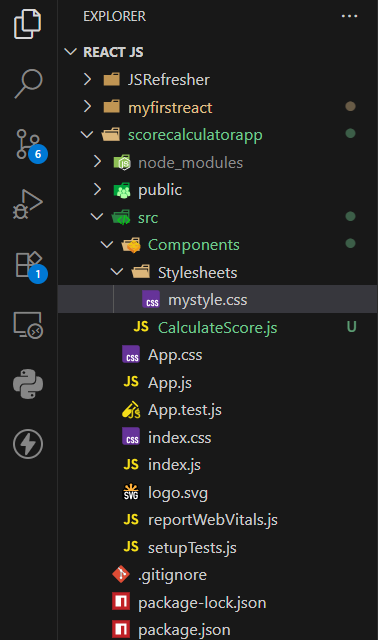
    </div>

  </div>

);

1. **Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components:-**

**OUTPUT:-**



**CODE :-**

.Name

{

    font-weight: 300;

    color: blue;

}

.School

{

    color: crimson;

}

.Total

{

    color: darkmagenta;

}

.formatstyle

{

    text-align: center;

    font-size: large;

}

.Score

{

    color: forestgreen;

}

1. **Edit the App.js to invoke the CalculateScore functional component as follows:-**

**CODE:-**

import { CalculateScore } from '../src/Components/CalculateScore';

function App()

{

  return (

    <div>

      <CalculateScore Name={"Steeve"}

                      School={"DNV Public School"}

                      total={284}

                      goal={3}

      />

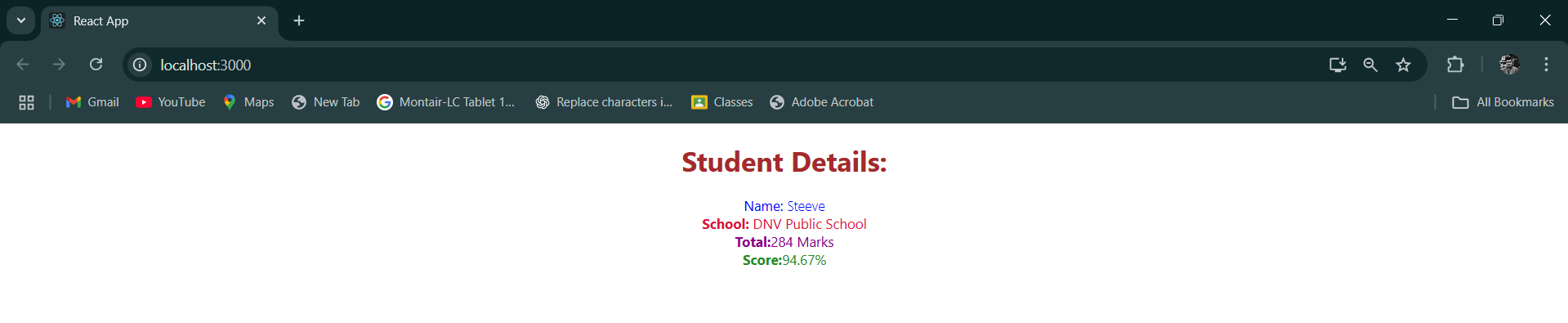
    </div>

  );

}

export default App;

**OUTPUT:-**



**(ReactJS - 4).**

**Objectives.**

1. ****Explain the Need and Benefits of Component Life Cycle.****

The component life cycle in frameworks such as Angular or React refers to the series of events or stages a component undergoes from its creation to its destruction. Understanding the component life cycle is crucial because it provides developers with strategic hook points to execute custom logic during different phases, such as initialization, change detection, rendering, and cleanup.

**Benefits include:-**

* Efficient resource management (e.g., subscribing and unsubscribing to services)
* Controlled DOM updates and manipulation.
* Performance optimization by reducing unnecessary re-renders.
* Better debugging and maintenance through predictable behavior.

1. ****Identify Various Life Cycle Hook Methods .****

In Angular, for instance, the following are key life cycle hook methods:-

* **ngOnChanges()** – Called when an input-bound property changes.
* **ngOnInit()** – Invoked once upon component initialization.
* **ngDoCheck()** – Detects and acts upon changes that Angular doesn’t detect on its own.
* **ngAfterContentInit()** – Called after content (ng-content) is projected into the component.
* **ngAfterContentChecked()** – Called after the projected content is checked.
* **ngAfterViewInit()** – Called after the component’s view (and child views) are initialized.
* **ngAfterViewChecked()** – Called after the component’s view has been checked.
* **ngOnDestroy()** – Called just before the component is destroyed.

1. ****List the Sequence of Steps in Rendering a Component.****

The typical sequence in Angular for rendering a component is:-

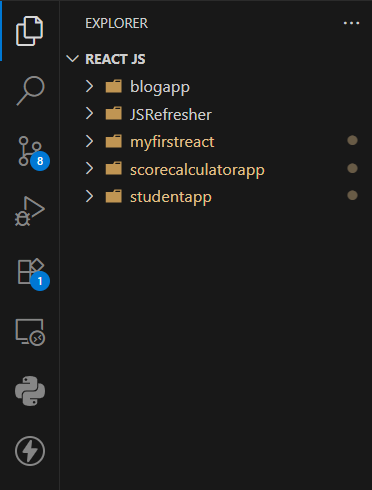
Constructor execution :-

* **ngOnChanges()** (if input properties are present)
* **ngOnInit()**
* **ngDoCheck()**
* **ngAfterContentInit()**
* **ngAfterContentChecked()**
* **ngAfterViewInit()**
* **ngAfterViewChecked()**
* **ngOnDestroy()** (when the component is removed)

**-: STEPS :-**

1. **Create a new react application using *create-react-app* tool with the name as “blogapp”.**

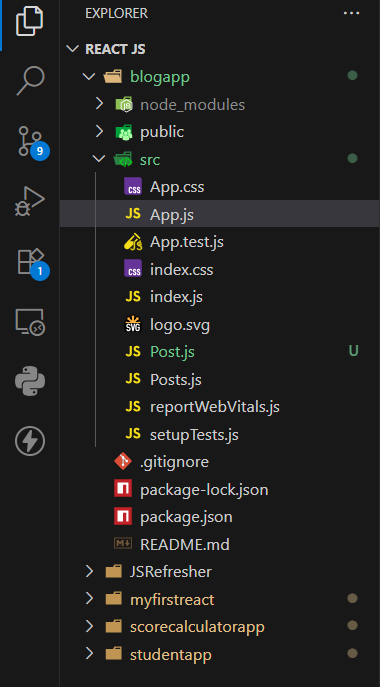
**OUTPUT:-**



1. **Open the application using VS Code.**
2. **Create a new file named as Post.js in src folder with following properties.**

1. **Create a new class based component named as Posts inside Posts.js file.**

**OUTPUT:-**



1. **Initialize the component with a list of Post in state of the component using the constructor.**
2. **Create a new method in component with the name as loadPosts() which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>).**
3. **Implement the componentDidMount() hook to make calls to loadPosts() which will fetch the posts.**
4. **Implement the render() which will display the title and post of posts in html page using heading and paragraphs respectively.**
5. **Define a componentDidCatch() method which will be responsible for displaying any error happing in the component as alert messages.**
6. **Add the Posts component to App component.**

**CODE :-**

// src/Posts.js

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

      hasError: false

    };

  }

  loadPosts() {

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

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

      .then(data => {

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

        this.setState({ posts: postObjects });

      })

      .catch(error => {

        this.setState({ hasError: true });

        alert("Failed to load posts: " + error.message);

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

    alert("An error occurred in the component: " + error.toString());

  }

  render() {

    return (

      <div>

        <h1>Blog Posts</h1>

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

          <div key={post.id}>

            <h2>{post.title}</h2>

            <p>{post.body}</p>

            <hr />

          </div>

        ))}

      </div>

    );

  }

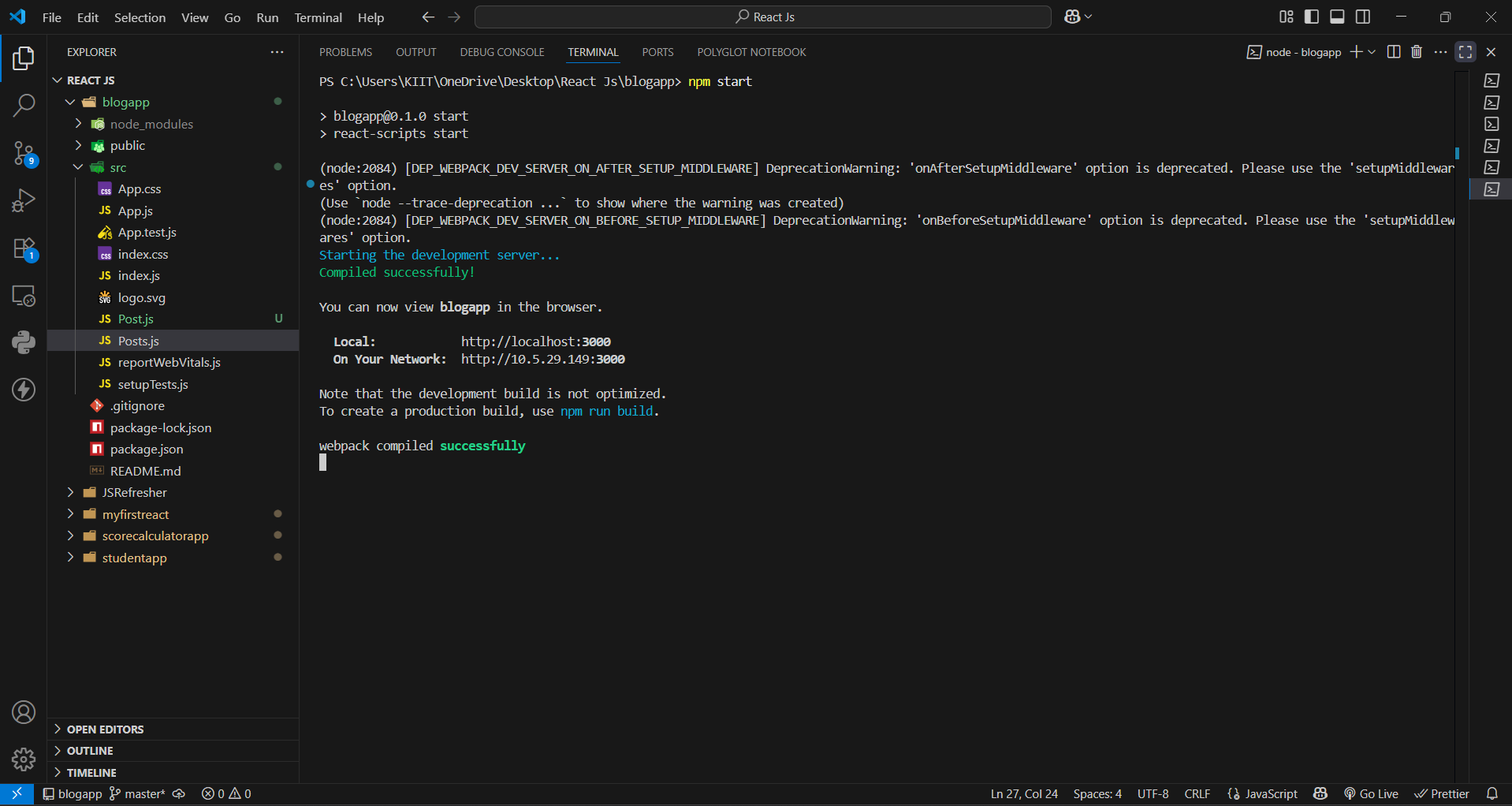
}

export default Posts;

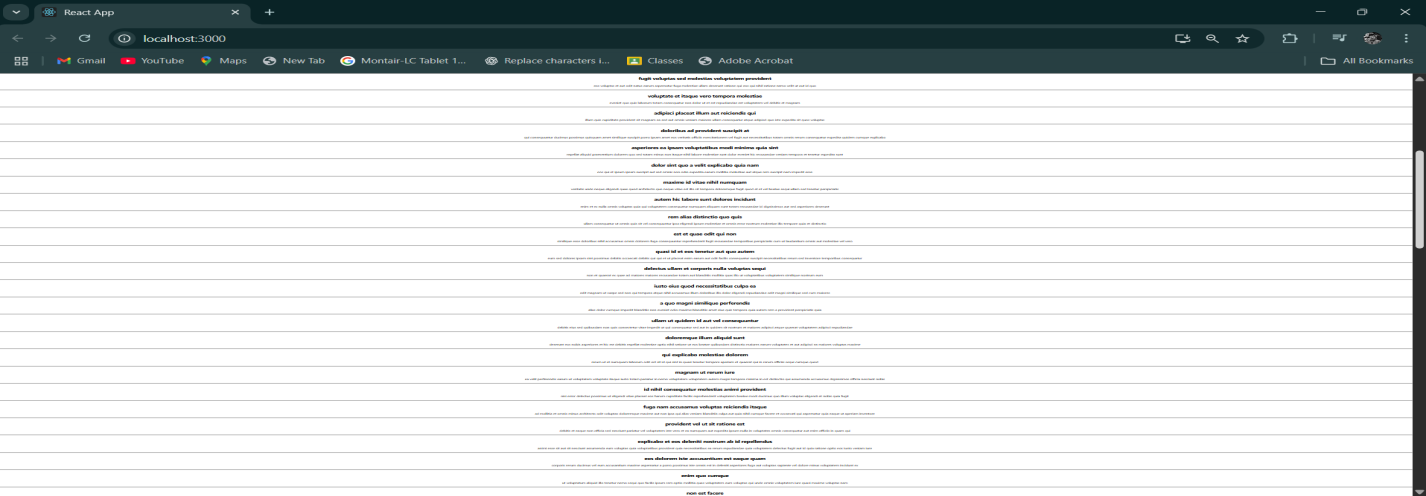
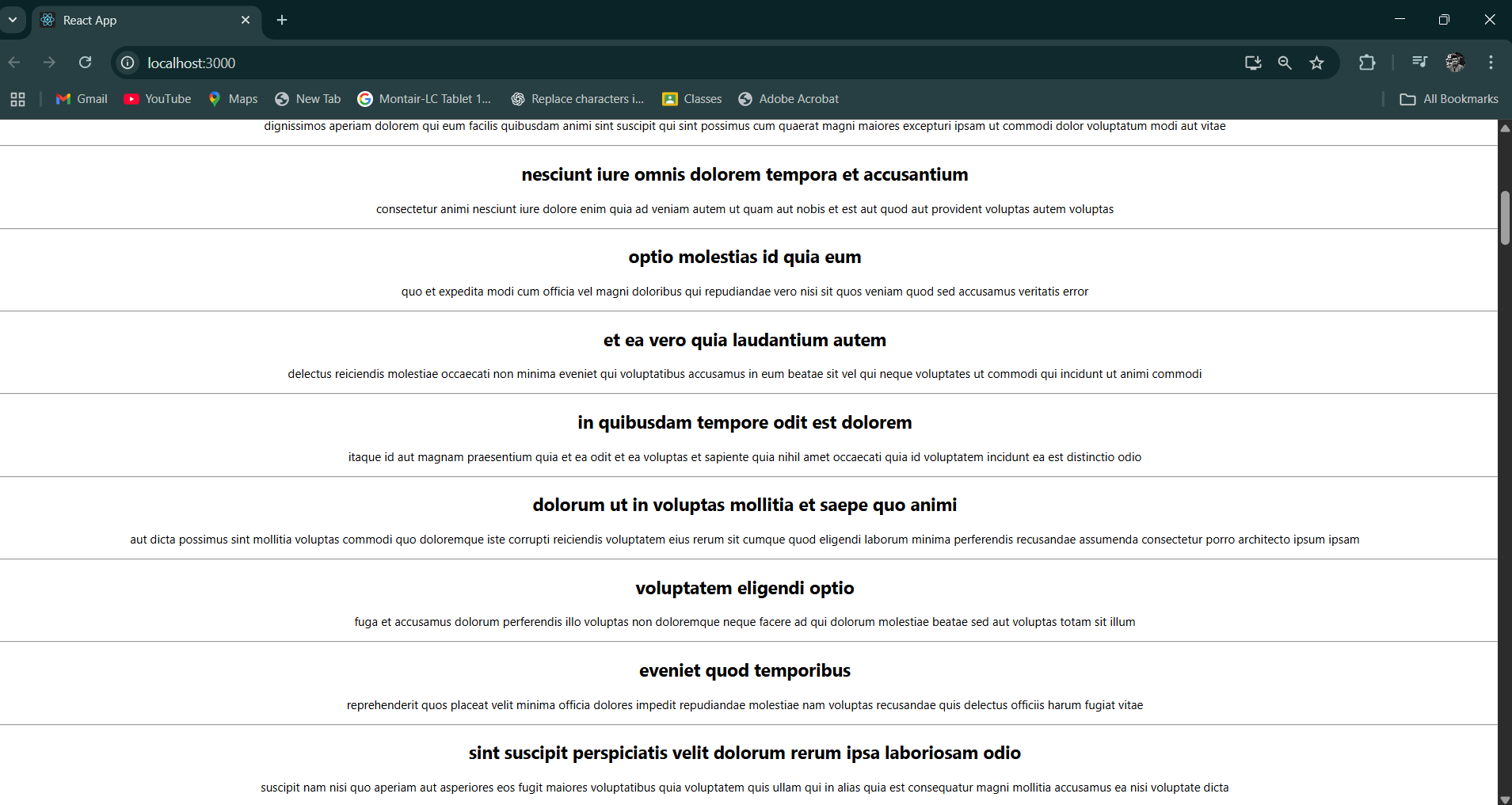
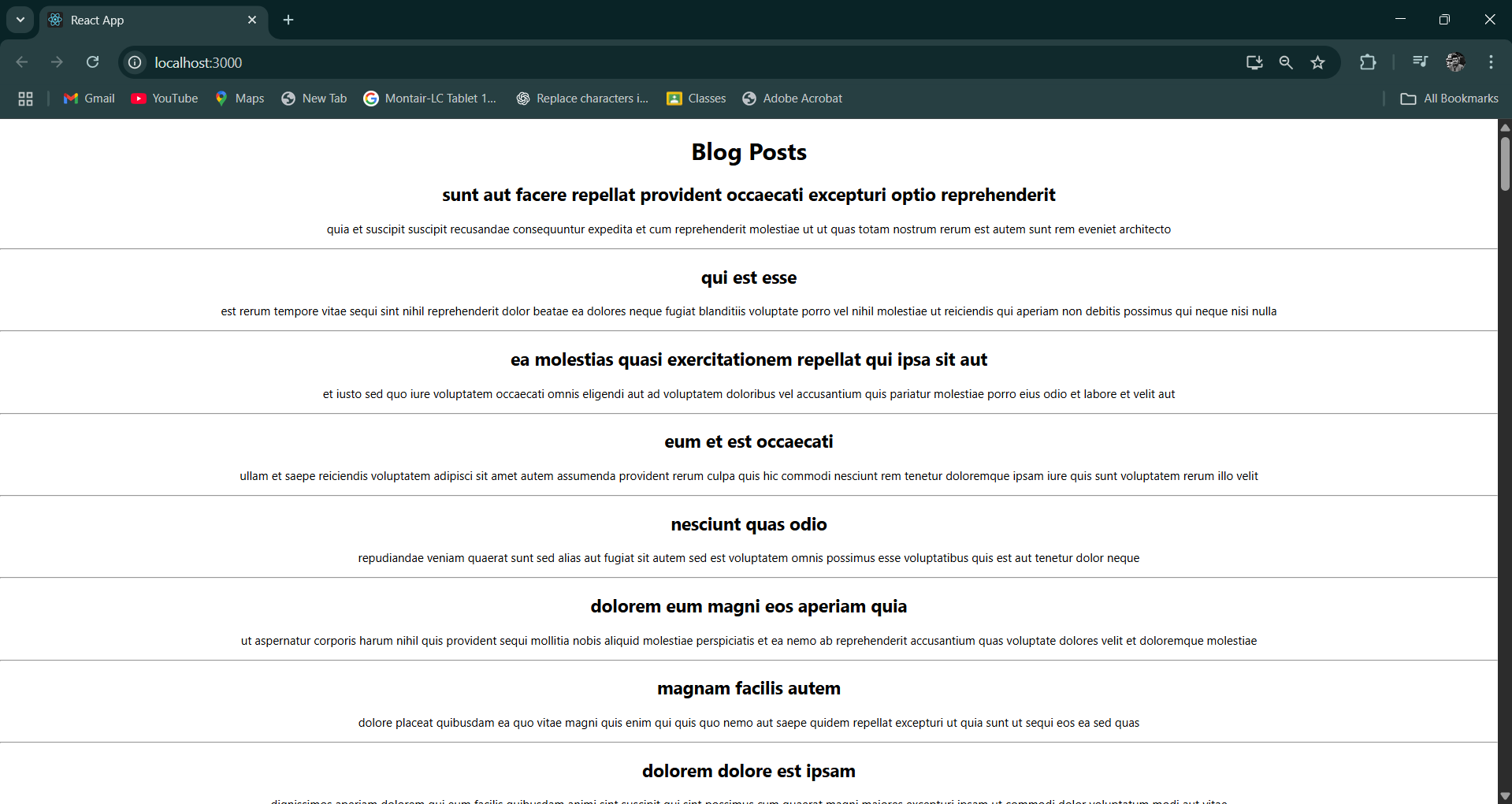
1. **Build and Run the application using *npm start* command.**

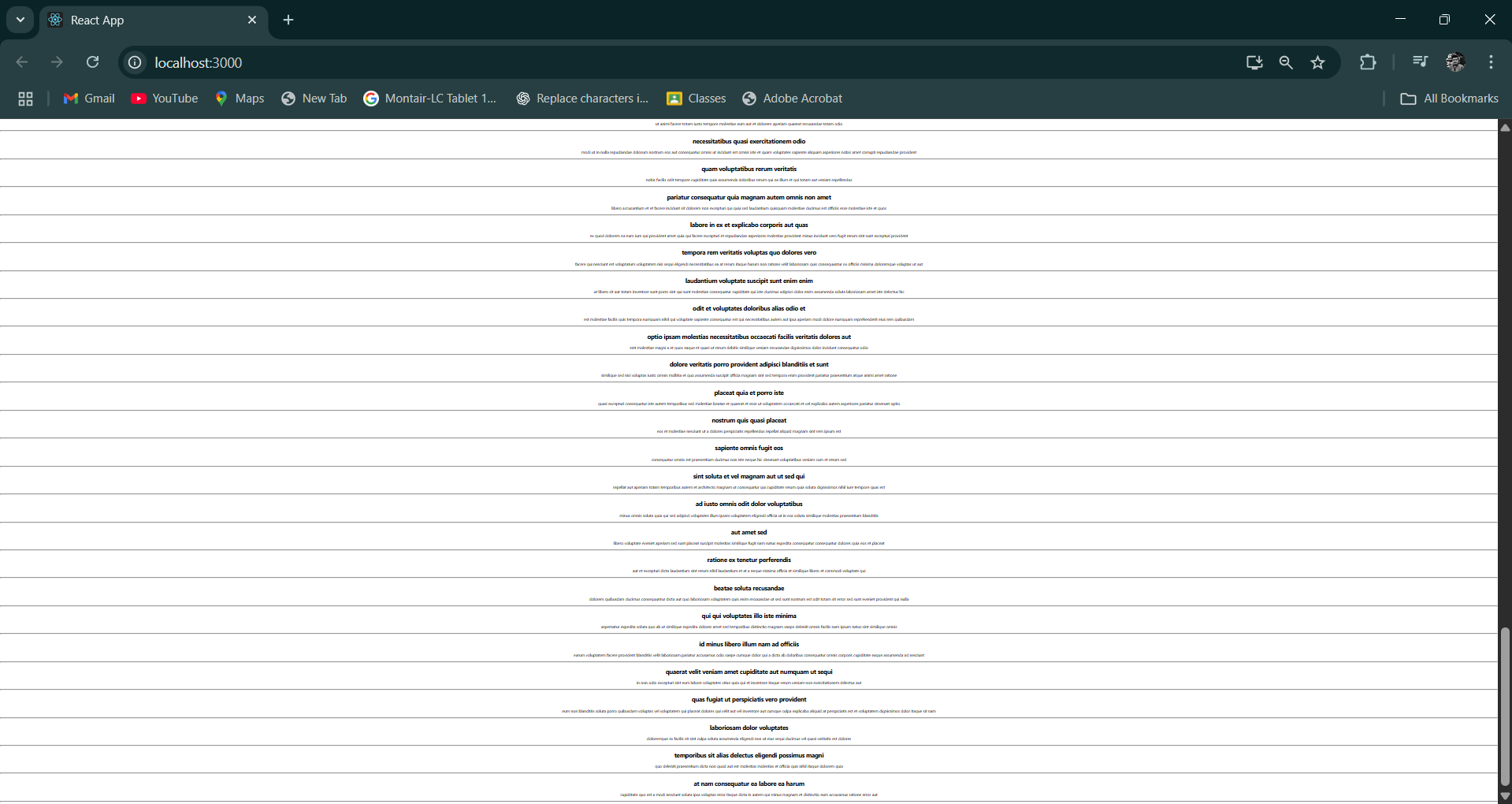
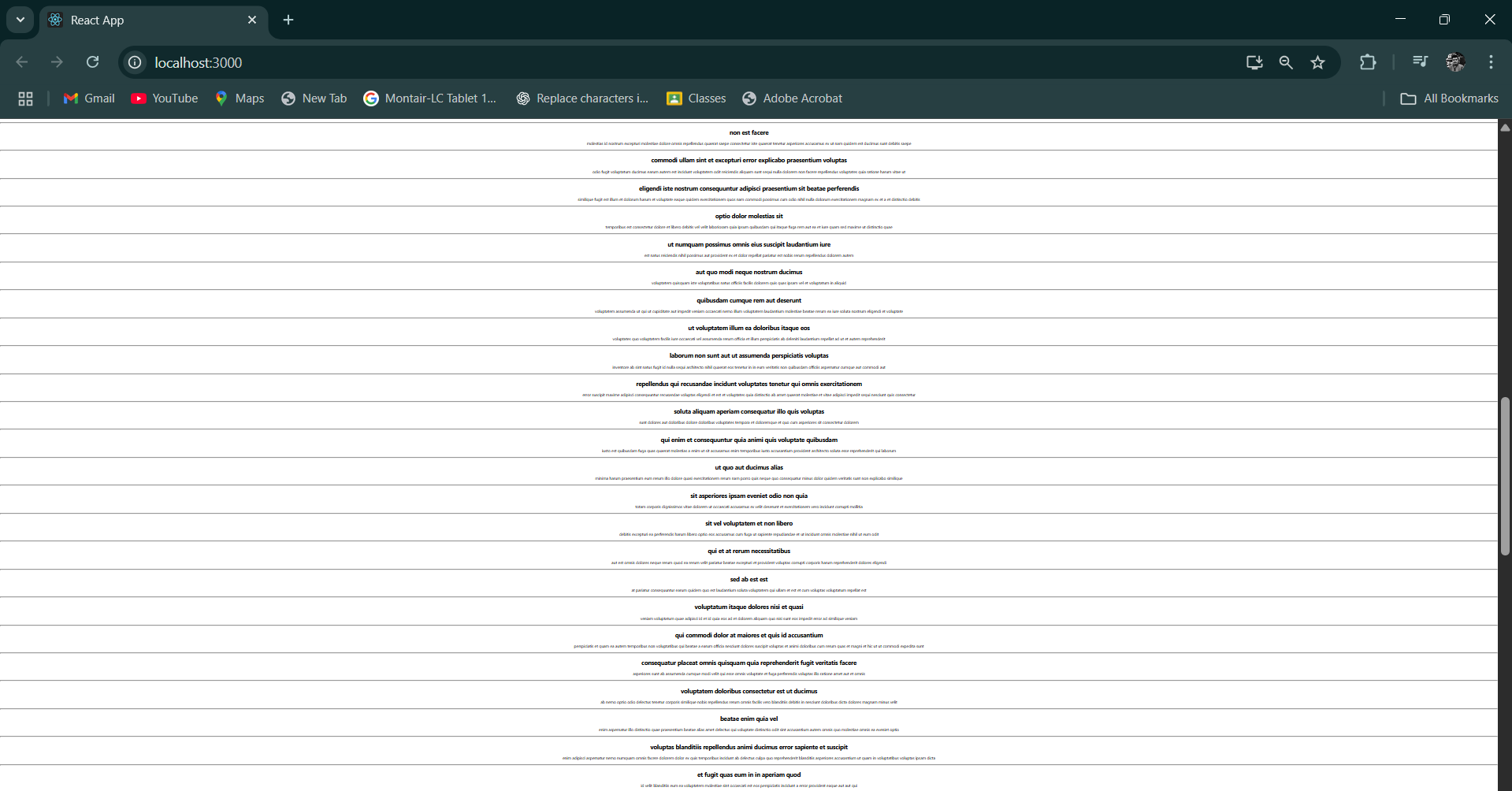
**CODE :-**

PS C:\Users\KIIT\OneDrive\Desktop\React Js\blogapp> npm start



**OUTPUT:-**





#### (ReactJS-5):-

**Objectives.**

#### 1. ****Understanding the Need for Styling React Components.****

* To improve the **visual appearance** and **user experience** of the app.
* Helps create **consistent UI** across components.
* Enables **separation of concerns** (logic and presentation).
* Makes components more **readable, maintainable, and reusable**.
* Allows **dynamic styling** based on props or state (e.g., error messages in red, success in green).

#### 2. ****Working with CSS Module and Inline Styles.****

* **CSS Modules**:-

Locally scoped CSS to avoid global naming conflicts.

Example:- import styles from './MyComponent.module.css'

Use: <div className={styles.container}></div>

* **Inline Styles**:-

Directly apply styles using JavaScript objects.

Useful for **dynamic styling** and **small tweaks**.

Example:-

Jsx:-

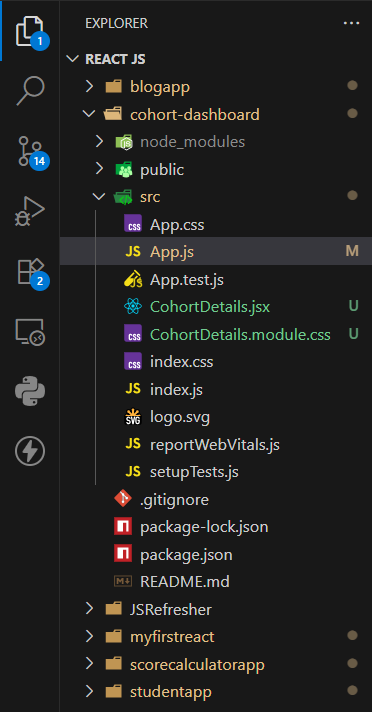
<div style={{ color: 'blue', fontSize: '18px' }}>Hello</div

**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.**

**-: STEPS :-**

1. **Unzip the react application in a folder.**
2. **Open command prompt and switch to the react application folder.**
3. **Restore the node packages using the following commands.**
4. **Open the application using VS Code.**
5. **Create a new CSS Module in a file called “CohortDetails.module.css”.**

**OUTPUT:-**



1. **Define a css class with the name as “box” with following properties.**

**CODE:-**

.box {

  width: 300px;

  display: inline-block;

  margin: 10px;

  padding: 10px 20px;

  border: 1px solid black;

  border-radius: 10px;

}

1. **Define a css style for html <dt> element using tag selector. Set the font weight to 500.**

**CODE:-**

dt {

  font-weight: 500;

}

1. **Open the cohort details component and import the CSS Module.**

**CODE:-**

import React from 'react';

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

1. **Apply the box class to the container div.**

**CODE:-**

<div className={styles.box}>

      <h3 style={{ color: titleColor }}>{cohort.name}</h3>

      <dl>

        <dt>Started On</dt>

        <dd>{cohort.start}</dd>

        <dt>Current Status</dt>

        <dd>{cohort.status}</dd>

        <dt>Coach</dt>

        <dd>{cohort.coach}</dd>

        <dt>Trainer</dt>

        <dd>{cohort.trainer}</dd>

      </dl>

    </div>

1. **Define the style for <h3> element to use “green” color font when cohort status is “ongoing” and “blue” color in all other scenarios.**

**CODE:-**

function CohortDetails({ cohort }) {

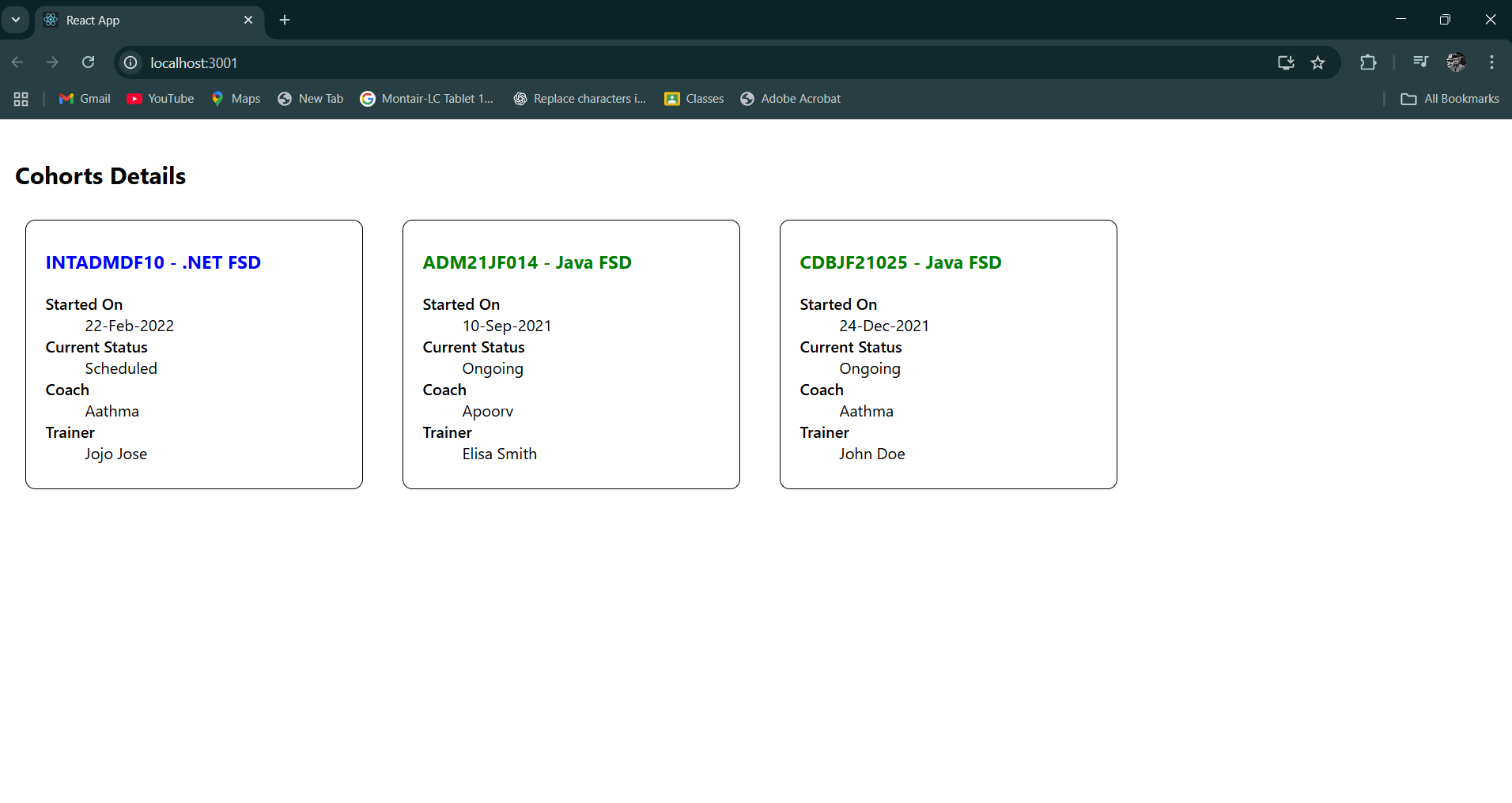
  const titleColor =

    cohort.status === 'Ongoing' ? 'green' : cohort.status === 'Scheduled' ? 'blue' : 'black';

<h3 style={{ color: titleColor }}>{cohort.name}</h3>

1. **Final result should look similar to the below image.**

**OUTPUT:-**



vbbb