**Week 6 \_React**

**Question:1**

**Project Title: myfirstreact  
Goal: Display "Welcome to the first session of React" on the home page using React.**

**React:**

React is an open-source JavaScript library used for building fast, interactive user interfaces (UI) in web applications and follows a component-based architecture.

**Single-Page Application (SPA):**

A Single-Page Application is a web app that dynamically rewrites the current page rather than loading entire new pages from the server. React is one of the most popular libraries used to create SPAs.

**Use React for SPAs:**

* Faster rendering via Virtual DOM
* Reusable UI components
* Real-time updates without full page reload
* Smooth user experience

**Step 1: Create the React App**

In the terminal of VS Code:

npx create-react-app myfirstreact

* npx → runs the tool temporarily without global install
* create-react-app → scaffolds a React project
* myfirstreact → project folder name

Output:

* 1300+ packages installed
* Git initialized
* Default folder structure generated

**Folder Structure Overview**

After creation:

**myfirstreact/**

node\_modules/ # All installed packages

public/ # Static assets like index.html

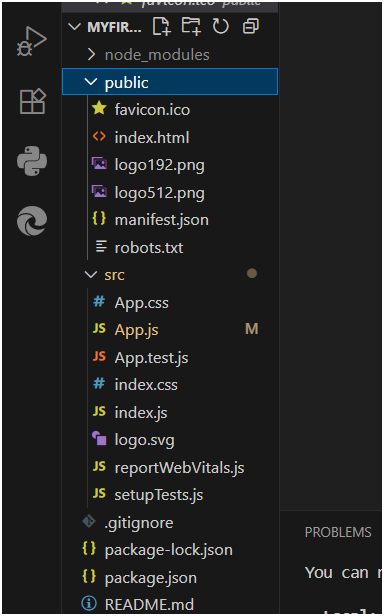
* index.html

src/ # Source folder containing React code

* App.js # Main component
* index.js # Entry point

package.json # Project config & dependencies

* README.md # Instructions



**Step 2: Navigate to the Project Directory**

cd myfirstreact

**Step 3: Open Project in VS Code**

Open File → Open Folder.

**Step 4: Modify the App.js File**

Go to src → App.js

import React from 'react';

function App() {

return (

<div>

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

</div>

);

}

export default App;

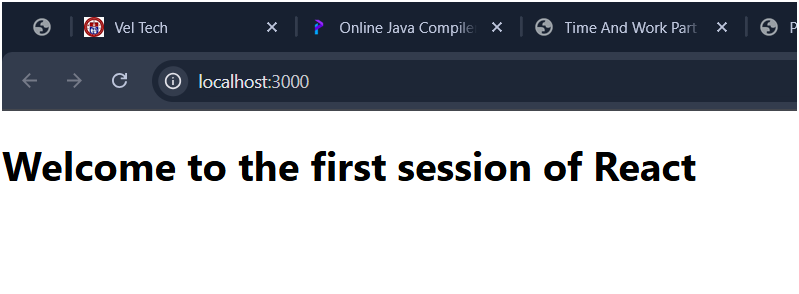
**Code Explanation:**

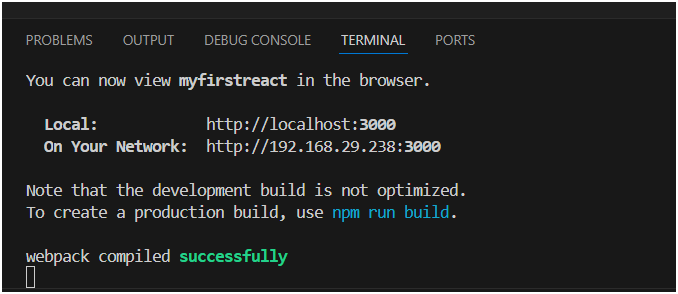
* import React → Required for JSX to work
* function App() → Defines a functional component
* <h1>...</h1> → React element returned as part of UI
* export default App → Exposes component to be used in index.js

**Step 5: Run the React App**

npm start

* Starts local development server
* Opens browser automatically at <http://localhost:3000>





**How It Works (Flow of working) :**

A[App.js - Defines UI] 🡪 B[index.js - Entry Point]

B 🡪 C[Virtual DOM]

C 🡪 D[Real DOM Update]

D 🡪 E[Browser Output]

**Question 2**

**Student Management Portal using Class Components**

Objectives

* Explain React components
* Identify the differences between components and JavaScript functions
* Identify the types of components
* Explain class components
* Explain function components
* Define the component constructor
* Define the render() function

**Step 1: Create a New React App**

In the terminal or command prompt, run:

npx create-react-app student-management-portal

**Step 2: Open the Project in VS Code**

cd student-management-portal

code .

**Step 3: Create a Components Folder**

Inside the src/ folder, create a new folder named:

Components

**Step 4: Create the Home Component**

**src/Components/Home.js**

import React from 'react';

function Home() {

return (

<div>

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

</div>

);

}

export default Home;

**Step 5: Create the About Component**

src/Components/About.js

import React from 'react';

function About() {

return (

<div>

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

</div>

);

}

export default About;

**Step 6: Create the Contact Component**

src/Components/Contact.js

import React from 'react';

function Contact() {

return (

<div>

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

</div>

);

}

export default Contact;

**Step 7: Edit the App.js File**

src/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 className="App">

<Home />

<About />

<Contact />

</div>

);

}

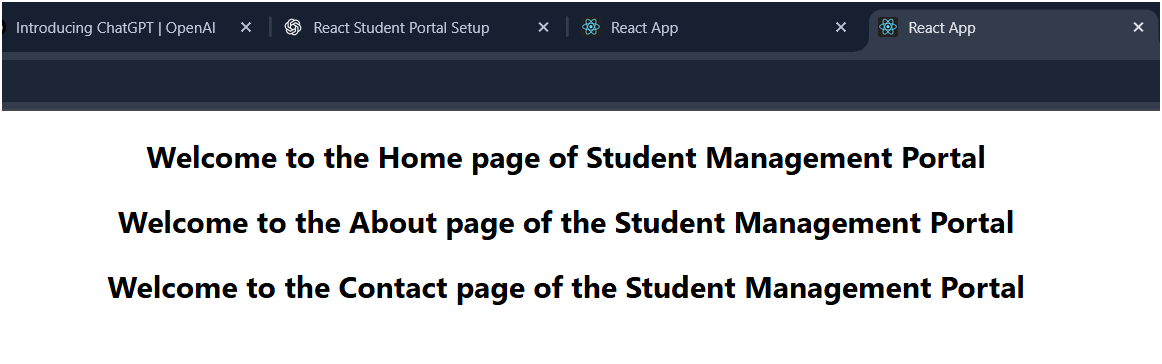
export default App;

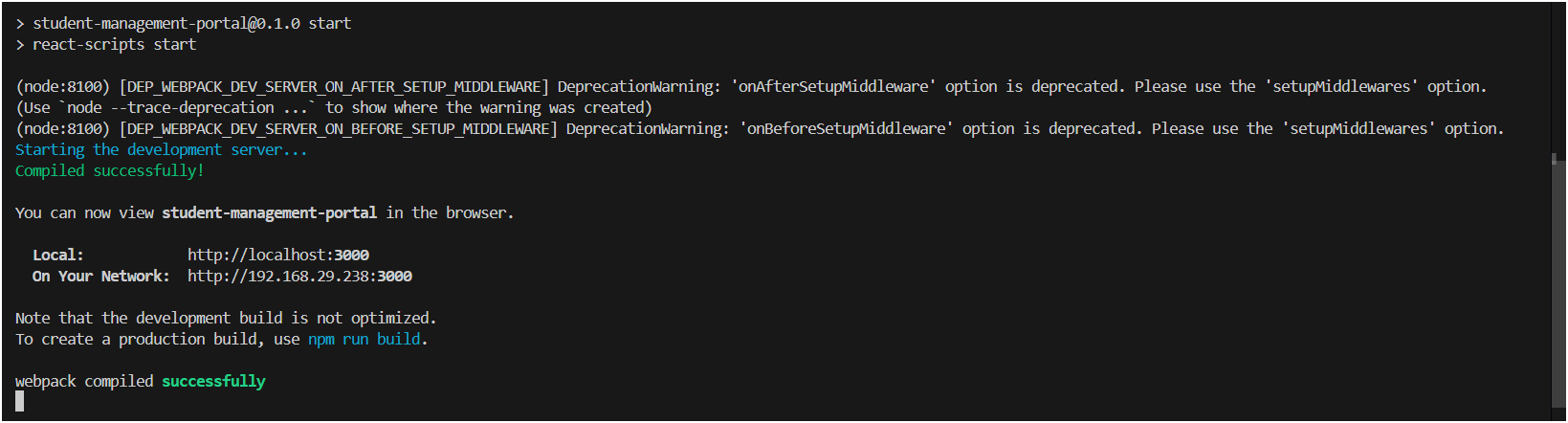
**Step 8: Start the React Server**

In terminal:

npm start

<http://localhost:3000>





**Question 3**

**Student Management Portal - Score Calculator App**

**Project Name: scorecalculatorapp**

**Objectives**

* Understand React function components.
* Learn how function components differ from JavaScript functions.
* Learn to style React components using external CSS.
* Calculate and render student score-related data using props.

**Step 1: Create React App**

Open terminal and run the following command:

npx create-react-app scorecalculatorapp

**Step 2: Create Component Folder and File**

Inside the src directory:

1. Create a folder named Components
2. Inside Components, create a file named: CalculateScore.js

**CalculateScore.js:**

import React from 'react';

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

function CalculateScore(props) {

const { name, school, total, goal } = props;

const average = total / goal;

return (

<div className="score-card">

<h2>Student Score Report</h2>

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

<p><strong>School:</strong> {school}</p>

<p><strong>Total Marks:</strong> {total}</p>

<p><strong>Number of Subjects:</strong> {goal}</p>

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

</div>

);

}

export default CalculateScore;

**Step 3: Add Styles using External CSS**

Under src,

Folder - Stylesheets

File - mystyle.css

**mystyle.css:**

.score-card {

background-color: #f9f9f9;

border: 2px solid #0077cc;

padding: 20px;

border-radius: 12px;

width: 400px;

margin: 20px auto;

box-shadow: 2px 4px 10px rgba(0, 0, 0, 0.1);

}

.score-card h2 {

color: #0077cc;

text-align: center;

}

.score-card p {

font-size: 1.1rem;

line-height: 1.6;

}

**Step 4: Update App.js**

**CalculateScore component:**

import React from 'react';

import './App.css';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div className="App">

<h1>Welcome to the Student Score Calculator Portal</h1>

<CalculateScore name="Deepika" school="ABC School" total={475} goal={5} />

</div>

);

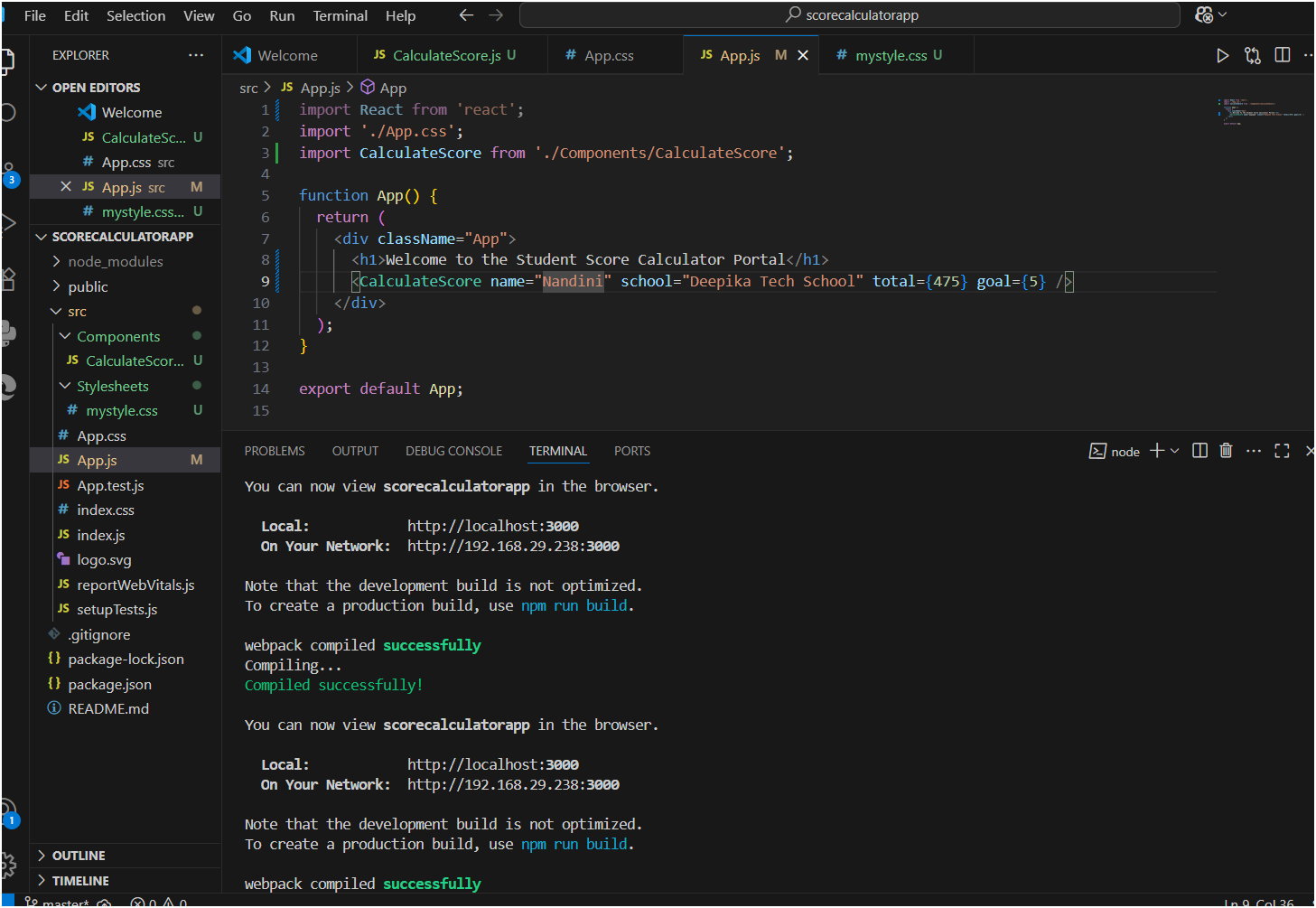
}

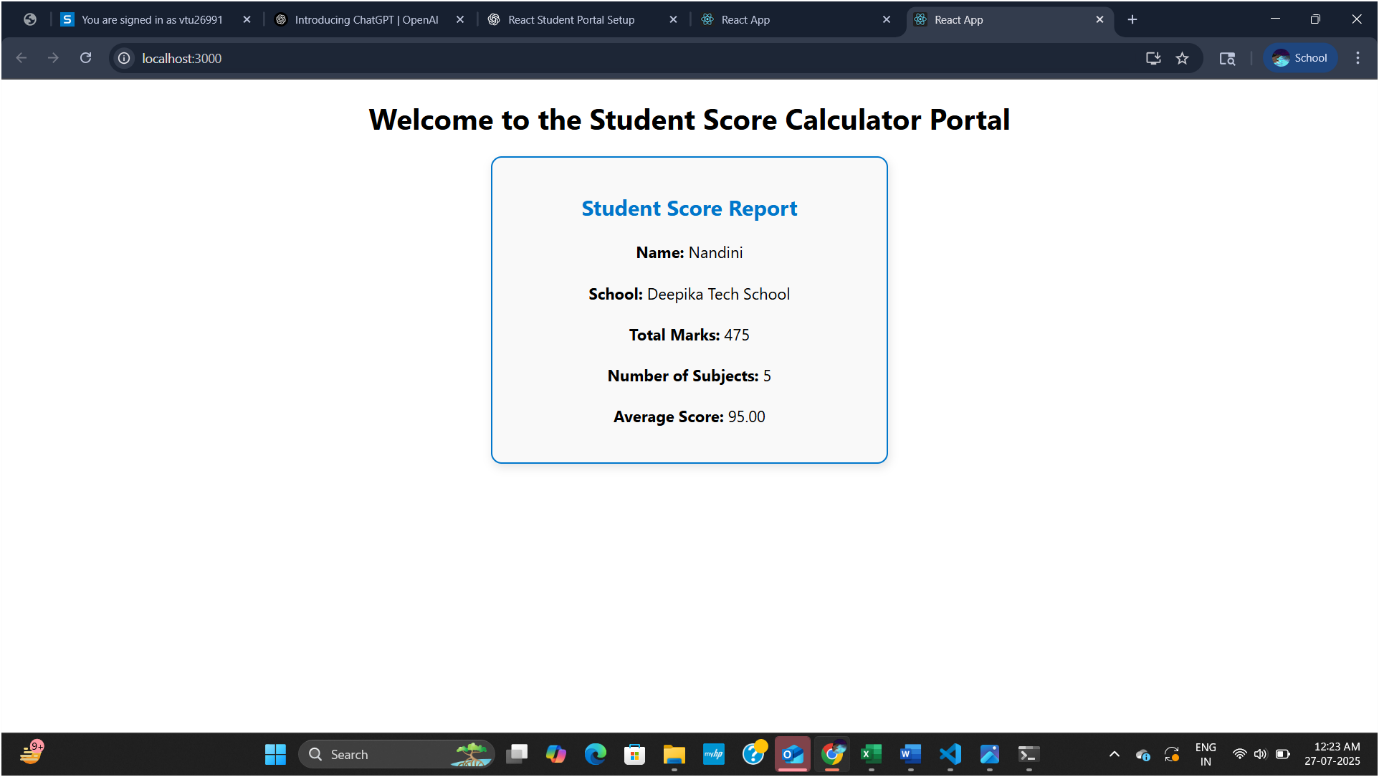
export default App;

**Step 5: Run Your App**

In terminal, run:

**npm start**





**Question 4**

**Project Title: My Blog App**

**Deepika's Blog Space – A Simple Blog Viewer Built with React**

**Objectives**

* Understand the importance of the React component lifecycle.
* Learn to implement componentDidMount() and componentDidCatch() lifecycle hooks.
* Fetch and display blog post data dynamically.
* Display custom error handling using lifecycle methods.
* Use stateful class components to manage blog data.

**1. Create React App**

npx create-react-app blogapp

cd blogapp

code .

**2. Create Post.js**

**File**: src/Post.js

import React from "react";

const Post = ({ title, body }) => {

return (

<div className="p-4 bg-white shadow-md rounded-lg mb-4">

<h2 className="text-xl font-bold text-blue-600">{title}</h2>

<p className="text-gray-700">{body.substring(0, 100)}...</p>

</div>

);

};

export default Post;

**3. Create Posts.js**

**File**: src/Posts.js

import React, { Component } from "react";

import Post from "./Post";

class Posts extends Component {

constructor(props) {

super(props);

this.state = {

posts: [

{ id: 1, title: "My First Blog", body: "This is my very first blog post!" },

{ id: 2, title: "Why I Love Fullstack", body: "Fullstack development makes building applications exciting and complete." },

{ id: 3, title: "NCC Life", body: "Discipline, adventure, and leadership — all in one!" },

],

error: null,

};

}

componentDidMount() {

// Simulated data fetch, could be replaced with real API later

this.loadPosts();

}

loadPosts = () => {

try {

// Simulating async behavior

this.setState({

posts: [...this.state.posts]

});

} catch (error) {

this.setState({ error });

}

};

componentDidCatch(error, info) {

alert("An error occurred while rendering the posts.");

console.error("Error boundary caught an error:", error, info);

}

render() {

const { posts } = this.state;

return (

<div className="p-6 bg-gray-100 min-h-screen">

<h1 className="text-3xl font-bold text-center mb-6">📚 My Blog App</h1>

<h2 className="text-2xl text-purple-700 font-semibold text-center mb-4">✨ Deepika's Blog Space ✨</h2>

<h3 className="text-xl font-medium mb-4">📝 Blog Posts</h3>

{posts.map((blog) => (

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

))}

</div>

);

}

}

export default Posts;

we can also use the url to get the posts (<https://jsonplaceholder.typicode.com/posts>) but I haven’t used this to create my own blog posts and added manually in posts.js

**4. Update App.js**

**File**: src/App.js

import React from "react";

import Posts from "./Posts";

function App() {

return (

<div>

<Posts />

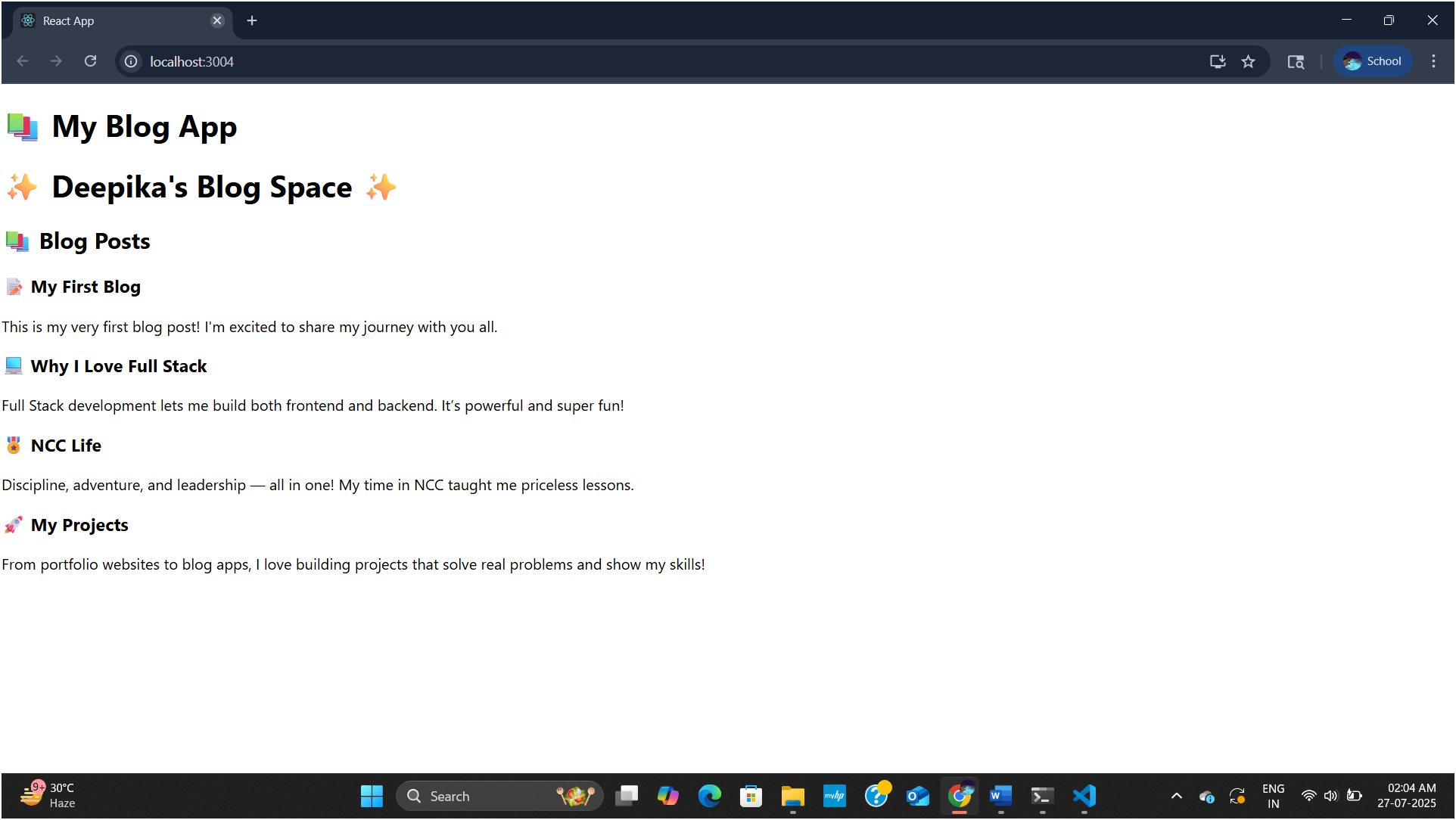
</div>

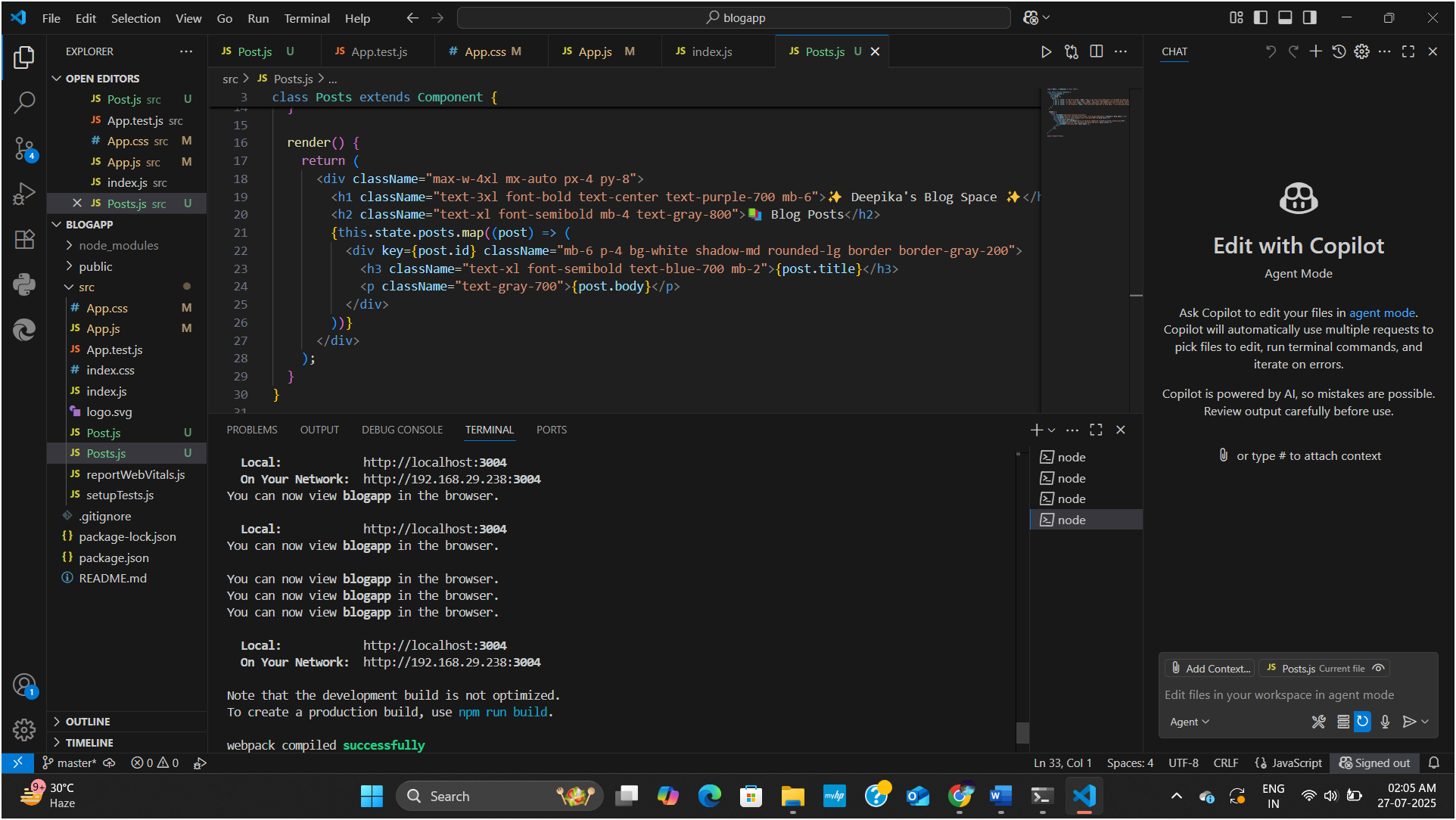
);

}export default App;

**5. Start the App**

npm start



****

**Question 5**

**Styling React Components**

**Objective:**

* Understand the **need for styling** React components.
* Use **CSS Modules** for component-level styling.
* Apply **inline styles** conditionally.

**1. Setup Project**

npx create-react-app cohort-dashboard

**2. Project Structure**

Inside src/ folder, create:

* components/CohortDetails.js
* components/CohortDetails.module.css

**3. 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;

}

**4. CohortDetails.js (Component)**

import React from 'react';

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

const CohortDetails = ({ cohort }) => {

const headingStyle = {

color: cohort.status === 'ongoing' ? 'green' : 'blue',

};

return (

<div className={styles.box}>

<h3 style={headingStyle}>{cohort.name}</h3>

<dl>

<dt>Technology:</dt>

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

<dt>Status:</dt>

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

</dl>

</div>

);

};

export default CohortDetails;

**5. App.js (Usage)**

import React from 'react';

import CohortDetails from './components/CohortDetails';

function App() {

const cohorts = [

{ name: "React Bootcamp", technology: "React", status: "ongoing" },

{ name: "Java Mastery", technology: "Java", status: "completed" }

];

return (

<div>

{cohorts.map((cohort, index) => (

<CohortDetails key={index} cohort={cohort} />

))}

</div>

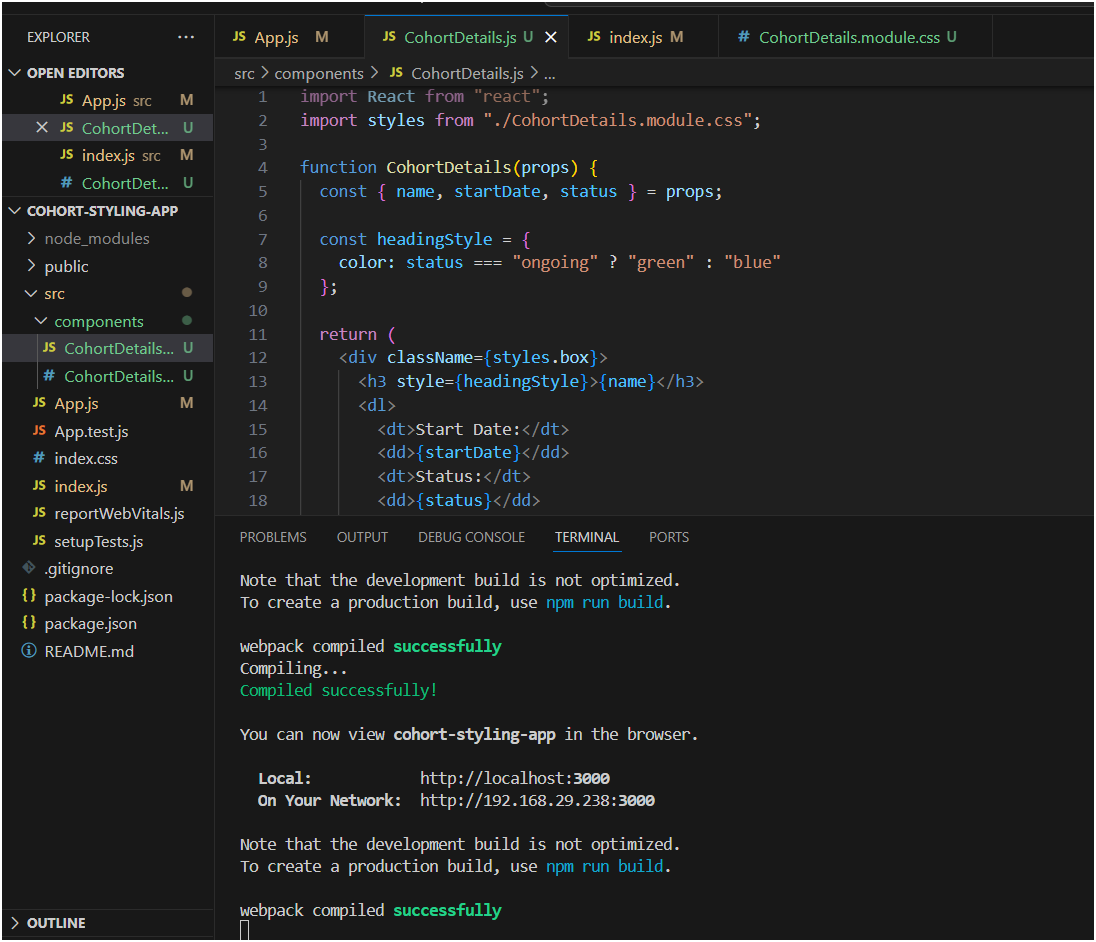
);

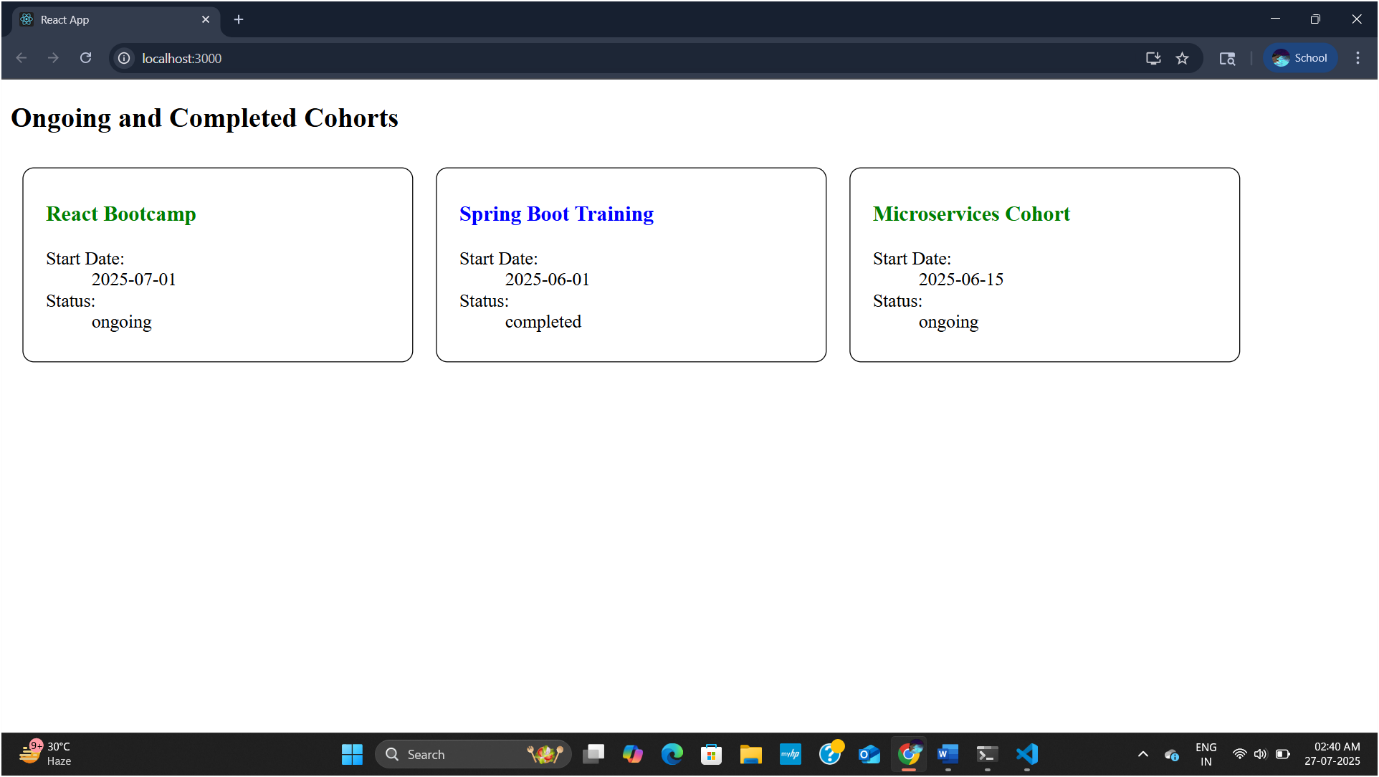
}

export default App;

**6.Run the App**

npm Start



****