SDM INSTITUTE OF TECHNOLOGY, UJIRE

(Affiliated to VTU, Belgaum, Approved by AICTE, New Delhi)

DEPARTMENT OF CSE

React

(BCSL657B)

Semester 6

Prepared by:

Adarsh Karanth

Table of Contents

Syllabus

React			
Course Code	**BCSL657B**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Total marks	100
Examination type (SEE)	PRACTICAL		

Course objectives:

- Enable students to develop React applications utilizing functional and class-based components, effectively managing state with hooks and lifecycle methods.
- Introduce, how to pass data dynamically between parent and child components using props, ensuring modular and reusable component design.
- Create dynamic and responsive applications, integrating forms, validation, task management systems, and styled components.
- Use React Router for navigation, external API integration for dynamic data handling, and CSS styling techniques for modern UI/UX design.

Sl.NO	**Lab Programs/Experiments**
1	Use create-react-app to set up a new project. Edit the App.js file to include a stateful component with useState. Add an input field and a <h1> element that displays text based on the input. Dynamically update the <h1> content as the user types.
2	Develop a React application that demonstrates the use of props to pass data from a parent component to child components. The application should include the parent component named App that serves as the central container for the application. Create two separate child components, Header: Displays the application title or heading. Footer: Displays additional information, such as copyright details or a tagline. Pass data (e.g., title, tagline, or copyright information) from the App component to the Header and Footer components using props. Ensure that the content displayed in the Header and Footer components is dynamically updated based on the data received from the parent component.
3	Create a Counter Application using React that demonstrates state management with the useState hook. Display the current value of the counter prominently on the screen. Add buttons to increase and decrease the counter value. Ensure the counter updates dynamically when the buttons are clicked. Use the useState hook to manage the counter's state within the component. Prevent the counter from going below a specified minimum value (e.g., 0). Add a "Reset" button to set the counter back to its initial value. Include functionality to specify a custom increment or decrement step value.
4	Develop a To-Do List Application using React functional components that demonstrates the use of the useState hook for state management. Create a functional component named ToDoFunction to manage and display the to-do list. Maintain a list of tasks using state. Provide an input field for users to add new tasks. Dynamically render the list of tasks below the input field. Ensure each task is displayed in a user-friendly manner. Allow users to delete tasks from the list. Mark tasks as completed or pending, and visually differentiate them.
5	Develop a React application that demonstrates component composition and the use of props to pass data. Create two components: FigureList: A parent component responsible for rendering multiple child components. BasicFigure: A child component designed to display an image and its associated caption. Use the FigureList component to dynamically render multiple BasicFigure components. Pass image URLs and captions as props from the FigureList component to each BasicFigure component. Style the BasicFigure components to display the image and caption in an aesthetically pleasing manner. Arrange the BasicFigure components within the FigureList in a grid or list format. Allow users to add or remove images dynamically. Add hover effects or animations to the images for an interactive experience.

6	Design and implement a React Form that collects user input for name, email, and password. Form Fields are Name, Email, Password. Ensure all fields are filled before allowing form submission.Validate the email field to ensure it follows the correct email format (e.g., example@domain.com). Optionally enforce a minimum password length or complexity. Display error messages for invalid or missing inputs. Provide visual cues (e.g., red borders) to highlight invalid fields. Prevent form submission until all fields pass validation. Log or display the entered data upon successful submission (optional). Add a "Show Password" toggle for the password field. Implement client-side sanitization to ensure clean input.
7	Develop a React Application featuring a ProfileCard component to display a user's profile information, including their name, profile picture, and bio. The component should demonstrate flexibility by utilizing both external CSS and inline styling for its design. Display the following information: Profile picture, User's name, A short bio or description Use an external CSS file for overall structure and primary styles, such as layout, colors, and typography. Apply inline styles for dynamic or specific styling elements, such as background colors or alignment. Design the ProfileCard to be visually appealing and responsive. Ensure the profile picture is displayed as a circle, and the name and bio are appropriately styled. Add hover effects or animations to enhance interactivity. Allow the background color of the card to change dynamically based on a prop or state.
8	Develop a Reminder Application that allows users to efficiently manage their tasks. The application should include the following functionalities: Provide a form where users can add tasks along with due dates. The form includes task name,Due date,An optional description. Display a list of tasks dynamically as they are added. Show relevant details like task name, due date, and completion status. Include a filter option to allow users to view all Tasks and Display all tasks regardless of status. Show only tasks marked as completed. Show only tasks that are not yet completed.
9	Design a React application that demonstrates the implementation of routing using the react-router-dom library. The application should include the Navigation Menu: Create a navigation bar with links to three distinct pages, Home, About, Contact. Develop separate components for each page (Home, About, and Contact) with appropriate content to differentiate them. Configure routes using react-router-dom to render the corresponding page component based on the selected link. Use BrowserRouter and Route components for routing. Highlight the active link in the navigation menu to indicate the current page.
10	Design a React application featuring a class-based component that demonstrates the use of lifecycle methods to interact with an external API. The component should fetch and update data dynamically based on user interactions or state changes. Use the componentDidMount lifecycle method to fetch data from an API when the component is initially rendered. Display the fetched data in a structured format, such as a table or list. Use the componentDidUpdate lifecycle method to detect changes in the component's state or props. Trigger additional API calls to update the displayed data based on user input or actions (e.g., filtering, searching, or pagination). Implement error handling to manage issues such as failed API requests or empty data responses. Display appropriate error messages to the user when necessary. Allow users to perform actions like filtering, searching, or refreshing the data. Reflect changes in the displayed data based on these interactions.

NOTE: Include necessary HTML elements and CSS for the above React programs.

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

1. Illustrate React basics and state components.
2. Develop React applications that utilize component composition, passing data through props.
3. Use dynamic state updates, event handling, and custom logic to increment, decrement, and reset state values.
4. Implement forms in React that collect and validate user input.

5. Demonstrate interaction with external APIs, dynamic content generation and manage state in real-time applications.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are 50 Marks.
The split-up of CIE marks for record/ journal and test are in the ratio 60:40.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability
- The marks scored shall be scaled down to 20 marks (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course are 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.
- The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. OR based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

- General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
- Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.
- The minimum duration of SEE is 02 hours

Suggested Learning Resources:
Books:

1. Beginning React JS Foundations Building User Interfaces with ReactJS: An Approachable Guide, Chris Minnick, Wiley publications , 2022.
2. Learning React Functional Web Development with React and Redux , Alex Banks, Eve Porcello · 2017

Web links and Video Lectures (e-Resources):

- https://www.youtube.com/watch?v=V9i3cGD-mts
- https://youtu.be/PHaECbrKgs0
- https://youtu.be/uvEAvxWvwOs
- https://www.geeksforgeeks.org/state-management-with-usestate-hook-in-react/
- https://youtu.be/KU-I2M9Jm68
- https://youtu.be/H63Pd_lXkeQ
- https://youtu.be/oTIJunBa6MA
- https://youtu.be/3EbYJrAOpUs

Lab Program #1

Objective

- Use create-react-app to set up a new project. Edit the App.js file to include a stateful component with useState. Add an input field and a <h1> element that displays text based on the input. Dynamically update the <h1> content as the user types..

Steps

- Create a New React Project.
  Run the following command to set up a new React project:
  ```
  npx create-react-app react-lab
  cd react-lab
  ```
- Update the App.js File
- Create Program specific content in programs/Pgm1.js
- Run the following command to start the application:
  ```
  npm start
  ```
- Verify the functionality.
- Go through the Code explanation to understand it.

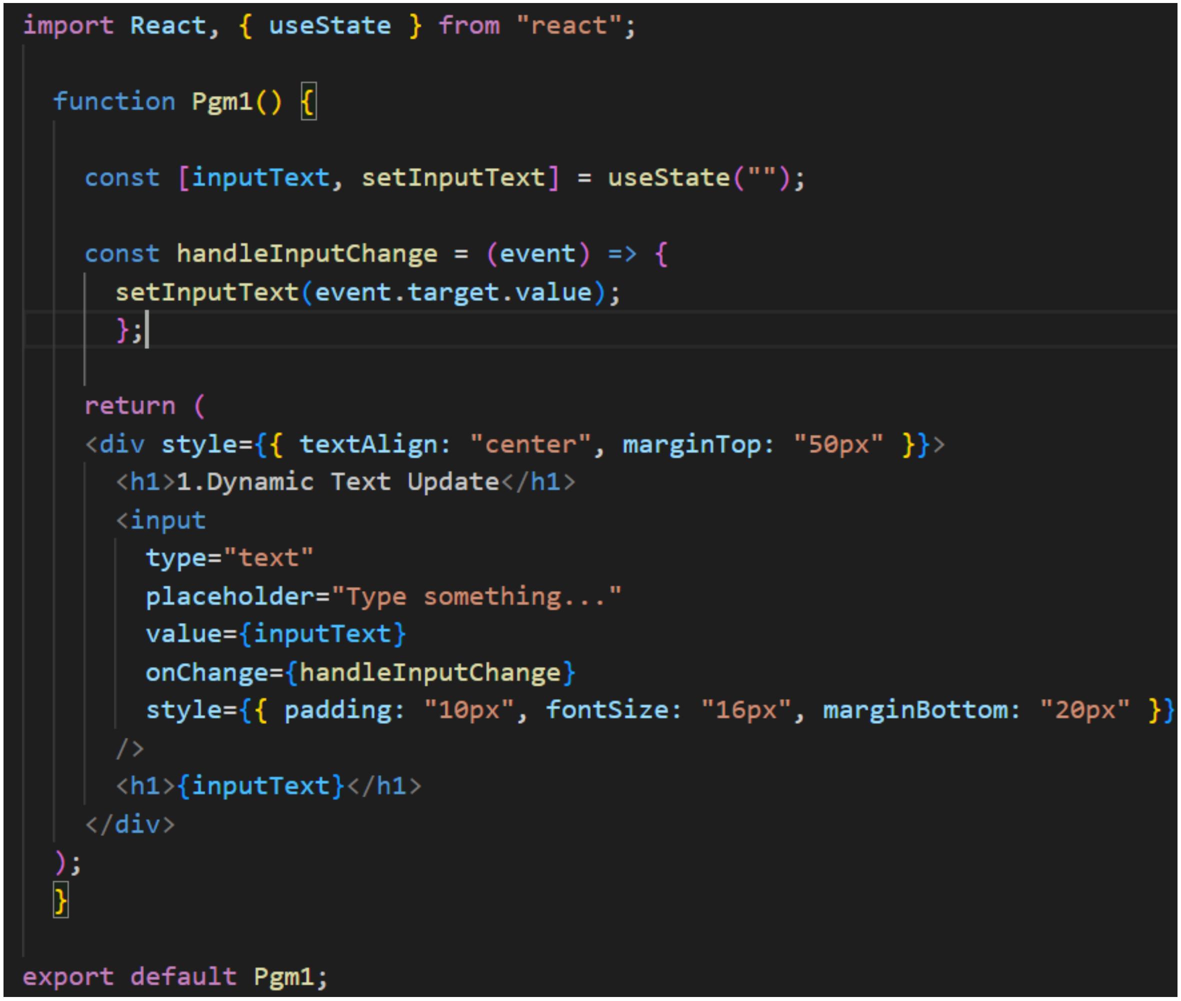
Files

File name	Changes
App.js	Include Pgm1 component.
programs/Pgm1.js	Create Program specific content.

Code

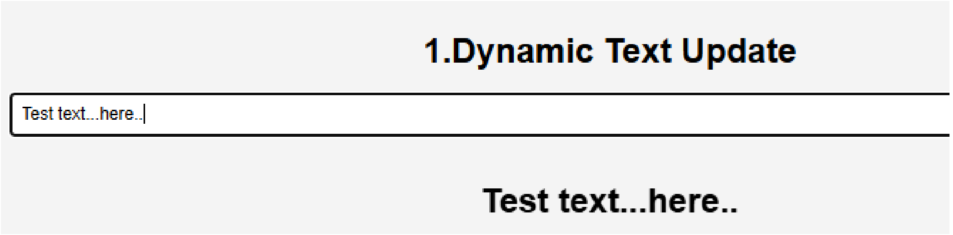
```
// App.js
import React  from "react";
import Pgm1 from "./programs/Pgm1";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm1/>
    </div>
  );
}

export default App;
```

```
//programs/Pgm1.js
```



```
import React, { useState } from "react";

function Pgm1() {

  const [inputText, setInputText] = useState("");

  const handleInputChange = (event) => {
    setInputText(event.target.value);
  };

  return (
  <div style={{ textAlign: "center", marginTop: "50px" }}>
    <h1>1.Dynamic Text Update</h1>
    <input
      type="text"
      placeholder="Type something..."
      value={inputText}
      onChange={handleInputChange}
      style={{ padding: "10px", fontSize: "16px", marginBottom: "20px" }}
    />
    <h1>{inputText}</h1>
  </div>
);
}

export default Pgm1;
```

Verification

URL	http://localhost:3000/
Browser View	 1.Dynamic Text Update Test text...here.. Test text...here..

Explanation

- **App.js**
  - **imports:**
    - imports the React library, which is necessary for creating React components.
    - import the React component Pgm1 from the programs directory.

- **Define App component**
    - App is a functional component that returns JSX (JavaScript XML).
    - <div> is used as a wrapper.
    - <Pgm1 /> → Renders the Pgm1 component inside the App component.
    - export default App; allows this component to be imported in index.js, to render it inside the root div of index.html.
- **Pgm1.js**
  - **useState:**
    - { useState } is imported to manage state in the functional component.
    - const [inputText, setInputText] = useState("");
      - useState("") initializes inputText with an empty string ("").
      - setInputText is a function that updates inputText.
  - **handleInputChange:**
    - handleInputChange is called when the user types in the input field.
    - event.target.value contains the current text in the input field.
    - setInputText updates the inputText state.

Lab Program #2

Objective

- Develop a React application that demonstrates the use of props to pass data from a parent component to child components. The application should include the parent component named App that serves as the central container for the application. Create two separate child components, Header: Displays the application title or heading. Footer: Displays additional information, such as copyright details or a tagline. Pass data (e.g., title, tagline, or copyright information) from the App component to the Header and Footer components using props. Ensure that the content displayed in the Header and Footer components is dynamically updated based on the data received from the parent component.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

  ```
  npm start
  ```
- Verify the functionality.
- Go through the Code explanation to understand it.

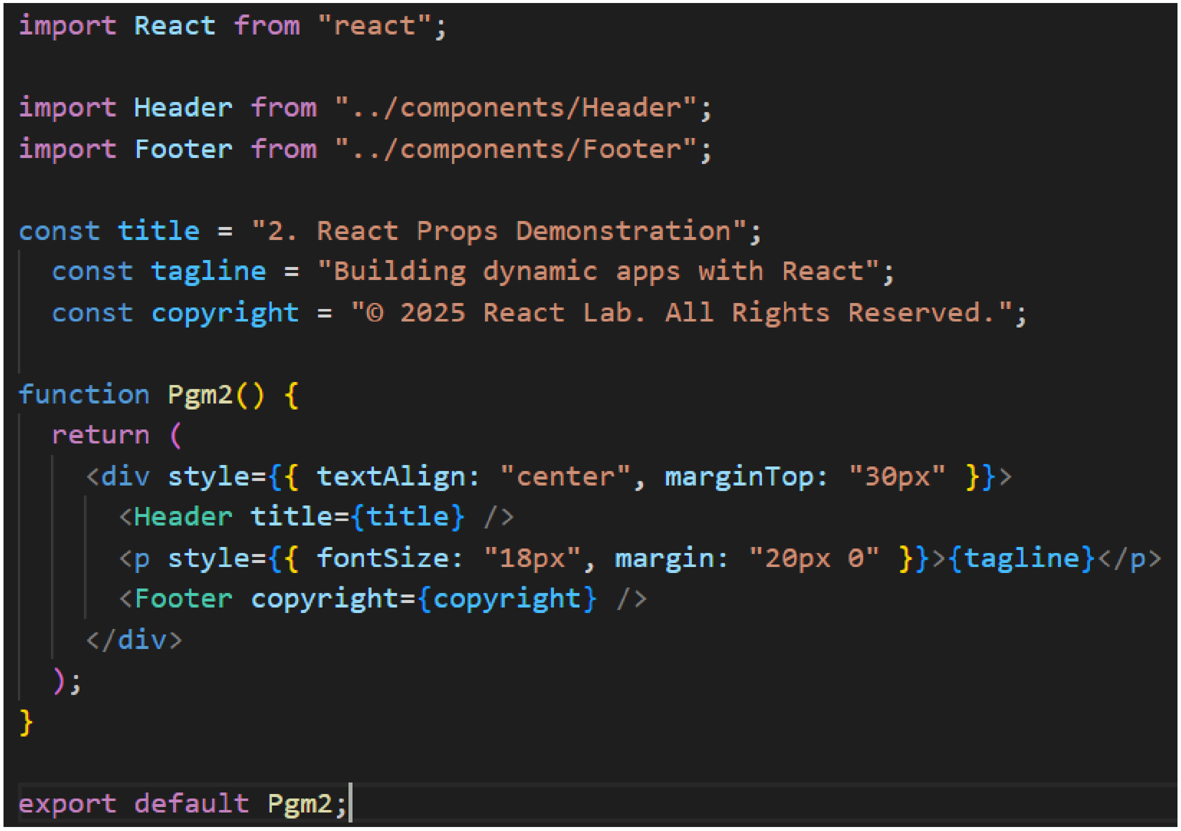
Files

File name	Changes
App.js	Include Pgm2 component.
programs/Pgm2.js	Create Program specific content.
components/Header.js	Write the Header code in it.
components/Footer.js	Write the Footer code in it.

Code

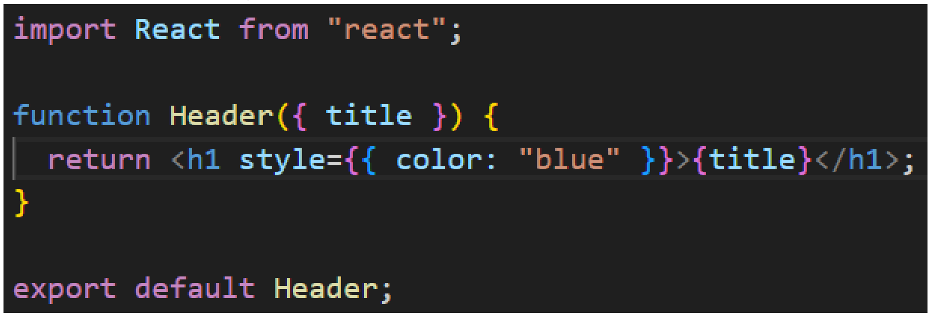
```
// App.js
import React  from "react";
import Pgm2 from "./programs/Pgm2";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm2/>
    </div>
  );
}

export default App;
```

```
//programs/Pgm2.js
```



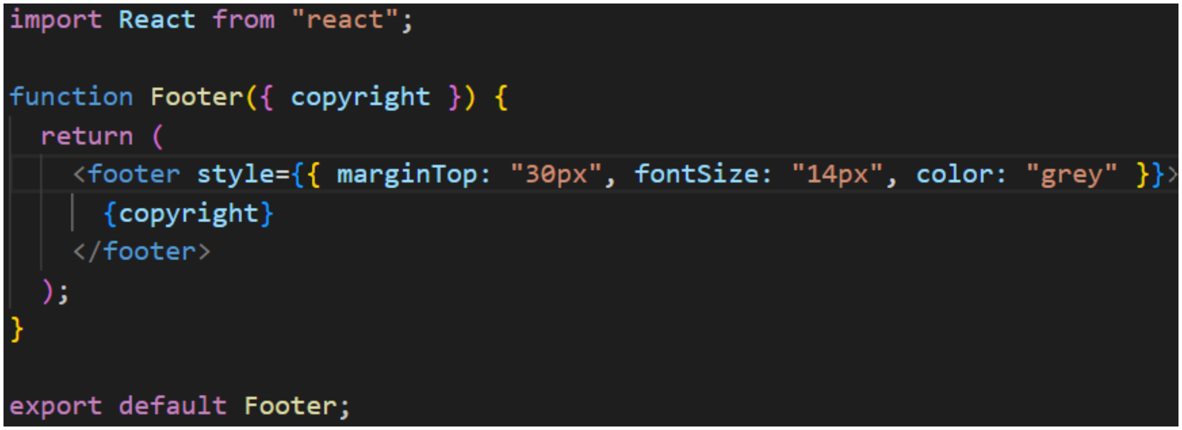
```
import React from "react";

import Header from "../components/Header";
import Footer from "../components/Footer";

const title = "2. React Props Demonstration";
  const tagline = "Building dynamic apps with React";
  const copyright = "© 2025 React Lab. All Rights Reserved.";

function Pgm2() {
  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
      <Header title={title} />
      <p style={{ fontSize: "18px", margin: "20px 0" }}>{tagline}</p>
      <Footer copyright={copyright} />
    </div>
  );
}

export default Pgm2;
```

/components/Header.js



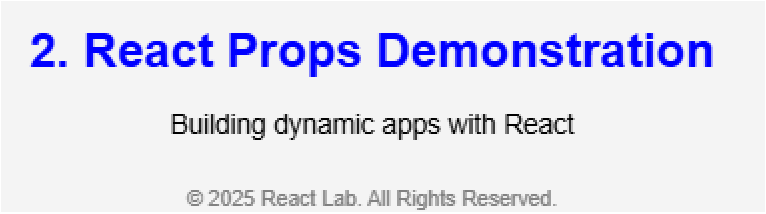
```
import React from "react";

function Header({ title }) {
  return <h1 style={{ color: "blue" }}>{title}</h1>;
}

export default Header;
```

/components/Footer.js



```
import React from "react";

function Footer({ copyright }) {
  return (
    <footer style={{ marginTop: "30px", fontSize: "14px", color: "grey" }}>
      {copyright}
    </footer>
  );
}

export default Footer;
```

Verification

URL	http://localhost:3000/
Browser View	 2. React Props Demonstration Building dynamic apps with React © 2025 React Lab. All Rights Reserved.

Explanation

- **App.js**
  - <Pgm2 /> → Renders the Pgm2 component inside the App component.
- **Pgm2.js**
  - **imports:**
    - Import the components to be included. ie Header and Footer.
  - **Define the functional components:**
    - It's done as like below:
      - <Header title={title} />
      - <Footer copyright={copyright} />

      The components should be enclosed withing angular brackets.
      And the props to be passed to the component should be specified as an argument within. Ex: title, copyright.
- **Header.js and Footer.js**
  - Return the DOM component with JSX.
  - Create the function name same as component name.
  - Specify the props as arguments; and use inside the component block.

Lab Program #3

Objective

- Create a Counter Application using React that demonstrates state management with the useState hook. Display the current value of the counter prominently on the screen. Add buttons to increase and decrease the counter value. Ensure the counter updates dynamically when the buttons are clicked. Use the useState hook to manage the counter's state within the component. Prevent the counter from going below a specified minimum value (e.g., 0). Add a "Reset" button to set the counter back to its initial value. Include functionality to specify a custom increment or decrement step value.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

```
npm start
```

- Verify the functionality.
- Go through the Code explanation to understand it.

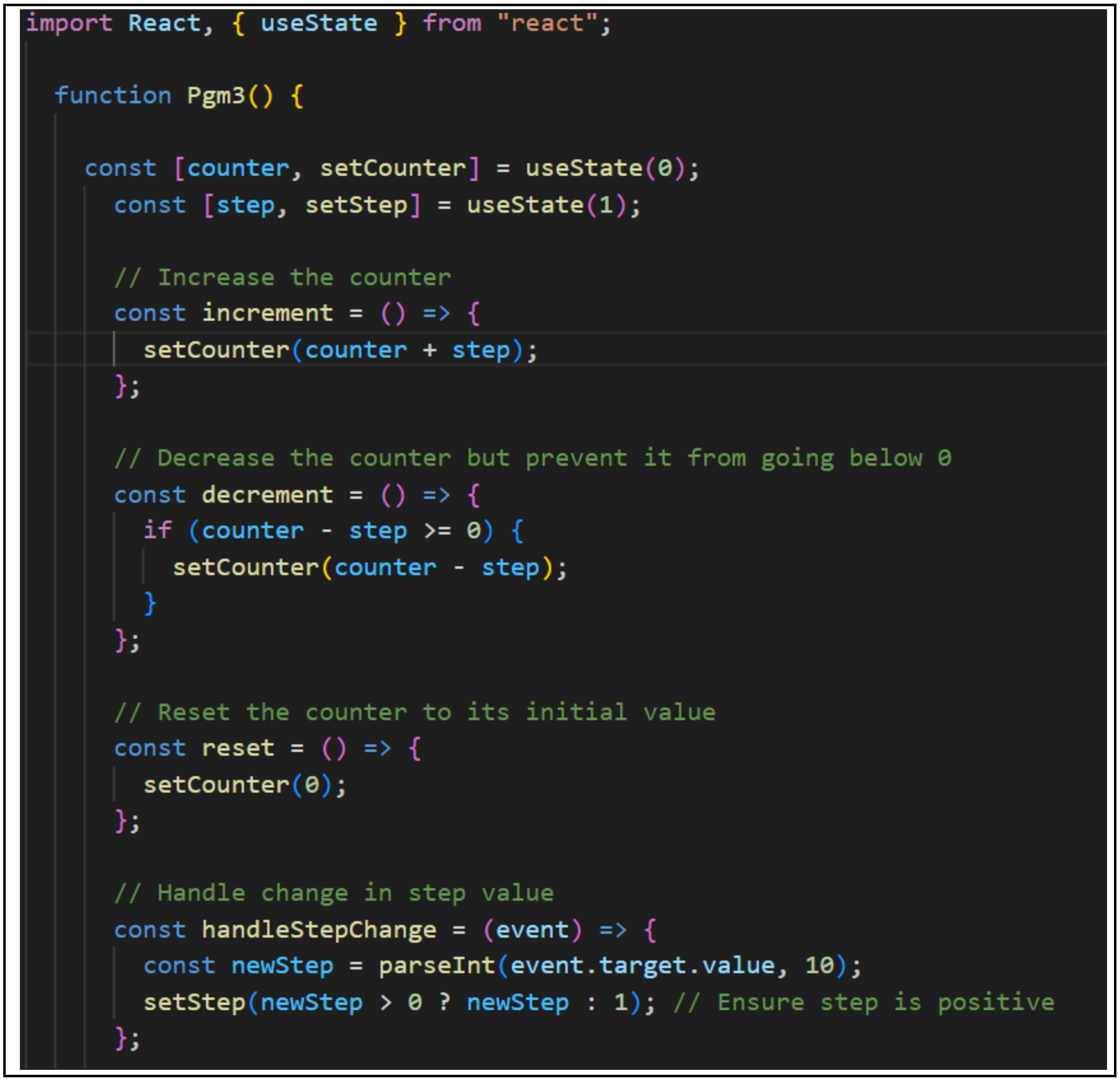
Files

File name	Changes
App.js	Include Pgm3 component.
programs/Pgm3.js	Create Program specific content.

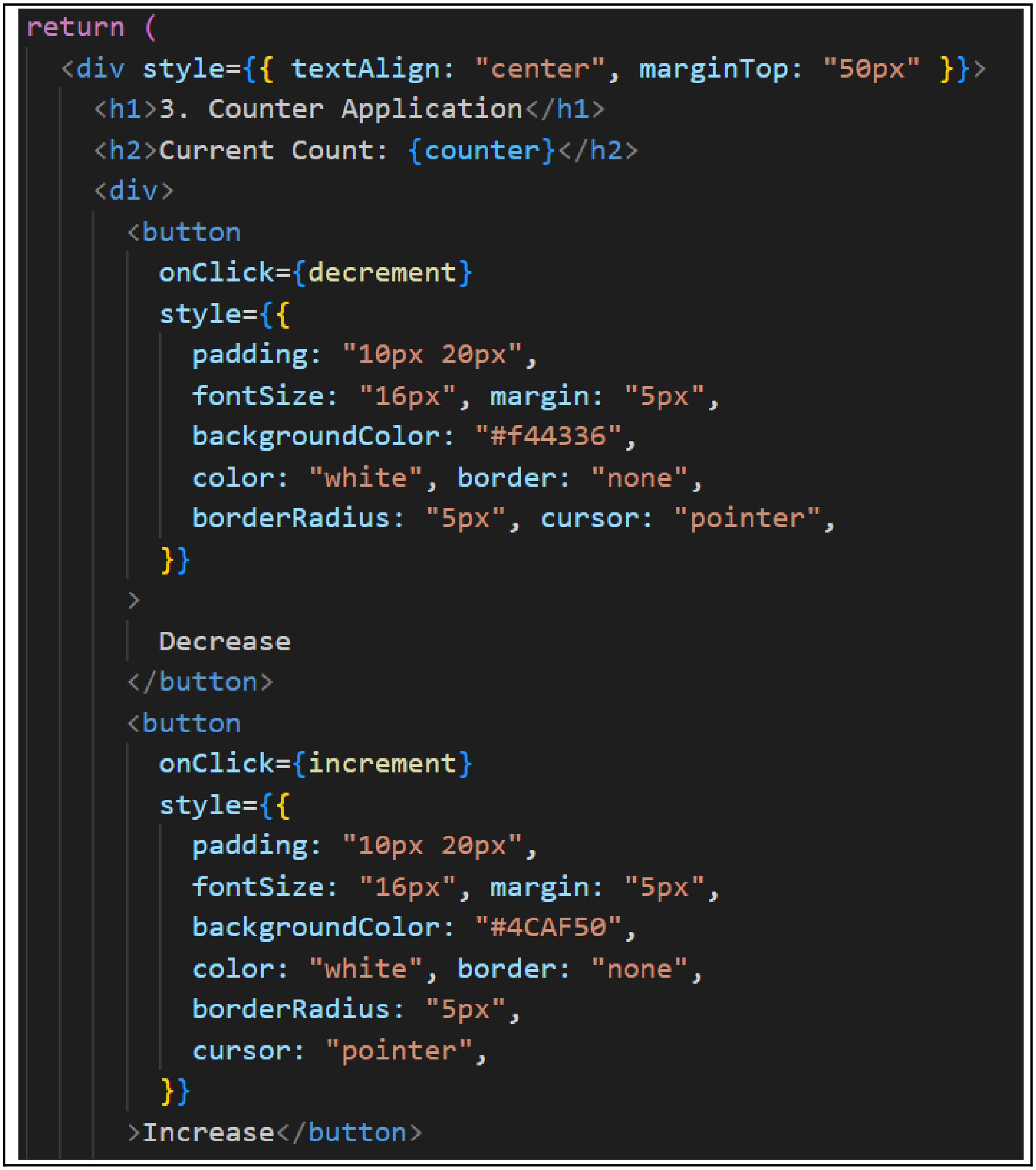
Code

```
// App.js
import React  from "react";
import Pgm3 from "./programs/Pgm3";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm3/>
    </div>
  );
}

export default App;
```

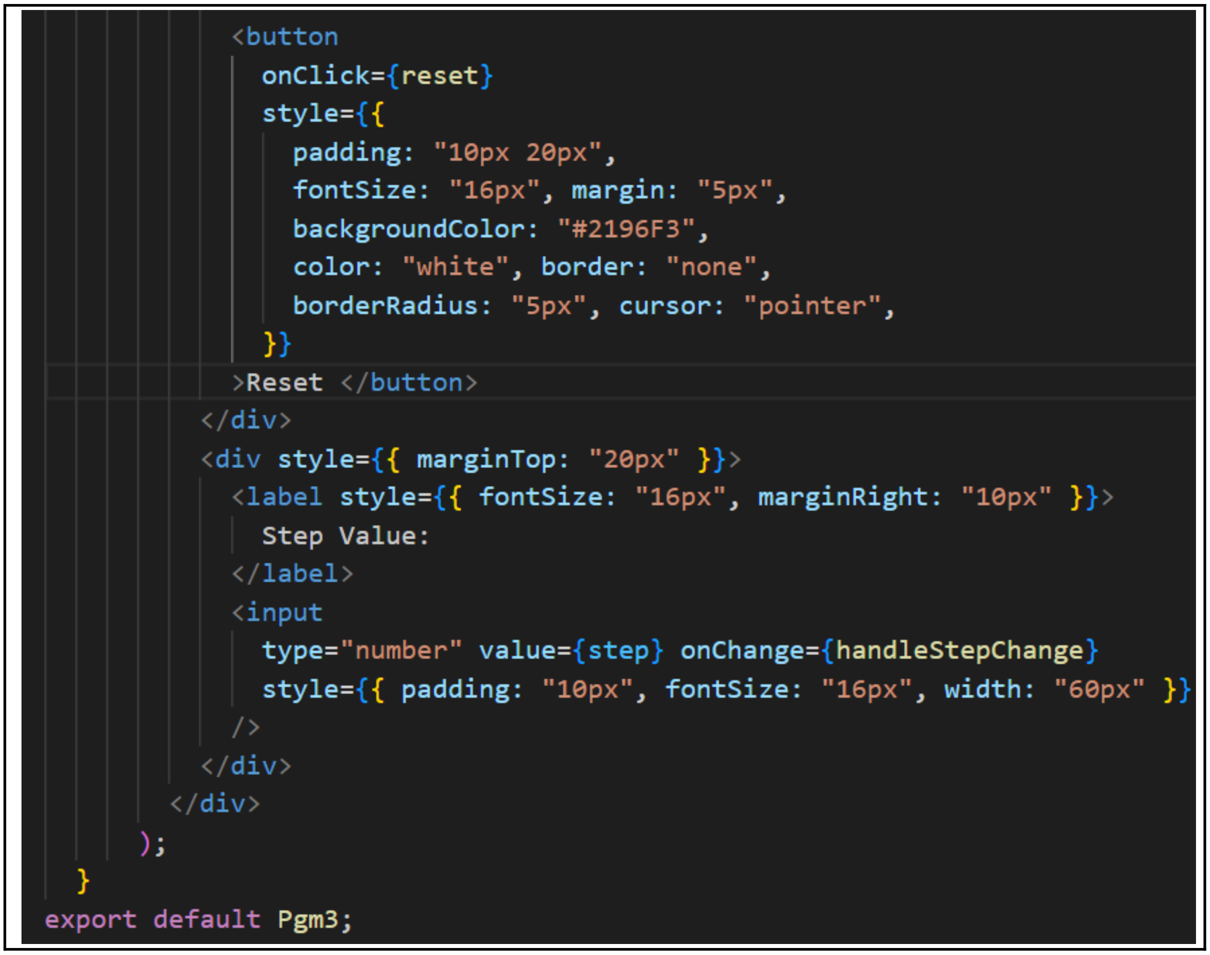
/programs/Pgm3.js



```
import React, { useState } from "react";

  function Pgm3() {

    const [counter, setCounter] = useState(0);
      const [step, setStep] = useState(1);

      // Increase the counter
      const increment = () => {
        setCounter(counter + step);
      };

      // Decrease the counter but prevent it from going below 0
      const decrement = () => {
        if (counter - step >= 0) {
          setCounter(counter - step);
        }
      };

      // Reset the counter to its initial value
      const reset = () => {
        setCounter(0);
      };

      // Handle change in step value
      const handleStepChange = (event) => {
        const newStep = parseInt(event.target.value, 10);
        setStep(newStep > 0 ? newStep : 1); // Ensure step is positive
      };
```

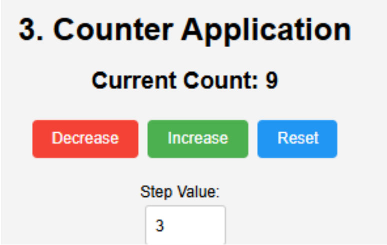


```
return (
  <div style={{ textAlign: "center", marginTop: "50px" }}>
    <h1>3. Counter Application</h1>
    <h2>Current Count: {counter}</h2>
    <div>
      <button
        onClick={decrement}
        style={{
          padding: "10px 20px",
          fontSize: "16px", margin: "5px",
          backgroundColor: "#f44336",
          color: "white", border: "none",
          borderRadius: "5px", cursor: "pointer",
        }}
      >
        Decrease
      </button>
      <button
        onClick={increment}
        style={{
          padding: "10px 20px",
          fontSize: "16px", margin: "5px",
          backgroundColor: "#4CAF50",
          color: "white", border: "none",
          borderRadius: "5px",
          cursor: "pointer",
        }}
      >Increase</button>
```



```
            <button
              onClick={reset}
              style={{
                padding: "10px 20px",
                fontSize: "16px", margin: "5px",
                backgroundColor: "#2196F3",
                color: "white", border: "none",
                borderRadius: "5px", cursor: "pointer",
              }}
            >Reset </button>
          </div>
          <div style={{ marginTop: "20px" }}>
            <label style={{ fontSize: "16px", marginRight: "10px" }}>
              Step Value:
            </label>
            <input
              type="number" value={step} onChange={handleStepChange}
              style={{ padding: "10px", fontSize: "16px", width: "60px" }}
            />
          </div>
        </div>
      );
  }
export default Pgm3;
```

Verification

URL	http://localhost:3000/
Browser View	 3. Counter Application Current Count: 9 Step Value: 3

Explanation

- **App.js**
  - <Pgm3 /> → Renders the Pgm3 component inside the App component.
- **Pgm3.js**
  - **Increment function:**
    - When called, it adds the step value to the current counter.
  - **Decrement function:**

    - Only decreases the counter if the result won't be negative.
- **Reset function:**
    - Resets the counter back to zero.
- **handleStepChange function:**
    - Reads the value from the input field.
    - Converts it to an integer using parseInt().
    - Makes sure the step is at least 1 (no zero or negative steps).

Lab Program #4

Objective

- Develop a To-Do List Application using React functional components that demonstrates the use of the useState hook for state management. Create a functional component named ToDoFunction to manage and display the to-do list. Maintain a list of tasks using state. Provide an input field for users to add new tasks. Dynamically render the list of tasks below the input field. Ensure each task is displayed in a user-friendly manner. Allow users to delete tasks from the list. Mark tasks as completed or pending, and visually differentiate them.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

```
npm start
```

- Verify the functionality.
- Go through the Code explanation to understand it.

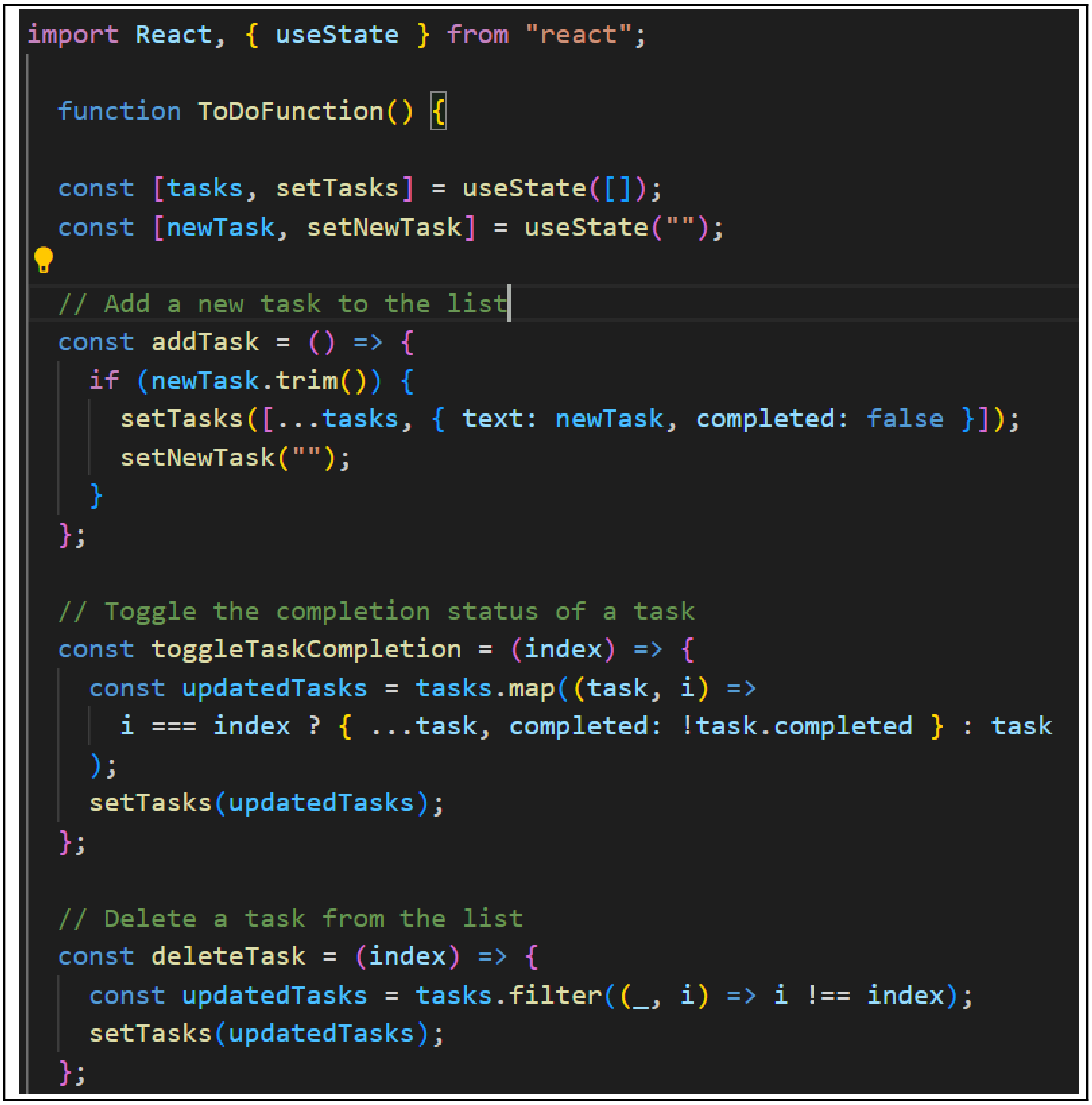
Files

File name	Changes
App.js	Include Pgm4 component.
programs/Pgm4.js	Create Program specific content.

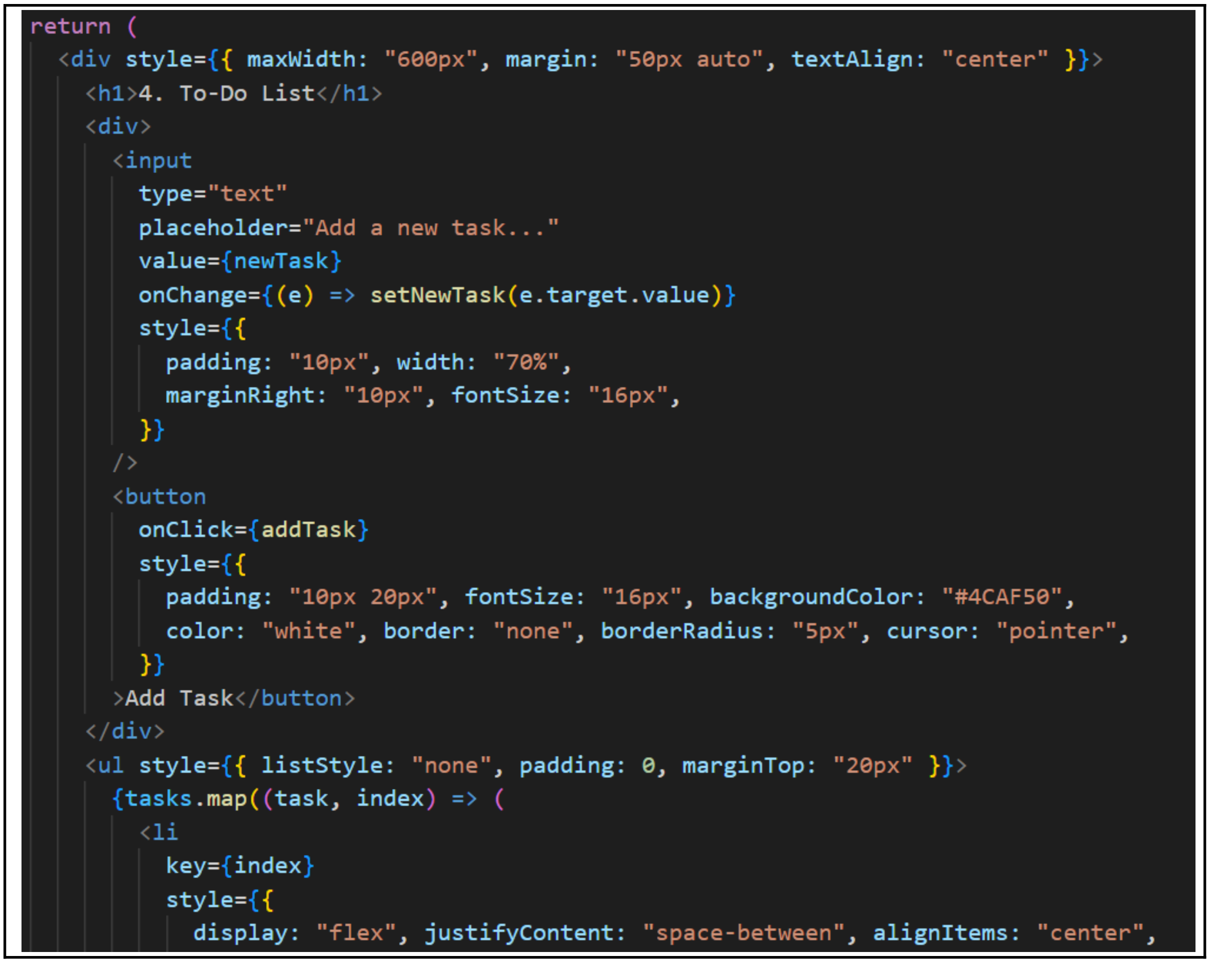
Code

```
// App.js
import React  from "react";
import Pgm4 from "./programs/Pgm4";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm4/>
    </div>
  );
}

export default App;
```

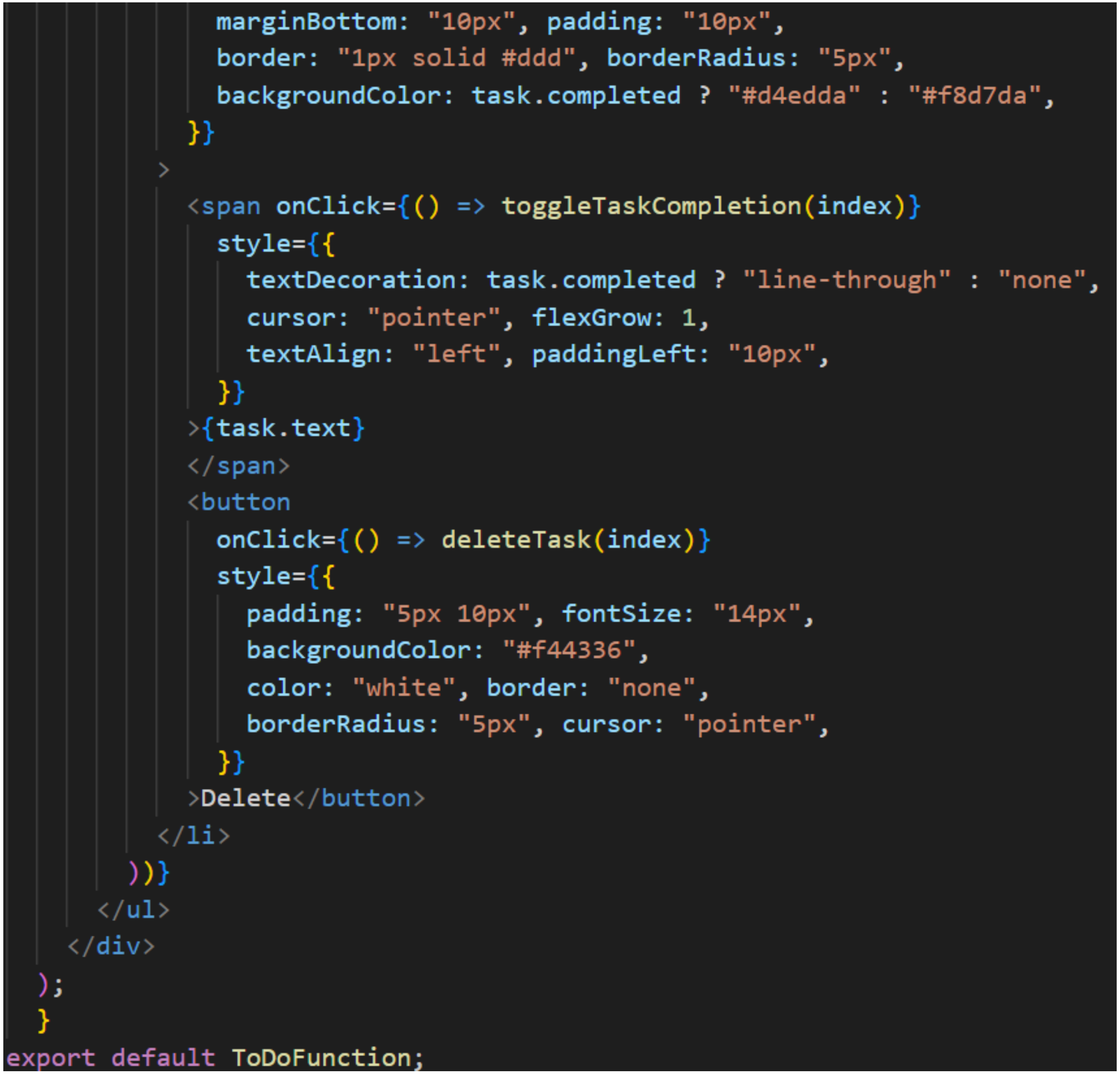
/programs/Pgm4.js



```
import React, { useState } from "react";

  function ToDoFunction() {

  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState("");

  // Add a new task to the list
  const addTask = () => {
    if (newTask.trim()) {
      setTasks([...tasks, { text: newTask, completed: false }]);
      setNewTask("");
    }
  };

  // Toggle the completion status of a task
  const toggleTaskCompletion = (index) => {
    const updatedTasks = tasks.map((task, i) =>
      i === index ? { ...task, completed: !task.completed } : task
    );
    setTasks(updatedTasks);
  };

  // Delete a task from the list
  const deleteTask = (index) => {
    const updatedTasks = tasks.filter((_, i) => i !== index);
    setTasks(updatedTasks);
  };
```

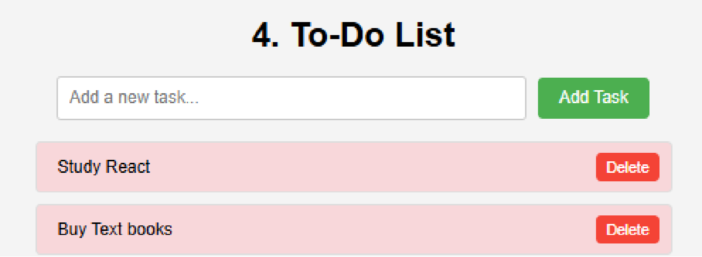


```
return (
  <div style={{ maxWidth: "600px", margin: "50px auto", textAlign: "center" }}>
    <h1>4. To-Do List</h1>
    <div>
      <input
        type="text"
        placeholder="Add a new task..."
        value={newTask}
        onChange={(e) => setNewTask(e.target.value)}
        style={{
          padding: "10px", width: "70%",
          marginRight: "10px", fontSize: "16px",
        }}
      />
      <button
        onClick={addTask}
        style={{
          padding: "10px 20px", fontSize: "16px", backgroundColor: "#4CAF50",
          color: "white", border: "none", borderRadius: "5px", cursor: "pointer",
        }}
      >Add Task</button>
    </div>
    <ul style={{ listStyle: "none", padding: 0, marginTop: "20px" }}>
      {tasks.map((task, index) => (
        <li
          key={index}
          style={{
            display: "flex", justifyContent: "space-between", alignItems: "center",
```



```
       marginBottom: "10px", padding: "10px",
       border: "1px solid #ddd", borderRadius: "5px",
       backgroundColor: task.completed ? "#d4edda" : "#f8d7da",
      }}
     >
      <span onClick={() => toggleTaskCompletion(index)}
       style={{
        textDecoration: task.completed ? "line-through" : "none",
        cursor: "pointer", flexGrow: 1,
        textAlign: "left", paddingLeft: "10px",
       }}
      >{task.text}
      </span>
      <button
       onClick={() => deleteTask(index)}
       style={{
        padding: "5px 10px", fontSize: "14px",
        backgroundColor: "#f44336",
        color: "white", border: "none",
        borderRadius: "5px", cursor: "pointer",
       }}
      >Delete</button>
     </li>
    ))}
   </ul>
  </div>
 );
 }
export default ToDoFunction;
```

Verification

URL	http://localhost:3000/
Browser View	 4. To-Do List Add a new task... Add Task Study React Delete Buy Text books Delete

Explanation

- **App.js**
  - <Pgm4 /> → Renders the Pgm3 component inside the App component.

- **Pgm4.js**
    - **imports:**
        - { useState } is used to manage the component's state.
    - **Defining State variables:**
        - const [tasks, setTasks] = useState([]);
            - tasks: Stores the list of to-do tasks (initially an empty array []).
            - setTasks: Updates the tasks state.
    - **addTask function**
        - creates a new task object with:
            - text: newTask → Stores the task description.
            - completed: false → Initially, the task is not completed.
            - Updates the tasks array using the spread operator (...tasks).
            - Clears the input field (setNewTask("")).
    - **Toggling Task Completion**
        - Loops through the tasks array and updates the completed status of the clicked task.
        - Uses .map() to return a new array with the modified task.
        - Toggles (!task.completed) between true and false.
    - **Deleting a Task**
        - Uses .filter() to remove the task at the given index.
        - Updates tasks state with the filtered list.

Lab Program #5

Objective

- Develop a React application that demonstrates component composition and the use of props to pass data. Create two components: FigureList: A parent component responsible for rendering multiple child components. BasicFigure: A child component designed to display an image and its associated caption. Use the FigureList component to dynamically render multiple BasicFigure components. Pass image URLs and captions as props from the FigureList component to each BasicFigure component. Style the BasicFigure components to display the image and caption in an aesthetically pleasing manner. Arrange the BasicFigure components within the FigureList in a grid or list format. Allow users to add or remove images dynamically. Add hover effects or animations to the images for an interactive experience.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

  npm start
- Verify the functionality.
- Go through the Code explanation to understand it.

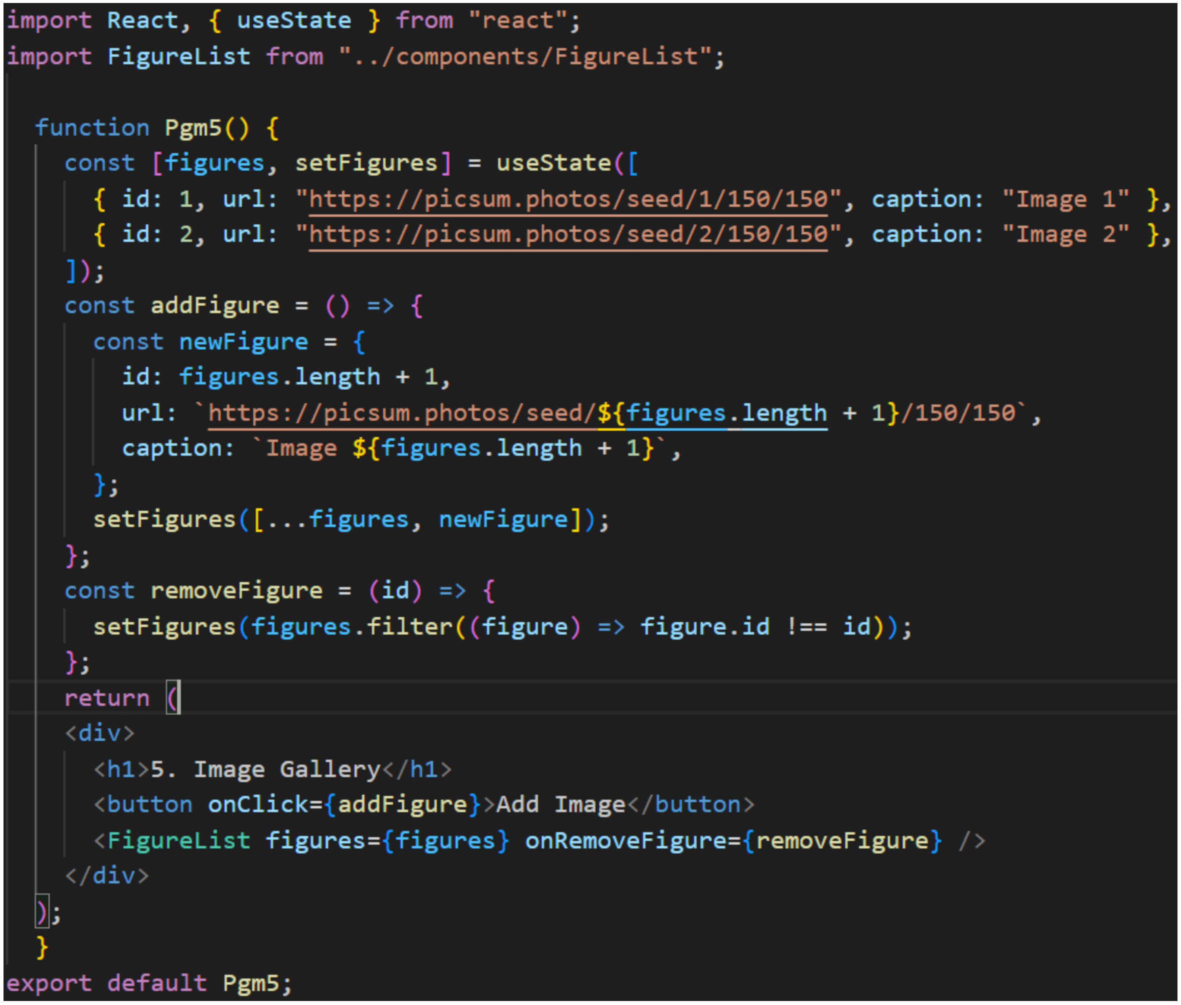
Files

File name	Changes
App.js	Include Pgm5 component.
programs/Pgm5.js	Create Program specific content.
/components/FigureList.js	New component
/components/BasicFigure.js	New component

Code

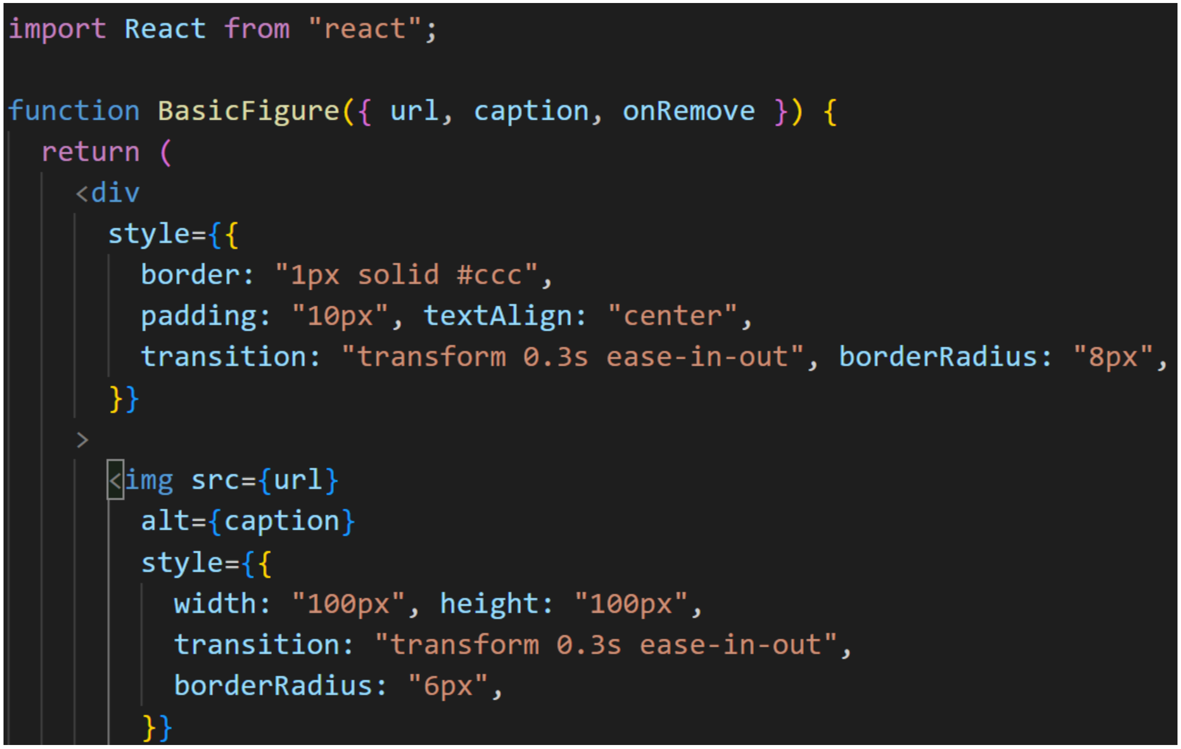
```
// App.js
import React  from "react";
import Pgm5 from "./programs/Pgm5";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm5/>
    </div>
  );
}

export default App;
```

/programs/Pgm5.js

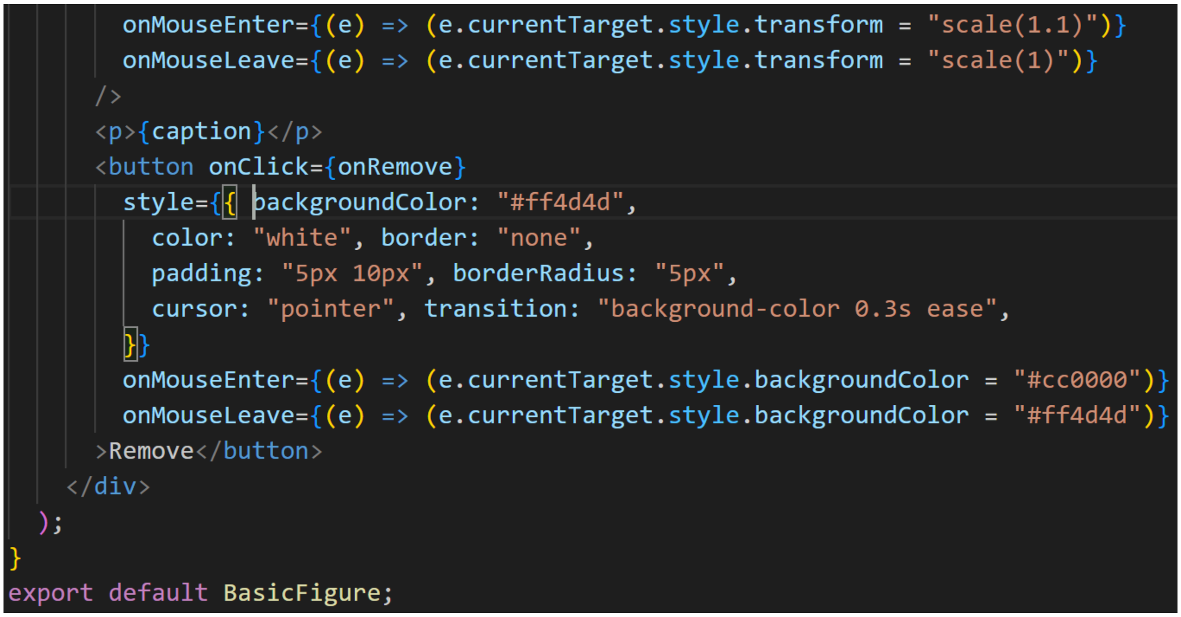


```
import React, { useState } from "react";
import FigureList from "../components/FigureList";

  function Pgm5() {
    const [figures, setFigures] = useState([
      { id: 1, url: "https://picsum.photos/seed/1/150/150", caption: "Image 1" },
      { id: 2, url: "https://picsum.photos/seed/2/150/150", caption: "Image 2" },
    ]);
    const addFigure = () => {
      const newFigure = {
        id: figures.length + 1,
        url: `https://picsum.photos/seed/${figures.length + 1}/150/150`,
        caption: `Image ${figures.length + 1}`,
      };
      setFigures([...figures, newFigure]);
    };
    const removeFigure = (id) => {
      setFigures(figures.filter((figure) => figure.id !== id));
    };
    return (
    <div>
      <h1>5. Image Gallery</h1>
      <button onClick={addFigure}>Add Image</button>
      <FigureList figures={figures} onRemoveFigure={removeFigure} />
    </div>
  );
  }
export default Pgm5;
```

/components/BasicFigure.js

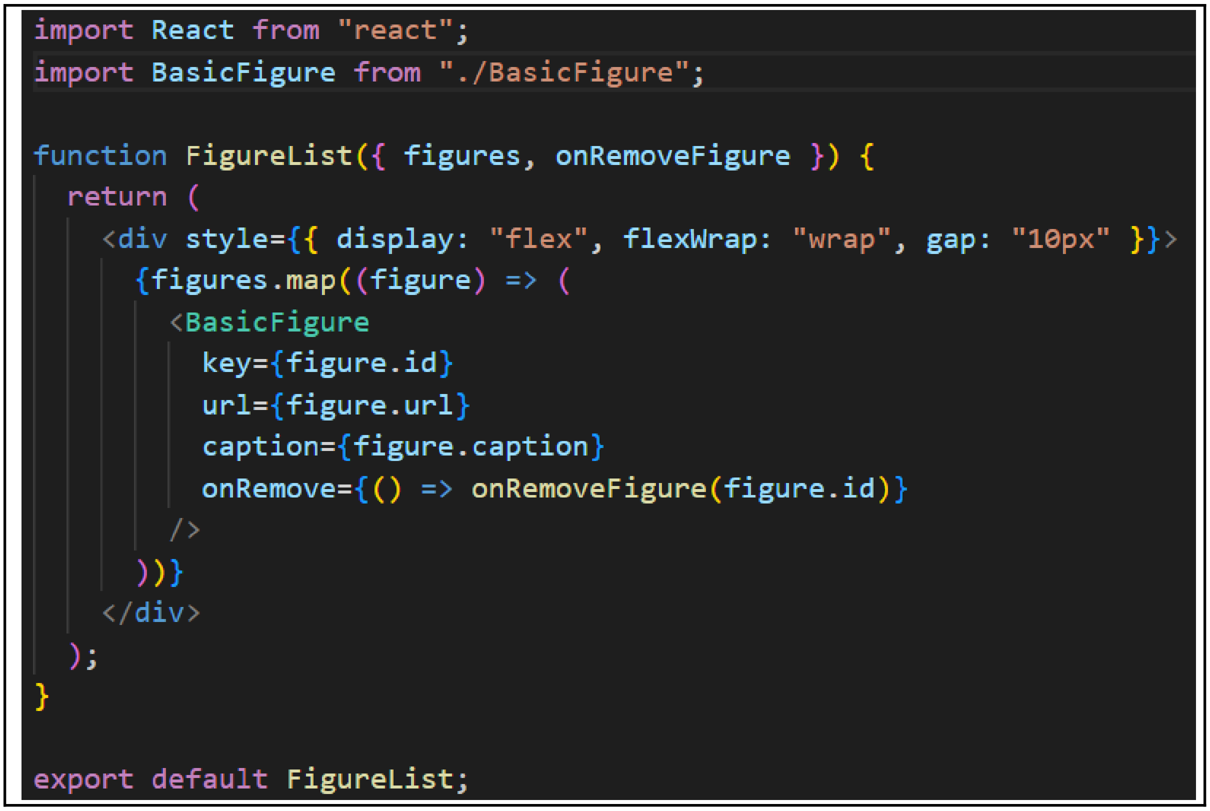


```
import React from "react";

function BasicFigure({ url, caption, onRemove }) {
  return (
    <div
      style={{
        border: "1px solid #ccc",
        padding: "10px", textAlign: "center",
        transition: "transform 0.3s ease-in-out", borderRadius: "8px",
      }}
    >
      <img src={url}
        alt={caption}
        style={{
          width: "100px", height: "100px",
          transition: "transform 0.3s ease-in-out",
          borderRadius: "6px",
        }}
```



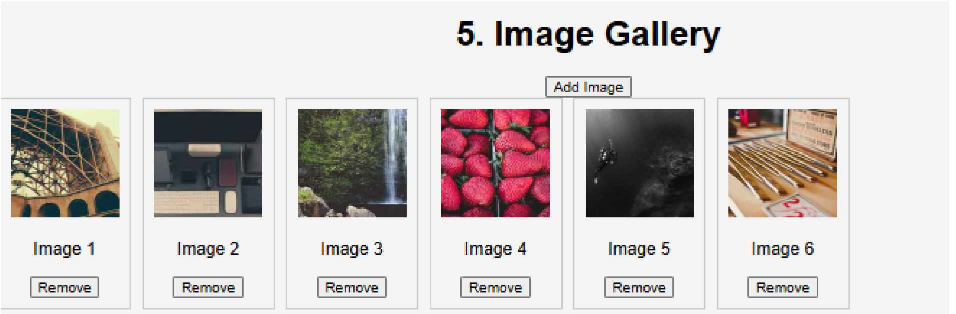
```
        onMouseEnter={(e) => (e.currentTarget.style.transform = "scale(1.1)")}
        onMouseLeave={(e) => (e.currentTarget.style.transform = "scale(1)")}
      />
      <p>{caption}</p>
      <button onClick={onRemove}
        style={{ backgroundColor: "#ff4d4d",
          color: "white", border: "none",
          padding: "5px 10px", borderRadius: "5px",
          cursor: "pointer", transition: "background-color 0.3s ease",
        }}
        onMouseEnter={(e) => (e.currentTarget.style.backgroundColor = "#cc0000")}
        onMouseLeave={(e) => (e.currentTarget.style.backgroundColor = "#ff4d4d")}
      >Remove</button>
    </div>
  );
}
export default BasicFigure;
```

/components/FigureList.js



```
import React from "react";
import BasicFigure from "./BasicFigure";

function FigureList({ figures, onRemoveFigure }) {
  return (
    <div style={{ display: "flex", flexWrap: "wrap", gap: "10px" }}>
      {figures.map((figure) => (
        <BasicFigure
          key={figure.id}
          url={figure.url}
          caption={figure.caption}
          onRemove={() => onRemoveFigure(figure.id)}
        />
      ))}
    </div>
  );
}

export default FigureList;
```

Verification

URL	http://localhost:3000/
Browser View	

Explanation

- **App.js**
  - <Pgm5 /> → Renders the Pgm5 component inside the App component.
- **Pgm5.js**
  - **function Pgm5():**
    - This function defines the Pgm5 component, which acts as an Image Gallery.
  - **Defining the figures State (Image List)**
    - Initial state (useState) contains an array of two image objects.

        - Each object has id, url, caption
    - **Adding a New Image (addFigure function)**
        - Creates a new image object:
        - Assigns a unique id (figures.length + 1).
        - Generates a new random image URL using Picsum.
        - Sets a caption like "Image 3", "Image 4", etc.
    - **Removing an Image (removeFigure function)**
        - Filters out the image with the given id.
        - Updates the state (setFigures) with the remaining images.

- **FigureList.js**
    - FigureList is a reusable component for displaying a list of images.
    - Props:
        - figures: An array of image objects (with id, url, and caption).
        - onRemoveFigure: A function to remove an image when called.
        - Clicking "Remove" calls onRemove(), which triggers onRemoveFigure(figure.id) in FigureList.
- **BasicFigure.js**
    - Receives 3 props: url, caption, onRemove

Lab Program #6

Objective

- Design and implement a React Form that collects user input for name, email, and password. Form Fields are Name, Email, Password. Ensure all fields are filled before allowing form submission.Validate the email field to ensure it follows the correct email format (e.g., example@domain.com). Optionally enforce a minimum password length or complexity. Display error messages for invalid or missing inputs. Provide visual cues (e.g., red borders) to highlight invalid fields. Prevent form submission until all fields pass validation. Log or display the entered data upon successful submission (optional). Add a "Show Password" toggle for the password field. Implement client-side sanitization to ensure clean input.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

```
npm start
```

- Verify the functionality.
- Go through the Code explanation to understand it.

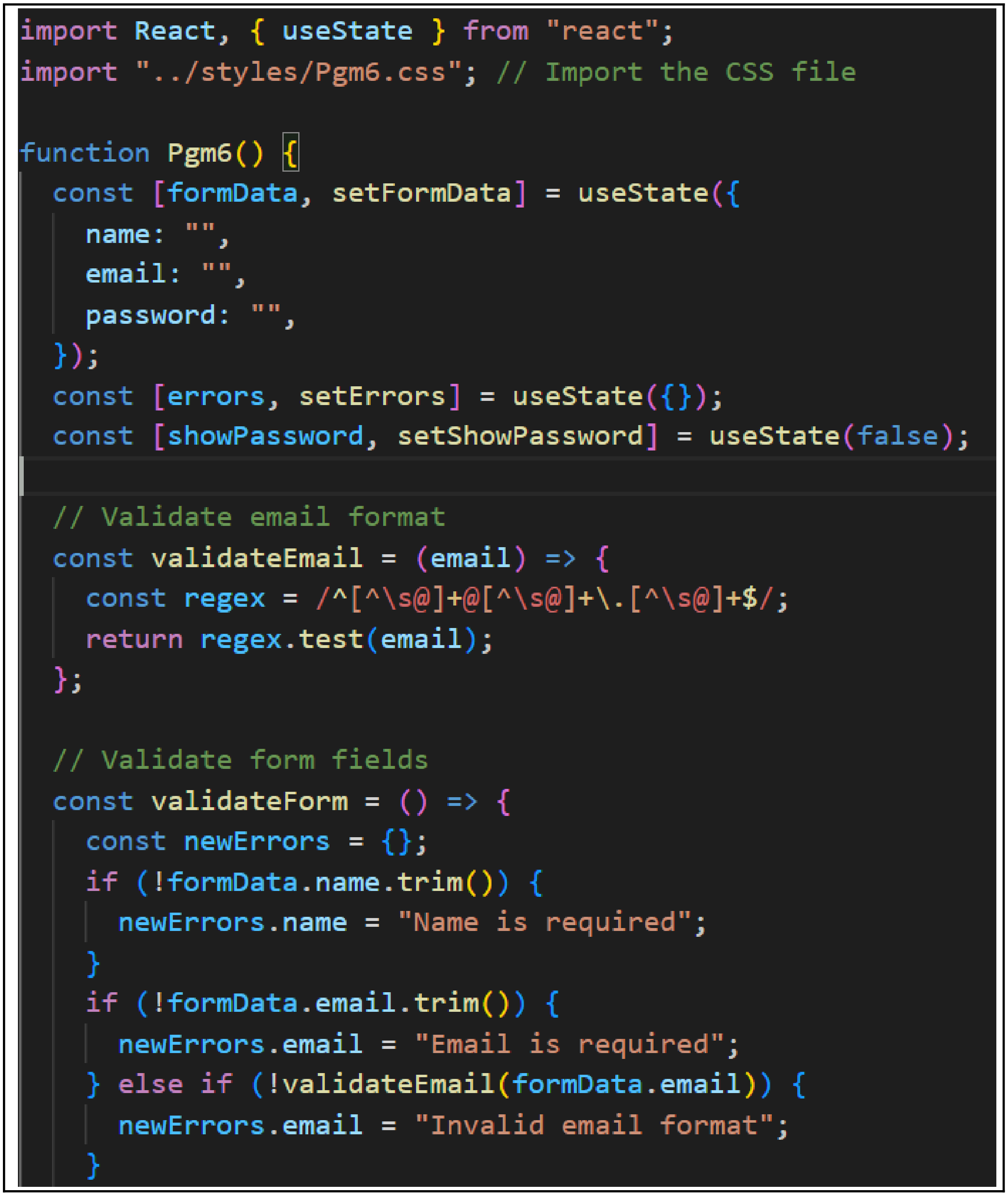
Files

File name	Changes
App.js	Include Pgm6 component.
programs/Pgm6.js	Create Program specific content.
styles/Pgm6.css	New CSS file.

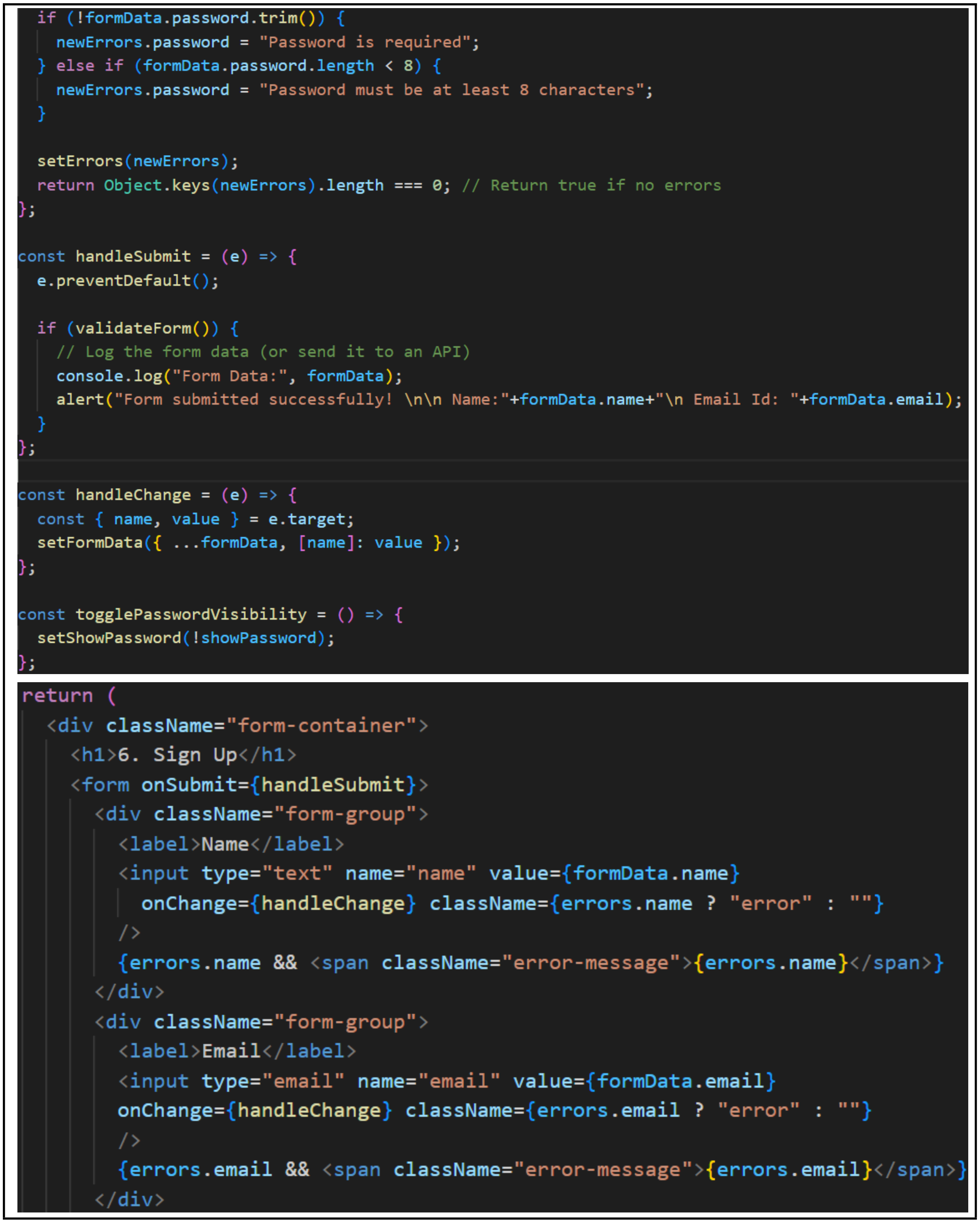
Code

```
// App.js
import React  from "react";
import Pgm6 from "./programs/Pgm6";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm6/>
    </div>
  );
}

export default App;
```

/programs/Pgm6.js

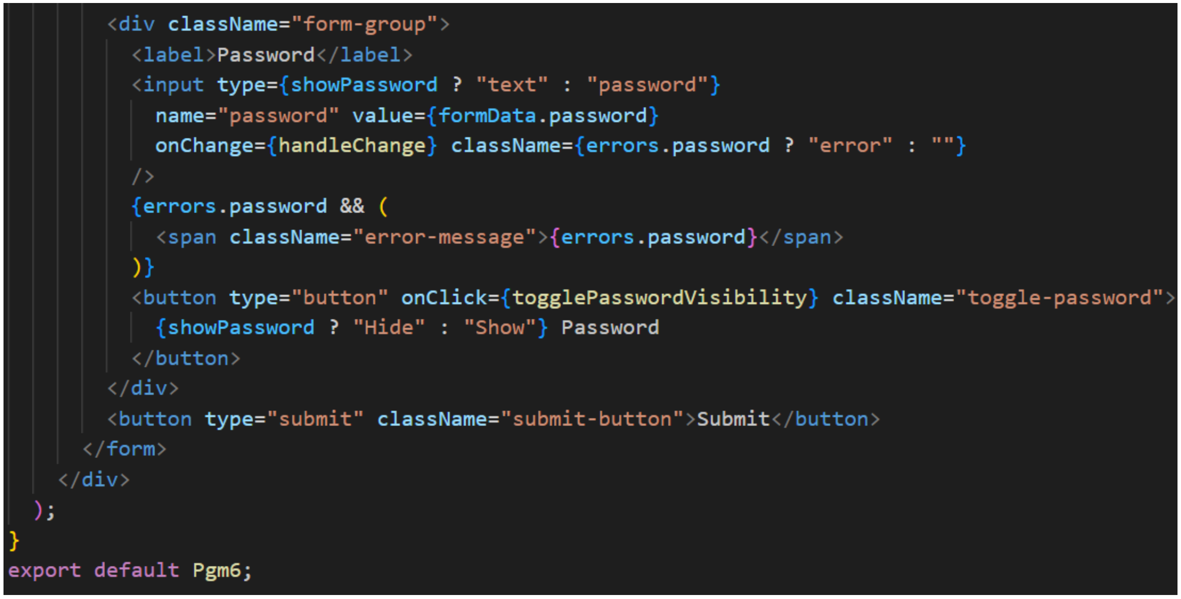


```
import React, { useState } from "react";
import "../styles/Pgm6.css"; // Import the CSS file

function Pgm6() {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
  });
  const [errors, setErrors] = useState({});
  const [showPassword, setShowPassword] = useState(false);

  // Validate email format
  const validateEmail = (email) => {
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return regex.test(email);
  };

  // Validate form fields
  const validateForm = () => {
    const newErrors = {};
    if (!formData.name.trim()) {
      newErrors.name = "Name is required";
    }
    if (!formData.email.trim()) {
      newErrors.email = "Email is required";
    } else if (!validateEmail(formData.email)) {
      newErrors.email = "Invalid email format";
    }
```



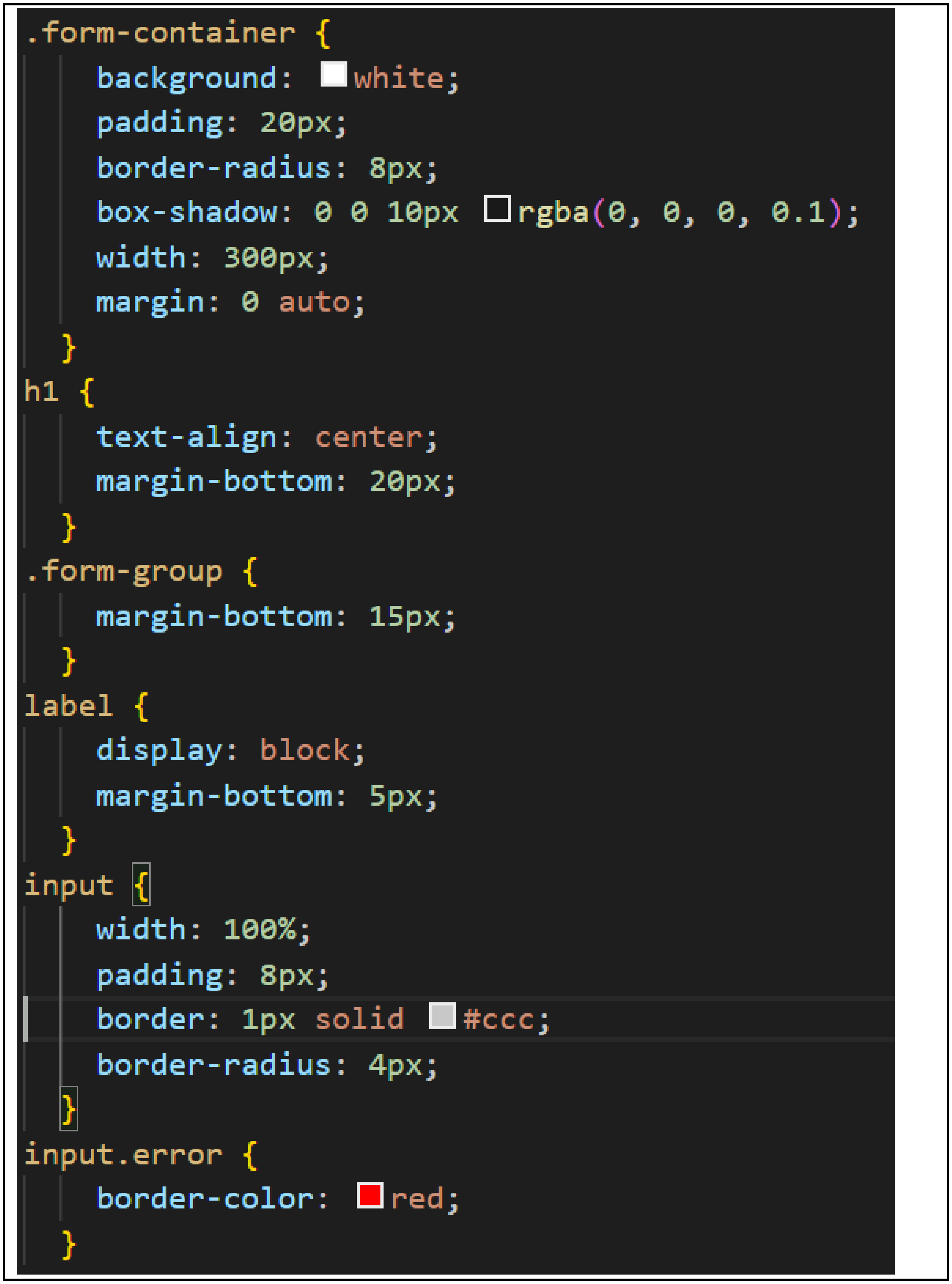
```
  if (!formData.password.trim()) {
    newErrors.password = "Password is required";
  } else if (formData.password.length < 8) {
    newErrors.password = "Password must be at least 8 characters";
  }

  setErrors(newErrors);
  return Object.keys(newErrors).length === 0; // Return true if no errors
};

const handleSubmit = (e) => {
  e.preventDefault();

  if (validateForm()) {
    // Log the form data (or send it to an API)
    console.log("Form Data:", formData);
    alert("Form submitted successfully! \n\n Name:"+formData.name+"\n Email Id: "+formData.email);
  }
};

const handleChange = (e) => {
  const { name, value } = e.target;
  setFormData({ ...formData, [name]: value });
};

const togglePasswordVisibility = () => {
  setShowPassword(!showPassword);
};
```

```
return (
  <div className="form-container">
    <h1>6. Sign Up</h1>
    <form onSubmit={handleSubmit}>
      <div className="form-group">
        <label>Name</label>
        <input type="text" name="name" value={formData.name}
          onChange={handleChange} className={errors.name ? "error" : ""}
        />
        {errors.name && <span className="error-message">{errors.name}</span>}
      </div>
      <div className="form-group">
        <label>Email</label>
        <input type="email" name="email" value={formData.email}
        onChange={handleChange} className={errors.email ? "error" : ""}
        />
        {errors.email && <span className="error-message">{errors.email}</span>}
      </div>
```

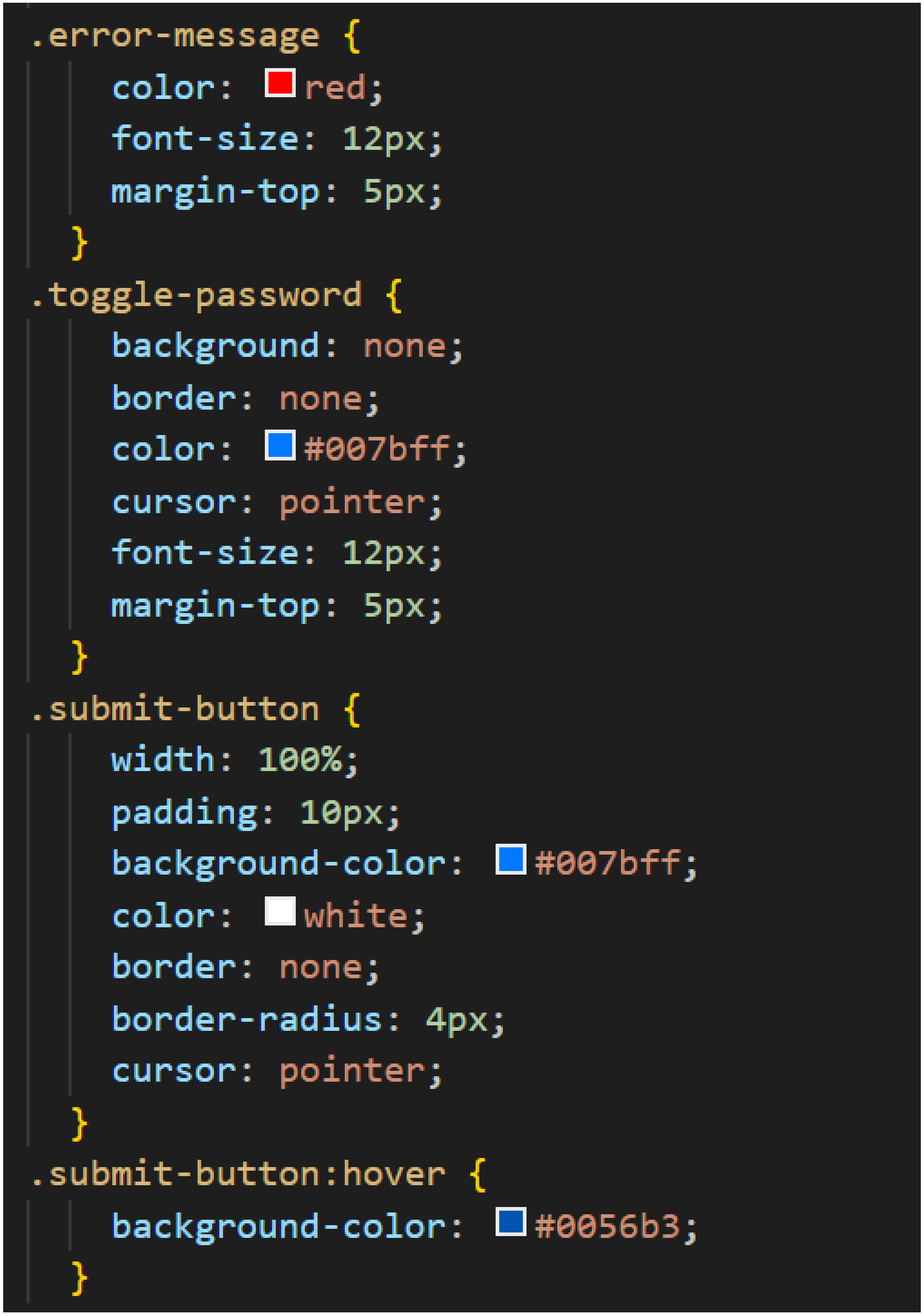


```
        <div className="form-group">
          <label>Password</label>
          <input type={showPassword ? "text" : "password"}
            name="password" value={formData.password}
            onChange={handleChange} className={errors.password ? "error" : ""}
          />
          {errors.password && (
            <span className="error-message">{errors.password}</span>
          )}
          <button type="button" onClick={togglePasswordVisibility} className="toggle-password">
            {showPassword ? "Hide" : "Show"} Password
          </button>
        </div>
        <button type="submit" className="submit-button">Submit</button>
      </form>
    </div>
  );
}
export default Pgm6;
```

/styles/Pgm6.css



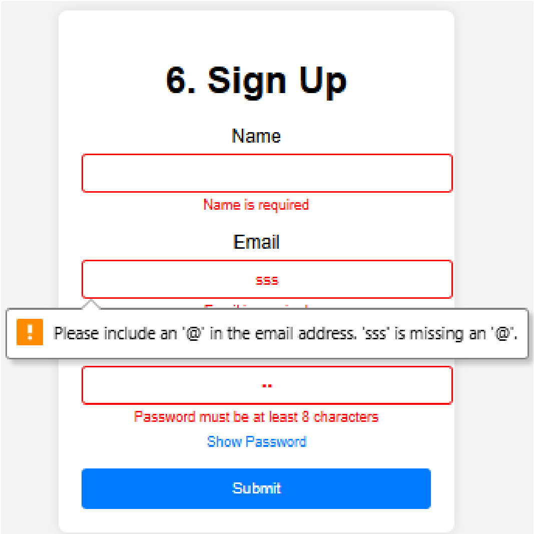
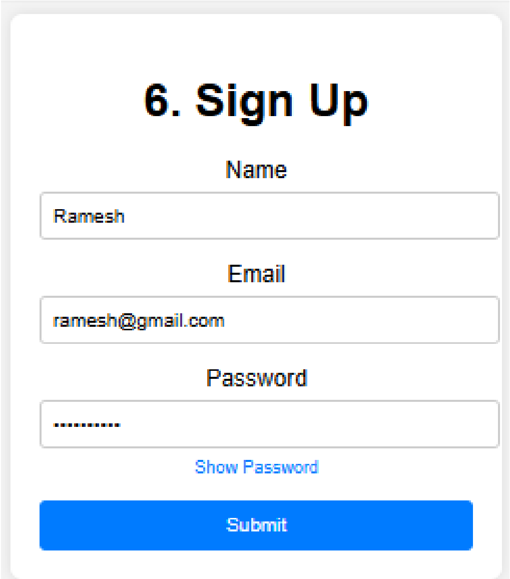
```css
.form-container {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 300px;
    margin: 0 auto;
  }
h1 {
    text-align: center;
    margin-bottom: 20px;
  }
.form-group {
    margin-bottom: 15px;
  }
label {
    display: block;
    margin-bottom: 5px;
  }
input {
    width: 100%;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 4px;
  }
input.error {
    border-color: red;
  }
```



```css
.error-message {
    color: red;
    font-size: 12px;
    margin-top: 5px;
  }
.toggle-password {
    background: none;
    border: none;
    color: #007bff;
    cursor: pointer;
    font-size: 12px;
    margin-top: 5px;
  }
.submit-button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }
.submit-button:hover {
    background-color: #0056b3;
  }
```

Verification

URL	http://localhost:3000/
Browser View	Validation failures:

	 6. Sign Up Name Name is required Email sss Please include an '@' in the email address. 'sss' is missing an '@'. •• Password must be at least 8 characters Show Password Submit Success:  6. Sign Up Name Ramesh Email ramesh@gmail.com Password •••••••••• Show Password Submit

Explanation

- **App.js**
    - <Pgm6 /> → Renders the Pgm6 component inside the App component.
- **Pgm6.js**
    - **Helper Functions**
        - Email Validation : validateEmail
            - Uses a regular expression (regex) to check if the email follows the correct format (example@domain.com).
        - Form Validation : validateForm
            - Checks for missing values (name, email, password).
            - Validates email format using validateEmail().
            - Ensures password has at least 8 characters.

        - If errors are found, they are stored in errors.
        - Returns true if the form is valid (no errors).

    - **Handling Form Submission : handleSubmit**
        - Prevents page reload (e.preventDefault()).
        - Calls validateForm() to check for errors.
        - If valid, logs form data and displays an alert message.
    - **Handling Input Changes : handleChange**
        - Updates formData dynamically when a user types into an input field.
    - **Toggling Password Visibility : togglePasswordVisibility**
        - Flips the showPassword state (true ↔ false).
    - **Rendering the Form**
        - The form is wrapped inside a container div (form-container).
        - onSubmit={handleSubmit} ensures validation runs when the form is submitted.
    - **Validation handling**
        - onChange={handleChange} updates the state on user input.
        - If an error exists, the class "error" is added to highlight the field.
        - The error message is shown if validation fails.

Lab Program #7

Objective

- Develop a React Application featuring a ProfileCard component to display a user's profile information, including their name, profile picture, and bio. The component should demonstrate flexibility by utilizing both external CSS and inline styling for its design. Display the following information: Profile picture, User's name, A short bio or description Use an external CSS file for overall structure and primary styles, such as layout, colors, and typography. Apply inline styles for dynamic or specific styling elements, such as background colors or alignment. Design the ProfileCard to be visually appealing and responsive. Ensure the profile picture is displayed as a circle, and the name and bio are appropriately styled. Add hover effects or animations to enhance interactivity. Allow the background color of the card to change dynamically based on a prop or state.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

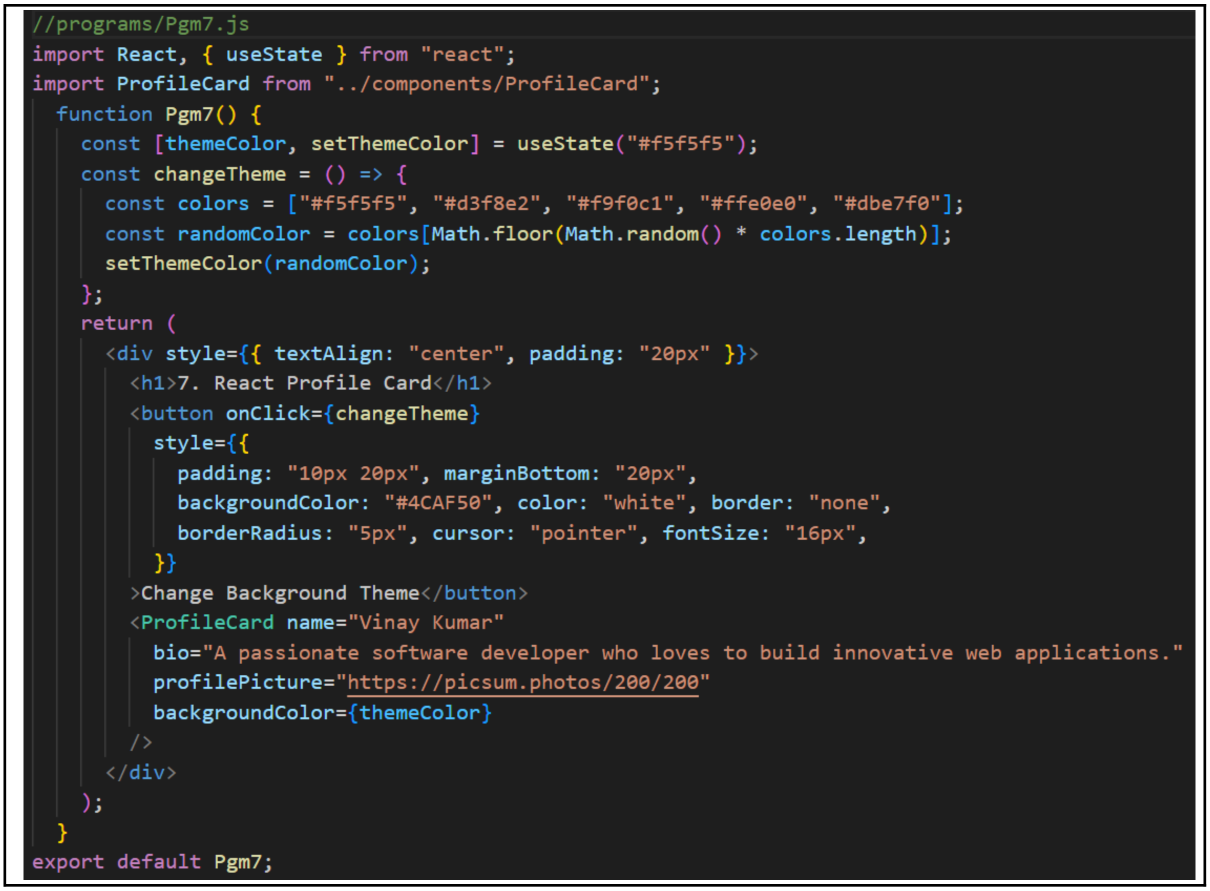
  ```
  npm start
  ```
- Verify the functionality.
- Go through the Code explanation to understand it.

Files

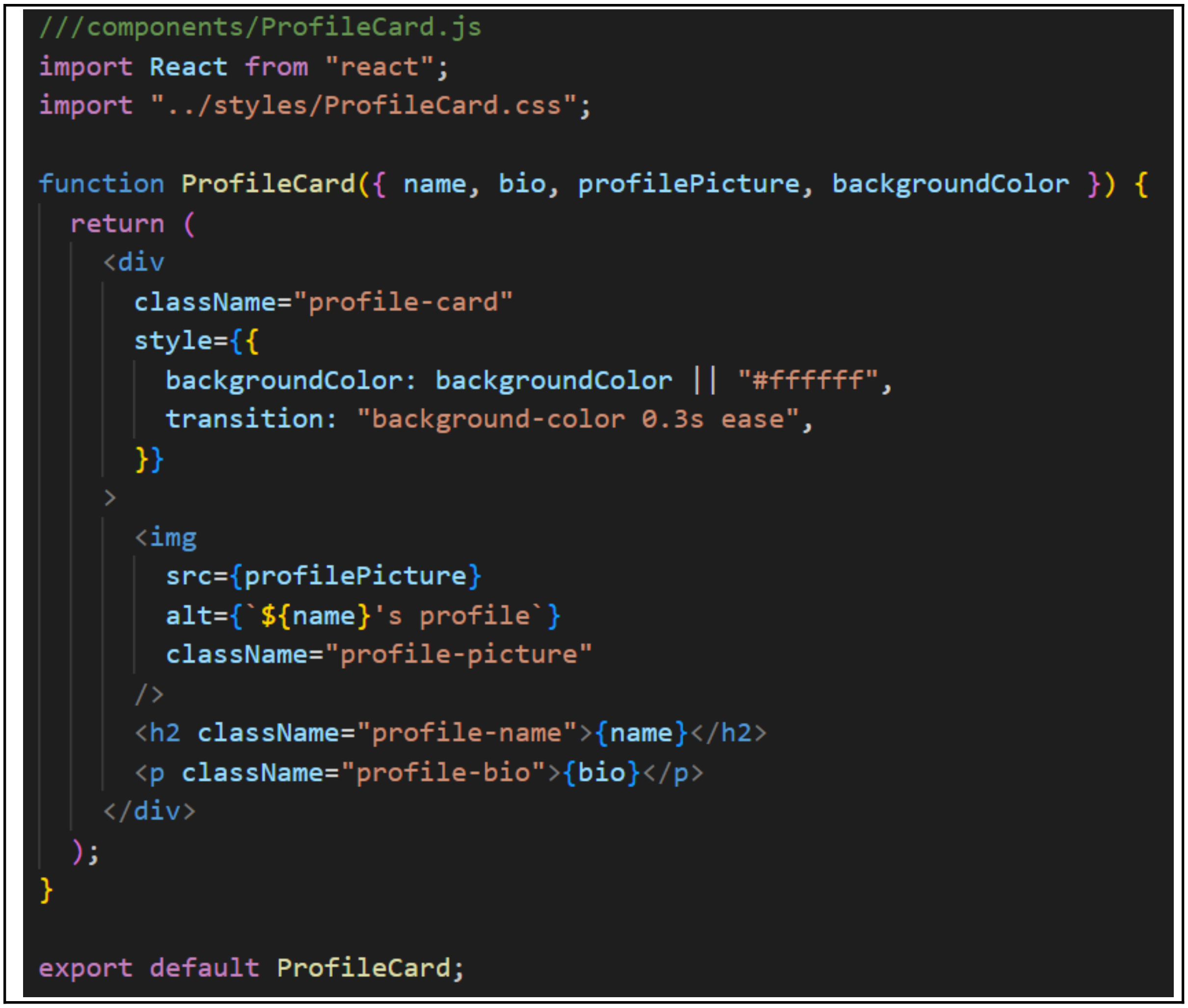
File name	Changes
App.js	Include Pgm7 component.
programs/Pgm7.js	Create Program specific content.
/components/ProfileCard.js	New component
/styles/ProfileCard.css	New css file

Code

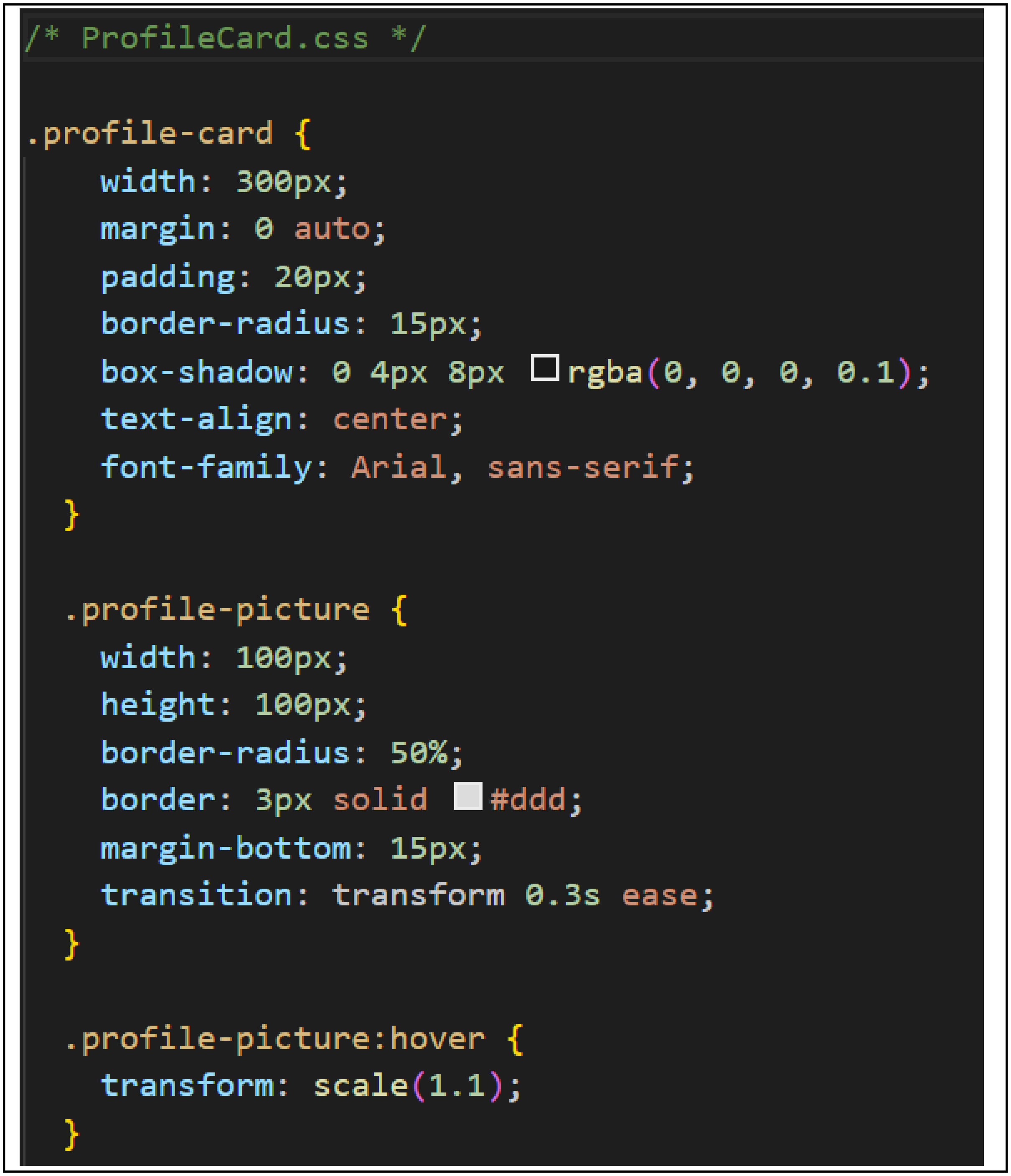
```
// App.js
import React  from "react";
import Pgm7 from "./programs/Pgm7";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm7/>
    </div>
  );
}

export default App;
```



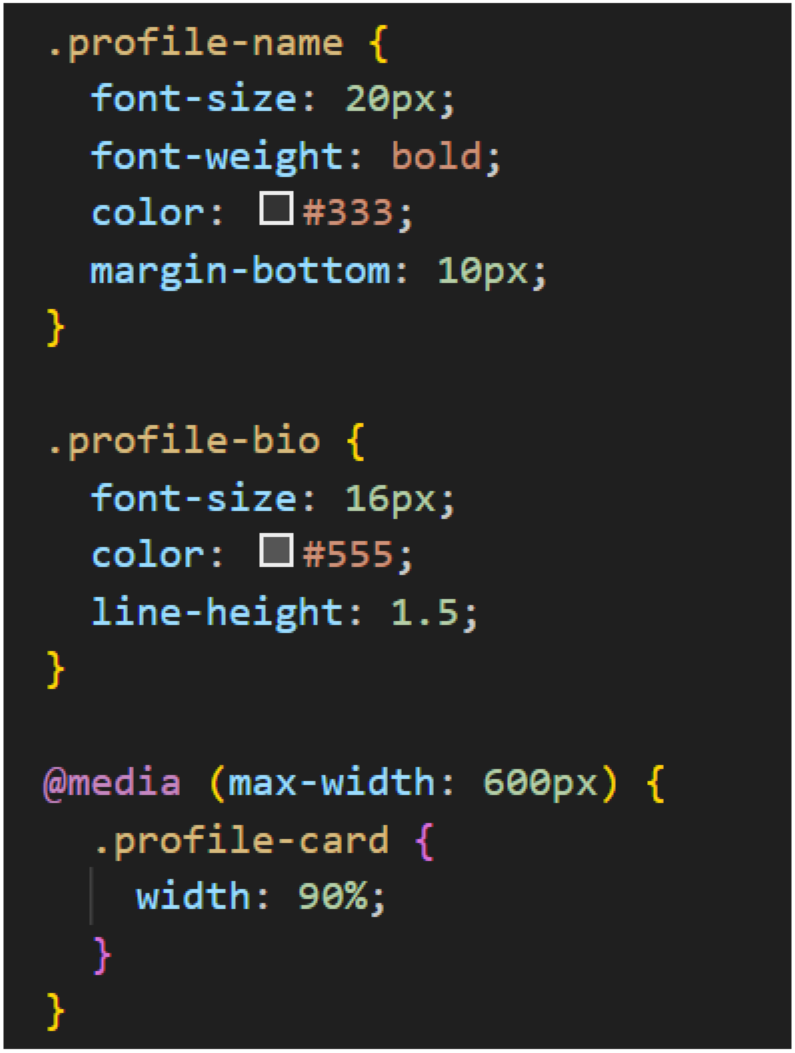
```
//programs/Pgm7.js
import React, { useState } from "react";
import ProfileCard from "../components/ProfileCard";
  function Pgm7() {
    const [themeColor, setThemeColor] = useState("#f5f5f5");
    const changeTheme = () => {
      const colors = ["#f5f5f5", "#d3f8e2", "#f9f0c1", "#ffe0e0", "#dbe7f0"];
      const randomColor = colors[Math.floor(Math.random() * colors.length)];
      setThemeColor(randomColor);
    };
    return (
      <div style={{ textAlign: "center", padding: "20px" }}>
        <h1>7. React Profile Card</h1>
        <button onClick={changeTheme}
          style={{
            padding: "10px 20px", marginBottom: "20px",
            backgroundColor: "#4CAF50", color: "white", border: "none",
            borderRadius: "5px", cursor: "pointer", fontSize: "16px",
          }}
        >Change Background Theme</button>
        <ProfileCard name="Vinay Kumar"
          bio="A passionate software developer who loves to build innovative web applications."
          profilePicture="https://picsum.photos/200/200"
          backgroundColor={themeColor}
        />
      </div>
    );
  }
export default Pgm7;
```



```
///components/ProfileCard.js
import React from "react";
import "../styles/ProfileCard.css";

function ProfileCard({ name, bio, profilePicture, backgroundColor }) {
  return (
    <div
      className="profile-card"
      style={{
        backgroundColor: backgroundColor || "#ffffff",
        transition: "background-color 0.3s ease",
      }}
    >
      <img
        src={profilePicture}
        alt={`${name}'s profile`}
        className="profile-picture"
      />
      <h2 className="profile-name">{name}</h2>
      <p className="profile-bio">{bio}</p>
    </div>
  );
}

export default ProfileCard;
```



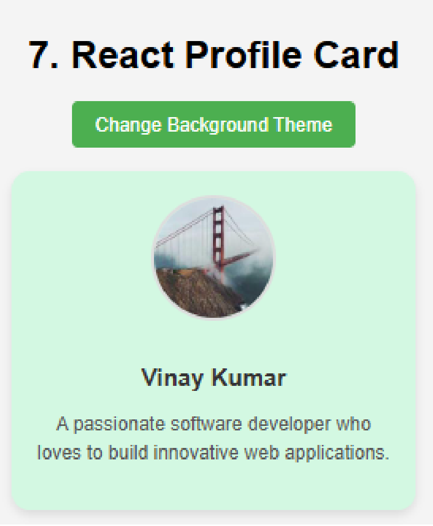
```css
/* ProfileCard.css */

.profile-card {
    width: 300px;
    margin: 0 auto;
    padding: 20px;
    border-radius: 15px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
    font-family: Arial, sans-serif;
  }

  .profile-picture {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    border: 3px solid #ddd;
    margin-bottom: 15px;
    transition: transform 0.3s ease;
  }

  .profile-picture:hover {
    transform: scale(1.1);
  }
```



```css
.profile-name {
  font-size: 20px;
  font-weight: bold;
  color: #333;
  margin-bottom: 10px;
}

.profile-bio {
  font-size: 16px;
  color: #555;
  line-height: 1.5;
}

@media (max-width: 600px) {
  .profile-card {
    width: 90%;
  }
}
```

Verification

URL	http://localhost:3000/

Browser View	 7. React Profile Card Change Background Theme Vinay Kumar A passionate software developer who loves to build innovative web applications.

Explanation

- **App.js**
  - <Pgm7 /> → Renders the Pgm7 component inside the App component.
- **Pgm7.js**
  - **Function to Change Theme Color: changeTheme**
    - Defines an array of colors.
    - Randomly selects a color from the array using Math.random().
    - Updates themeColor state using setThemeColor().
  - **ProfileCard Component with Dynamic Background**
    - Passes props to ProfileCard:
      - name: "Vinay Kumar"
      - bio: "A passionate software developer..."
      - profilePicture: Random 200x200 image from Picsum.
      - backgroundColor: Uses themeColor from state, which changes dynamically.
- **ProfileCard.js**
  - **ProfileCard Function Definition and Props Usage**
    - Accepts four props:
      - name: User's name (string).
      - bio: Short description about the user.
      - profilePicture: URL of the user's profile picture.
      - backgroundColor: Background color for the card (dynamic).

Lab Program #8

Objective

- Develop a Reminder Application that allows users to efficiently manage their tasks. The application should include the following functionalities: Provide a form where users can add tasks along with due dates. The form includes task name,Due date,An optional description. Display a list of tasks dynamically as they are added. Show relevant details like task name, due date, and completion status. Include a filter option to allow users to view all Tasks and Display all tasks regardless of status. Show only tasks marked as completed. Show only tasks that are not yet completed.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

```
npm start
```

- Verify the functionality.
- Go through the Code explanation to understand it.

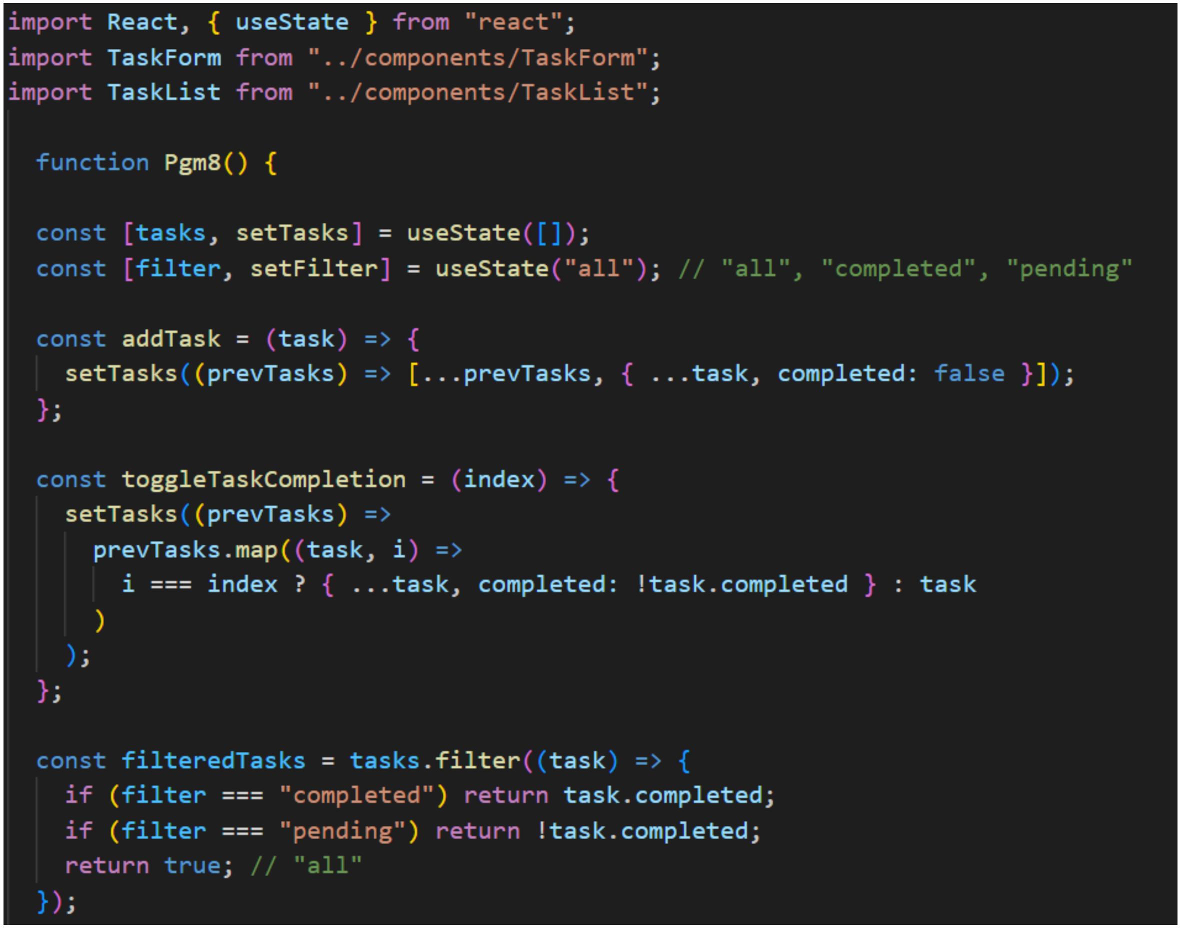
Files

File name	Changes
App.js	Include Pgm8 component.
programs/Pgm8.js	Create Program specific content.
components/TaskForm.js components/TaskList.js	New components

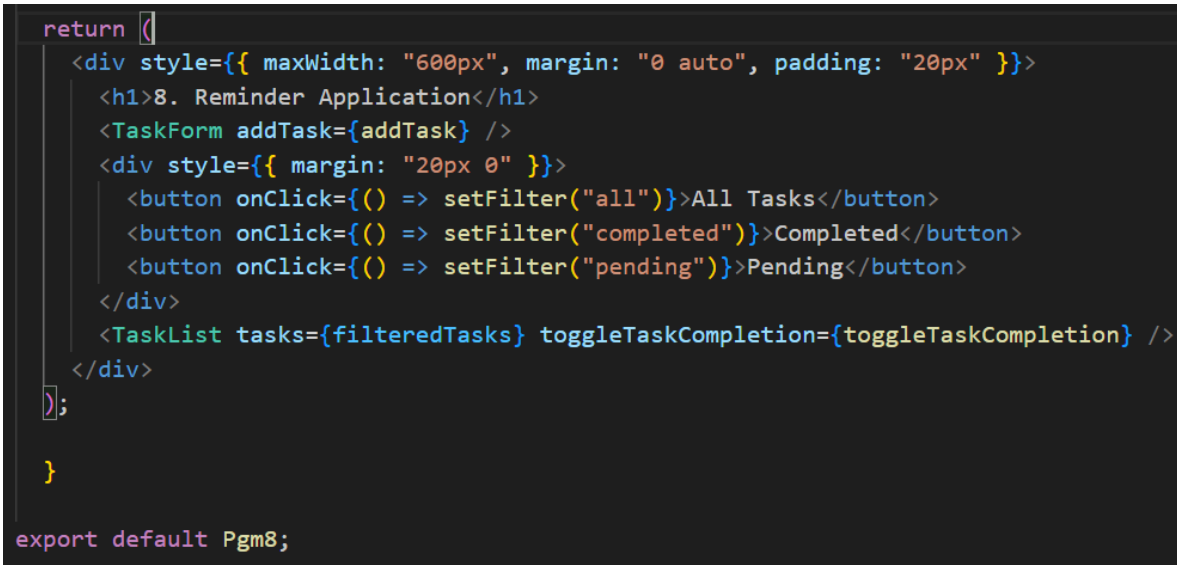
Code

```
// App.js
import React  from "react";
import Pgm8 from "./programs/Pgm8";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm8/>
    </div>
  );
}

export default App;
```

```
/programs/Pgm8.js
```

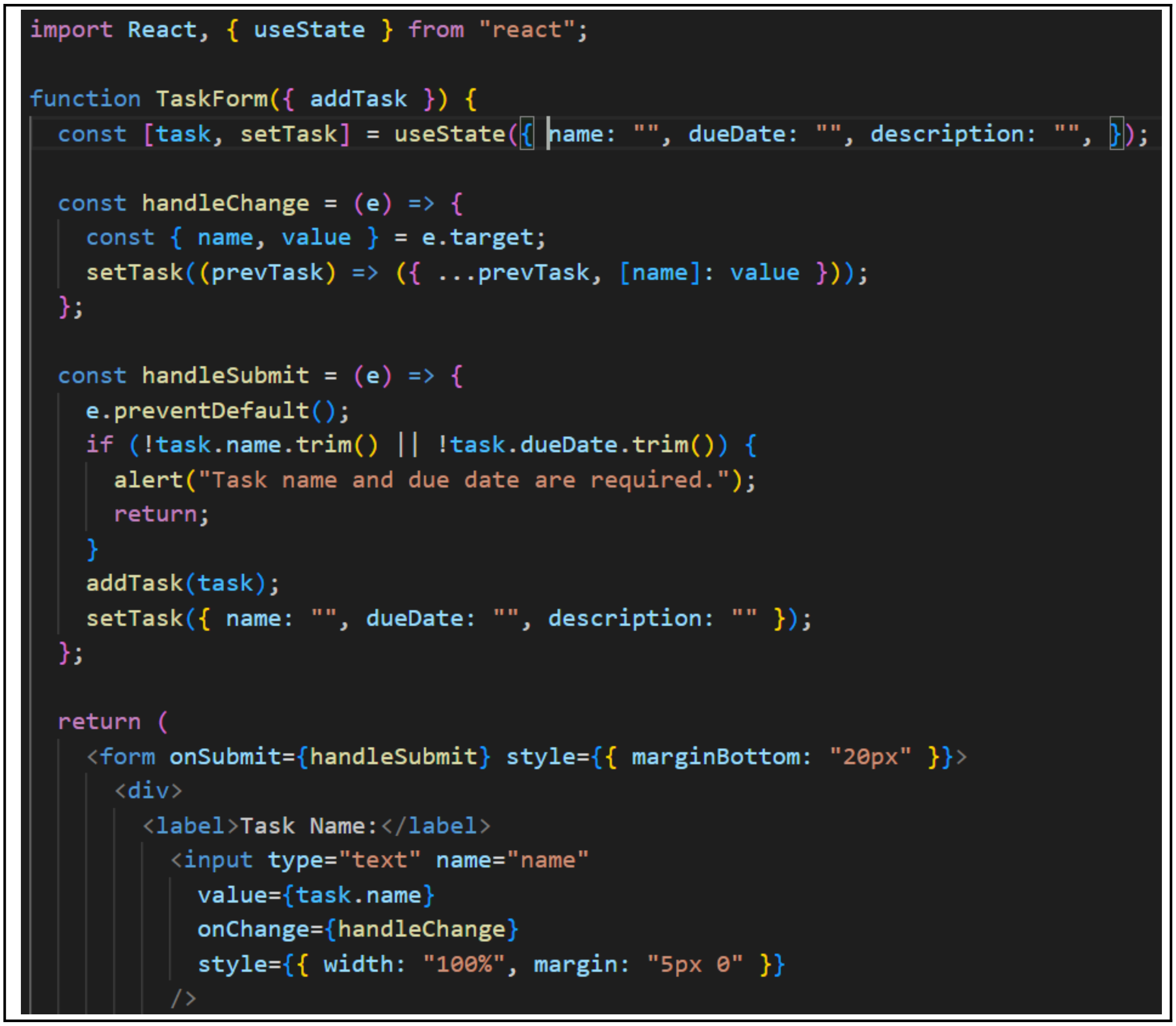


```
import React, { useState } from "react";
import TaskForm from "../components/TaskForm";
import TaskList from "../components/TaskList";

  function Pgm8() {

  const [tasks, setTasks] = useState([]);
  const [filter, setFilter] = useState("all"); // "all", "completed", "pending"

  const addTask = (task) => {
    setTasks((prevTasks) => [...prevTasks, { ...task, completed: false }]);
  };

  const toggleTaskCompletion = (index) => {
    setTasks((prevTasks) =>
      prevTasks.map((task, i) =>
        i === index ? { ...task, completed: !task.completed } : task
      )
    );
  };

  const filteredTasks = tasks.filter((task) => {
    if (filter === "completed") return task.completed;
    if (filter === "pending") return !task.completed;
    return true; // "all"
  });
```

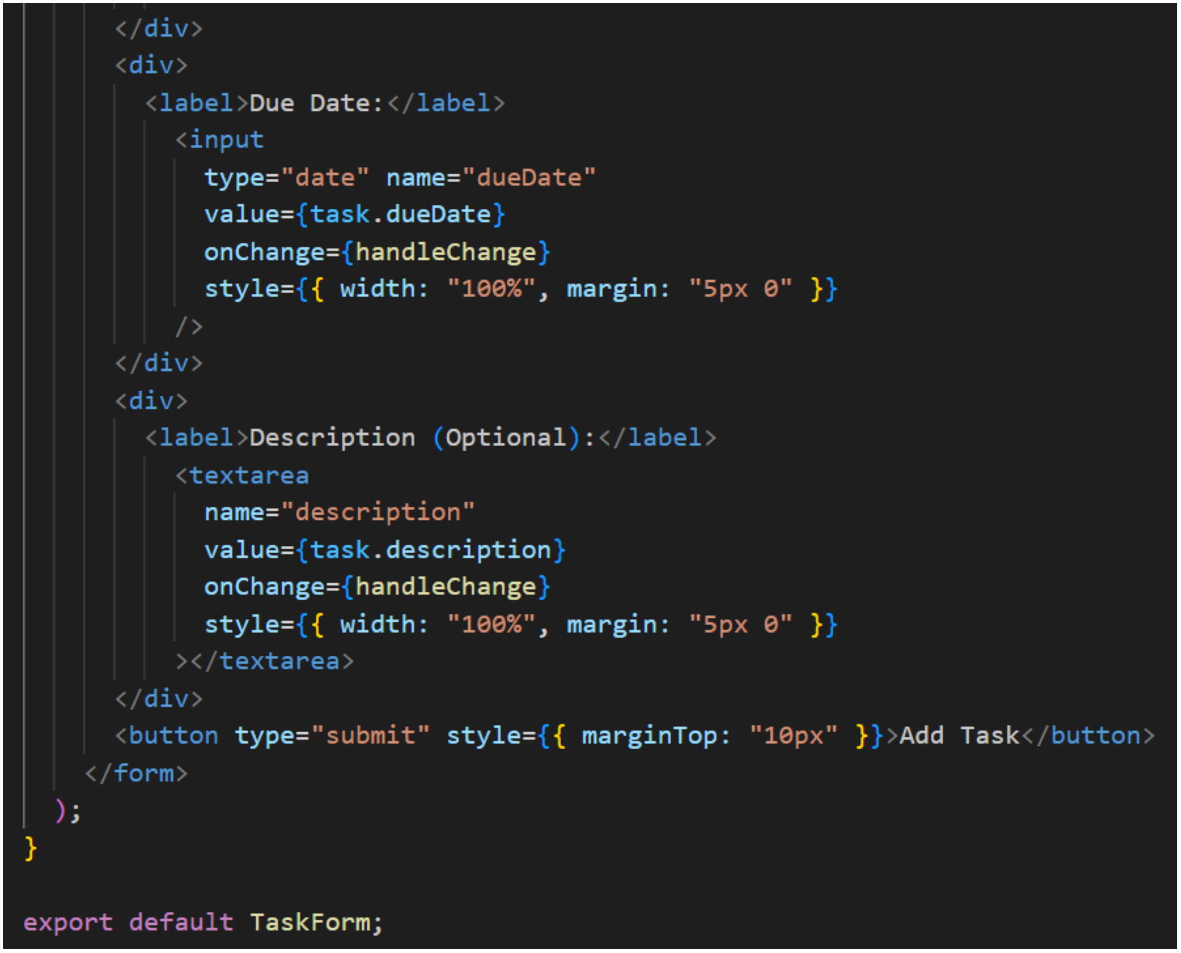


```
  return (
    <div style={{ maxWidth: "600px", margin: "0 auto", padding: "20px" }}>
      <h1>8. Reminder Application</h1>
      <TaskForm addTask={addTask} />
      <div style={{ margin: "20px 0" }}>
        <button onClick={() => setFilter("all")}>All Tasks</button>
        <button onClick={() => setFilter("completed")}>Completed</button>
        <button onClick={() => setFilter("pending")}>Pending</button>
      </div>
      <TaskList tasks={filteredTasks} toggleTaskCompletion={toggleTaskCompletion} />
    </div>
  );

  }

export default Pgm8;
```

/components/TaskForm.js

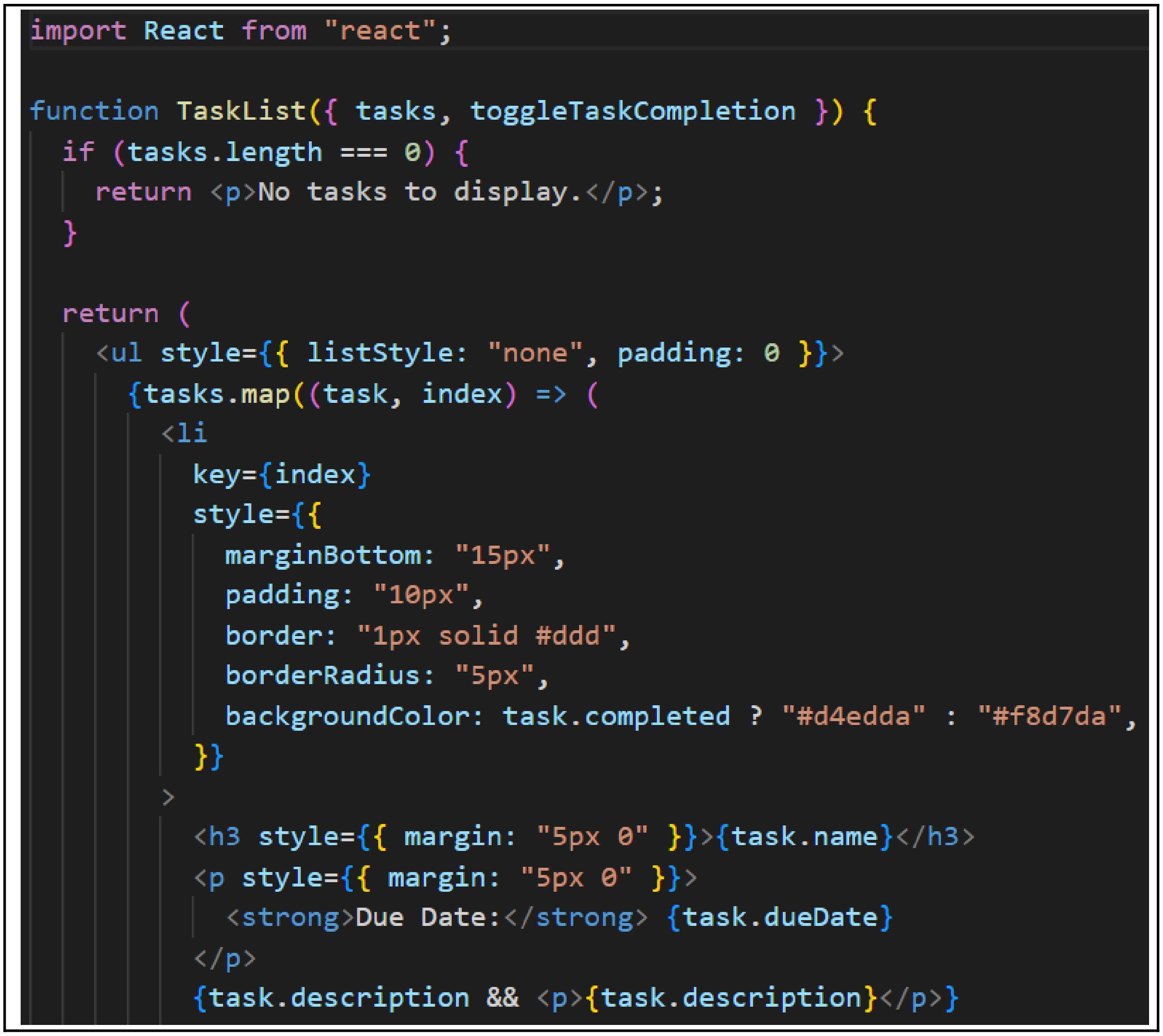


```
import React, { useState } from "react";

function TaskForm({ addTask }) {
  const [task, setTask] = useState({ name: "", dueDate: "", description: "", });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setTask((prevTask) => ({ ...prevTask, [name]: value }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!task.name.trim() || !task.dueDate.trim()) {
      alert("Task name and due date are required.");
      return;
    }
    addTask(task);
    setTask({ name: "", dueDate: "", description: "" });
  };

  return (
    <form onSubmit={handleSubmit} style={{ marginBottom: "20px" }}>
      <div>
        <label>Task Name:</label>
          <input type="text" name="name"
            value={task.name}
            onChange={handleChange}
            style={{ width: "100%", margin: "5px 0" }}
          />
```

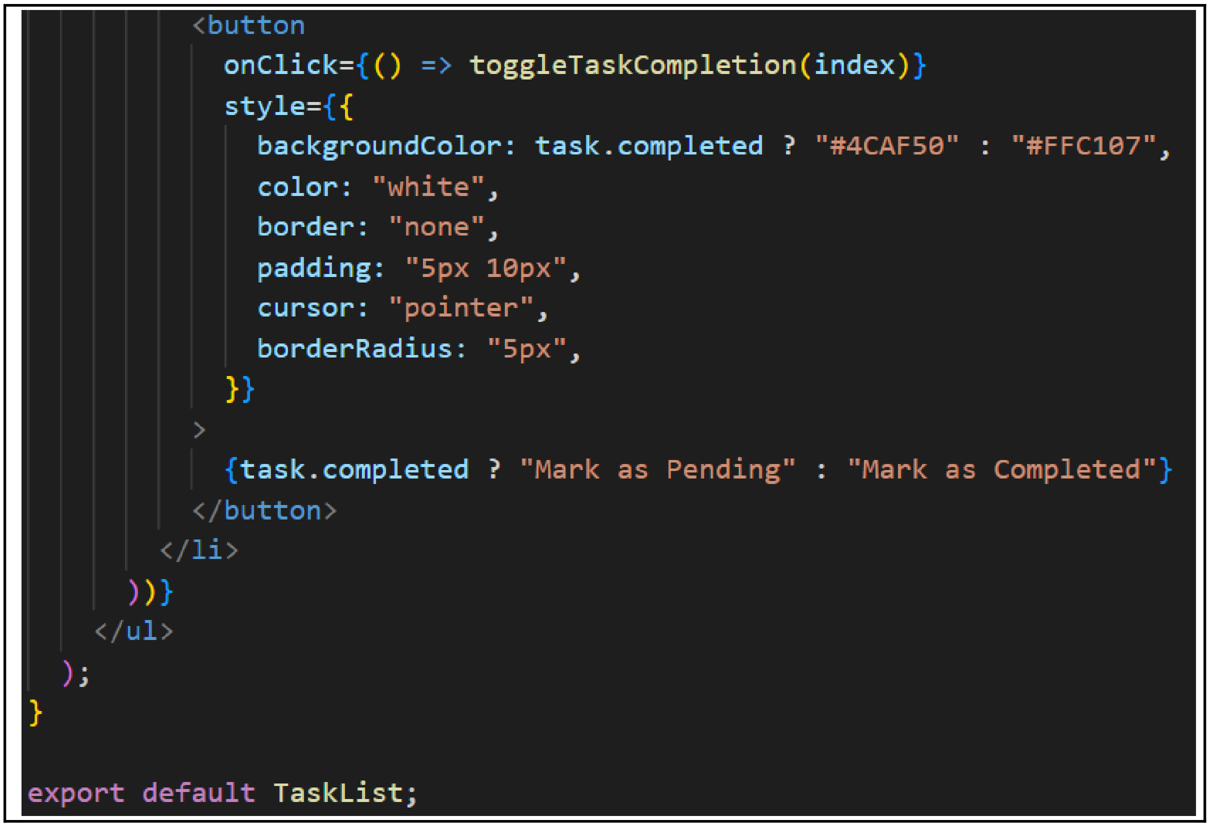


```
      </div>
      <div>
        <label>Due Date:</label>
          <input
            type="date" name="dueDate"
            value={task.dueDate}
            onChange={handleChange}
            style={{ width: "100%", margin: "5px 0" }}
          />
      </div>
      <div>
        <label>Description (Optional):</label>
          <textarea
            name="description"
            value={task.description}
            onChange={handleChange}
            style={{ width: "100%", margin: "5px 0" }}
          ></textarea>
      </div>
      <button type="submit" style={{ marginTop: "10px" }}>Add Task</button>
    </form>
  );
}

export default TaskForm;
```

/components/TaskList.js



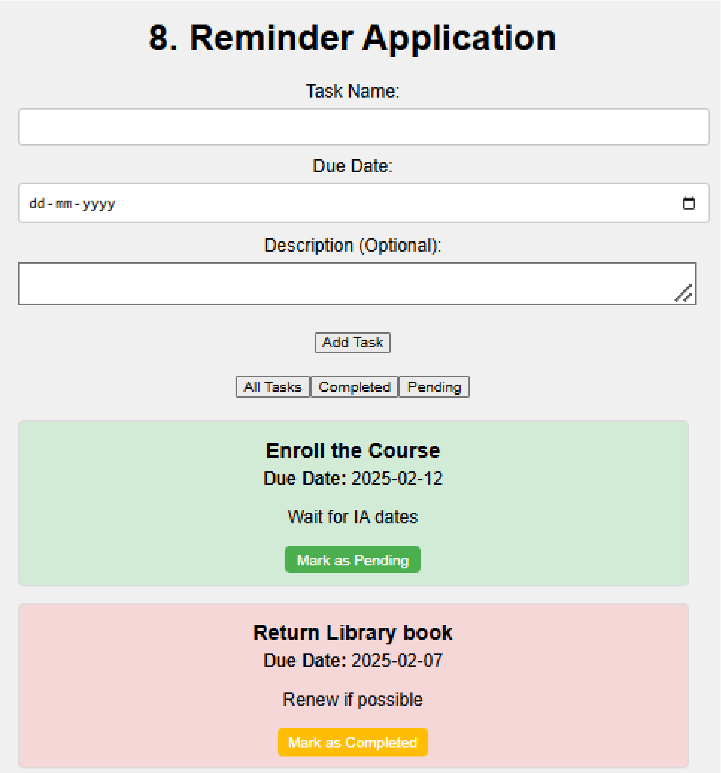
```jsx
import React from "react";

function TaskList({ tasks, toggleTaskCompletion }) {
  if (tasks.length === 0) {
    return <p>No tasks to display.</p>;
  }

  return (
    <ul style={{ listStyle: "none", padding: 0 }}>
      {tasks.map((task, index) => (
        <li
          key={index}
          style={{
            marginBottom: "15px",
            padding: "10px",
            border: "1px solid #ddd",
            borderRadius: "5px",
            backgroundColor: task.completed ? "#d4edda" : "#f8d7da",
          }}
        >
          <h3 style={{ margin: "5px 0" }}>{task.name}</h3>
          <p style={{ margin: "5px 0" }}>
            <strong>Due Date:</strong> {task.dueDate}
          </p>
          {task.description && <p>{task.description}</p>}
```



```
          <button
            onClick={() => toggleTaskCompletion(index)}
            style={{
              backgroundColor: task.completed ? "#4CAF50" : "#FFC107",
              color: "white",
              border: "none",
              padding: "5px 10px",
              cursor: "pointer",
              borderRadius: "5px",
            }}
          >
            {task.completed ? "Mark as Pending" : "Mark as Completed"}
          </button>
        </li>
      ))}
    </ul>
  );
}

export default TaskList;
```

Verification

URL	http://localhost:3000/

Browser View	

8. Reminder Application

Task Name:

Due Date:

dd-mm-yyyy

Description (Optional):

Enroll the Course

Due Date: 2025-02-12

Wait for IA dates

Return Library book

Due Date: 2025-02-07

Renew if possible

Explanation

- **Pgm8.js**
  - **Filters:**
    - Filters the tasks based on the selected filter:
      - "completed" → Shows only completed tasks.
      - "pending" → Shows only pending tasks.
      - "all" → Shows all tasks.
    - The default value is set to “all”
  - **Adding a New Task :addTask**
    - This function adds a new task to the task list.
    - The new task starts as completed: false by default.
  - **Toggling Task Completion**
    - Toggles the completed status of a task when clicked.
    - If the task is completed, it is marked as pending, and vice versa.
  - **Rendering the Task List**
    - Passes the filteredTasks (based on the selected filter) to the TaskList component.
    - Passes the toggleTaskCompletion function to allow tasks to be marked as completed/pending.

- **TaskForm.js**
  - Manages form state using useState to store task details (name, dueDate, description).
  - Handles input changes dynamically using handleChange:
    - Updates the corresponding field (name, dueDate, description) in the state.
  - **Validates user input:**
    - Ensures name and dueDate are not empty before submitting.
    - Alerts the user if required fields are missing.
  - **Submits form data when the user clicks "Add Task":**
    - Calls addTask(task) to send the new task to the parent component.
    - Resets the form fields after submission.
- **TaskList.js**
  - **Displays Task List Dynamically**
    - Accepts tasks as a prop and iterates over them using .map().
    - If there are no tasks, it displays "No tasks to display."
  - **Conditional Styling Based on Task Completion**
    - Completed tasks have a green background (#d4edda).
    - Pending tasks have a red background (#f8d7da).
  - **Toggle Task Completion**
    - Clicking the button toggles task status between Completed and Pending.
    - Uses toggleTaskCompletion(index) from the parent component.

Lab Program #9

Objective

- Design a React application that demonstrates the implementation of routing using the react-router-dom library. The application should include the Navigation Menu: Create a navigation bar with links to three distinct pages, Home, About, Contact. Develop separate components for each page (Home, About, and Contact) with appropriate content to differentiate them. Configure routes using react-router-dom to render the corresponding page component based on the selected link. Use BrowserRouter and Route components for routing. Highlight the active link in the navigation menu to indicate the current page.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

```
npm start
```

- Verify the functionality.
- Go through the Code explanation to understand it.

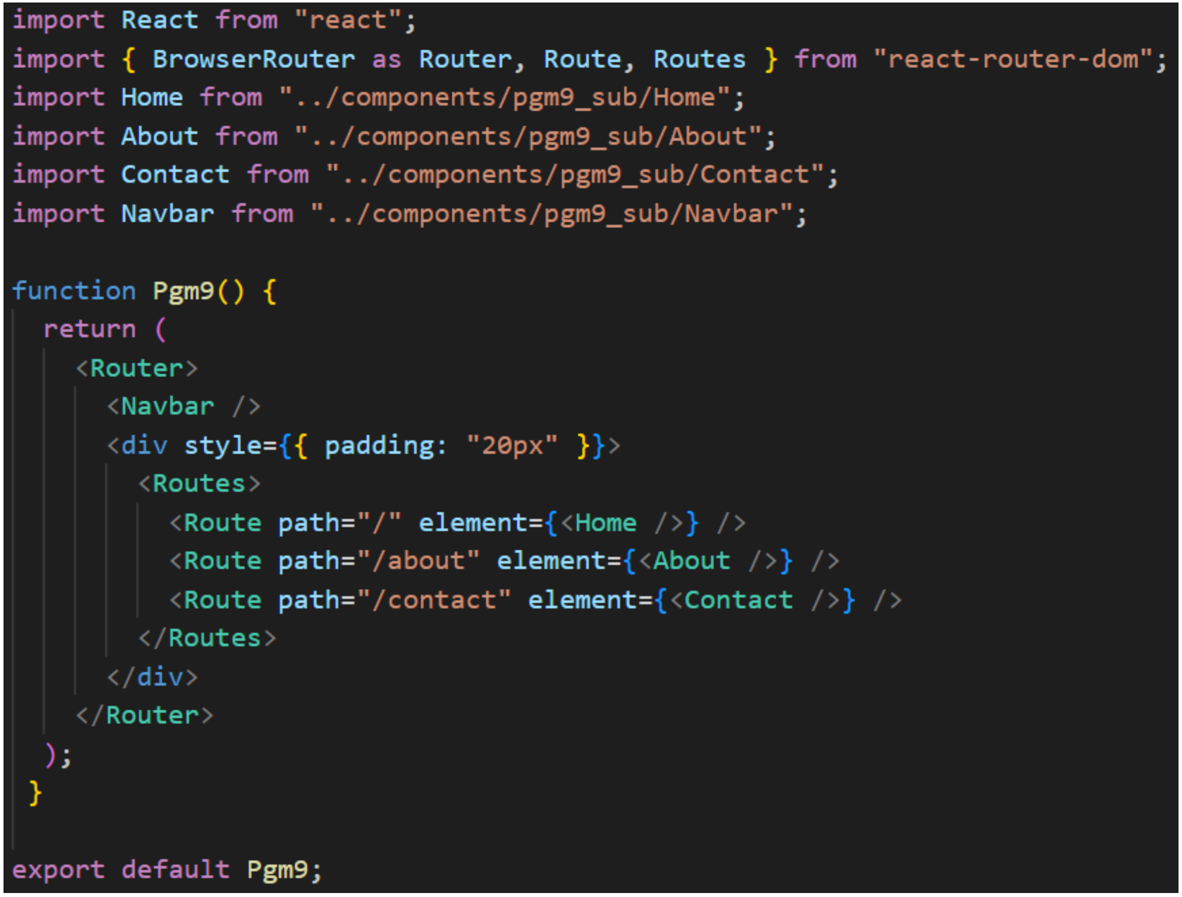
Files

File name	Changes
App.js	Include Pgm9 component.
programs/Pgm9.js	Create Program specific content.
/components/pgm9_sub/Home /components/pgm9_sub/About /components/pgm9_sub/Contact /components/pgm9_sub/Navbar	New components
/styles/Navbar.css	New css file

Code

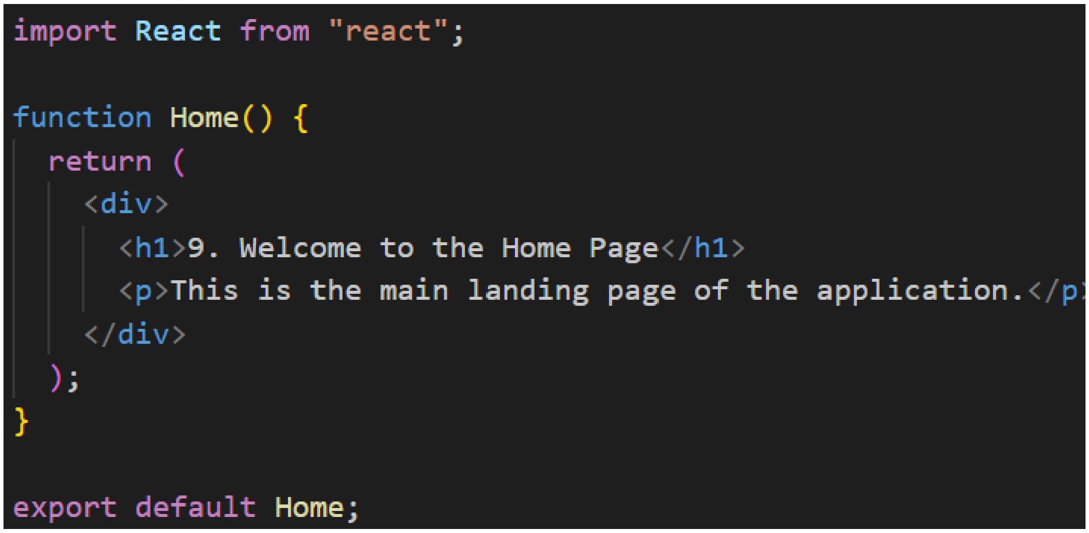
```
// App.js
import React  from "react";
import Pgm9 from "./programs/Pgm9";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm9/>
    </div>
  );
}

export default App;
```

/programs/Pgm9.js



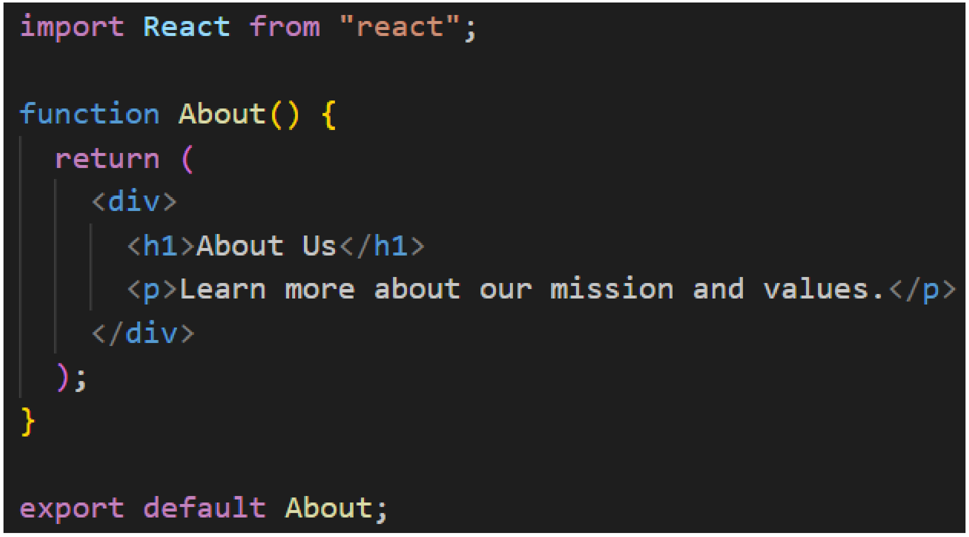
```
import React from "react";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import Home from "../components/pgm9_sub/Home";
import About from "../components/pgm9_sub/About";
import Contact from "../components/pgm9_sub/Contact";
import Navbar from "../components/pgm9_sub/Navbar";

function Pgm9() {
  return (
    <Router>
      <Navbar />
      <div style={{ padding: "20px" }}>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contact" element={<Contact />} />
        </Routes>
      </div>
    </Router>
  );
 }

export default Pgm9;
```

/components/pgm9_sub/Home



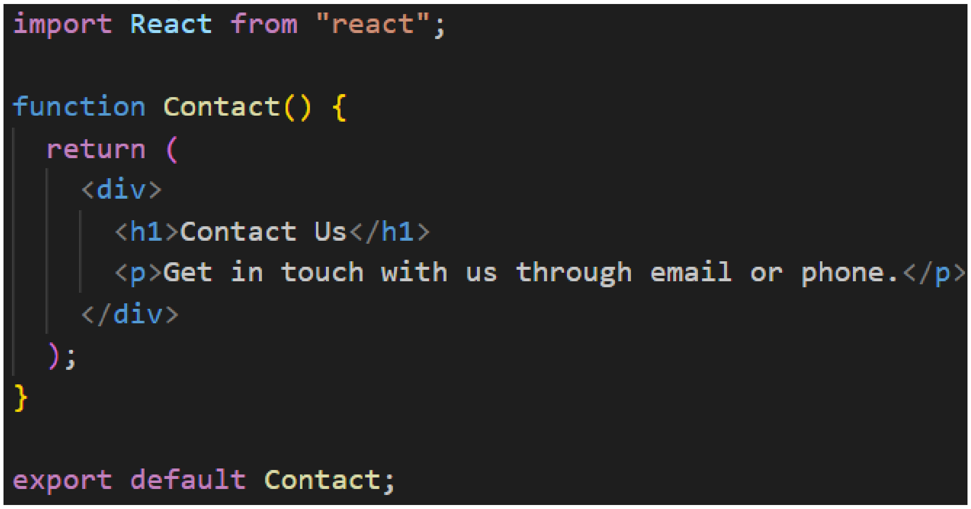
```
import React from "react";

function Home() {
  return (
    <div>
      <h1>9. Welcome to the Home Page</h1>
      <p>This is the main landing page of the application.</p>
    </div>
  );
}

export default Home;
```

/components/pgm9_sub/About



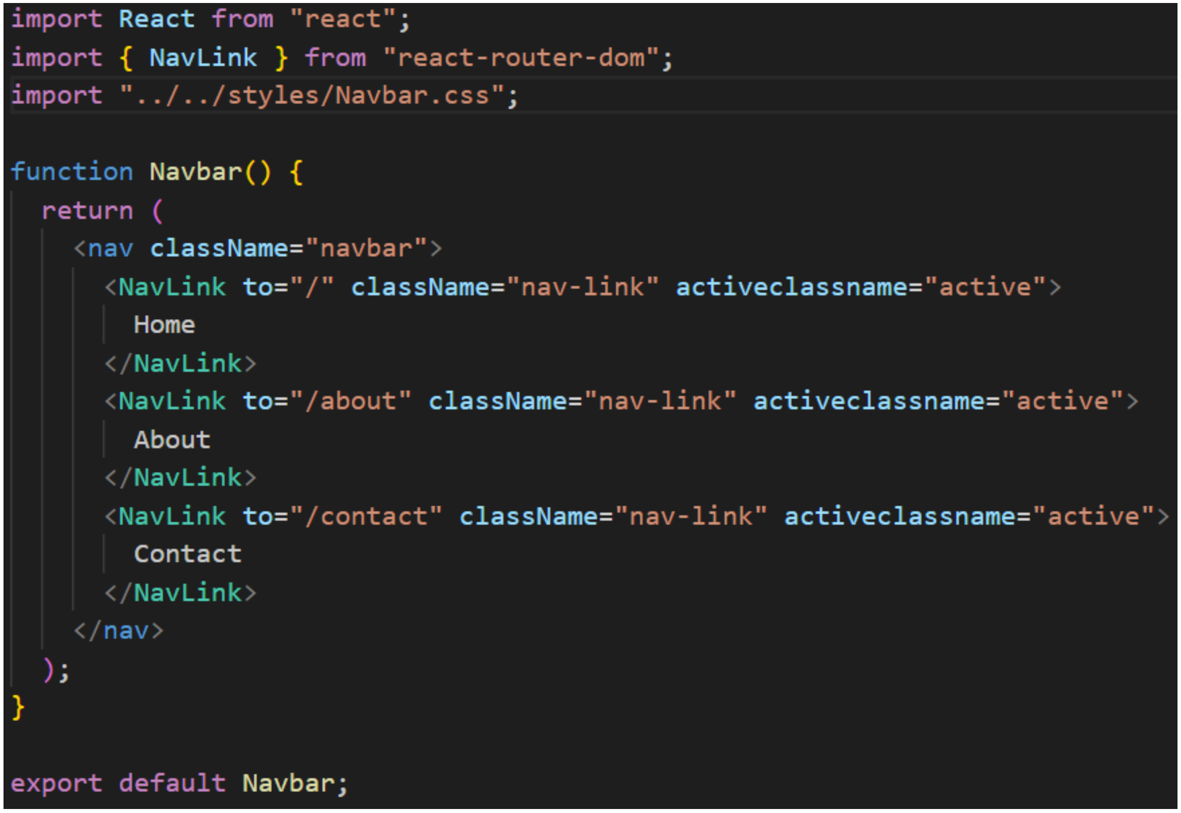
```
import React from "react";

function About() {
  return (
    <div>
      <h1>About Us</h1>
      <p>Learn more about our mission and values.</p>
    </div>
  );
}

export default About;
```

/components/pgm9_sub/Contact



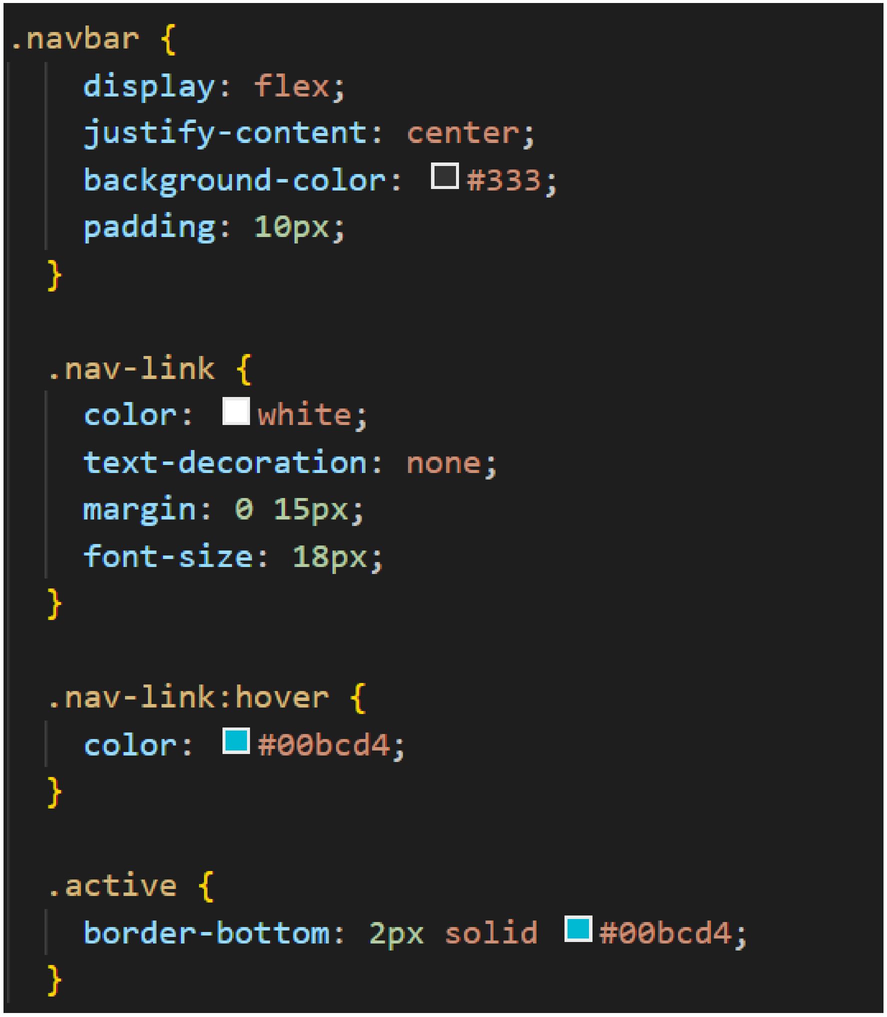
```
import React from "react";

function Contact() {
  return (
    <div>
      <h1>Contact Us</h1>
      <p>Get in touch with us through email or phone.</p>
    </div>
  );
}

export default Contact;
```

/components/pgm9_sub/Navbar



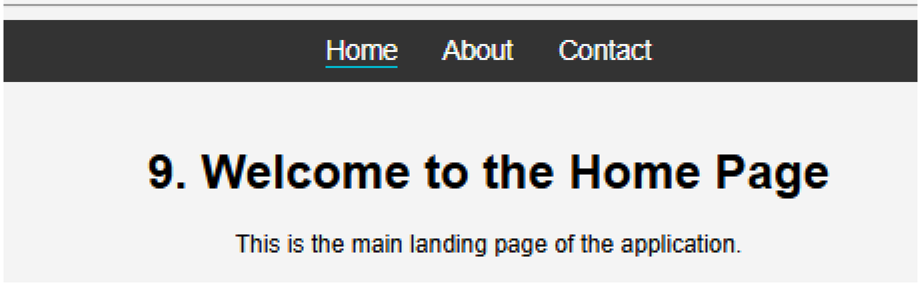
```
import React from "react";
import { NavLink } from "react-router-dom";
import "../../styles/Navbar.css";

function Navbar() {
  return (
    <nav className="navbar">
      <NavLink to="/" className="nav-link" activeclassname="active">
        Home
      </NavLink>
      <NavLink to="/about" className="nav-link" activeclassname="active">
        About
      </NavLink>
      <NavLink to="/contact" className="nav-link" activeclassname="active">
        Contact
      </NavLink>
    </nav>
  );
}

export default Navbar;
```

/styles/Navbar.css



```css
.navbar {
    display: flex;
    justify-content: center;
    background-color: #333;
    padding: 10px;
  }

  .nav-link {
    color: white;
    text-decoration: none;
    margin: 0 15px;
    font-size: 18px;
  }

  .nav-link:hover {
    color: #00bcd4;
  }

  .active {
    border-bottom: 2px solid #00bcd4;
  }
```

Verification

URL	http://localhost:3000/
Browser View	 Home About Contact 9. Welcome to the Home Page This is the main landing page of the application.

Explanation

- **Pgm9.js**
  - **Imports**
    - The BrowserRouter (Router) is wrapped around the application.

    - It enables client-side navigation without reloading the page.

- **Includes a Navigation Bar**
    - The Navbar component is included outside the Routes, so it stays visible on all pages.
- **<Router> <Navbar />**
    - This means users can navigate between pages using the Navbar.
- **Defines Three Routes**
    - The Routes component defines three pages:

/	<Home />
/about	<About />
/contact	<Contact />

    - Each Route renders a different component based on the URL.

Lab Program #10

Objective

- Design a React application featuring a class-based component that demonstrates the use of lifecycle methods to interact with an external API. The component should fetch and update data dynamically based on user interactions or state changes. Use the componentDidMount lifecycle method to fetch data from an API when the component is initially rendered. Display the fetched data in a structured format, such as a table or list. Use the componentDidUpdate lifecycle method to detect changes in the component's state or props. Trigger additional API calls to update the displayed data based on user input or actions (e.g., filtering, searching, or pagination). Implement error handling to manage issues such as failed API requests or empty data responses. Display appropriate error messages to the user when necessary. Allow users to perform actions like filtering, searching, or refreshing the data. Reflect changes in the displayed data based on these interactions.

Steps

- Create or Update the below listed files.
- Write the code in relevant files.
- Run the following command to start the application:

```
npm start
```

- Verify the functionality.
- Go through the Code explanation to understand it.

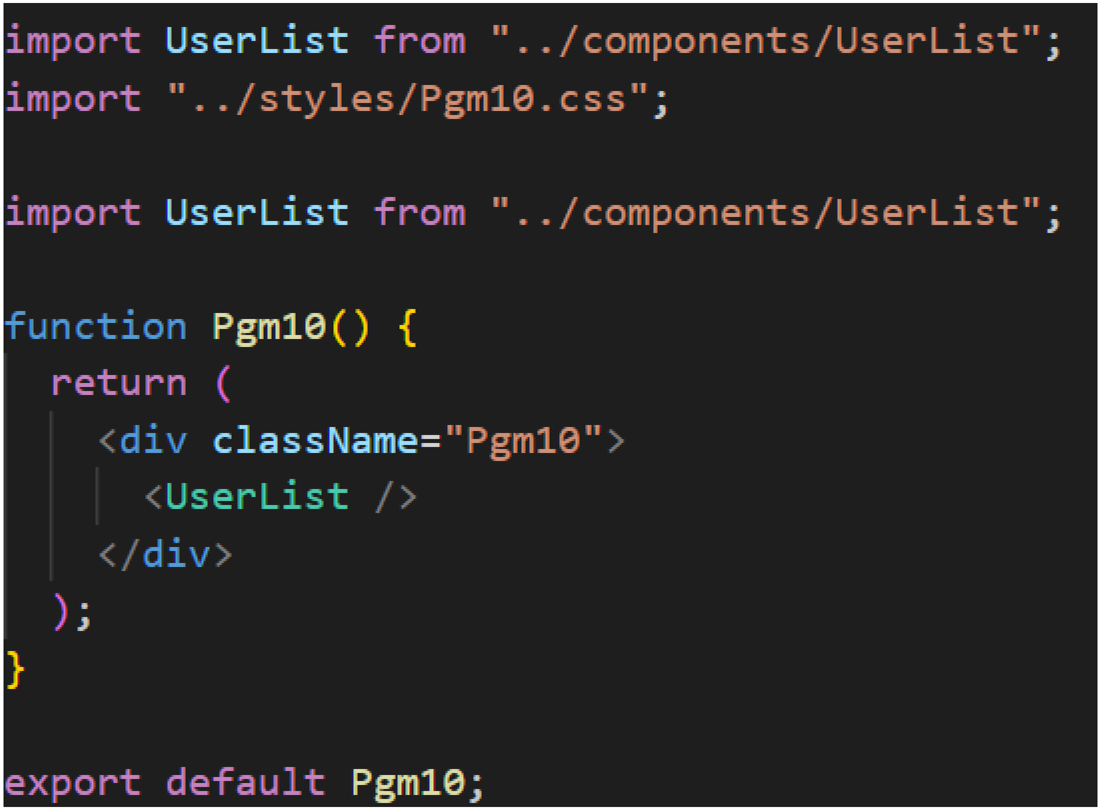
Files

File name	Changes
App.js	Include Pgm10 component.
programs/Pgm10.js	Create Program specific content.
/components/UserList.js	New child component

Code

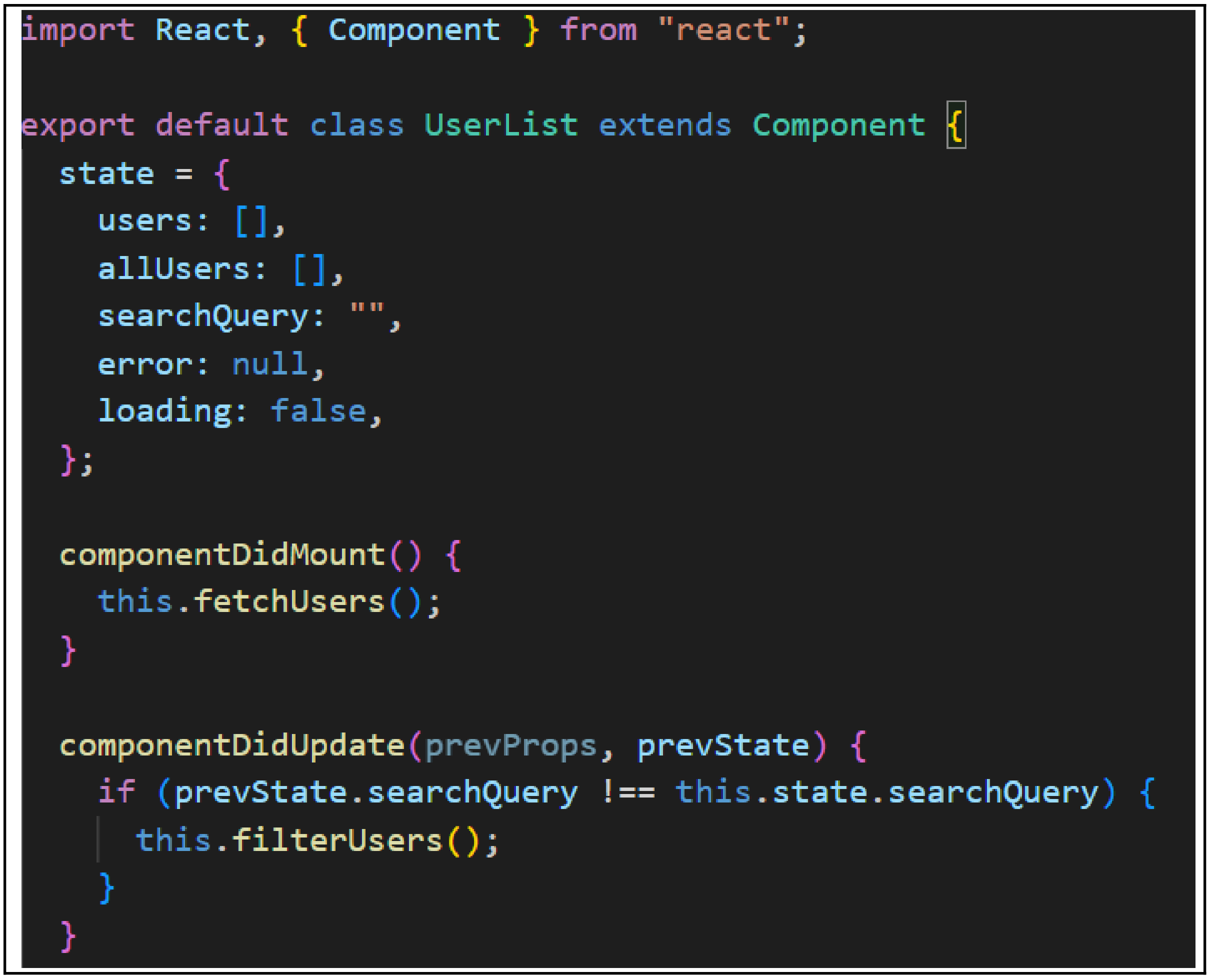
```
// App.js
import React  from "react";
import Pgm10 from "./programs/Pgm10";
function App() {

  return (
    <div style={{ textAlign: "center", marginTop: "30px" }}>
    <Pgm10/>
    </div>
  );
}

export default App;
```

/programs/Pgm10.js

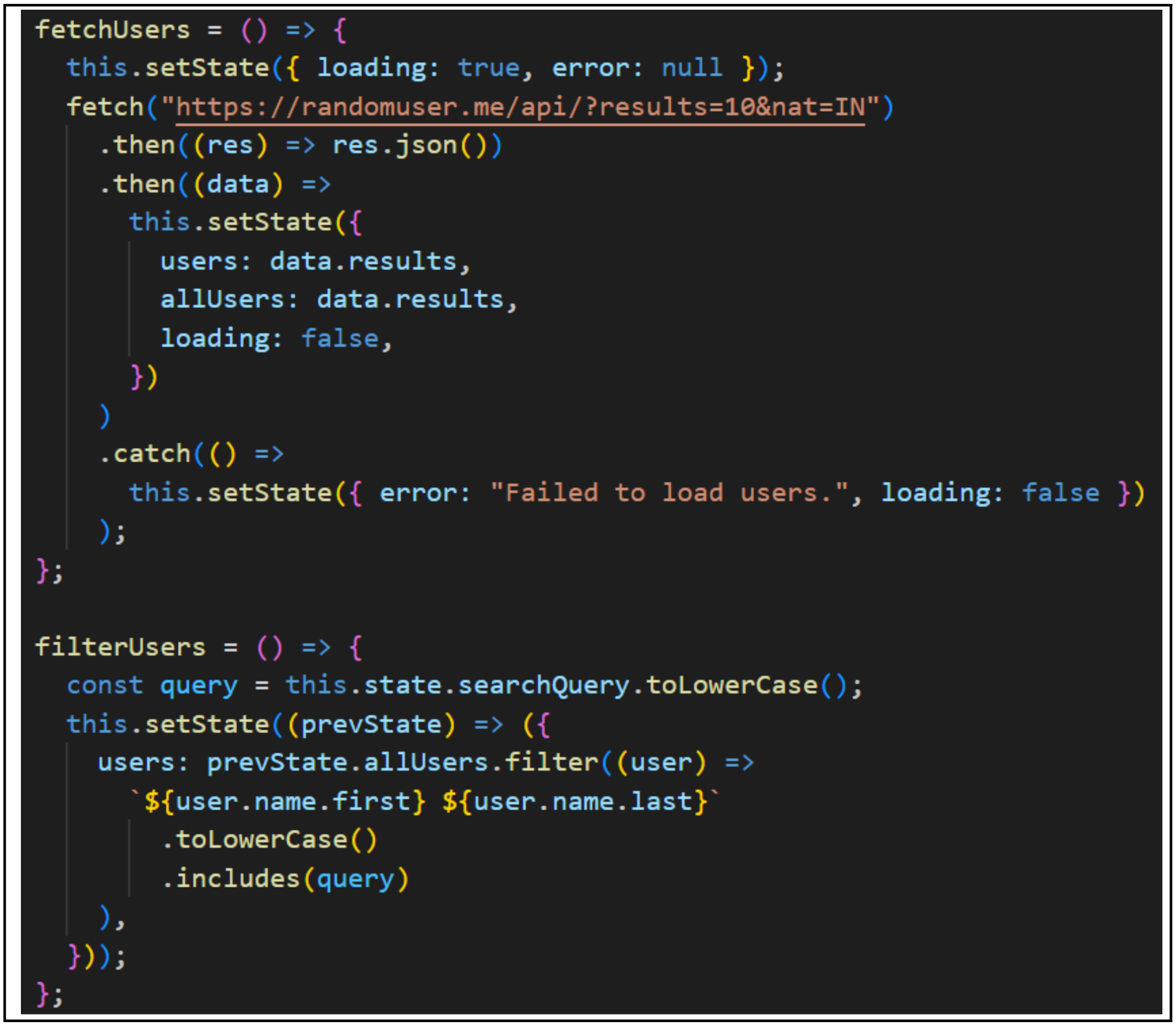


```
import UserList from "../components/UserList";
import "../styles/Pgm10.css";

import UserList from "../components/UserList";

function Pgm10() {
  return (
    <div className="Pgm10">
      <UserList />
    </div>
  );
}

export default Pgm10;
```

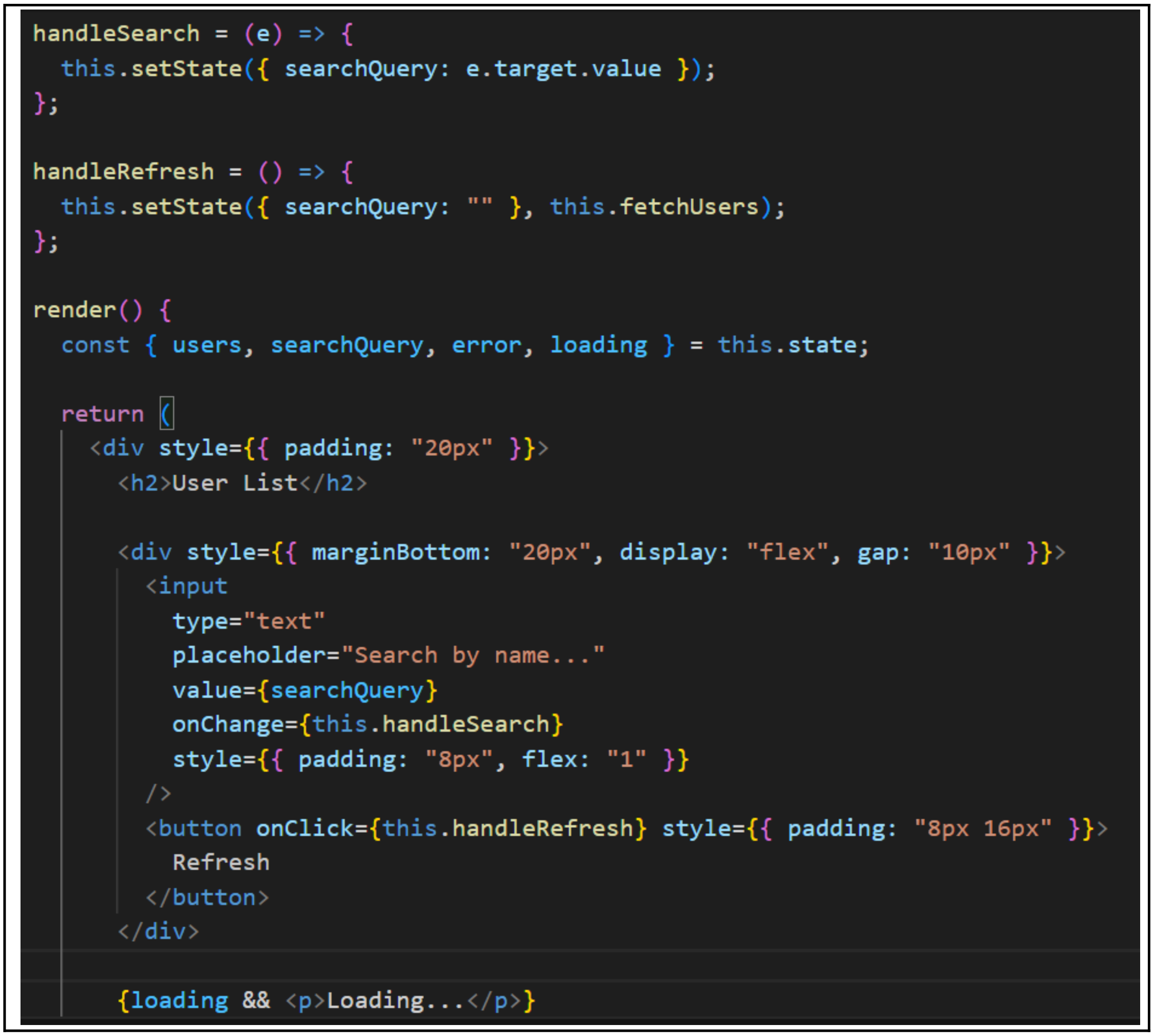
/components/UserList.js



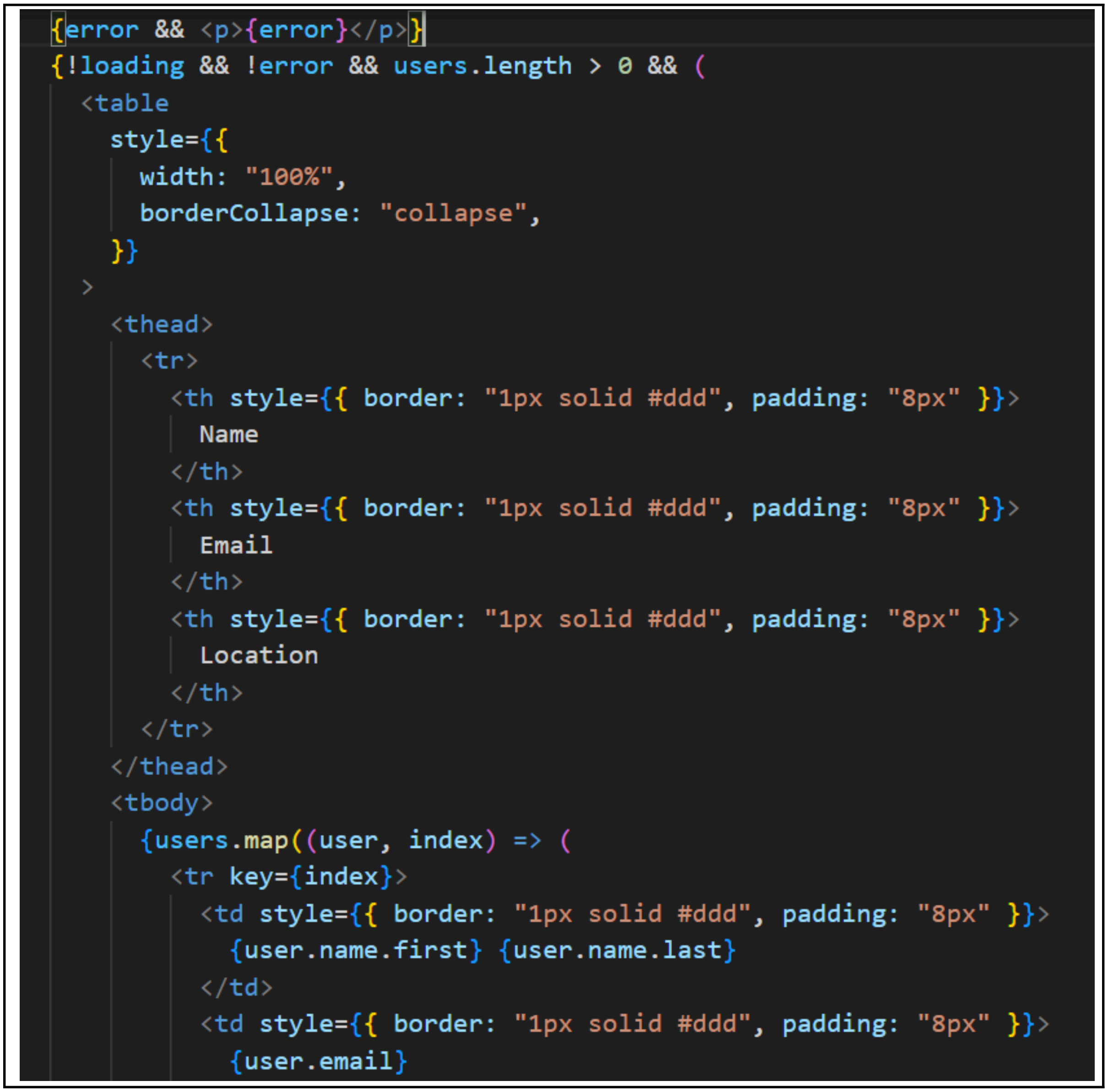
```
import React, { Component } from "react";

export default class UserList extends Component {
  state = {
    users: [],
    allUsers: [],
    searchQuery: "",
    error: null,
    loading: false,
  };

  componentDidMount() {
    this.fetchUsers();
  }

  componentDidUpdate(prevProps, prevState) {
    if (prevState.searchQuery !== this.state.searchQuery) {
      this.filterUsers();
    }
  }
```



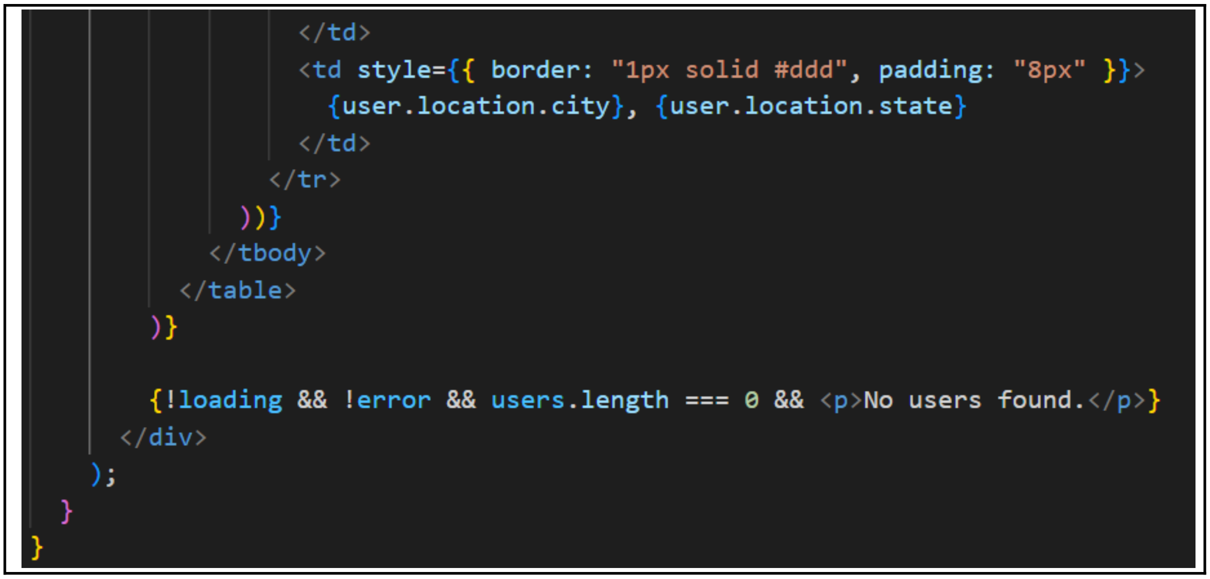
```
fetchUsers = () => {
  this.setState({ loading: true, error: null });
  fetch("https://randomuser.me/api/?results=10&nat=IN")
    .then((res) => res.json())
    .then((data) =>
      this.setState({
        users: data.results,
        allUsers: data.results,
        loading: false,
      })
    )
    .catch(() =>
      this.setState({ error: "Failed to load users.", loading: false })
    );
};

filterUsers = () => {
  const query = this.state.searchQuery.toLowerCase();
  this.setState((prevState) => ({
    users: prevState.allUsers.filter((user) =>
      `${user.name.first} ${user.name.last}`
        .toLowerCase()
        .includes(query)
    ),
  }));
};
```



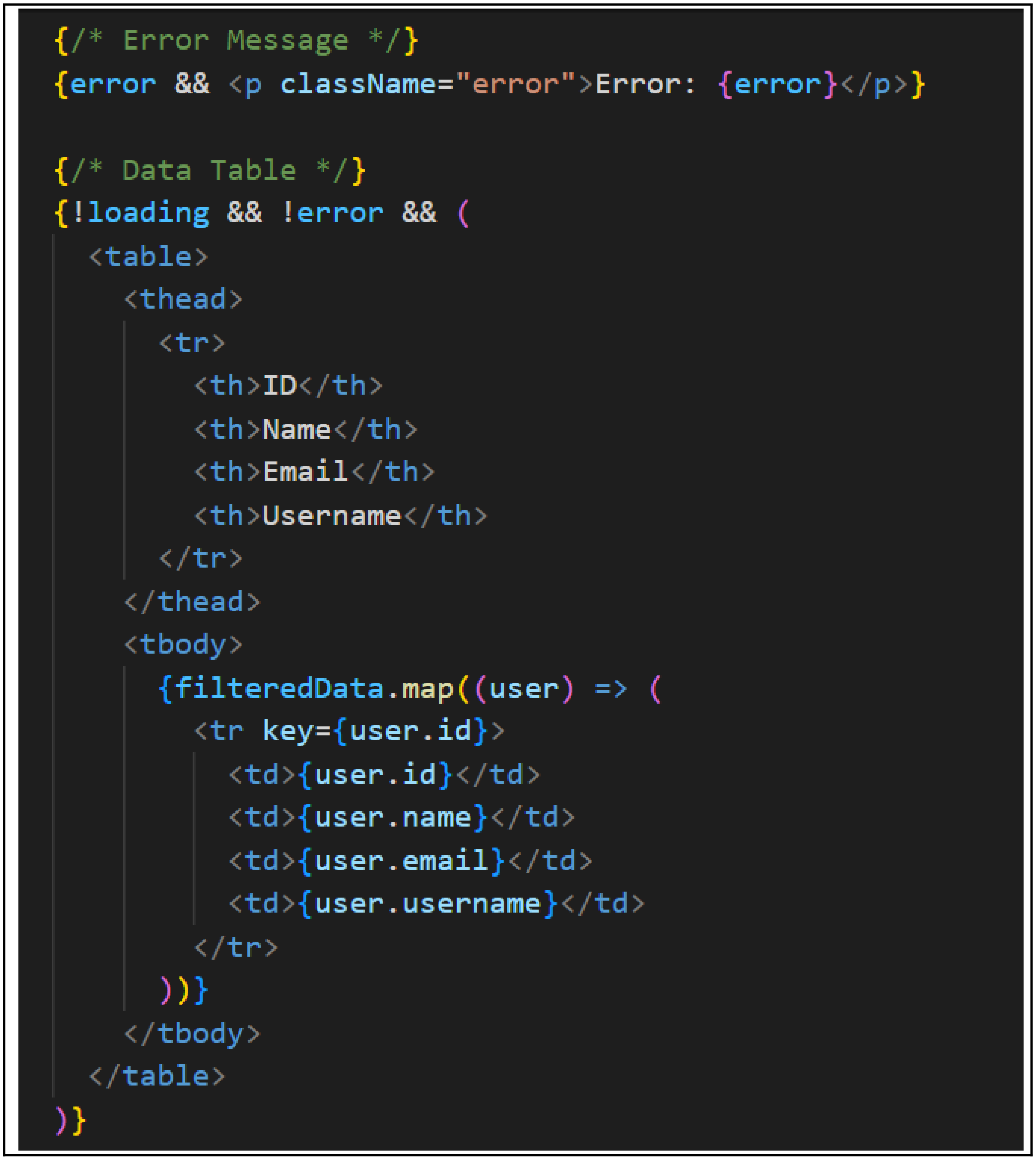
```
handleSearch = (e) => {
  this.setState({ searchQuery: e.target.value });
};

handleRefresh = () => {
  this.setState({ searchQuery: "" }, this.fetchUsers);
};

render() {
  const { users, searchQuery, error, loading } = this.state;

  return (
    <div style={{ padding: "20px" }}>
      <h2>User List</h2>

      <div style={{ marginBottom: "20px", display: "flex", gap: "10px" }}>
        <input
          type="text"
          placeholder="Search by name..."
          value={searchQuery}
          onChange={this.handleSearch}
          style={{ padding: "8px", flex: "1" }}
        />
        <button onClick={this.handleRefresh} style={{ padding: "8px 16px" }}>
          Refresh
        </button>
      </div>

      {loading && <p>Loading...</p>}
```



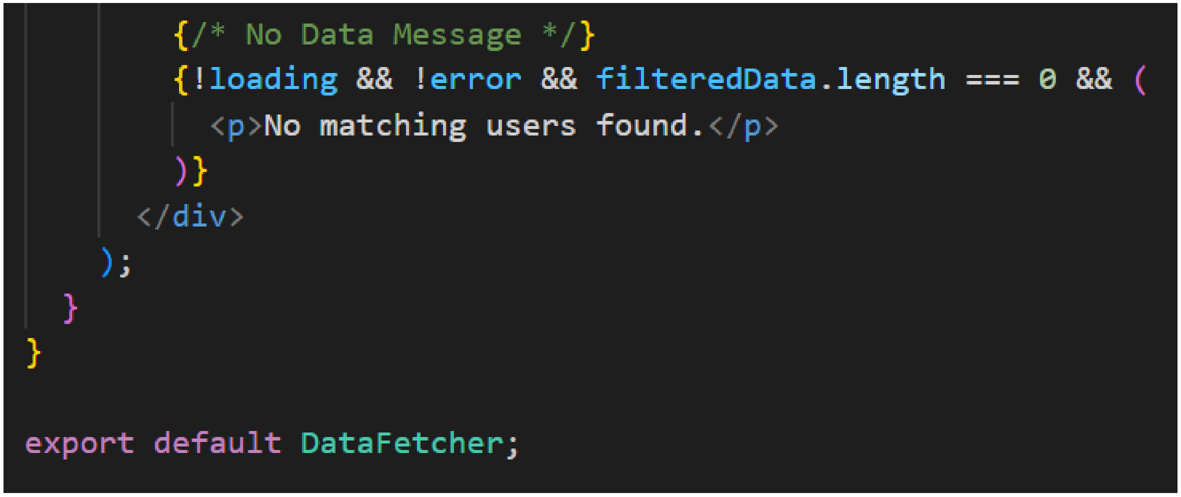
```
{error && <p>{error}</p>}
{!loading && !error && users.length > 0 && (
  <table
    style={{
      width: "100%",
      borderCollapse: "collapse",
    }}
  >
    <thead>
      <tr>
        <th style={{ border: "1px solid #ddd", padding: "8px" }}>
          Name
        </th>
        <th style={{ border: "1px solid #ddd", padding: "8px" }}>
          Email
        </th>
        <th style={{ border: "1px solid #ddd", padding: "8px" }}>
          Location
        </th>
      </tr>
    </thead>
    <tbody>
      {users.map((user, index) => (
        <tr key={index}>
          <td style={{ border: "1px solid #ddd", padding: "8px" }}>
            {user.name.first} {user.name.last}
          </td>
          <td style={{ border: "1px solid #ddd", padding: "8px" }}>
            {user.email}
```



```
         </td>
         <td style={{ border: "1px solid #ddd", padding: "8px" }}>
          {user.location.city}, {user.location.state}
         </td>
        </tr>
       ))}
      </tbody>
     </table>
    )}

    {!loading && !error && users.length === 0 && <p>No users found.</p>}
   </div>
  );
 }
}
```

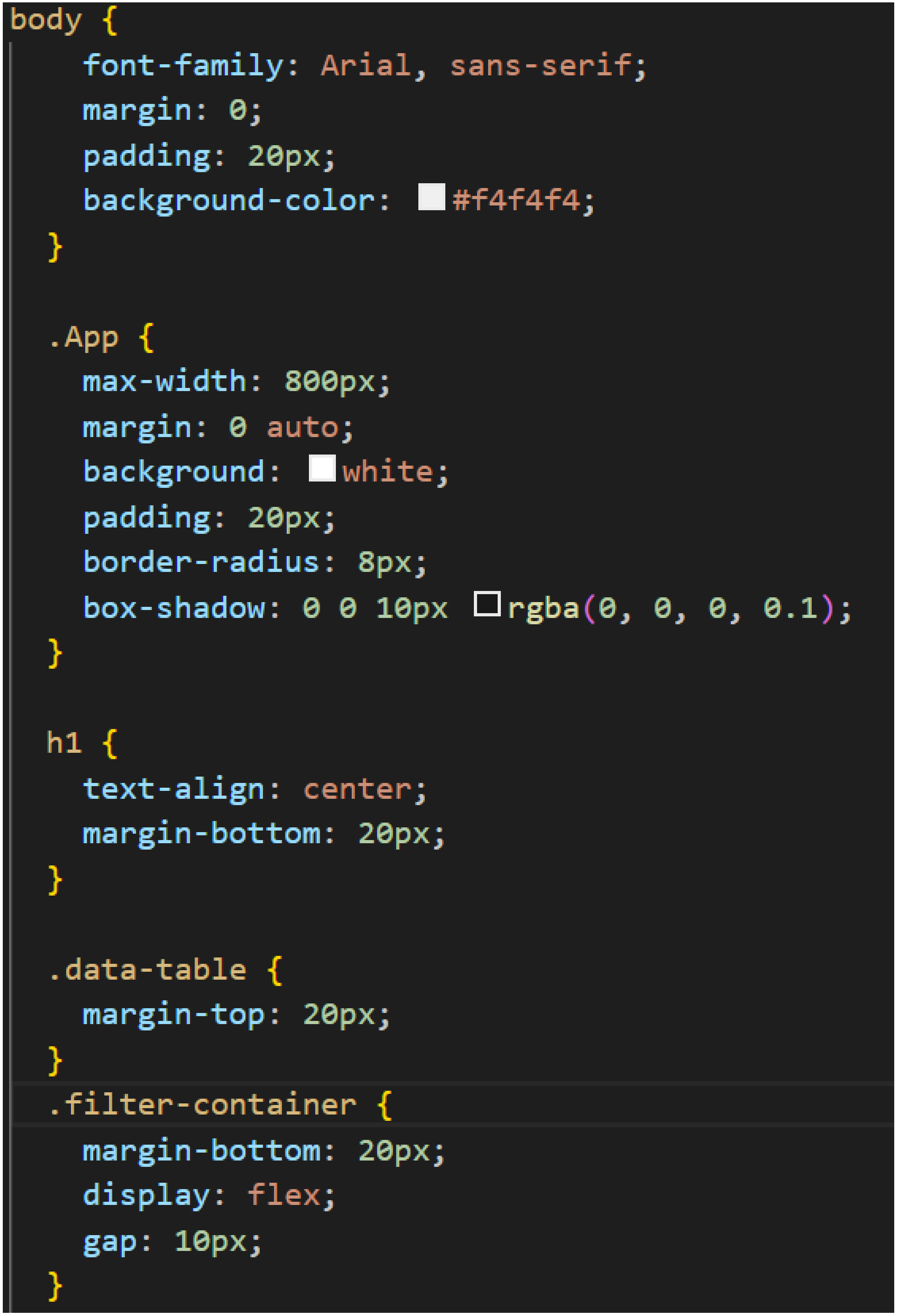


```
{/* Error Message */}
{error && <p className="error">Error: {error}</p>}

{/* Data Table */}
{!loading && !error && (
  <table>
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Email</th>
        <th>Username</th>
      </tr>
    </thead>
    <tbody>
      {filteredData.map((user) => (
        <tr key={user.id}>
          <td>{user.id}</td>
          <td>{user.name}</td>
          <td>{user.email}</td>
          <td>{user.username}</td>
        </tr>
      ))}
    </tbody>
  </table>
)}
```

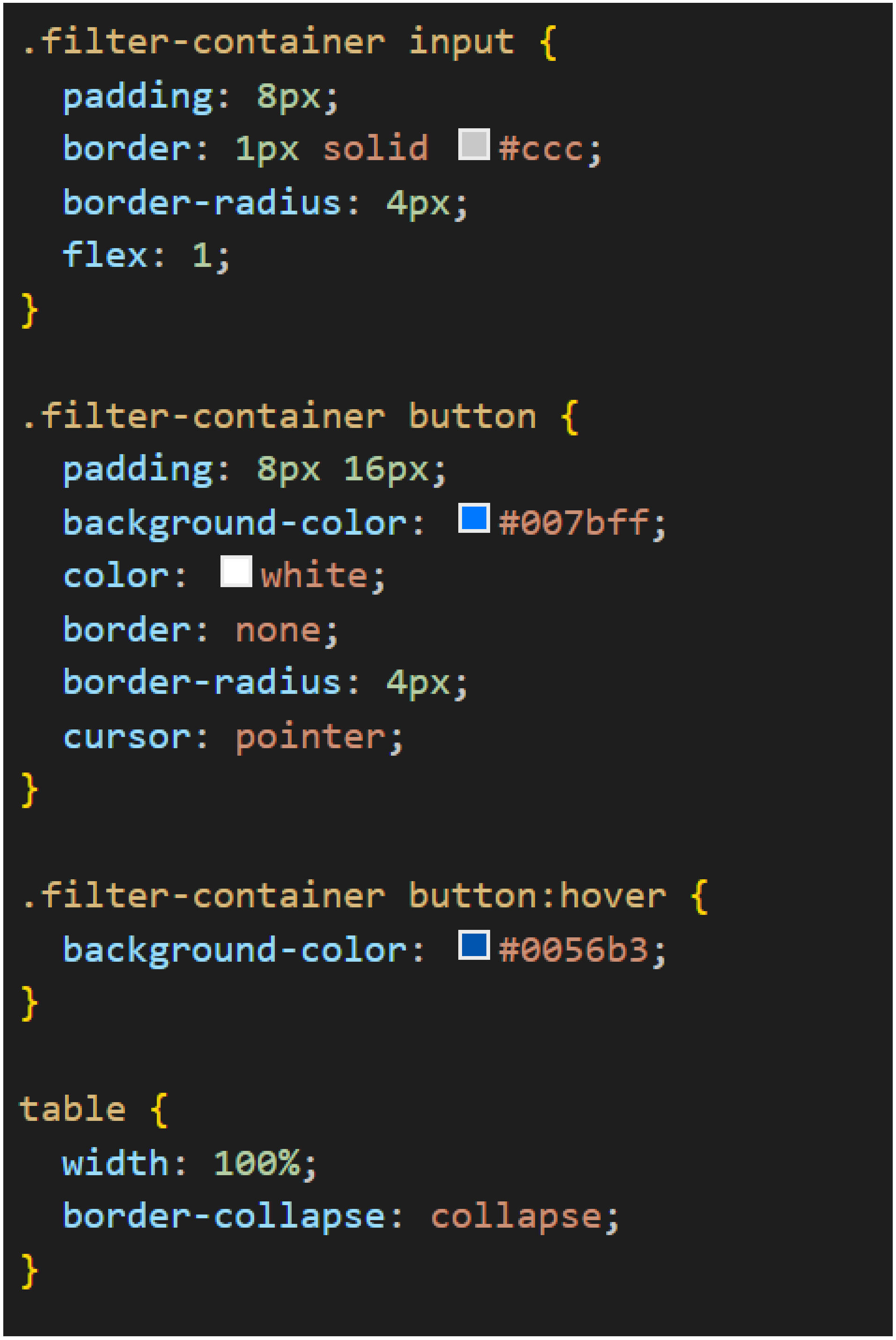


```
        {/* No Data Message */}
        {!loading && !error && filteredData.length === 0 && (
          <p>No matching users found.</p>
        )}
      </div>
    );
  }
}

export default DataFetcher;
```

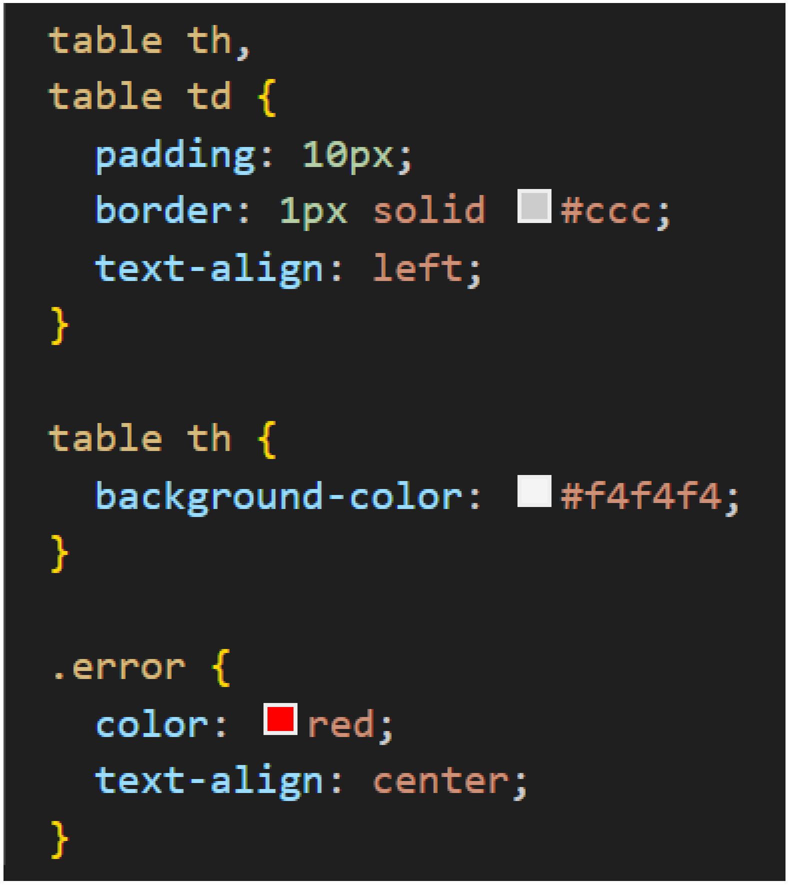
/styles/Pgm10.css



```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
  }

  .App {
    max-width: 800px;
    margin: 0 auto;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }

  h1 {
    text-align: center;
    margin-bottom: 20px;
  }

  .data-table {
    margin-top: 20px;
  }
  .filter-container {
    margin-bottom: 20px;
    display: flex;
    gap: 10px;
  }
```

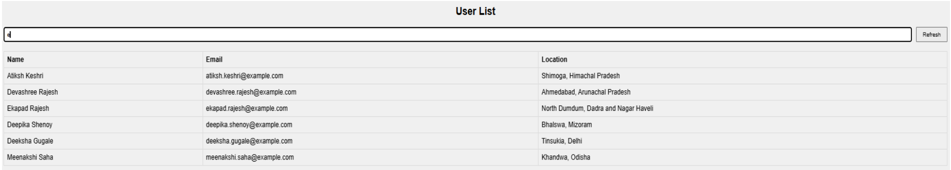


```
.filter-container input {
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
  flex: 1;
}

.filter-container button {
  padding: 8px 16px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.filter-container button:hover {
  background-color: #0056b3;
}

table {
  width: 100%;
  border-collapse: collapse;
}
```



```
table th,
table td {
  padding: 10px;
  border: 1px solid #ccc;
  text-align: left;
}

table th {
  background-color: #f4f4f4;
}

.error {
  color: red;
  text-align: center;
}
```

Verification

URL	http://localhost:3000/
Browser View	

User List

e

Name	Email	Location
Atiksh Keshri	atiksh.keshri@example.com	Shimoga, Himachal Pradesh
Devashree Rajesh	devashree.rajesh@example.com	Ahmedabad, Arunachal Pradesh
Ekapad Rajesh	ekapad.rajesh@example.com	North Dumdum, Dadra and Nagar Haveli
Deepika Shenoy	deepika.shenoy@example.com	Bhalswa, Mizoram
Deeksha Gugale	deeksha.gugale@example.com	Tinsukia, Delhi
Meenakshi Saha	meenakshi.saha@example.com	Khandwa, Odisha

Explanation

- **UserList component**
  - Fetches random Indian user data from the randomuser.me API.
  - Displays the data in a table with Name, Email, and Location.
  - Provides features like search and refresh.
  - Handles loading and error states.
- **State variables**
  - state = {

```
users: [],        // Current displayed users (filtered or unfiltered)
allUsers: [],     // Original full list of users (for resetting search)
 searchQuery: "",  // Search input value
error: null,      // Error message (if fetching fails)
```

```
loading: false,    // Loading status (while fetching)
};
```

- **Lifecycle Methods**
  - **componentDidMount()**
    - Runs once when the component is first loaded.
    - Calls fetchUsers() to get user data from the API.
  - **componentDidUpdate()**
    - Runs after every state update.
    - Checks if the searchQuery changed.
      - If yes, it calls filterUsers() to update the list based on the search.
- **API Fetching**
  - **fetchUsers()**
    - Fetches 10 random Indian users from: **https://randomuser.me/api/?results=10&nat=IN**
    - On success: Saves the data into both users and allUsers.
    - On failure: Sets an error message.
    - Updates the loading status.
- **Searching**
  - **handleSearch()**
    - Updates searchQuery whenever the user types in the search box.
  - **filterUsers()**
    - Filters allUsers by checking if the full name (first + last) matches the searchQuery.
- **Refresh Button**
  - **handleRefresh()**
    - Clears the searchQuery.
    - Re-fetches new random Indian users from the API.