

PROGRAM-01: 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 element that displays text based on the input. Dynamically update the content as the user types.

App.js

```
import React, { useState } from 'react';
import './App.css';

function App() { const [text, setText] = useState("");

const handleChange = (event) => {
  setText(event.target.value);
};

return (
  <div className="App">
    <h1>Dynamic Text Display</h1>
    <input
      type="text"
      value={text}
      onChange={handleChange}
      placeholder="Type something..."
    />
    <p>You typed: {text}</p>
  </div>
);
}

export default App;
```

OUTPUT

Dynamic Text Display

You typed: Hello, world!

PROGRAM-02: 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.

App.js

```
import React from 'react';
import Header from './Header';
import Footer from './Footer';
import './App.css';

function App() {
  const title = "Welcome to My React App";
  const tagline = "Building great apps with React";
  const copyright = "© 2025 MyApp, All Rights Reserved";

  return (
    <div className="App">
      <Header title={title} />
      <Footer tagline={tagline} copyright={copyright} />
    </div>
  );
}
```

```
export default App;
```

Header.js

```
import React from 'react';
```

```
function Header(props) {  
  return (  
    <header>  
      <h1>{props.title}</h1>  
    </header>  
  );  
}
```

```
export default Header;
```

Footer.js

```
import React from 'react';
```

```
function Footer(props) {  
  return (  
    <footer>  
      <p>{props.tagline}</p>  
      <p>{props.copyright}</p>  
    </footer>  
  );  
}
```

```
export default Footer;
```

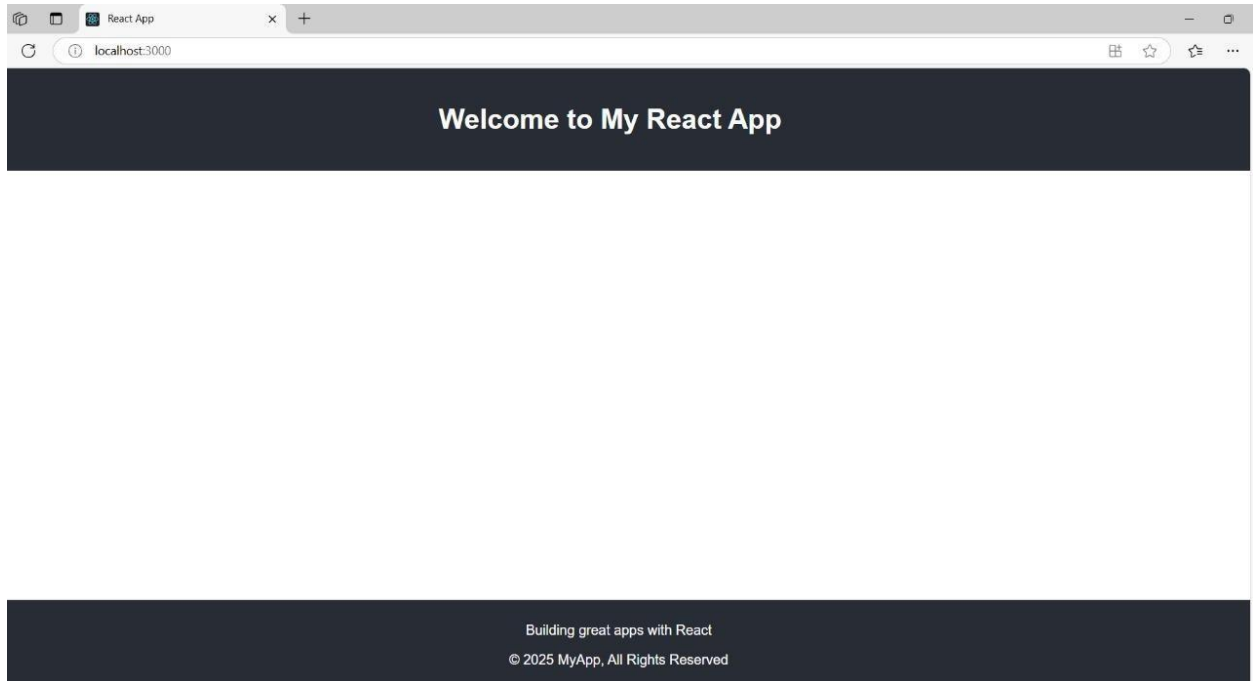
App.css

```
.App {  
  text-align: center;  
  font-family: Arial, sans-serif;  
}
```

```
header {  
  background-color: #282c34;  
  padding: 20px;  
  color: white;  
}
```

```
footer {  
  background-color: #282c34;  
  padding: 10px;  
  color: white;  
  position: absolute;  
  bottom: 0;  
  width: 100%;  
  text-align: center;  
}
```

OUTPUT



PROGRAM-03: 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.

App.js

```
import React, { useState } from 'react';
import './App.css';

function App() {
  const [counter, setCounter] = useState(0);
  const [step, setStep] = useState(1);
  const minVal = 0;

  const handleIncrement = () => {
    setCounter(prevCounter => prevCounter + step);
  };

  const handleDecrement = () => {
    if (counter - step >= minVal) {
      setCounter(prevCounter => prevCounter - step);
    }
  };

  const handleReset = () => {
    setCounter(0);
  };
}
```

```
const handleStepChange = (event) => {  
  setStep(Number(event.target.value));  
};  
  
return (  
  <div style={{ textAlign: 'center', marginTop: '50px' }}>  
    <h1>Counter Application</h1>  
    <div style={{ fontSize: '48px', margin: '20px' }}>  
      <span>{counter}</span>  
    </div>  
  
    <div>  
      <button onClick={handleIncrement}>Increase by {step}</button>  
      <button onClick={handleDecrement}>Decrease by {step}</button>  
      <button onClick={handleReset}>Reset</button>  
    </div>  
  
    <div style={{ marginTop: '20px' }}>  
      <label>  
        Set Increment/Decrement Step:  
        <input  
          type="number"  
          value={step}  
          onChange={handleStepChange}  
          min="1"  
          style={{ marginLeft: '10px' }}  
        />  
      </label>  
    </div>  
  </div>
```



```
);  
}
```

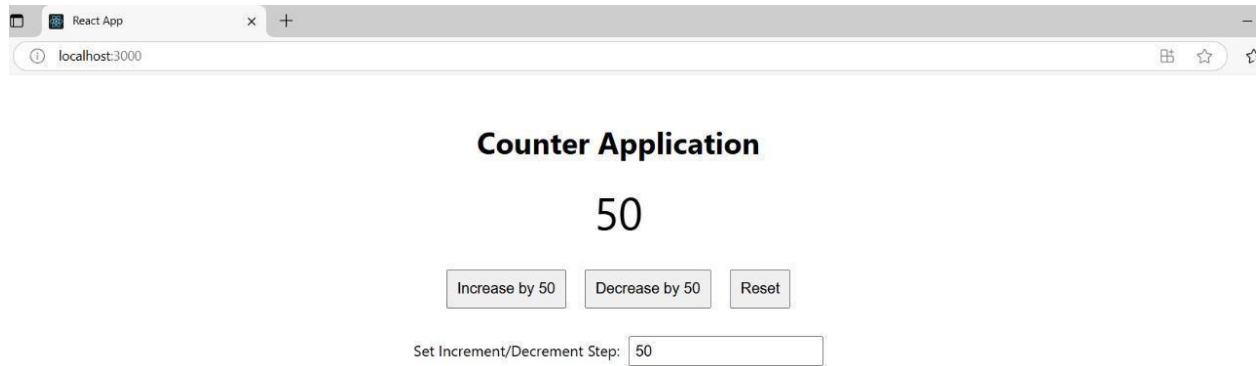
```
export default App;
```

App.css

```
.App {  
text-align: center;  
}
```

```
button {  
margin: 10px;  
padding: 10px;  
font-size: 16px;  
cursor: pointer;  
}
```

```
input {  
padding: 5px;  
font-size: 16px;  
}
```

OUTPUT

PROGRAM-04: 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.

App.js

```
import React, { useState } from 'react';
import './App.css';

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

  const addTask = () => {
    if (newTask.trim()) {
      setTasks([
        ...tasks,
        { id: Date.now(), text: newTask, completed: false },
      ]);
      setNewTask("");
    }
  };

  const deleteTask = (taskId) => {
    setTasks(tasks.filter(task => task.id !== taskId));
  };

  const toggleTaskCompletion = (taskId) => {
```

```
setTasks(tasks.map(task =>
  task.id === taskId
  ? { ...task, completed: !task.completed }
  : task
));
};

return (
  <div className="todo-container">
    <h2 className="todo-header">To-Do List</h2>

    <div className="todo-input-wrapper">
      <input
        type="text"
        value={newTask}
        onChange={(e) => setNewTask(e.target.value)}
        placeholder="Add a new task..."
        className="todo-input"
      />
      <button className="add-task-button" onClick={addTask}>Add Task</button>
    </div>

    <ul className="todo-list">
      {tasks.map((task) => (
        <li
          key={task.id}
          className={`todo-item ${task.completed ? 'completed' : ''}`}
        >
          <span
            className="task-text"

```

```
onClick={() => toggleTaskCompletion(task.id)}
>
{task.text}
</span>
<button
className="delete-button"
onClick={() => deleteTask(task.id)}
>
+
</button>
</li>
)}}
</ul>
</div>
);
};
```

```
export default ToDoFunction;
```

App.css

```
.todo-container {
font-family: 'Arial', sans-serif;
max-width: 500px;
margin: 50px auto;
padding: 20px;
border: 1px solid #e0e0e0;
border-radius: 8px;
background-color: #f9f9f9;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
```

```
text-align: center;
}
```

```
.todo-header {
color: #4A90E2;
font-size: 2rem;
margin-bottom: 20px;
}
```

```
.todo-input-wrapper {
display: flex;
justify-content: center;
margin-bottom: 20px;
}
```

```
.todo-input {
width: 70%;
padding: 10px;
border-radius: 4px;
border: 1px solid #ccc;
font-size: 1rem;
outline: none;
}
```

```
.add-task-button {
padding: 10px 15px;
margin-left: 10px;
background-color: #4CAF50;
color: white;
border: none;
```

```
border-radius: 4px;  
font-size: 1rem;  
cursor: pointer;  
transition: background-color 0.3s;  
}
```

```
.add-task-button:hover {  
background-color: #45a049;  
}
```

```
.todo-list {  
list-style-type: none;  
padding-left: 0;  
margin: 0;  
}
```

```
.todo-item {  
display: flex;  
align-items: center;  
justify-content: space-between;  
background-color: #fff;  
padding: 12px;  
margin: 10px 0;  
border-radius: 5px;  
border: 1px solid #ddd;  
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
transition: transform 0.2s ease-in-out;  
}
```

```
.todo-item:hover {
```

```
transform: scale(1.03);  
}
```

```
.todo-item.completed {  
background-color: #f1f1f1;  
text-decoration: line-through;  
color: #aaa;  
}
```

```
.task-text {  
cursor: pointer;  
font-size: 1.1rem;  
color: #333;  
transition: color 0.3s;  
}
```

```
.task-text:hover {  
color: #4CAF50;  
}
```

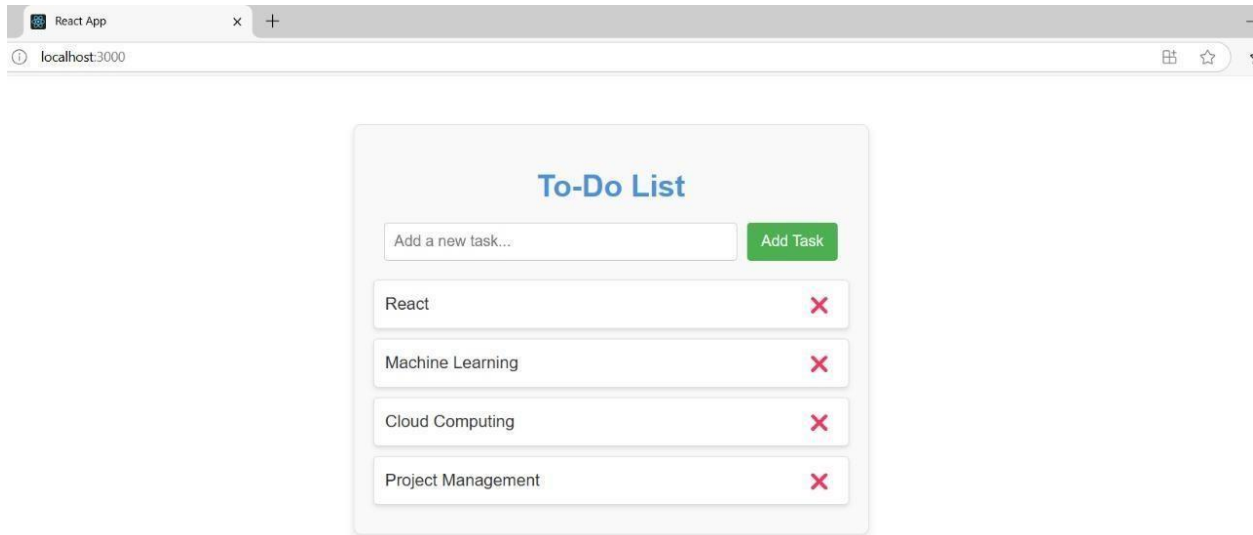
```
.delete-button {  
background: none;  
border: none;  
font-size: 1.1rem;  
color: #ff6347;  
cursor: pointer;  
transition: color 0.3s;  
}
```

```
.delete-button:hover {
```



```
color: #ff4500;  
}
```

OUTPUT



PROGRAM-05: 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.

BasicFigure.js:

```
// BasicFigure.js
import React from 'react';

const BasicFigure = ({ imageUrl, caption }) => {
  return (
    <div className="figure">
      <img src={imageUrl} alt={caption} className="figure-image" />
      <p className="figure-caption">{caption}</p>
    </div>
  );
};

export default BasicFigure;
```

FigureList.js:

```
// FigureList.js
import React, { useState } from 'react';
import BasicFigure from './BasicFigure';
```

```
const FigureList = () => {
  const [figures, setFigures] = useState([
    { imageUrl: 'https://picsum.photos/400', caption: 'Random Image 1' },
    { imageUrl: 'https://picsum.photos/400', caption: 'Random Image 2' },
    { imageUrl: 'https://picsum.photos/400', caption: 'Random Image 3' },
    { imageUrl: 'https://picsum.photos/400', caption: 'Random Image 4' },
  ]);

  const addFigure = () => {
    const newFigure = {
      imageUrl: `https://picsum.photos/400?random=${figures.length + 1}`,
      caption: `Random Image ${figures.length + 1}`,
    };
    setFigures([...figures, newFigure]);
  };

  const removeFigure = () => {
    const updatedFigures = figures.slice(0, -1);
    setFigures(updatedFigures);
  };

  return (
    <div className="figure-list-container">
      <div className="button-box">
        <button onClick={addFigure} className="action-button">Add Image</button>
        <button onClick={removeFigure} className="action-button">Remove Image</button>
      </div>
      <div className="figure-list">
```

```
{ figures.map((figure, index) => (  
  <BasicFigure key={index} imageUrl={figure.imageUrl} caption={figure.caption} />  
))}  
</div>  
</div>  
);  
};  
  
export default FigureList;
```

App.js

```
// App.js  
import React from 'react';  
import FigureList from './components/FigureList';  
import './App.css';  
  
const App = () => {  
  return (  
    <div className="app">  
      <h1>Dynamic Image Gallery</h1>  
      <FigureList />  
    </div>  
  );  
};  
  
export default App;
```

App.css

```
*{  
padding: 0;  
margin: 0;  
box-sizing: border-box;  
}
```

```
h1 {  
background: #000;  
color: #fff;  
padding: 10px;  
text-align: center;  
}
```

```
.figure-list-container {  
display: flex;  
flex-direction: column;  
align-items: center;  
margin: 20px;  
}
```

```
.button-box {  
display: block;  
text-align: center;  
padding: 10px;  
margin-bottom: 20px;  
}
```

```
.action-button {
```

```
padding: 10px 20px;
margin: 10px;
background-color: #4CAF50;
color: white;
border: none;
border-radius: 5px;
cursor: pointer;
font-size: 16px;
transition: background-color 0.3s ease;
}
```

```
.action-button:hover {
background-color: #45a049;
}
```

```
.figure-list {
display: flex;
flex-wrap: wrap;
justify-content: center;
gap: 15px;
}
```

```
.figure-list img {
max-width: 200px;
max-height: 200px;
border: 2px solid #ccc;
border-radius: 8px;
}
```

```
figure {
```

```
display: flex;
flex-direction: column;
align-items: center;
}
```

```
figcaption {
margin-top: 8px;
font-size: 14px;
color: #555;
}
```

```
.figure {
display: flex;
flex-direction: column;
align-items: center;
border: 2px solid #ddd;
border-radius: 8px;
padding: 10px;
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
transition: transform 0.2s ease, box-shadow 0.2s ease;
}
```

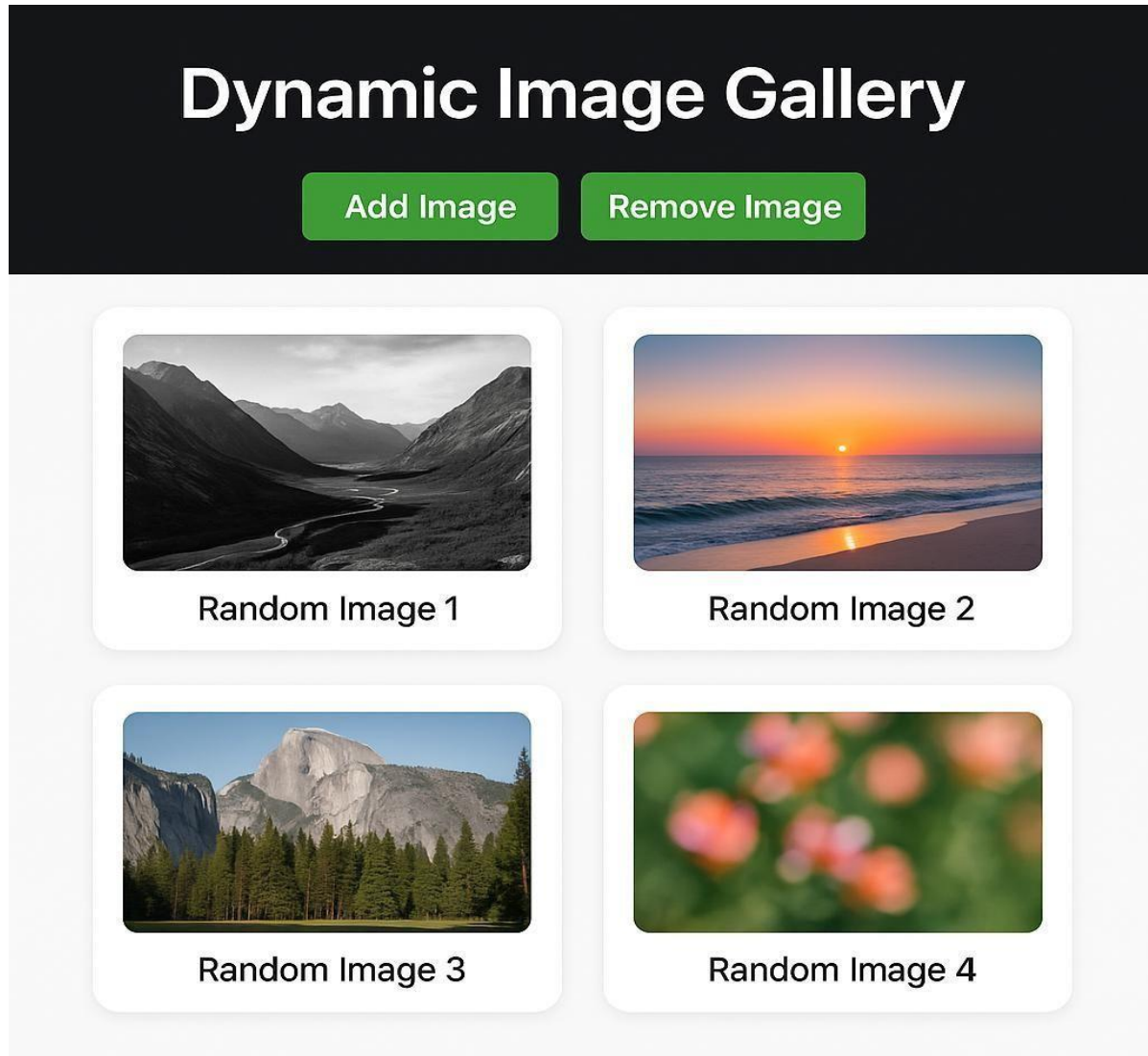
```
.figure:hover {
transform: translateY(-5px);
box-shadow: 0 6px 12px rgba(0, 0, 0, 0.2);
}
```

```
.figure-image {
max-width: 200px;
max-height: 200px;
```

```
border-radius: 8px;  
object-fit: cover;  
}
```

```
.figure-caption {  
margin-top: 10px;  
font-size: 14px;  
color: #555;  
text-align: center;  
}
```

OUTPUT



PROGRAM-06: 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.

Form.js

```
import React, { useState, useEffect, useCallback } from 'react';
import './Form.css';

const Form = () => {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
  });

  const [errors, setErrors] = useState({
    name: "",
    email: "",
    password: "",
  });

  const [showPassword, setShowPassword] = useState(false);
  const [isFormValid, setIsFormValid] = useState(false);

  const handleChange = (e) => {
```

```
const { name, value } = e.target;
setFormData((prevState) => ({
  ...prevState,
  [name]: value.trim(),
}));
};
```

```
const validateForm = useCallback(() => {
  let isValid = true;
  const newErrors = { name: "", email: "", password: "" };
```

```
  if (!formData.name) {
    newErrors.name = 'Name is required.';
    isValid = false;
  }
```

```
  const emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
  if (!formData.email || !emailPattern.test(formData.email)) {
    newErrors.email = 'Please enter a valid email address.';
    isValid = false;
  }
```

```
  if (!formData.password) {
    newErrors.password = 'Password is required.';
    isValid = false;
  } else if (formData.password.length < 6) {
    newErrors.password = 'Password must be at least 6 characters long.';
    isValid = false;
  }
```

```
setErrors(newErrors);
setIsFormValid(isValid);
}, [formData]);

useEffect(() => {
  validateForm();
}, [formData, validateForm]);

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

  if (isValid) {
    console.log('Form Data:', formData);
    setFormData({
      name: "",
      email: "",
      password: "",
    });
  }
};

return (
  <div className="form-container">
    <h2 className="form-title">Registration Form</h2>
    <form onSubmit={handleSubmit} className="form">
      <div className="form-group">
        <label htmlFor="name" className="form-label">Name</label>
        <input
          type="text"
          id="name"
```

```
name="name"
value={formData.name}
onChange={handleChange}
className={`form-input ${errors.name ? 'error' : ''}`}
placeholder="Enter your name"
/>
{errors.name && <div className="error-message">{errors.name}</div>}
</div>
```

```
<div className="form-group">
  <label htmlFor="email" className="form-label">Email</label>
  <input
    type="email"
    id="email"
    name="email"
    value={formData.email}
    onChange={handleChange}
    className={`form-input ${errors.email ? 'error' : ''}`}
    placeholder="Enter your email"
  />
  {errors.email && <div className="error-message">{errors.email}</div>}
</div>
```

```
<div className="form-group">
  <label htmlFor="password" className="form-label">Password</label>
  <input
    type={showPassword ? 'text' : 'password'}
    id="password"
    name="password"
    value={formData.password}
```

```
onChange={handleChange}
className={`form-input ${errors.password ? 'error': ''}`}
placeholder="Enter your password"
/>
{errors.password && <div className="error-message">{errors.password}</div>}
</div>

<div className="form-group password-toggle">
  <label>
    <input
      type="checkbox"
      checked={showPassword}
      onChange={() => setShowPassword(!showPassword)}
    />
    Show Password
  </label>
</div>

<div className="form-group">
  <button type="submit" className="form-submit" disabled={!isFormValid}>
    Submit
  </button>
</div>
</form>
</div>
);
};

export default Form;
```

App.js

```
import React from 'react';
import './App.css';
import Form from './components/Form';

function App() {
  return (
    <div className="App">
      <Form />
    </div>
  );
}

export default App;
```

App.css

```
.form-container {
  width: 100%;
  max-width: 500px;
  margin: 0 auto;
  padding: 20px;
  background-color: #f7f7f7;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
```

```
.form-title {  
text-align: center;  
font-size: 24px;  
margin-bottom: 20px;  
color: #333;  
}
```

```
.form {  
display: flex;  
flex-direction: column;  
}
```

```
.form-group {  
margin-bottom: 15px;  
}
```

```
.form-label {  
font-size: 14px;  
font-weight: 600;  
color: #555;  
}
```

```
.form-input {  
width: 100%;  
padding: 12px;  
margin-top: 5px;  
border: 1px solid #ddd;  
border-radius: 4px;  
font-size: 16px;  
box-sizing: border-box;
```

```
}
```

```
.form-input.error {  
border-color: red;  
}
```

```
.error-message {  
color: red;  
font-size: 12px;  
margin-top: 5px;  
}
```

```
.password-toggle {  
margin-bottom: 20px;  
}
```

```
.form-submit {  
padding: 12px;  
background-color: #4CAF50;  
color: white;  
border: none;  
border-radius: 4px;  
font-size: 16px;  
cursor: pointer;  
transition: background-color 0.3s;  
}
```

```
.form-submit.disabled {  
background-color: #ccc;  
cursor: not-allowed;
```



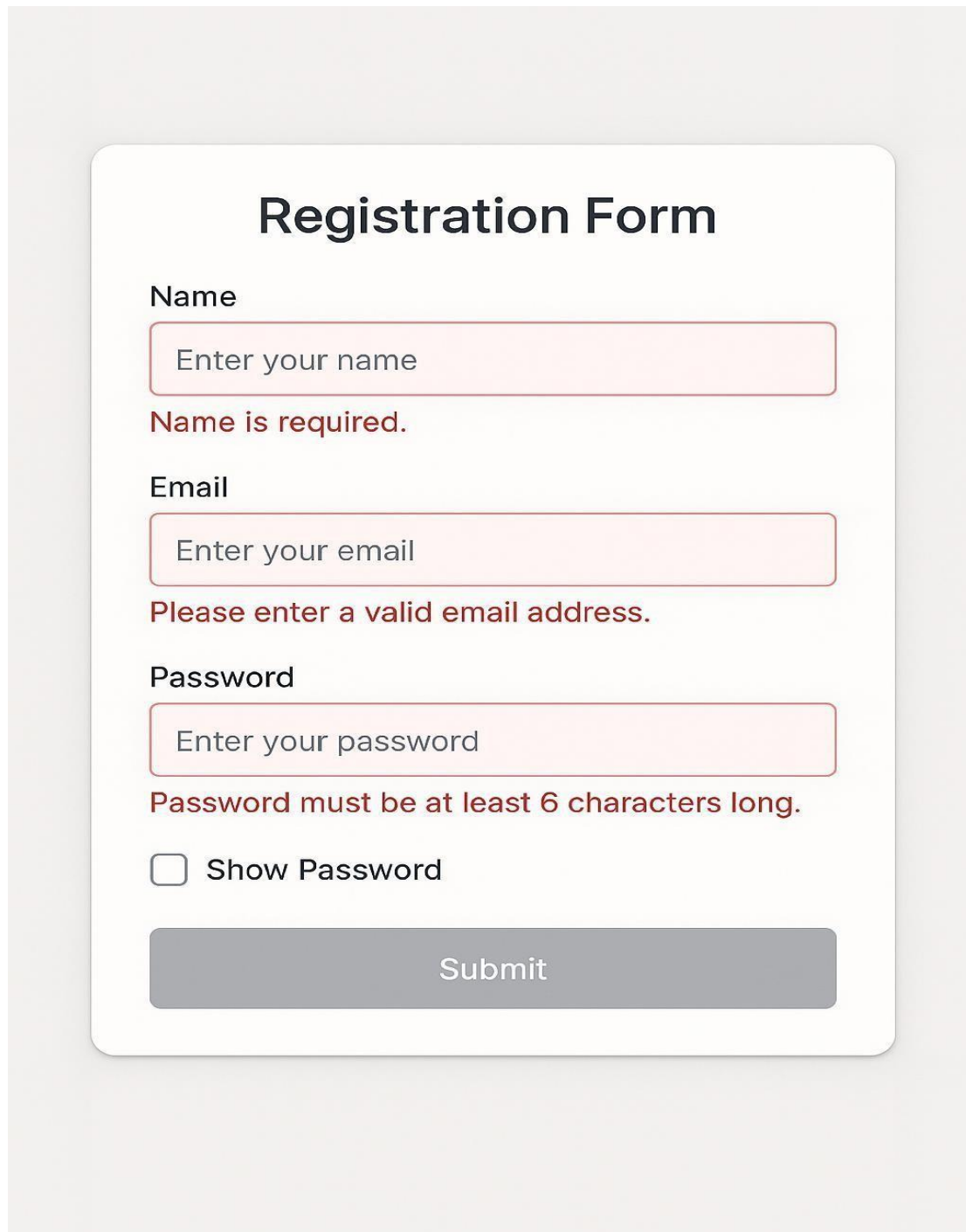
```
}
```

```
.form-submit:hover:not(:disabled) {
```

```
background-color: #45a049;
```

```
}
```

OUTPUT

A registration form UI mockup. It features a white rounded rectangle centered on a light gray background. The form has a title "Registration Form" in bold black text. Below the title are three input fields, each with a red border and a light pink background. The first field is labeled "Name" and contains the placeholder text "Enter your name". Below it is a red error message "Name is required.". The second field is labeled "Email" and contains the placeholder text "Enter your email". Below it is a red error message "Please enter a valid email address.". The third field is labeled "Password" and contains the placeholder text "Enter your password". Below it is a red error message "Password must be at least 6 characters long.". Below the password field is a checkbox labeled "Show Password". At the bottom of the form is a gray "Submit" button.

PROGRAM-07: 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.

ProfileCard.js

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

const ProfileCard = ({ name, bio, profilePicture }) => {
  const [bgColor, setBgColor] = useState('#f0f0f0');

  const handleMouseEnter = () => {
    setBgColor('#d1c4e9');
  };

  const handleMouseLeave = () => {
    setBgColor('#f0f0f0');
  };

  return (
    <div
      className="profile-card"
      style={{ backgroundColor: bgColor }}
      onMouseEnter={handleMouseEnter}
      onMouseLeave={handleMouseLeave}
    >
```

```
<img
src={profilePicture}
alt={` ${name}'s profile` }
className="profile-picture"
/>

<div className="profile-info">
  <h2 className="profile-name">{name}</h2>
  <p className="profile-bio">{bio}</p>
</div>
</div>
);
};

export default ProfileCard;
```

App.js

```
import React from 'react';
import ProfileCard from './ProfileCard';
import './App.css'

const App = () => {
  return (
    <div className="App">
      <ProfileCard
        name="Jyothy Institute of
        Technology"
        bio="Jyothy"
        profilePicture="https://vtucircle.com/wp-content/uploads/2024/11/cropped-vtucircle_icon-1.png"
      />
    </div>
  )
}
```

```
</div>
```

```
);
```

```
};
```

```
export default App;
```

App.css

```
body {  
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
  background-color: #f4f7fa;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height: 100vh;  
  margin: 0;  
}
```

```
.profile-card {  
  width: 320px;  
  padding: 30px;  
  border-radius: 15px;  
  text-align: center;  
  background-color: #ffffff;  
  box-shadow: 0 6px 12px rgba(0, 0, 0, 0.1);  
  transition: transform 0.3s ease, box-shadow 0.3s ease, background-color 0.3s ease;  
  cursor: pointer;  
  overflow: hidden;  
  margin: 20px;
```

```
}
```

```
.profile-card-container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height: 100vh;  
  width: 100%;  
}
```

```
.profile-card:hover {  
  transform: translateY(-10px);  
  box-shadow: 0 12px 24px rgba(0, 0, 0, 0.2);  
  background-color: #f3f4f6;  
}
```

```
.profile-picture {  
  width: 130px;  
  height: 130px;  
  border-radius: 50%;  
  object-fit: cover;  
  border: 4px solid #fff;  
  transition: transform 0.3s ease, box-shadow 0.3s ease;  
}
```

```
.profile-card:hover .profile-picture {  
  transform: scale(1.1);  
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
}
```

```
.profile-info {  
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
}
```

```
.profile-name {  
font-size: 1.8rem;  
font-weight: 600;  
color: #333;  
margin-bottom: 15px;  
transition: color 0.3s ease;  
}
```

```
.profile-card:hover .profile-name {  
color: #5e35b1;  
}
```

```
.profile-bio {  
font-size: 1.1rem;  
color: #555;  
line-height: 1.5;  
margin-bottom: 0;  
transition: color 0.3s ease;  
}
```

```
.profile-card:hover .profile-bio {  
color: #444;  
}
```

```
.profile-card-container {  
display: flex;
```

```
justify-content: center;  
align-items: center;  
height: 100vh;  
width: 100%;  
background-color: #f4f7fa;}
```

OUTPUT



PROGRAM-08: 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

TaskForm.js

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

function TaskForm({ addTask }) {
  const [taskName, setTaskName] = useState("");
  const [dueDate, setDueDate] = useState("");
  const [description, setDescription] = useState("");

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

    if (taskName && dueDate) {
      const newTask = {
        id: Date.now(),
        name: taskName,
        dueDate: dueDate,
        description,
        completed: false,
      };
      addTask(newTask);

      setTaskName("");
      setDueDate("");
    }
  }
}
```



```
setDescription("");
}
};

return (
  <form onSubmit={handleSubmit}>
    <div>
      <input
        type="text"
        placeholder="Task Name"
        value={taskName}
        onChange={(e) => setTaskName(e.target.value)}
      />
    </div>
    <div>
      <input
        type="date"
        value={dueDate}
        onChange={(e) => setDueDate(e.target.value)}
      />
    </div>
    <div>
      <textarea
        placeholder="Description (optional)"
        value={description}
        onChange={(e) => setDescription(e.target.value)}
      />
    </div>
    <button type="submit">Add Task</button>
  </form>
)
```

```
);  
}
```

```
export default TaskForm;
```

Filter.js

```
import React from 'react';
```

```
function Filter({ setFilter }) {  
  return (  
    <div>  
      <button onClick={() => setFilter('all')}>All Tasks</button>  
      <button onClick={() => setFilter('completed')}>Completed Tasks</button>  
      <button onClick={() => setFilter('not-completed')}>Pending Tasks</button>  
    </div>  
  );  
}
```

```
export default Filter;
```

TaskList.js

```
import React from 'react';
```

```
function TaskList({ tasks, setTasks }) {  
  const toggleTaskCompletion = (taskId) => {  
    setTasks(  
      tasks.map((task) =>  
        task.id === taskId ? { ...task, completed: !task.completed } : task
```

```
)  
);  
};  
  
const deleteTask = (taskId) => {  
  setTasks(tasks.filter((task) => task.id !== taskId));  
};  
  
return (  
  <div>  
    {tasks.length > 0 ? (  
      <ul>  
        {tasks.map((task) => (  
          <li key={task.id}>  
            <h3>{task.name}</h3>  
            <p>Due Date: {task.dueDate}</p>  
            {task.description && <p>Description: {task.description}</p>  
            <p>Status: {task.completed ? 'Completed' : 'Not Completed'}</p>  
            <button onClick={() => toggleTaskCompletion(task.id)}>  
              {task.completed ? 'Mark as Not Completed' : 'Mark as Completed'}  
            </button>  
            <button onClick={() => deleteTask(task.id)}>Delete</button>  
          </li>  
        )]}  
      </ul>  
    ) : (  
      <p>No tasks available!</p>  
    )}  
  </div>  
);
```

```
}
```

```
export default TaskList;
```

App.js

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

```
import TaskForm from './components/TaskForm';
```

```
import TaskList from './components/TaskList';
```

```
import Filter from './components/Filter';
```

```
import './App.css';
```

```
function App() {
```

```
  const [tasks, setTasks] = useState([]);
```

```
  const [filter, setFilter] = useState('all');
```

```
  const addTask = (task) => {
```

```
    setTasks([...tasks, task]);
```

```
  };
```

```
  const handleFilterChange = (status) => {
```

```
    setFilter(status);
```

```
  };
```

```
  const filteredTasks = tasks.filter((task) => {
```

```
    if (filter === 'completed') return task.completed;
```

```
    if (filter === 'not-completed') return !task.completed;
```

```
    return true;
```

```
  });
```

```
return (  
  <div className="App">  
    <h1>Task Reminder</h1>  
    <TaskForm addTask={addTask} />  
    <Filter setFilter={handleFilterChange} />  
    <TaskList tasks={filteredTasks} setTasks={setTasks} />  
  </div>  
);  
}  
  
export default App;
```

App.css

```
body {  
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
  margin: 0;  
  padding: 0;  
  background-color: #f0f4f8;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  min-height: 100vh;  
}  
  
.App {  
  width: 550px;  
  padding: 30px;  
  background-color: #ffffff;  
  border-radius: 12px;
```

```
box-shadow: 0 4px 16px rgba(0, 0, 0, 0.1);  
transition: transform 0.3s ease, box-shadow 0.3s ease;  
}
```

```
.App:hover {  
  transform: translateY(-5px);  
  box-shadow: 0 8px 24px rgba(0, 0, 0, 0.2);  
}
```

```
h1 {  
  font-size: 2.2rem;  
  color: #333;  
  text-align: center;  
  margin-bottom: 10px;  
  margin-top: 0;  
}
```

```
form {  
  display: flex;  
  flex-direction: column;  
  gap: 20px;  
}
```

```
input,  
textarea {  
  padding: 12px;  
  font-size: 1rem;  
  border: 1px solid #ccc;  
  border-radius: 8px;  
  transition: border-color 0.3s ease;
```

```
}
```

```
input:focus,  
textarea:focus {  
border-color: #4CAF50;  
outline: none;  
}
```

```
button {  
background-color: #4CAF50;  
color: white;  
border: none;  
padding: 12px;  
font-size: 1rem;  
border-radius: 8px;  
cursor: pointer;  
transition: background-color 0.3s ease, transform 0.3s ease;  
}
```

```
button:hover {  
background-color: #45a049;  
}
```

```
button:active {  
transform: scale(0.98);  
}
```

```
textarea {  
resize: vertical;  
min-height: 120px;
```

```
}
```

```
input[type="date"] {  
padding: 12px;  
}
```

```
div {  
display: flex;  
flex-direction: column;  
gap: 10px;  
}
```

```
ul {  
list-style-type: none;  
padding: 0;  
}
```

```
li {  
background-color: #fafafa;  
margin: 15px 0;  
padding: 20px;  
border-radius: 12px;  
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);  
transition: transform 0.3s ease, box-shadow 0.3s ease;  
}
```

```
li:hover {  
transform: translateY(-5px);  
box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);  
}
```



```
h3 {  
  margin: 0;  
  font-size: 1.5rem;  
  color: #333;  
  font-weight: 600;  
}
```

```
p {  
  margin: 5px 0;  
  color: #666;  
}
```

```
button {  
  background-color: #007BFF;  
  color: white;  
  border: none;  
  padding: 8px 15px;  
  font-size: 1rem;  
  border-radius: 8px;  
  cursor: pointer;  
  transition: background-color 0.3s ease, transform 0.3s ease;  
  margin-right: 10px;  
}
```

```
button:hover {  
  background-color: #0056b3;  
}
```

```
button:active {
```

```
background-color: #003f8d;  
}
```

```
button:last-child {  
background-color: #e74c3c;  
}
```

```
button:last-child:hover {  
background-color: #c0392b;  
}
```

```
button:last-child:active {  
background-color: #7f1c1c;  
}
```

```
.completed {  
text-decoration: line-through;  
color: #bbb;  
}
```

```
div {  
display: flex;  
gap: 20px;  
justify-content: center;  
}
```

```
button {  
background-color: #f1f1f1;  
color: #333;
```

```
padding: 12px 18px;  
font-size: 1rem;  
border: 1px solid #ccc;  
border-radius: 8px;  
cursor: pointer;  
transition: background-color 0.3s ease, transform 0.3s ease;  
}
```

```
button:hover {  
  background-color: #ddd;  
}
```

```
button:active {  
  transform: scale(0.98);  
}
```

```
button:focus {  
  outline: none;  
  border-color: #007BFF;  
}
```

OUTPUT

Task Reminder

Add Task

All Tasks

Completed Tasks

Pending Tasks

Submit Assignment

May 1, 2024

Mark as Completed

Prepare Presentation

May 5, 2024

Work on slides

Delete

Read Chapter 3

April 28, 2024

Delete

PROGRAM-09: 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.

Home.js

```
import React from 'react';

const Home = () => {
  return (
    <div>
      <h2>Home Page</h2>
      <p>Welcome to the Home Page!</p>
    </div>
  );
};

export default Home;
```

About.js

```
import React from 'react';

const About = () => {

  return (

    <div>

      <h2>About Page</h2>

      <p>Learn more about us on the About Page!</p>

    </div>

  );

};

export default About;
```

Contact.js

```
import React from 'react';

const Contact = () => {

  return (

    <div>

      <h2>Contact Page</h2>

      <p>Get in touch with us through the Contact Page!</p>

    </div>

  );

};
```

```
};
```

```
export default Contact;
```

Navbar.js

```
import React from 'react';
```

```
import { NavLink } from 'react-router-dom';
```

```
const Navbar = () => {
```

```
  return (
```

```
    <nav>
```

```
      <ul>
```

```
        <li>
```

```
          <NavLink
```

```
            to="/"
```

```
            className={({ isActive }) => (isActive ? 'active' : '')}
```

```
          >
```

```
            Home
```

```
          </NavLink>
```

```
        </li>
```

```
      </ul>
```

```
    <NavLink
```

```
      to="/about"
```

```
      className={({ isActive }) => (isActive ? 'active' : '')}
```

```
    >
```

```
      About
```

```
</NavLink>

</li>

<li>

<NavLink
  to="/contact"
  className={({ isActive }) => (isActive ? 'active' : '')}
>
  Contact
</NavLink>
</li>
</ul>
</nav>

);

};
```

```
export default Navbar;
```

App.js

```
import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import Navbar from './components/Navbar';

import Home from './components/Home';

import About from './components/About';

import Contact from './components/Contact';

import './App.css'
```



```
const App = () => {  
  return (  
    <Router>  
      <div>  
        <Navbar />  
        <div style={{ padding: '20px' }}>  
          <Routes>  
            <Route path="/" element={ <Home /> } />  
            <Route path="/about" element={ <About /> } />  
            <Route path="/contact" element={ <Contact /> } />  
          </Routes>  
        </div>  
      </div>  
    </Router>  
  );  
};  
  
export default App;
```

App.css

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

```
}
```

```
div {  
  
margin: 0 auto;  
  
max-width: 960px;  
  
padding: 20px;  
  
}
```

```
h2 {  
  
color: #333;  
  
padding-bottom: 20px;  
  
}
```

```
nav {  
  
background-color: #333;  
  
padding: 10px;  
  
border-radius: 5px;  
  
margin-bottom: 20px;  
  
}
```

```
ul {  
  
list-style: none;  
  
display: flex;  
  
gap: 15px;  
  
justify-content: center;  
  
margin: 0;
```

```
padding: 0;  
}
```

```
li {  
display: inline;  
}
```

```
a {  
text-decoration: none;  
color: white;  
padding: 8px 16px;  
border-radius: 4px;  
}
```

```
a:hover {  
background-color: #444;  
}
```

```
a.active {  
background-color: #1e90ff;  
color: white;  
font-weight: bold;  
}
```

```
p {  
color: #555;
```

```
font-size: 1.1rem;
```

```
line-height: 1.6;
```

```
}
```

```
Index.js
```

```
import React from 'react';
```

```
import ReactDOM from 'react-dom/client';
```

```
import App from './App';
```

```
const rootElement = document.getElementById('root');
```

```
const root = ReactDOM.createRoot(rootElement);
```

```
root.render(<App />);
```

OUTPUT

Routing a roouting ude monstrouting using reart–
router–dom library

[Home](#) [About](#) [Contact](#)

Home Page

Welcome to the Home Page!

PROGRAM-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.

App.js

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

const API_URL = 'https://jsonplaceholder.typicode.com/users';

class DataFetcher extends Component {
  constructor(props) {
    super(props);
    this.state = {
      data: [],
      filteredData: [],
      searchQuery: '',
      error: null,
      loading: false,
```

```
};
```

```
}
```

```
componentDidMount() {
```

```
  this.fetchData();
```

```
}
```

```
fetchData = async () => {
```

```
  this.setState({ loading: true, error: null });
```

```
  try {
```

```
    const response = await fetch(API_URL);
```

```
    if (!response.ok) {
```

```
      throw new Error('Failed to fetch data');
```

```
    }
```

```
    const data = await response.json();
```

```
    this.setState({ data, filteredData: data, loading: false });
```

```
  } catch (error) {
```

```
    this.setState({ error: error.message, loading: false });
```

```
  }
```

```
};
```

```
componentDidUpdate(prevProps, prevState) {
```

```
  if (prevState.searchQuery !== this.state.searchQuery) {
```

```
    this.filterData();
```

```
  }
```

```
}
```

```
handleSearchChange = (event) => {  
  this.setState({ searchQuery: event.target.value });  
};  
  
filterData = () => {  
  const { data, searchQuery } = this.state;  
  if (searchQuery.trim() === "") {  
    this.setState({ filteredData: data });  
  } else {  
    const filteredData = data.filter((item) =>  
      item.name.toLowerCase().includes(searchQuery.toLowerCase())  
    );  
    this.setState({ filteredData });  
  }  
};  
  
renderError = () => {  
  const { error } = this.state;  
  return error ? <div className="error">{`Error: ${error}`}</div> : null;  
};  
  
render() {  
  const { filteredData, searchQuery, loading } = this.state;  
  
  return (  

```



```
<div className="data-fetcher">
```

```
<h1>User Data</h1>
```

```
{this.renderError()}
```

```
<div className="search-bar">
```

```
<input
```

```
type="text"
```

```
value={searchQuery}
```

```
onChange={this.handleSearchChange}
```

```
placeholder="Search by name"
```

```
/>
```

```
</div>
```

```
{loading ? (
```

```
<div>Loading...</div>
```

```
): (
```

```
<table>
```

```
<thead>
```

```
<tr>
```

```
<th>Name</th>
```

```
<th>Email</th>
```

```
<th>City</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
{ filteredData.length > 0 ? (  
  filteredData.map((item) => (  
    <tr key={item.id}>  
      <td>{item.name}</td>  
      <td>{item.email}</td>  
      <td>{item.address.city}</td>  
    </tr>  
  ))  
  ) : (  
    <tr>  
      <td colspan="3">No results found.</td>  
    </tr>  
  )}  
</tbody>  
</table>  
)  
  
<button onClick={this.fetchData}>Refresh Data</button>  
</div>  
);  
}  
}  
  
export default DataFetcher;
```

Index.js

```
import React from 'react';  
  
import ReactDOM from 'react-dom/client';  
  
import './App.css';  
  
import DataFetcher from './App';  
  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
  
  
root.render(  
  
  <React.StrictMode>  
  
    <DataFetcher />  
  
  </React.StrictMode>  
  
);
```

App.css

```
* {  
  
padding: 0;  
  
margin: 0;  
  
box-sizing: border-box;  
  
}  
  
  
body {  
  
font-family: Arial, sans-serif;  
  
margin: 0;
```

```
padding: 0;
background-color: #f4f4f4;
}
```

```
button {
border-radius: 5px;
border: none;
cursor: pointer;
color: #fff;
font-weight: bold;
background: red;
margin-top: 20px;
padding: 10px;
}
```

```
.data-fetcher {
width: 80%;
margin: 0 auto;
padding: 20px;
background-color: #fff;
border-radius: 8px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
```

```
h1 {  
  text-align: center;  
  color: #333;  
}
```

```
.search-bar {  
  margin: 20px 0;  
  text-align: center;  
}
```

```
.search-bar input {  
  padding: 8px;  
  width: 60%;  
  font-size: 16px;  
  border: 1px solid #000;  
  border-radius: 4px;  
}
```

```
table {  
  width: 100%;  
  margin-top: 20px;  
  border-collapse: collapse;  
}
```

```
table th,  
table td {
```

```
padding: 10px;  
text-align: left;  
border-bottom: 1px solid #ddd;  
}
```

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

OUTPUT

User Data

Name	Email	City
Leanne Graham	Sincere@april.blz	Gwenborough
Ervin Howell	Shanna@melissa.tv	Wisokyburgh
Clementine Bauch	Nathan@yesenia.net	McKenziehaven
Patricia Lebsack	Julianne.OConner@kory.org	South Elvis
Chelsey Dietrich	Lucio_Hettinger@annie.ca	Roscoeview
Mrs. Dennis Schulist	Karley_Dach@jasper.info	South Christy
Kurtis Weissnat	Telly.Hoeger@billy.tv	Howemouth
Nicholas Runolfsdóttir V	Sherwood@rosamond.mee	Aliyaview
Glenna Reichert	Chaim_McDermott@dana.io	Bartholomebury

[Refresh Data](#)