**Name - Gaurang A Raorane Roll No - 49**

**Class - D15A Batch - C**

**EXPERIMENT NO - 7**

**Aim:-** Experiment to study the basics of React.

**Theory:-**

* **Introduction to React:**

React is a JavaScript library developed by Facebook for building user interfaces.

React allows you to create interactive and dynamic web applications by using reusable components.

It employs a Virtual DOM (Document Object Model) to efficiently update the actual DOM, improving performance.

* **Setting up the Environment:**

Before starting, ensure you have Node.js and npm (Node Package Manager) installed on your computer.

Use create-react-app to set up a new React project. This tool simplifies the initial project configuration.

* **Components:**

React applications are built using components, which are reusable, self-contained pieces of a user interface.

Components can be functional or class-based. Both have their use cases.

* **JSX (JavaScript XML):**

JSX is a syntax extension for JavaScript used in React.

It allows you to write HTML-like code within your JavaScript files.

* **Rendering Components:**

Use the ReactDOM.render() method to render a React component into the HTML DOM.

The entry point for your application is usually an element with an id in your HTML file.

* **Props:**

Props (short for properties) are used to pass data from parent components to child components.

They are read-only and help create dynamic and reusable components.

* **State:**

State allows a component to manage and handle its data that can change over time.

State is essential for creating interactive and responsive user interfaces.

* **Event Handling:**

React handles events using synthetic events, similar to standard JavaScript events but with some differences.

* **Conditional Rendering:**

You can conditionally render components or elements based on specific conditions or user interactions.

* **Lists and Keys:**

Rendering lists of elements efficiently using the map function and providing unique keys to help React identify elements.

* **Hooks (Optional):**

In addition to class-based components, React offers hooks like useState and useEffect to manage state and side effects in functional components.

Input:-

App.js

import React, { Component } from "react";

class App extends Component {

constructor(props) {

super(props);

this.state = {

contacts: [

{ id: 1, name: "John Doe", email: "john@example.com" },

{ id: 2, name: "Jane Smith", email: "jane@example.com" },

],

newContact: {

name: "",

email: "",

},

};

}

handleInputChange = (event) => {

const { name, value } = event.target;

this.setState((prevState) => ({

newContact: {

...prevState.newContact,

[name]: value,

},

}));

};

handleAddContact = () => {

const { name, email } = this.state.newContact;

if (name && email) {

this.setState((prevState) => ({

contacts: [

...prevState.contacts,

{ id: Date.now(), ...prevState.newContact },

],

newContact: {

name: "",

email: "",

},

}));

}

};

handleDeleteContact = (contactId) => {

this.setState((prevState) => ({

contacts: prevState.contacts.filter(

(contact) => contact.id !== contactId

),

}));

};

render() {

return (

<div className="App">

<h1>Contact Management</h1>

<div>

<h2>Add Contact</h2>

<input

type="text"

name="name"

placeholder="Name"

value={this.state.newContact.name}

onChange={this.handleInputChange}

/>

<input

type="text"

name="email"

placeholder="Email"

value={this.state.newContact.email}

onChange={this.handleInputChange}

/>

<button onClick={this.handleAddContact}>Add</button>

</div>

<div>

<h2>Contact List</h2>

<ul>

{this.state.contacts.map((contact) => (

<li key={contact.id}>

{contact.name} - {contact.email}

<button onClick={() => this.handleDeleteContact(contact.id)}>

Delete

</button>

</li>

))}

</ul>

</div>

</div>

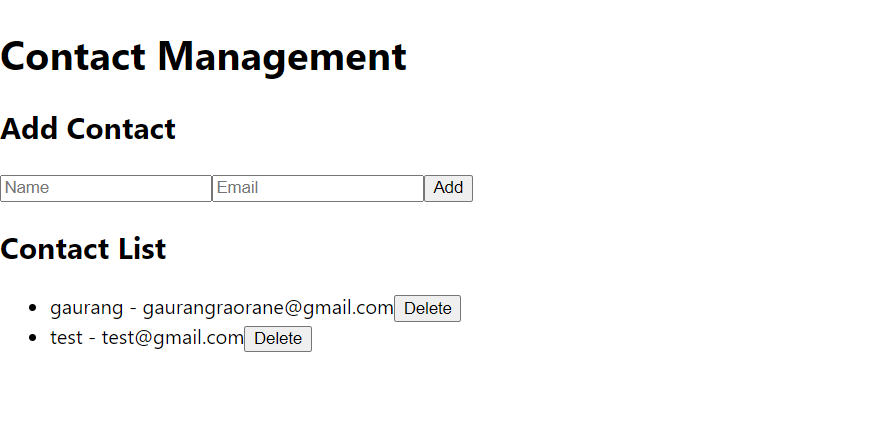
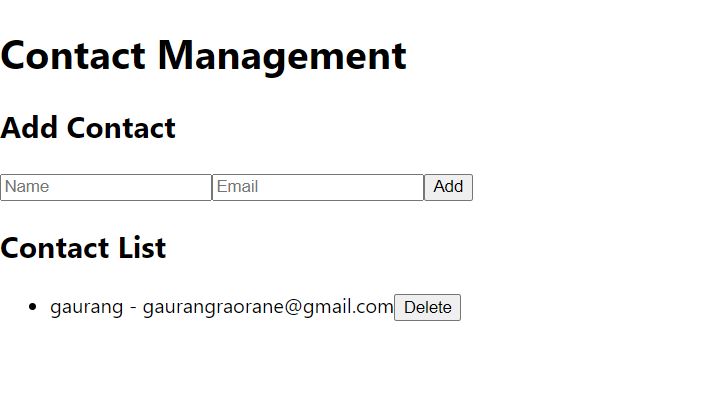
);

}

}

export default App;

**Output:-**

****