Ex. No.: 07	Pushing Up The Data
Date : 04/09/2024	

Aim:

To create a React application that collects user data (name, age, email, and address) using a form and displays it upon submission. This application demonstrates form handling, state management, and user interaction in React.

Procedure:

1. Setup the Environment:

- Ensure Node.js and npm are installed on your system.
- Create a new React application using Create React App:

bash

npx create-react-app data-collection-app

cd data-collection-app

2. Create the App Component:

- Replace the code in src/App.js with the provided code for data collection.

3. Import useState:

- Import the useState hook from React to manage the state of each input field.

4. Define State Variables:

- Use useState to create four state variables:
- name for the user's name.
- age for the user's age.
- email for the user's email.
- address for the user's address.
- Each state variable is initialized with an empty string ('').

5. Handle Form Submission:

- Define a handleSubmit function that prevents the page from reloading using event.preventDefault().
- Log each state variable to the console.
- Display an alert with the user's data.
- Optionally clear each input field by resetting the state variables to empty strings.

6. Render the Form UI:

- Create a form with four input fields for name, age, email, and address, and a submit button:
- Each input field is connected to its respective state variable, and updates its value via an onChange event.
- Style the form for a clean and user-friendly interface**.**

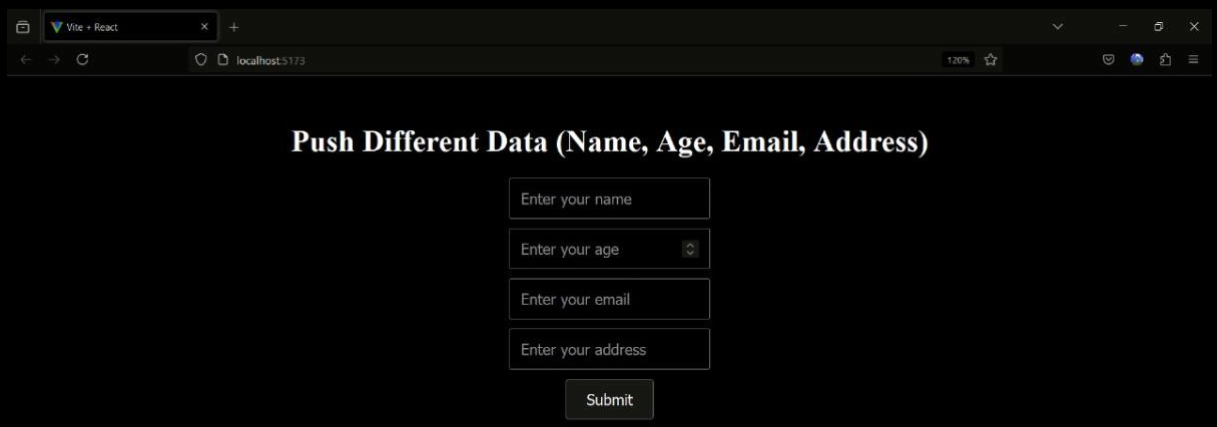
Source Code:

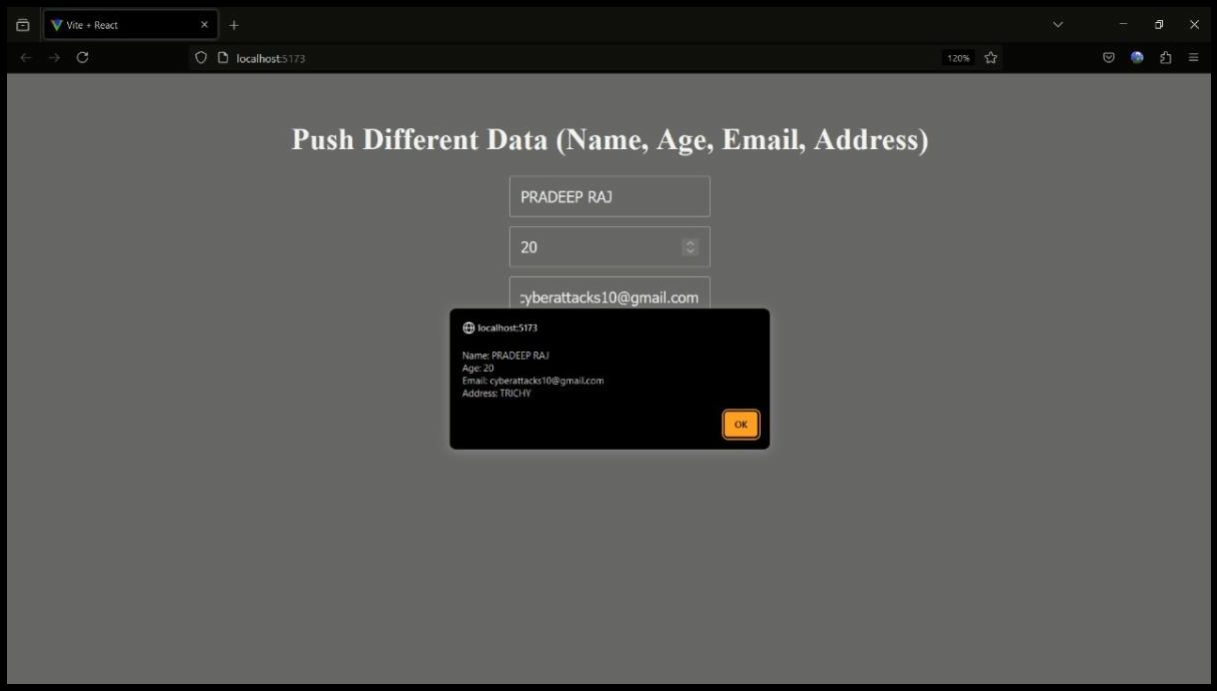
```
import { useState } from 'react';


function App() {
  const [name, setName] = useState('');       // State for storing name
  const [age, setAge] = useState('');         // State for storing age
  const [email, setEmail] = useState('');     // State for storing email
  const [address, setAddress] = useState(''); // State for storing address


 // Controller function to handle form submission
 const handleSubmit = (event) => {
   event.preventDefault(); // Prevents page reload on form submission


   // Log the data (you can also push to a backend API)
   console.log("Name:", name);
   console.log("Age:", age);
   console.log("Email:", email);
   console.log("Address:", address);


   // Display the data in an alert (or handle it further)
   alert(Name: ${name}\nAge: ${age}\nEmail: ${email}\nAddress: ${address});


   // Optionally clear the form after submission
   setName('');
   setAge('');
   setEmail('');
   setAddress('');
 };
```

```
return (
  <div style={{ textAlign: 'center', marginTop: '50px' }}>
    <h1>Push Different Data (Name, Age, Email, Address)</h1>

    {/* Form for collecting data */}
    <form onSubmit={handleSubmit}>
      {/* Input for Name */}
      <input
        type="text"
        placeholder="Enter your name"
        value={name}
        onChange={(event) => setName(event.target.value)} // Update name state
        style={{ padding: '10px', fontSize: '16px', marginBottom: '10px' }}
      />
      <br />

      {/* Input for Age */}
      <input
        type="number"
        placeholder="Enter your age"
        value={age}
        onChange={(event) => setAge(event.target.value)} // Update age state
        style={{ padding: '10px', fontSize: '16px', marginBottom: '10px' }}
      />
      <br />

      {/* Input for Email */}
      <input
        type="email"
        placeholder="Enter your email"
        value={email}
        onChange={(event) => setEmail(event.target.value)} // Update email state
```

```
          style={{ padding: '10px', fontSize: '16px', marginBottom: '10px' }}
        />
        <br />

        {/* Input for Address */}
        <input
          type="text"
          placeholder="Enter your address"
          value={address}
          onChange={(event) => setAddress(event.target.value)} // Update address state
          style={{ padding: '10px', fontSize: '16px', marginBottom: '10px' }}
        />
        <br />

        {/* Submit Button */}
        <button type="submit" style={{ padding: '10px 20px', fontSize: '16px' }}>
          Submit
        </button>
      </form>
    </div>
  );
}

export default App;
```

Output:





Result:

The program successfully creates the student100 table, inserts student records, and retrieves specific and all student details using PL/SQL blocks with cursors. The output confirms that the program works as intended, showcasing how to use cursors and DBMS_OUTPUT for displaying results in PL/SQL.