TÀI LIỆU HƯỚNG DẪN LẬP TRÌNH REACT CƠ BẢN (CRUD)

I. MỤC TIÊU BÀI HỌC

- Hiểu cấu trúc ứng dụng React gồm nhiều Functional Component.
- Sử dụng Hooks (useState, useEffect) để quản lý trạng thái và vòng đời.
- Áp dụng mô hình CRUD (Create, Read, Update, Delete).
- Thực hành Luồng dữ liệu một chiều giữa Component cha và con
- Thiết kế Form thêm/sửa, giao diện thân thiện.

II. TRÌNH TỰ THỰC HIỆN

BƯỚC 1. THIẾT LẬP CẤU TRÚC REACT CƠ BẢN

1. Tạo file HTML với khung cơ bản và thẻ `<div id='root'></div>`
2. Thêm các thư viện cần thiết (React, ReactDOM, Babel – JS Compiler) qua CDN

```
//trong head
<script
src='https://unpkg.com/react@18/umd/react.development.js'></script>
<script src='https://unpkg.com/react-dom@18/umd/react-
dom.development.js'></script>
<script src='https://unpkg.com/@babel/standalone/babel.min.js'></script>
```

3. Viết mã React trong thẻ `<script type= "text/babel">`
4. Tạo component gốc App và render bằng ReactDOM:

```
function App() {
  return <div><h1>Quản lý người dùng</h1></div>;
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

BƯỚC 2. TỔ CHỨC COMPONENT VÀ STATE TẬP TRUNG

1. Ứng dụng gồm các component: App, SearchForm, AddUser, ResultTable.

Component	Vai trò
App	Quản lý toàn bộ state và truyền props xuống con

SearchForm	Nhận input từ người dùng để tìm kiếm
AddUser	Form thêm người dùng mới
ResultTable	Hiển thị, lọc, sửa, xóa người dùng

Nguyên tắc: Dữ liệu cần chia sẻ (Từ khóa tìm kiếm, Người dùng mới) phải được lưu trữ ở component cha (App) và truyền xuống qua props. Các hàm cập nhật state được truyền xuống component con thông qua Props (State Lifting)**.**

2. Mã minh họa

```
function App() {
  const [kw, setKeyword] = React.useState("");
  const [newUser, setNewUser] = React.useState(null);
  return (
    <div>
     <SearchForm onChangeValue={setKeyword} />
     <AddUser onAdd={setNewUser} />
     <ResultTable    keyword={kw}    user={newUser}    onAdded={()    =>
setNewUser(null)} />
    </div>
  );
}
```

BƯỚC 3. CHỨC NĂNG TÌM KIẾM (SEARCHFORM)

1. **Mục tiêu:** Truyền dữ liệu tìm kiếm từ con → cha → bảng kết quả.

Logic:

- SearchForm nhận hàm onChangeValue (tức là setKeyword từ App).
- Sử dụng sự kiện onChange để kích hoạt hàm callback, cập nhật kw ở App.

2. **Mã minh họa**

```
function SearchForm({ onChangeValue }) {
  return (
    <input
      type="text"
      placeholder="Tìm theo name, username"
      onChange={(e) => onChangeValue(e.target.value)}
    />
  );
}
```

Khi người dùng nhập, App cập nhật kw, và ResultTable tự lọc lại danh sách qua keyword.

BƯỚC 4. HIỂN THỊ DANH SÁCH (RESULTTABLE)

1. **Mục tiêu:** Tải dữ liệu từ API, lưu trong state, và render ra bảng.
2. **Logic**

- Tải dữ liệu: Dùng React.useEffect (với dependency array []) để gọi API **một lần** khi component được mount.
- Lọc dữ liệu: Dùng Array.filter() để lọc danh sách users theo keyword nhận được từ props.

3. **Mã minh họa**

```
function ResultTable({ keyword, user, onAdded }) {
  const [users, setUsers] = React.useState([]);
  const [loading, setLoading] = React.useState(true);

  // Tải dữ liệu 1 lần khi component mount
  React.useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/users")
      .then(res => res.json())
      .then(data => { setUsers(data); setLoading(false); });
  }, []);
```

4. **Lọc danh sách theo keyword**

```
const filteredUsers = users.filter(
  (u) =>
    u.name.toLowerCase().includes(keyword.toLowerCase()) ||
    u.username.toLowerCase().includes(keyword.toLowerCase())
);
```

5. **Hiển thị dữ liệu bằng map**

```
<tbody>
  {filteredUsers.map((u) => (
    <tr key={u.id}>
      <td>{u.id}</td>
      <td>{u.name}</td>
      <td>{u.username}</td>
      <td>{u.email}</td>
      <td>{u.address.city}</td>
      <td>
        <button onClick={() => editUser(u)}>Sửa</button>
        <button onClick={() => removeUser(u.id)}>Xóa</button>
      </td>
    </tr>
  ))}
</tbody>
```

➕ BƯỚC 5. THÊM NGƯỜI DÙNG (ADDUSER)

1. **Mục tiêu:** Tạo form thêm người dùng mới, xử lý state lồng nhau (address), và gửi dữ liệu lên App.
2. **Logic**
   - State Form**:** Dùng useState để quản lý các trường nhập liệu.
   - Xử lý State Lồng Nhau (Nested State): Khi cập nhật các trường như street (nằm trong address), phải sử dụng Spread Operator (...) để sao chép cả đối tượng cha (address) để tránh mất dữ liệu cũ và đảm bảo React nhận ra sự thay đổi.
   - Thao tác Thêm**:** Gọi onAdd(user) để truyền dữ liệu lên App.
   - Cập nhật danh sách**:** Trong ResultTable, dùng useEffect([user]) để lắng nghe người dùng mới và thêm vào danh sách users.

3. **Mã minh họa**
   - ***Component con - AddUser***

```
function AddUser({ onAdd }) {
  const [adding, setAdding] = React.useState(false);
  const [user, setUser] = React.useState({
    name: "", username: "", email: "",
    address: { street: "", suite: "", city: "" },
    phone: "", website: ""
  });

  const handleChange = (e) => {
    const { id, value } = e.target;
    if (["street", "suite", "city"].includes(id)) {
      setUser({ ...user, address: { ...user.address, [id]: value } });
    } else {
      setUser({ ...user, [id]: value });
    }
  };

  const handleAdd = () => {
    if (user.name === "" || user.username === "") {
      alert("Vui lòng nhập Name và Username!");
      return;
    }
    onAdd(user);
    setUser({ name: "", username: "", email: "", address: { street: "",
suite: "", city: "" }, phone: "", website: "" });
    setAdding(false);
  };

---- UI ----
<div>
```

```
<button onClick={() => setAdding(true)}>Thêm</button>
 {adding && (
    <div>
       <div>
           <h4>Thêm người dùng</h4>
            <label htmlFor="name"> Name: </label>
            <input id="name" type="text" value={user.name}
             onChange={handleChange}/>
….
)}
```

4. Khi prop user thay đổi → thêm vào danh sách trong ResultTable:
   Component ResultTable

```
React.useEffect(() => {
  if (user) {
    setUsers((prev) => [...prev, { ...user, id: prev.length + 1 }]);
    onAdded();
  }
}, [user]);
```

✏️ BƯỚC 6. SỬA NGƯỜI DÙNG (EDIT)

1. **Logic:**
   - Kích hoạt form Sửa**:** Khi nhấn **Sửa**, lưu dữ liệu người dùng đó vào state **editing**.
   - Kỹ thuật quan trọng**:** Deep Copy**.** Phải sao chép (...) đối tượng user và các đối tượng lồng nhau (address) vào state **editing** để tránh việc thay đổi dữ liệu trong form làm thay đổi dữ liệu gốc trong bảng (do tham chiếu).
   - Lưu Dữ liệu: Dùng Array.map() để tìm id tương ứng và thay thế bằng đối tượng editing đã được cập nhật.
2. **Khai báo state trong Component ResultTable**

```
const [editing, setEditing] = React.useState(null);
```

3. **Sao chép dữ liệu user sang state editing**

```
function editUser(user) {
  setEditing({ ...user, address: { ...user.address } });
}
```

4. **Mỗi khi người dùng sửa trên form thì cập nhật dữ liệu**

```
{editing && (
 <label htmlFor="name"> Name </label>
  <input id="name" type="text" value={editing.name}
  onChange={(e) => handleEditChange("name", e.target.value)  />
```

5. **Lưu sau khi chỉnh sửa:**

```
function saveUser() {
 setUsers(prev => prev.map(u => u.id === editing.id ? editing : u));
 setEditing(null);
}
```

❌ BƯỚC 7. XÓA NGƯỜI DÙNG (DELETE)

1. **Mục tiêu:** Xóa người dùng trực tiếp khỏi danh sách, không cần xác nhận.
2. Trong component ResultTable, việc xóa người dùng được thực hiện bằng cách sử dụng hàm **filter()** để tạo ra một mảng mới mà không chứa phần tử có id trùng với người cần xóa.

```
function removeUser(id) {
  // Giữ lại tất cả người dùng có id khác với id cần xóa
  setUsers((prev) => prev.filter((u) => u.id != id));
}

<button className="btn-delete" onClick={() => removeUser(user.id)}>
  Xóa
</button>
```

3. **Lưu ý**

- React không cho phép chúng ta chỉnh sửa trực tiếp state; do đó, cần tạo **một mảng mới** bằng filter().
- Hàm setUsers((prev) => ...) đảm bảo luôn dùng phiên bản users mới nhất, tránh lỗi đồng bộ hóa.

🎨 BƯỚC 8. GIAO DIỆN (CSS) VÀ MODAL FORM

1. **Mục tiêu:** Tạo giao diện bảng dữ liệu, form nhập liệu và modal thân thiện với người dùng, sử dụng CSS cơ bản.
2. **Mã tham khảo – Modal khi thêm hoặc chỉnh sửa user**

```
/* Modal Overlay */
.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.4);
  display: flex;
```

```
  justify-content: center;
  align-items: center;
  z-index: 999;
}

/* Modal Content */
.modal-content {
  background: #fff;
  padding: 6px;
  border-radius: 6px;
  width: 300px;
}
```

BƯỚC 9. KIỂM THỬ VÀ TỔNG KẾT

✅ Kiểm tra đầy đủ CRUD: Thêm, Sửa, Xóa, Tìm kiếm.

✅ Luồng dữ liệu chuẩn React:

- AddUser → App → ResultTable
- SearchForm → App → ResultTable

✅ Nắm vững useState, useEffect, State Lifting, Controlled Components.