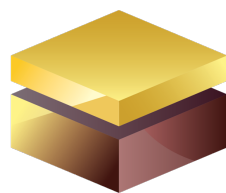
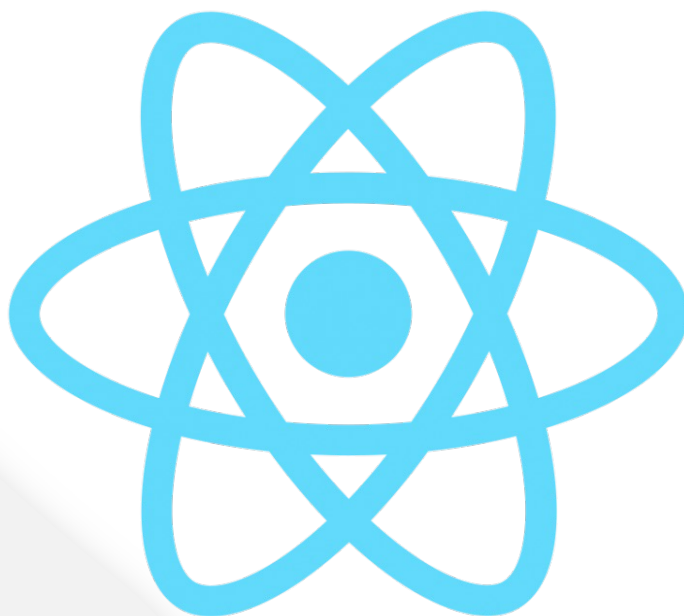
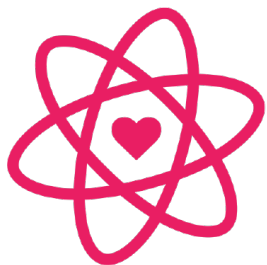


React js



CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

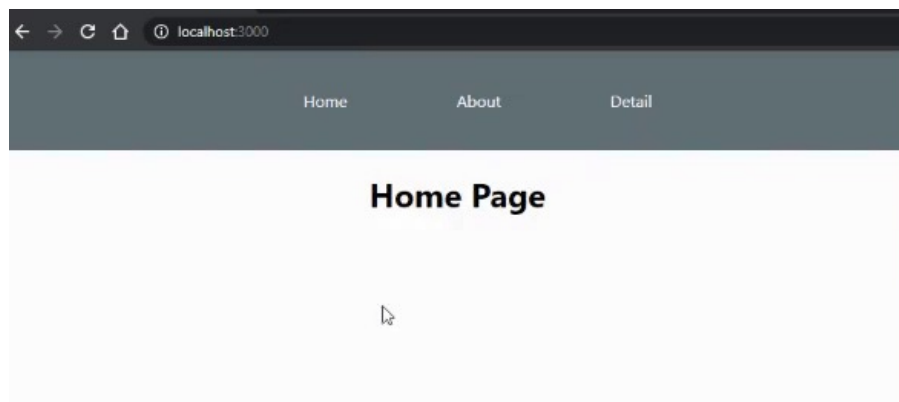
Hướng dẫn : Trương Tấn Khải



React JS

Routing React router dom Ver.6

React Form - Validation

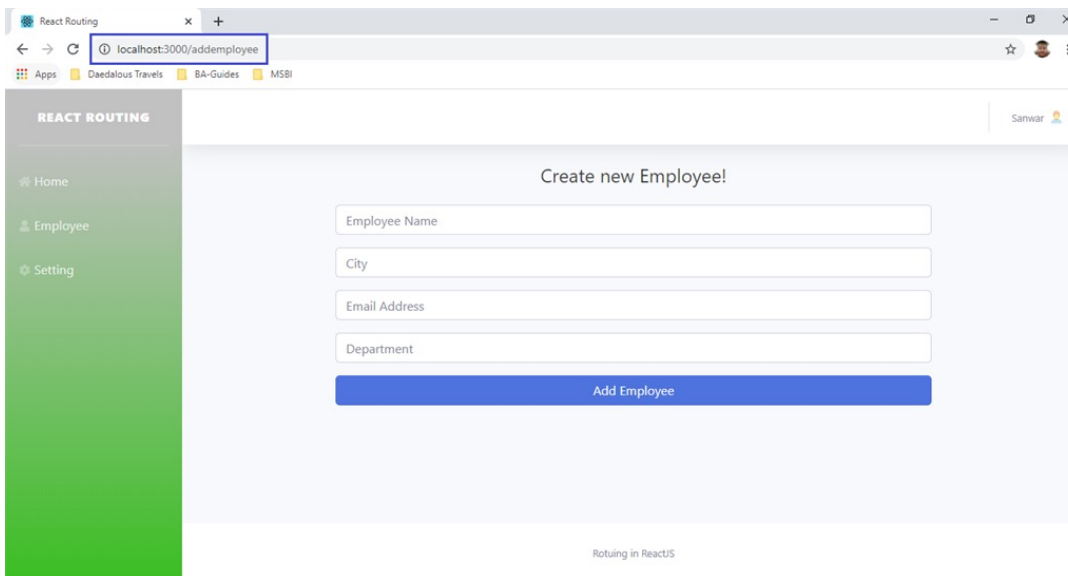


Routing (react-router-dom)

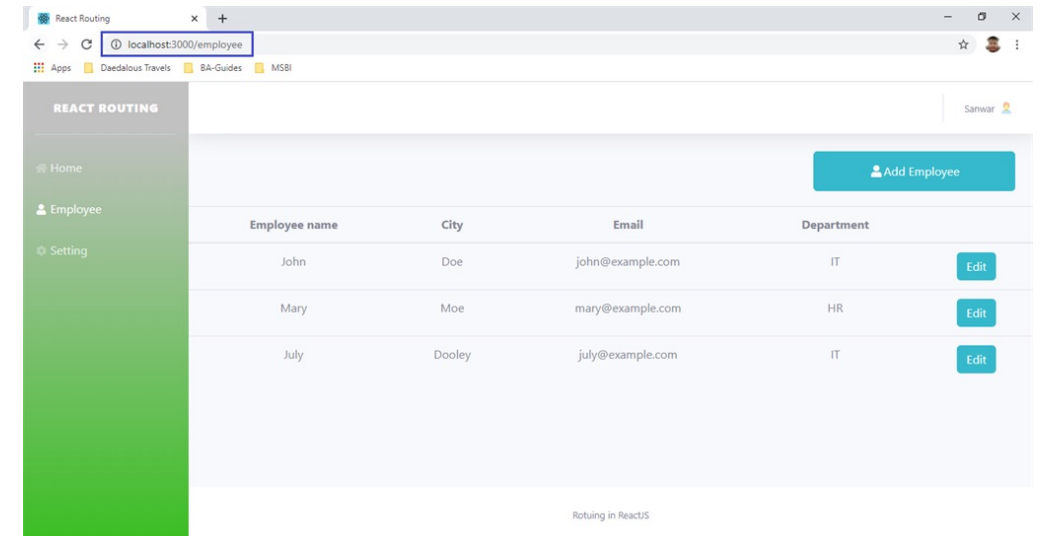
Routing là gì ?

- ❖ Routing là cơ chế trong single page giúp ta chuyển đổi qua lại giữa các component
 - ❖ Để sử dụng được routing với reactjs, ta cần package hỗ trợ đó là React-router-dom
 - ❖ Tiến hành cài đặt: `npm install --save react-router-dom`
 - ❖ Sử dụng: tại app.js, ta đang có 2 component không thể hiện cùng lúc, đó là danh sách khóa học, và danh sách sinh viên . Do đó ta sẽ dùng routing để quản lý, dựa theo đường dẫn url để hiển thị component tương ứng
- Document: <https://reactrouter.com/docs/en/v6/getting-started/overview>

Ví dụ:



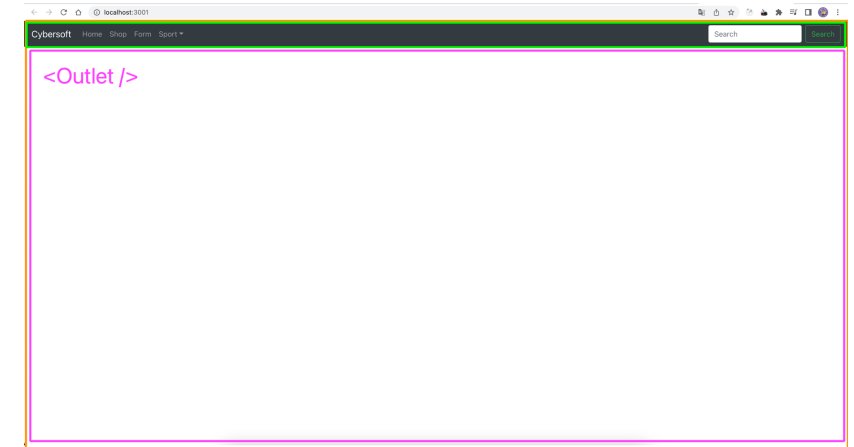
url: <https://localhost:3000/addemployee>



url: <https://localhost:3000/employee>

Routing (react-router-dom)

- Đối với react router dom phiên bản 6. Cơ chế hoạt động như sau:
- React router dom sẽ dùng thẻ BrowserRouter để định nghĩa phần phạm vi nội dung được load ra thành bởi thành url.
- Mỗi phần nội dung load ra được định nghĩa trên 1 thẻ Routes
- Trong mỗi thẻ **Routes** sẽ có các thẻ **Route** quy định nội dung được load.
- Ví dụ bên phải:
 - Thẻ **Route** quy định khi người dùng gõ path là “(rỗng)” thì nội dung mặc định load ra trên trình duyệt là Component **HomeLayout**. (Viền cam bên hình mà chúng ta thấy)
 - Bên trong nội dung của **HomeLayout** sẽ được chia nhỏ các phần con như **shop** và **form** theo đường dẫn / tiếp theo

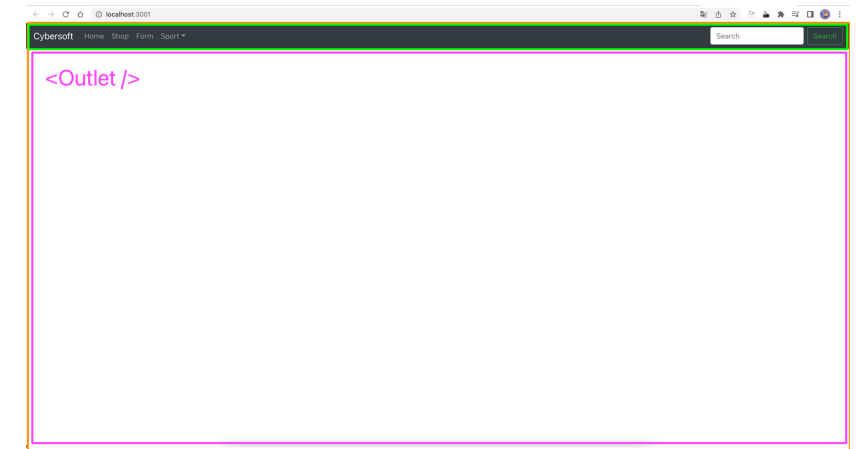


```
import './App.css';
import { Routes, Route, BrowserRouter } from 'react-router-dom'
import Shop from './Pages/Shop';
import HomeLayout from './templates/HomeLayout';
import ReactForm from './Pages/ReactForm/ReactForm';
function App(props) {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<HomeLayout />} >
          <Route path="shop" element={<Shop />} />
          <Route path="form" element={<ReactForm />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Routing (react-router-dom)

- Nội dung home layout sẽ được load ra với **Header** và 1 thẻ tên **<Outlet />**
- Thẻ header là thẻ cố định còn thẻ Outlet sẽ đại diện khi người dùng tiếp tục gõ /link
- Localhost:3000 / shop => Thẻ outlet sẽ được thế = thẻ ShopComponent
- Localhost:3000/ form => Thẻ outlet sẽ được thế = thẻ FormComponent



```
<Route path="/" element={<HomeLayout />} >
  <Route path="shop" element={<Shop />} />
  <Route path="form" element={<ReactForm />}
/>
</Route>
```

App.jsx

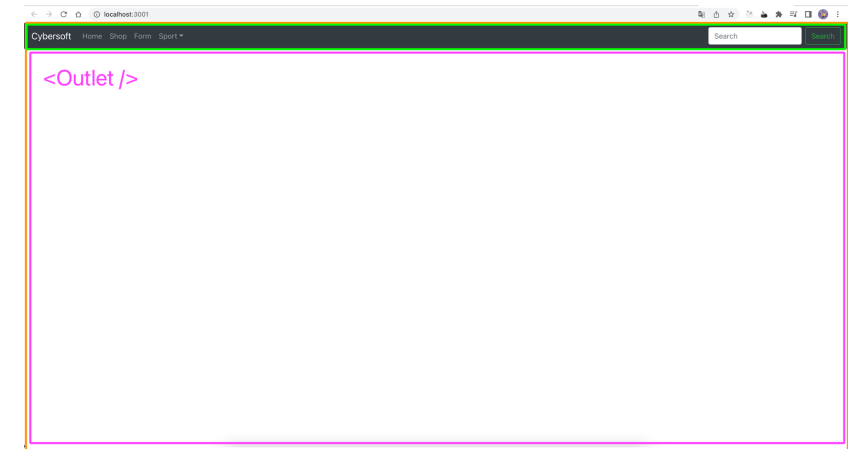
```
import React from 'react'
import { Outlet } from 'react-router-dom'
import HeaderHome from '../components/HeaderHome'

export default function HomeLayout(props) {
  return (
    <>
      <HeaderHome />
      <Outlet />
    </>
  )
}
```

HomeLayout.jsx

Routing (react-router-dom)

- Nội dung home layout sẽ được load ra với **Header** và 1 thẻ tên **<Outlet />**
- Thẻ header là thẻ cố định còn thẻ Outlet sẽ đại diện khi người dùng tiếp tục gõ /link
- Localhost:3000 / shop => Thẻ outlet sẽ được thế = thẻ ShopComponent
- Localhost:3000/ form => Thẻ outlet sẽ được thế = thẻ FormComponent



```
<Route path="/" element={<HomeLayout />} >
  <Route path="shop" element={<Shop />} />
  <Route path="form" element={<ReactForm />}
/>
</Route>
```

App.jsx

```
import React from 'react'
import { Outlet } from 'react-router-dom'
import HeaderHome from '../components/HeaderHome'

export default function HomeLayout(props) {
  return (
    <>
      <HeaderHome />
      <Outlet />
    </>
  )
}
```

HomeLayout.jsx

Routing (react-router-dom)

- Tương tự như vậy ở 1 hệ thống website ta có thể định nghĩa nhiều Routes để build nhiều template khác nhau. Và phân Route theo đặc trưng của từng hệ thống.

```
import './App.css';
import { Routes, Route, BrowserRouter } from 'react-router-dom'
import Shop from './Pages/Shop';
import HomeLayout from './templates/HomeLayout';
import ReactForm from './Pages/ReactForm/ReactForm';
import AdminLayout from './templates/AdminLayout';
import UserManagement from './Pages/Admin/UsersManagement/UserManagement';
import ProductManagement from './Pages/Admin/ProductManagement/ProductManagement';
function App(props) {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<HomeLayout />} >
          <Route path="shop" element={<Shop />} />
          <Route path="form" element={<ReactForm />} />
        </Route>
      </Routes>

      <Routes>
        <Route path="/admin" element={<AdminLayout />} >
          <Route path="users" element={<UserManagement />} />
          <Route path="products" element={<ProductManagement />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Routing (react-router-dom)

➤ NavLink (Hoặc <Link>)

- Thư viện react router dom hỗ trợ ta thẻ **NavLink** thay thế cho thẻ **<a>** với **href** đổi thành **to**, giúp cho ta có thể chuyển đổi qua lại giữa các trang mà không cần load lại toàn bộ html của trang đó.
- Ngoài ra **<NavLink>** còn cho phép ta custom link css, khi người dùng gõ đúng đường dẫn thì thẻ NavLink sẽ được active class hoặc style thông qua thuộc tính tương ứng **className, style**.
Style và class name sẽ trả ra cho ta thuộc tính active = true (Đối với class ta truyền vào function trả về string Class, đối với style ta truyền vào function trả về object style).

```
<NavLink  
  className={({isActive}) => isActive ? 'nav-link bg-warning text-white' : 'nav-link'}  
  to="/home"  
  style={({isActive}) => isActive ? {color:'red'} : {}}>  
    Home  
</NavLink>
```


Routing (react-router-dom)

➤ Navigate (react-class-component)

- Component navigate được cung cấp bởi react-router-dom hỗ trợ ta điều hướng từ path này tới path khác

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

export default class UserManagement extends Component {
  render() {
    if(!localStorage.getItem('userLogin')) {
      return <Navigate to="/" replace={false} />
    }
    return (
      <div>
        <h3>admin/users</h3>
      </div>
    )
  }
}
```

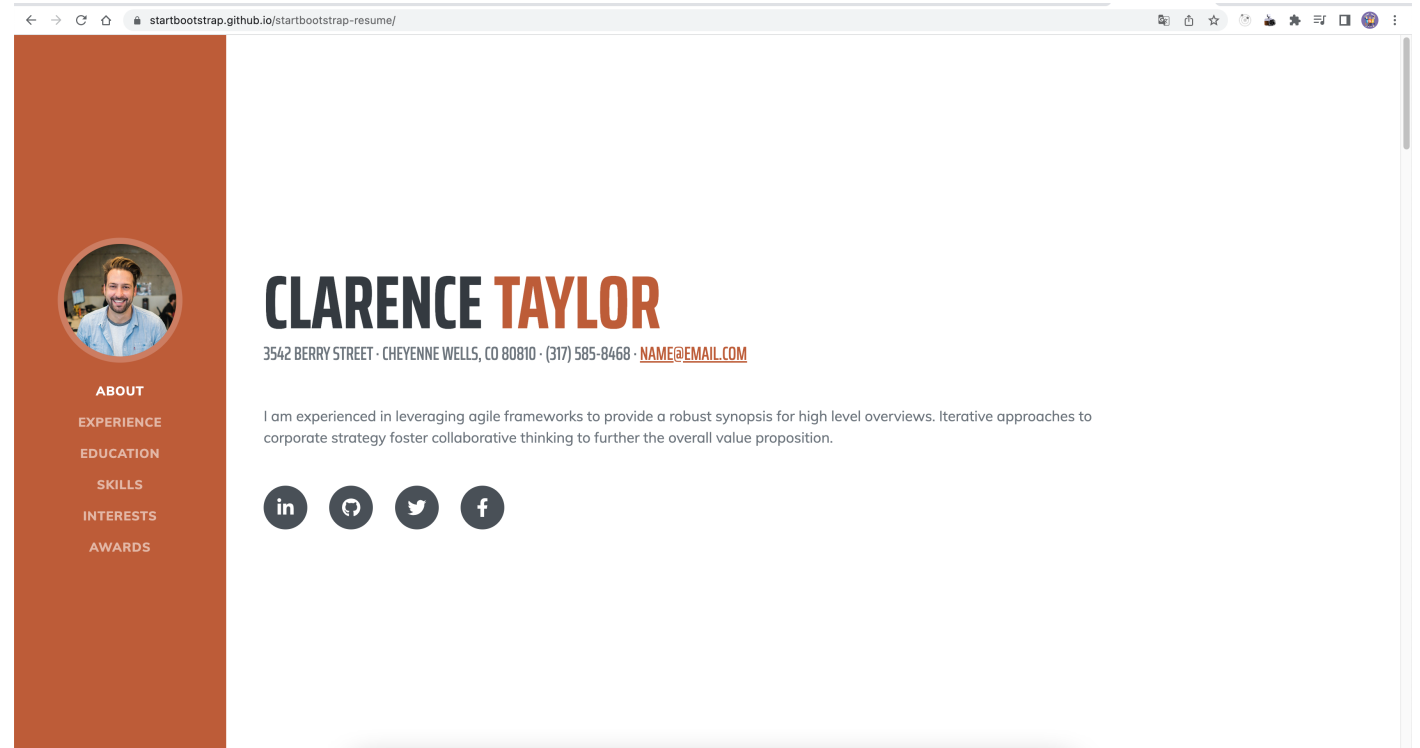
- Khi nội dung component gặp đối tượng component `<Navigate />` thì sẽ chuyển hướng về path tương ứng. Lưu ý: có thể chuyển hướng dạng replace

Tóm tắt và bài tập (Tạo lại dự án mới tự setup)

- ❖ <BrowserRouter> Là thẻ dùng để bọc phần hiển thị nội dung của thẻ route
 - ❖ Thay vì tất cả component cùng được load lên trình duyệt → ta sẽ định nghĩa các Route link tương ứng thông qua (path, component)
 - ❖ Ngoài ra 1 component được load từ thẻ Router còn được cung cấp thẻ 1 số props như : history, location, match, ... Để tương tác với các công cụ trên url.

Bài tập:

- Xây dựng cho mình trang cv với react router dom



React Form - Validation

- ❑ Khác với javascript ta thường sử dụng để lấy giá trị thông qua 1 tag <input> ta phải thực hiện dom thông qua các selector sau đó truy xuất đến thuộc tính value để lấy giá trị ... mất khá nhiều thao tác. Tuy nhiên:
- ❑ Trong react chúng ta sẽ sử dụng thuộc tính state của component kết hợp cùng các sự kiện của control input để thực hiện việc lấy dữ liệu từ form dễ dàng hơn qua ví dụ sau:
- ❑ Css: https://drive.google.com/drive/folders/19pRgMaTFw3wNLnwSkynv_oXfZSZ7welq?usp=sharing

The screenshot displays a web application interface for managing products. At the top, there is a navigation bar with the logo 'Cybersoft' and links for 'Home', 'Shop', 'Form', and 'Sport'. A search bar is located on the right side of the navigation bar. Below the navigation bar, the breadcrumb 'Home/Products/Create' is visible. The main section is titled 'Create product' and contains a 'Product info' form. The form has several input fields: 'id' (with placeholder 'product id'), 'name' (with placeholder 'product name'), 'price' (with placeholder 'product price'), 'image' (with placeholder 'product image link'), 'ProductType' (a dropdown menu currently showing 'mobile'), and 'price' (with placeholder 'product price'). Below the form are two buttons: 'Create' (blue) and 'Update' (green). At the bottom of the page, there is a table listing existing products. The table has columns for 'id', 'image', 'name', 'price', 'description', and 'type'. The first row shows a product with id '1', a small image, name 'Product 1', price '1.000', description 'Lorem ipsum dolor sit amet consectetur adipisicing elit.', and type 'Laptop'. There are also delete and edit icons for this product.

id	image	name	price	description	type
1		Product 1	1.000	Lorem ipsum dolor sit amet consectetur adipisicing elit.	Laptop

React Form - Validation

- Xây dựng giao diện như hình bằng bootstrap 4 như hình

- Xây dựng cấu trúc component như sau:

```
<FormProduct>  
  <Giao diện form />  
  <TableProducts />  
</FormProduct>
```

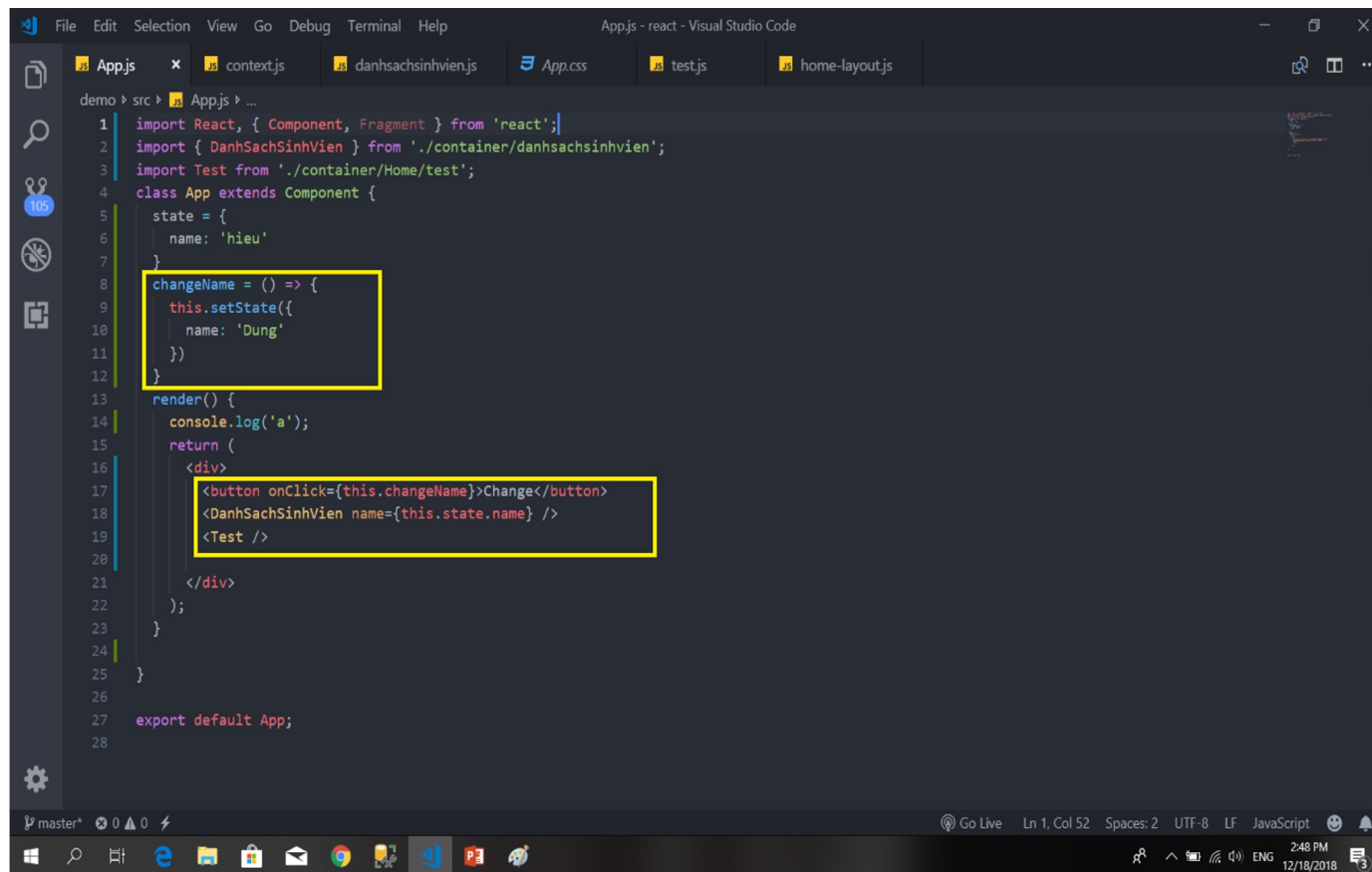
- Xác định state là gì ? (Array, object hay ...)
- Tổ chức lưu trữ state
- Thực hiện các tính năng CRUD
- Thực hiện kiểm tra validation

The screenshot shows a web application interface for managing products. At the top, there is a navigation bar with links: Cybersoft, Home, Shop, Form, Sport. A search bar is located on the right. Below the navigation bar, the breadcrumb path is 'Home/Products/Create'. The main section is titled 'Create product' and contains a 'Product info' form. The form has two columns of input fields: 'id' (product id), 'image' (product image link), 'name' (product name), 'ProductType' (a dropdown menu with 'mobile' selected), 'price' (product price), and 'price' (product description). Below the form are 'Create' and 'Update' buttons. At the bottom, there is a table with columns: id, image, name, price, description, type. The table contains one row with the following data: id: 1, image: [product image], name: Product 1, price: 1.000, description: Lorem ipsum dolor sit amet consectetur adipiscing elit., type: Laptop. There are also delete and edit icons for this row.

Giao diện mẫu

Pure component

- ❑ Trong một vài trường hợp, dù props của component không hề thay đổi nhưng vẫn bị update, dẫn tới ảnh hưởng performance của app
- ❑ Ví dụ :
- ❑ Ở đây ta có thể thấy, khi click vào nút change, state sẽ được set lại, dẫn tới hàm render() sẽ chạy lại và các component con bên trong app cũng được render lại.
- ❑ Vấn đề ở đây là state chỉ ảnh hưởng tới component DanhSachSinhVien, còn component Test vốn ko hề thay đổi, do đó việc render lại nó là ko cần thiết



```
1 import React, { Component, Fragment } from 'react';
2 import { DanhSachSinhVien } from './container/danhsachsinhvien';
3 import Test from './container/Home/test';
4 class App extends Component {
5   state = {
6     name: 'hieu'
7   }
8   changeName = () => {
9     this.setState({
10       name: 'Dung'
11     })
12   }
13   render() {
14     console.log('a');
15     return (
16       <div>
17         <button onClick={this.changeName}>Change</button>
18         <DanhSachSinhVien name={this.state.name} />
19         <Test />
20       </div>
21     );
22   }
23 }
24 export default App;
```

Pure component

- Để xử lý vấn đề này, ta có cách giải quyết sau
- Ở component Test, thay vì class Test extends Component, ta sẽ cho nó extends từ PureComponent, lúc này component Test chỉ render lại khi mà props của nó thật sự thay đổi

```
src ▸ container ▸ Home ▸ JS test.js ▸ ...
import React, { PureComponent } from 'react'

export default class test extends PureComponent {

  render() {

    return (
      <div>
        aaa
      </div>
    )
  }
}
```

➔ Lưu ý: chỉ sử dụng PureComponent, ko nên lạm dụng vì có thể dẫn tới lỗi . Bản chất của PureComponent là tự động kiểm tra xem nếu props và state của component đó thay đổi thì sẽ render lại, không thì thôi. Nhưng sự so sánh thay đổi của react là so sánh tham chiếu (shallow comparison – so sánh nguyên thủy) , nếu như ta truyền một object dưới dạng props, và thay đổi một thuộc tính nào đó thì react ko so sánh đc, vì căn bản là cùng 1 object

Tổng hợp kiến thức

- **Routing** là cơ chế chia các **component** thành **page** trong react, với mỗi page là mỗi **url** (đường dẫn khác nhau). Có thể dùng **NavLink** hoặc **navigate** để điều hướng trang.
- **NavLink** được sử dụng thay cho thẻ `<a>`
- **Navigate** được sử dụng để chuyển hướng trong các xử lý.
- **React form** là cách sử dụng **this.state** trong react hoặc thuộc tính của **class** để lấy dữ liệu từ **form**. Tương tự **validation** cũng vậy sử dụng state hoặc thuộc tính của form để quản lý.
- **Pure component** dùng để tối ưu render **component** khi **props** của component không thay đổi. (Lưu ý: **PureComponent** chỉ nhận biết props khi là giá trị **primitive value (boolean, string, number, null, undefined)**. Còn đối với **Reference (Object, array)** value thì purecomponent không nhận biết được mà ta phải clone ra 1 object hoặc array mới.