

  
  
**Báo Cáo học phần  
Thiết kế và lập trình giao diện Web nâng cao**

***ĐỀ TÀI***

**<<Thư viện xe hơi>>**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **Th.S Trịnh Văn Chung** |
| **Sinh viên thực hiện:** | **Nguyễn Trực Kiên** |
| **Mã sinh viên:** | **2210900033** |
| **Các thành viên:** |  |
| **Lớp:** | **K22CNT3** |
| **Khóa:** | **CNTT** |

**@FITNTU-** 09/2025

**MỤC LỤC**

[**DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ** 4](#_Toc209786146)

[**CHƯƠNG 1: NGHIÊN CỨU CƠ SỞ LÝ THUYẾT** 1](#_Toc209786147)

[1. Tổng quan về Reactjs 1](#_Toc209786148)

[1.1.1. Đặc điểm nổi bật của ReactJS 1](#_Toc209786149)

[1.1.2. Ưu điểm 1](#_Toc209786150)

[1.1.3. Nhược điểm 1](#_Toc209786151)

[1.1.4. Ứng dụng thực tế 2](#_Toc209786152)

[1.1.5. Tổng quan về ngôn ngữ lập trình Javascript 2](#_Toc209786153)

[1.2. Một số khái niệm quan trọng trong ReactJS 2](#_Toc209786154)

[1.2.1. Virtual DOM 2](#_Toc209786155)

[1.2.2. JSX 2](#_Toc209786156)

[1.2.3. Components 2](#_Toc209786157)

[1.2.4. Props và State 3](#_Toc209786158)

[1.2.5. React Lifecycle 3](#_Toc209786159)

[1.2.6. Event – Form – List & Key 3](#_Toc209786160)

[1.2.7. React Hooks 3](#_Toc209786161)

[1.2.8. Class Components 3](#_Toc209786162)

[1.2.9. Function Components 4](#_Toc209786163)

[1.2.10. React Router 5](#_Toc209786164)

[**CHƯƠNG 2 CHƯƠNG TRÌNH ỨNG DỤNG** 7](#_Toc209786165)

[2.1. Đặt vấn đề bài toán 7](#_Toc209786166)

[2.2. Hệ thống đề xuất 7](#_Toc209786167)

[2.2.1. Chức năng dành cho người dùng (User) 7](#_Toc209786168)

[2.2.2. Yêu cầu phi chức năng 7](#_Toc209786169)

[2.2.3. Công nghệ đề xuất sử dụng 8](#_Toc209786170)

[2.3. Chức năng của hệ thống 8](#_Toc209786171)

[2.4. Xây dựng phần mềm 8](#_Toc209786172)

[2.4.1. Một số mã nguồn chương trình 8](#_Toc209786173)

[NtkApp.js 8](#_Toc209786174)

[NtkNavBar.js 9](#_Toc209786175)

[2.4.2. Một số giao diện người dùng và chức năng cụ thể 10](#_Toc209786176)

[1. Giao diện trang chủ 10](#_Toc209786177)

[2. Giao diện trang quản lý xe 11](#_Toc209786178)

[2.1. Giao diện Thêm xe 11](#_Toc209786179)

[2.2. Giao diện Sửa xe 12](#_Toc209786180)

[3. Giao diện quản lý Users 12](#_Toc209786181)

[3.1. Giao diện Thêm user 13](#_Toc209786182)

[3.2. Giao diện Sửa user 13](#_Toc209786183)

[**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 15](#_Toc209786184)

[3.1. Kết luận 15](#_Toc209786185)

[3.2. Hướng phát triển: 15](#_Toc209786186)

[**DANH MỤC TÀI LIỆU THAM KHẢO** 17](#_Toc209786187)

# **DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ**

# **CHƯƠNG 1: NGHIÊN CỨU CƠ SỞ LÝ THUYẾT**

## 1. Tổng quan về Reactjs

* **ReactJS** là một thư viện JavaScript do **Facebook** (nay là Meta) phát triển, ra mắt vào năm 2013.
* Dùng để xây dựng **giao diện người dùng (UI)**, đặc biệt là **ứng dụng web đơn trang (SPA – Single Page Application)**.
* React hướng đến việc phát triển các **component** (thành phần giao diện) có thể **tái sử dụng**, giúp tăng hiệu suất và dễ quản lý.
* Đặc điểm nổi bật của ReactJS:

### 1.1.1. Đặc điểm nổi bật của ReactJS

* **Component-based**: Ứng dụng được chia thành nhiều component nhỏ, độc lập → dễ tái sử dụng.
* **Virtual DOM**: Cơ chế so sánh và cập nhật thông minh, chỉ render lại phần thay đổi → tăng tốc độ.
* **One-way Data Binding**: Dữ liệu truyền theo 1 chiều từ cha → con, dễ kiểm soát luồng dữ liệu.
* **JSX (JavaScript XML)**: Kết hợp JavaScript và HTML trong cùng một file → viết code trực quan hơn.
* **Hỗ trợ thư viện/phần mở rộng phong phú**: Redux, React Router, Next.js...

Nhờ các đặc điểm trên, ReactJS hiện là một trong những thư viện front-end phổ biến nhất, được sử dụng bởi nhiều công ty công nghệ lớn như Facebook, Instagram, Netflix, Airbnb,…

### 1.1.2. Ưu điểm

- Hiệu suất cao nhờ **Virtual DOM**  
- Tái sử dụng component → tiết kiệm thời gian phát triển  
- Cộng đồng lớn, tài liệu phong phú  
- Tương thích tốt với nhiều thư viện khác  
- SEO-friendly hơn so với nhiều framework JS khác (khi dùng Next.js)

### 1.1.3. Nhược điểm

- React chỉ tập trung vào UI → cần kết hợp với thư viện khác (Redux, React Router) để làm - SPA hoàn chỉnh  
- JSX khá khó làm quen với người mới  
- Tốc độ thay đổi nhanh, cần cập nhật liên tục

### 1.1.4. Ứng dụng thực tế

ReactJS hiện đang được dùng trong các ứng dụng web lớn như:

* Facebook, Instagram
* Netflix
* Airbnb
* WhatsApp Web
* Shopee, Tiki (một số phần)

### 1.1.5. Tổng quan về ngôn ngữ lập trình Javascript

JavaScript (JS) là ngôn ngữ lập trình kịch bản phía client, được Brendan Eich phát triển năm 1995. Ban đầu JavaScript chỉ dùng để xử lý các tác vụ đơn giản trên trình duyệt, nhưng đến nay đã trở thành ngôn ngữ đa năng, ứng dụng ở cả **front-end** và **back-end** (Node.js).

Đặc điểm chính của JavaScript:

* Hỗ trợ **lập trình hướng sự kiện** (event-driven).
* Có thể chạy trên nhiều nền tảng khác nhau.
* Cú pháp dễ học, tương tự C/Java.
* Kết hợp với HTML, CSS để tạo ra các ứng dụng web tương tác, hiện đại.

## 1.2. Một số khái niệm quan trọng trong ReactJS

### 1.2.1. Virtual DOM

Virtual DOM là bản sao của DOM thật, được React quản lý. Khi dữ liệu thay đổi, React sẽ so sánh Virtual DOM mới với bản cũ, từ đó chỉ cập nhật những phần tử thay đổi lên DOM thật. Điều này giúp tối ưu hiệu năng, giảm số lần thao tác DOM vốn rất tốn tài nguyên.

### 1.2.2. JSX

JSX là cú pháp mở rộng cho JavaScript, cho phép lập trình viên viết code giống HTML ngay trong file JS. JSX được trình biên dịch Babel dịch sang JavaScript thuần.

Ví dụ:

const element = <h1>Hello, React!</h1>;

### 1.2.3. Components

Component là khối xây dựng cơ bản trong React, có thể hiểu là một “mảnh” giao diện độc lập. Có hai loại:

* **Class Components**: Được khai báo dưới dạng class, có hỗ trợ lifecycle methods.
* **Function Components**: Viết dưới dạng hàm, thường kết hợp với Hooks để quản lý state và lifecycle.

### 1.2.4. Props và State

* **Props**: Thuộc tính được truyền từ component cha xuống component con (dữ liệu chỉ đọc).
* **State**: Trạng thái cục bộ bên trong component, có thể thay đổi bằng hàm setState hoặc useState.

### 1.2.5. React Lifecycle

Vòng đời của component trong React bao gồm các giai đoạn:

* Mounting (Khởi tạo)
* Updating (Cập nhật)
* Unmounting (Hủy bỏ)

### 1.2.6. Event – Form – List & Key

* **Event**: React cung cấp cách xử lý sự kiện tương tự DOM, nhưng sử dụng camelCase (ví dụ: onClick).
* **Form**: Quản lý dữ liệu nhập vào thông qua state hoặc controlled components.
* **List & Key**: Khi hiển thị danh sách, mỗi phần tử nên có key duy nhất để React tối ưu việc render.

### 1.2.7. React Hooks

Hooks là tính năng mới từ React 16.8, cho phép dùng state và các tính năng khác trong Function Component. Các hook thường dùng:

* useState (quản lý trạng thái)
* useEffect (quản lý vòng đời)
* useContext (quản lý dữ liệu toàn cục)

### 1.2.8. ****Class Components****

* Được viết dưới dạng **class** (ES6).
* Có thể sử dụng **state** và **lifecycle methods** (componentDidMount, componentDidUpdate, componentWillUnmount,…).
* Phải kế thừa từ React.Component.

**Ví dụ:**

import React, { Component } from "react";

class Welcome extends Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

componentDidMount() {

console.log("Component đã được render lần đầu");

}

render() {

return (

<div>

<h1>Xin chào, {this.props.name}</h1>

<p>Bạn đã bấm {this.state.count} lần</p>

<button onClick={() => this.setState({ count: this.state.count + 1 })}>

Bấm tôi

</button>

</div>

);

}

}

export default Welcome;

### 1.2.9. Function Components

* Được viết dưới dạng **hàm JavaScript**.
* Trước đây chỉ dùng để hiển thị UI (stateless).
* Từ React 16.8 trở đi, có thể sử dụng **Hooks** (như useState, useEffect) → nên Function Components ngày nay thay thế dần Class Components.

**Ví dụ:**

import React, { useState, useEffect } from "react";

function Welcome(props) {

const [count, setCount] = useState(0);

useEffect(() => {

console.log("Component đã render hoặc cập nhật");

}, [count]);

return (

<div>

<h1>Xin chào, {props.name}</h1>

<p>Bạn đã bấm {count} lần</p>

<button onClick={() => setCount(count + 1)}>Bấm tôi</button>

</div>

);

}

export default Welcome;

### 1.2.10. React Router

**Khái niệm:**  
React Router là một thư viện định tuyến dành cho ReactJS, giúp xây dựng các ứng dụng web đơn trang (Single Page Application – SPA). Với React Router, việc điều hướng giữa các trang trong ứng dụng sẽ diễn ra **mượt mà, không cần tải lại toàn bộ trang web**, từ đó nâng cao trải nghiệm người dùng.

**Chức năng chính:**

* **Routing**: Điều hướng giữa các trang trong ứng dụng (Home, About, Contact,…).
* **Dynamic Routing**: Cho phép sử dụng tham số động trong đường dẫn, ví dụ /products/:id.
* **Nested Routing**: Hỗ trợ lồng các route bên trong nhau, tổ chức ứng dụng theo cấu trúc module.
* **Navigation**: Cung cấp các component <Link>, <NavLink> để tạo menu điều hướng thay cho thẻ <a>.
* **Protected Route**: Cho phép kiểm soát quyền truy cập (ví dụ: chỉ người đã đăng nhập mới vào được trang quản trị).

**Cài đặt:**

npm install react-router-dom --save

**Ví dụ sử dụng cơ bản (React Router v6):**

import React from "react";

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

function Home() {

return <h2>Trang chủ</h2>;

}

function About() {

return <h2>Giới thiệu</h2>;

}

function Contact() {

return <h2>Liên hệ</h2>;

}

export default function App() {

return (

<Router>

<div>

<nav>

<ul>

<li><Link to="/">Trang chủ</Link></li>

<li><Link to="/about">Giới thiệu</Link></li>

<li><Link to="/contact">Liên hệ</Link></li>

</ul>

</nav>

<Routes>

<Route path="/" element={<Home />} />

<Route path="/about" element={<About />} />

<Route path="/contact" element={<Contact />} />

</Routes>

</div>

</Router>

);

}

**Ưu điểm:**

* Giúp tạo ứng dụng SPA mượt mà, không tải lại toàn trang.
* Dễ sử dụng, hỗ trợ đầy đủ tính năng điều hướng hiện đại.
* Có thể mở rộng với lazy loading để tối ưu hiệu suất.
* Tích hợp tốt với các tính năng khác của React như Hooks, Context API.

**Ứng dụng thực tế:**  
React Router được sử dụng rộng rãi trong các website và ứng dụng thương mại điện tử, hệ thống quản trị, blog, mạng xã hội,… nơi có nhiều trang và yêu cầu trải nghiệm điều hướng nhanh chóng.

# **CHƯƠNG 2 CHƯƠNG TRÌNH ỨNG DỤNG**

## Đặt vấn đề bài toán

Trong kỷ nguyên công nghệ số, việc tra cứu và quản lý thông tin trở nên cần thiết và phổ biến trong nhiều lĩnh vực, trong đó có ngành **ô tô**. Hiện nay, có rất nhiều dòng xe hơi trên thị trường với đa dạng về mẫu mã, giá cả, thông số kỹ thuật, xuất xứ, hãng sản xuất… Người dùng (khách hàng, sinh viên, nhà nghiên cứu hoặc người yêu xe) thường gặp khó khăn khi:

1. **Tìm kiếm thông tin xe**: Các nguồn thông tin rời rạc, không tập trung, khó so sánh.
2. **Quản lý dữ liệu xe**: Đối với quản trị viên, việc cập nhật, chỉnh sửa hoặc xoá thông tin xe nếu dùng cách thủ công (excel, văn bản) dễ bị sai sót và khó mở rộng.
3. **Tiện ích trực tuyến**: Người dùng có nhu cầu tra cứu nhanh thông tin về một dòng xe (giá bán, năm sản xuất, hãng, tốc độ tối đa, động cơ, hình ảnh minh hoạ…) nhưng chưa có một nền tảng trực tuyến thân thiện, dễ dùng.

Chính vì vậy, cần xây dựng một **Website Thư viện xe hơi** với các chức năng cơ bản:

* Quản lý danh sách xe hơi (thêm, sửa, xoá, tìm kiếm).
* Hiển thị thông tin chi tiết của từng xe (tên xe, hãng sản xuất, năm sản xuất, động cơ, tốc độ, giá, hình ảnh).
* Quản lý tài khoản người dùng (đăng nhập, đăng ký, phân quyền user/admin).
* Giao diện trực quan, dễ sử dụng, cho phép người dùng cuối dễ dàng tra cứu.

## Hệ thống đề xuất

Để giải quyết các vấn đề đã nêu, nhóm đề xuất xây dựng một **Website Thư viện xe hơi** với các chức năng chính như sau:

### 2.2.1. Chức năng dành cho người dùng (User)

* **Xem danh sách xe hơi**: hiển thị các xe theo dạng bảng hoặc card, có hình ảnh minh họa.
* **Quản lý xe**: thêm mới, chỉnh sửa, xoá thông tin xe hơi.
* **Quản lý người dùng**: xem danh sách user, thay đổi quyền truy cập hoặc trạng thái tài khoản.
* **Quản lý dữ liệu**: đồng bộ dữ liệu hình ảnh và thông tin kỹ thuật của xe.

### 2.2.2. Yêu cầu phi chức năng

* **Giao diện thân thiện, trực quan**: sử dụng ReactJS để thiết kế giao diện, dễ sử dụng cho cả người dùng phổ thông.
* **Hiệu năng và tốc độ tải nhanh**: tối ưu hoá việc gọi API bằng Axios, kết hợp React Router cho điều hướng mượt mà.

### 2.2.3. Công nghệ đề xuất sử dụng

* **Frontend**: ReactJS (Component, Hook, React Router DOM, Axios).
* **Api**: MockAPI

## Chức năng của hệ thống

* **Xem các loại xe**
* Quản lý xe
* Quản lý người dùng
* Thêm sửa xoá

## Xây dựng phần mềm

## Một số mã nguồn chương trình

### NtkApp.js

import './App.css';

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

import NtkNavBar from './components/NtkNavBar';

import NtkHome from './pages/NtkHome';

import NtkAbout from './pages/NtkAbout';

import NtkContact from './pages/NtkContact'

import Ntk404 from './pages/Ntk404'

import NtkUsers from './pages/NtkUsers';

import NtkUserAdd from './pages/NtkUserAdd';

import NtkUserEdit from './pages/NtkUserEdit';

import NtkCar from './pages/NtkCar';

export default function NtkApp() {

  return (

    <div className="container border">

      <h1>React router - Demo [Nguyễn Trực Kiên]</h1>

      <hr/>

      <BrowserRouter>

        <NtkNavBar/>

        <Routes>

          <Route path='/' element={<NtkHome />}/>

          <Route path='/about' element={<NtkAbout />}/>

          <Route path='/contact' element={<NtkContact/>}/>

          {/\* Nested route \*/}

          <Route path='/car' element={<NtkCar/>}>

          </Route>

          <Route path='/users' element={<NtkUsers/>}> </Route>

          <Route path="/users/add" element={<NtkUserAdd/>}/>

          <Route path="/users/edit/:id" element={<NtkUserEdit/>}/>

          {/\* 404 page \*/}

          <Route path='\*' element={<Ntk404 />}/>

        </Routes>

      </BrowserRouter>

    </div>

  );

}

### NtkNavBar.js

import React from 'react'

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

import '../NavBar.css'

export default function NtkNavBar() {

  return (

    <nav className="navbar">

      <NavLink to="/" className="nav-item">Home</NavLink>

      <NavLink to="/about" className="nav-item">About</NavLink>

      <NavLink to="/contact" className="nav-item">Contact</NavLink>

      <NavLink to="/car" className="nav-item">Cars</NavLink>

      <NavLink to="/users" className="nav-item">Users</NavLink>

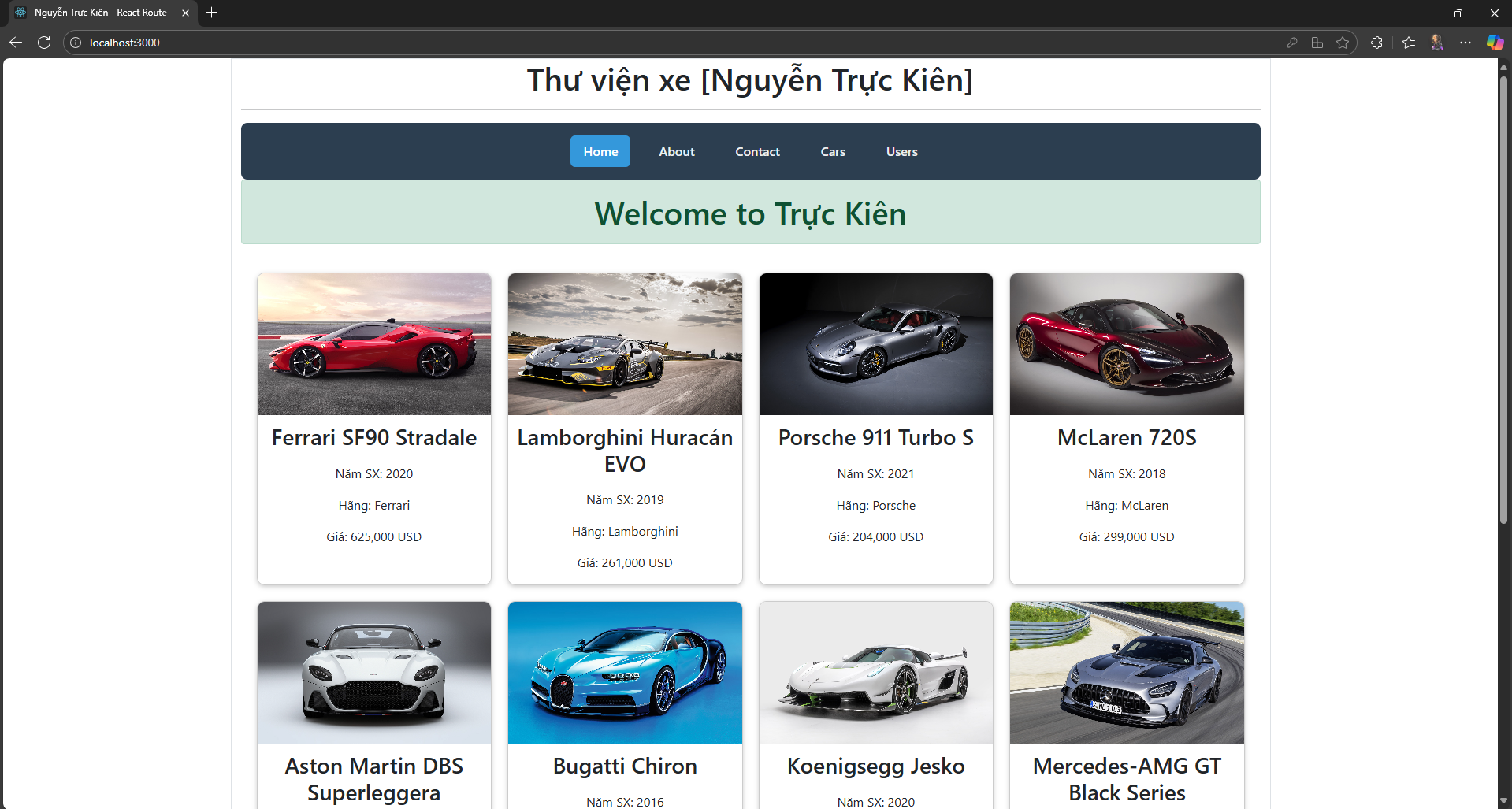
    </nav>

  )

}

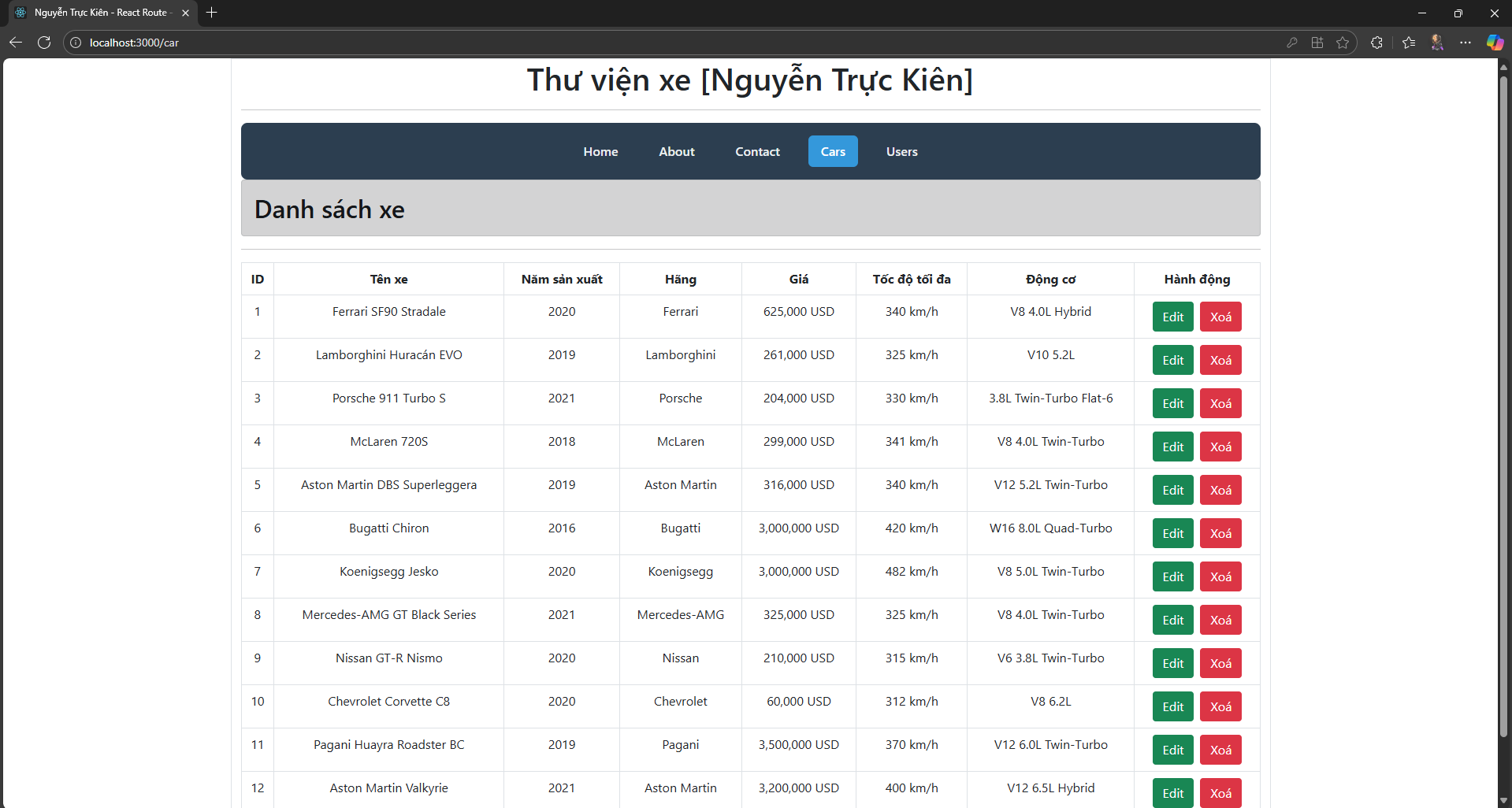
## Một số giao diện người dùng và chức năng cụ thể

## Giao diện trang chủ

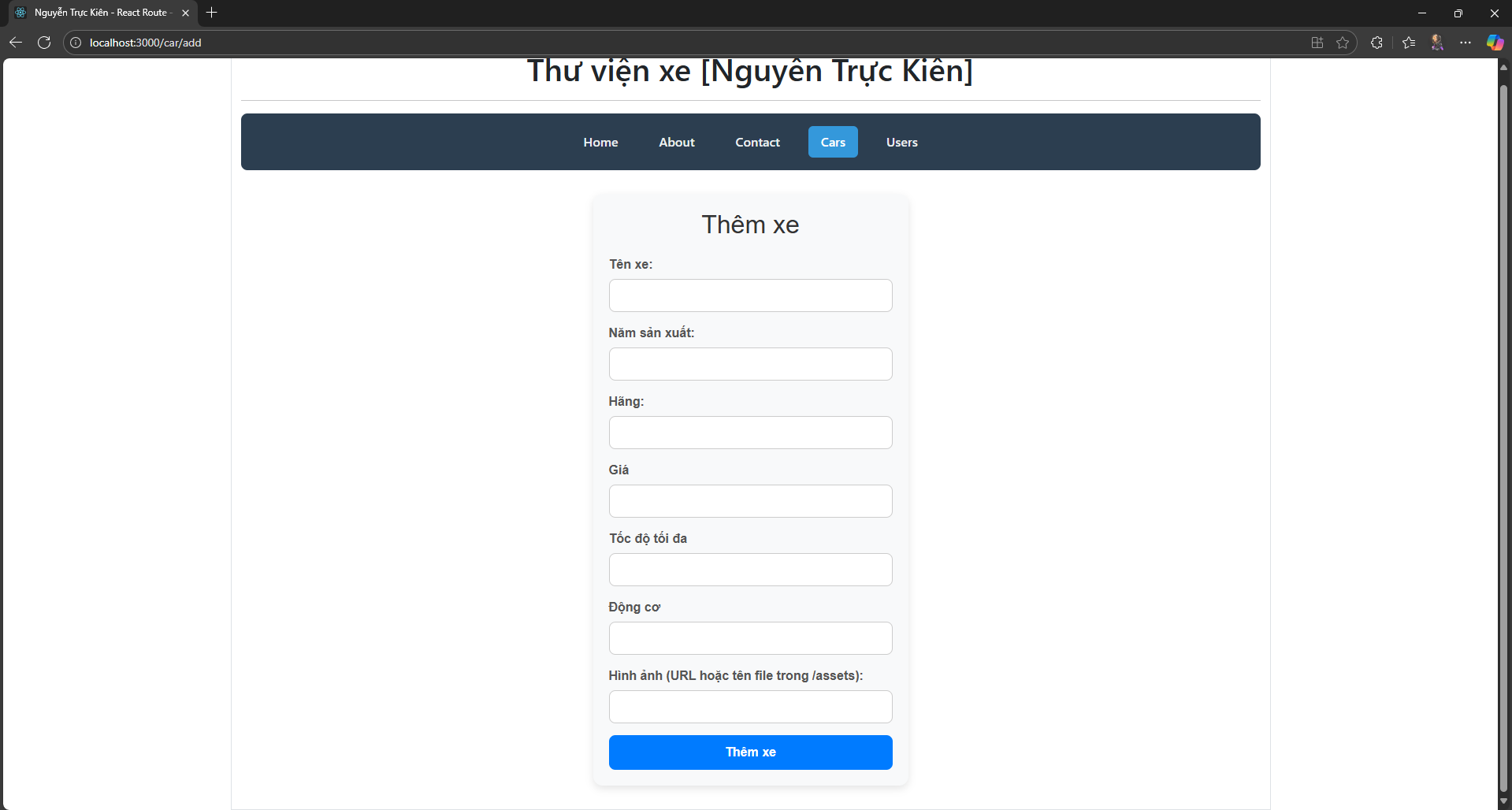


Hiển thị từ API

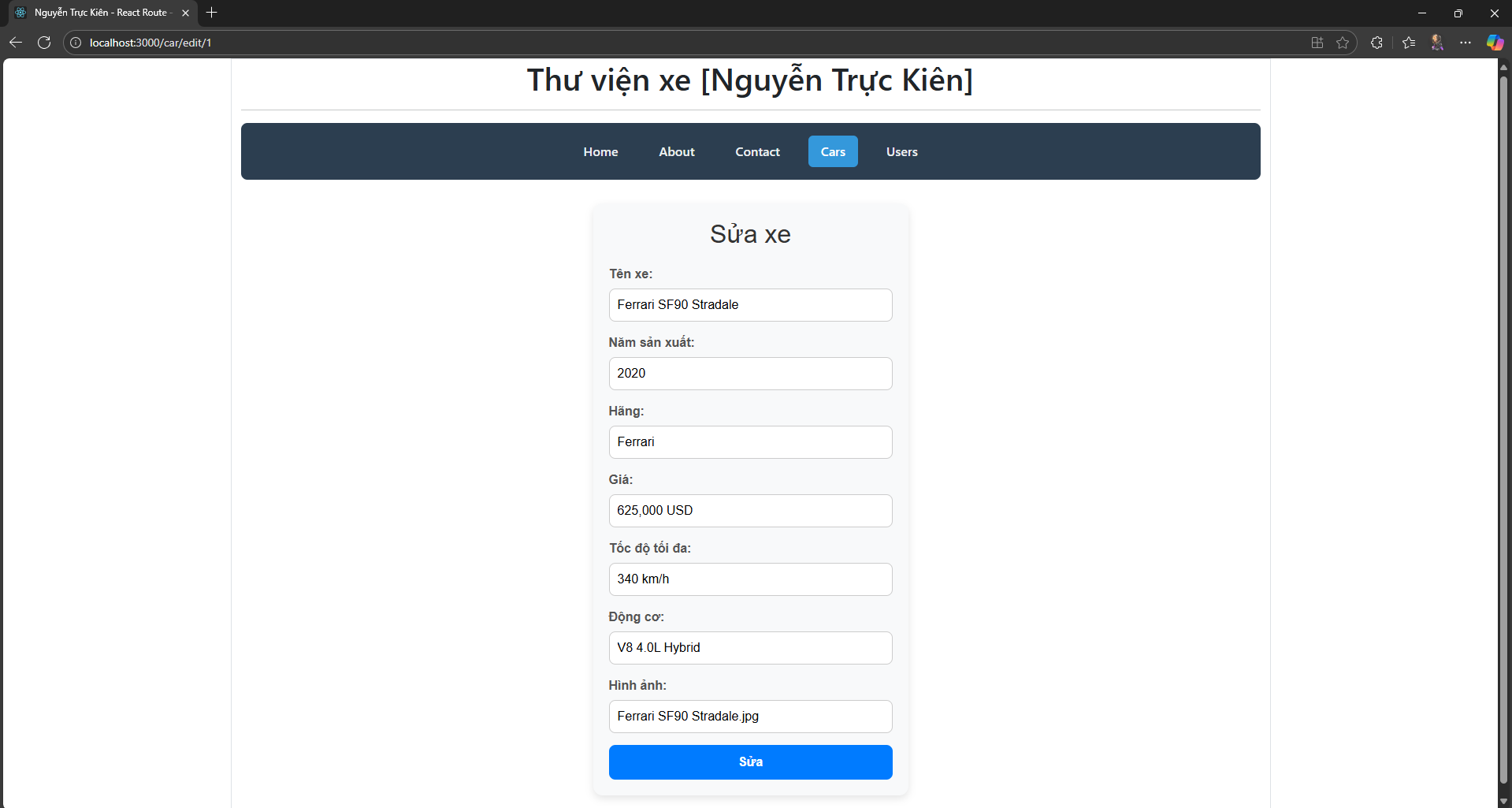
## Giao diện trang quản lý xe



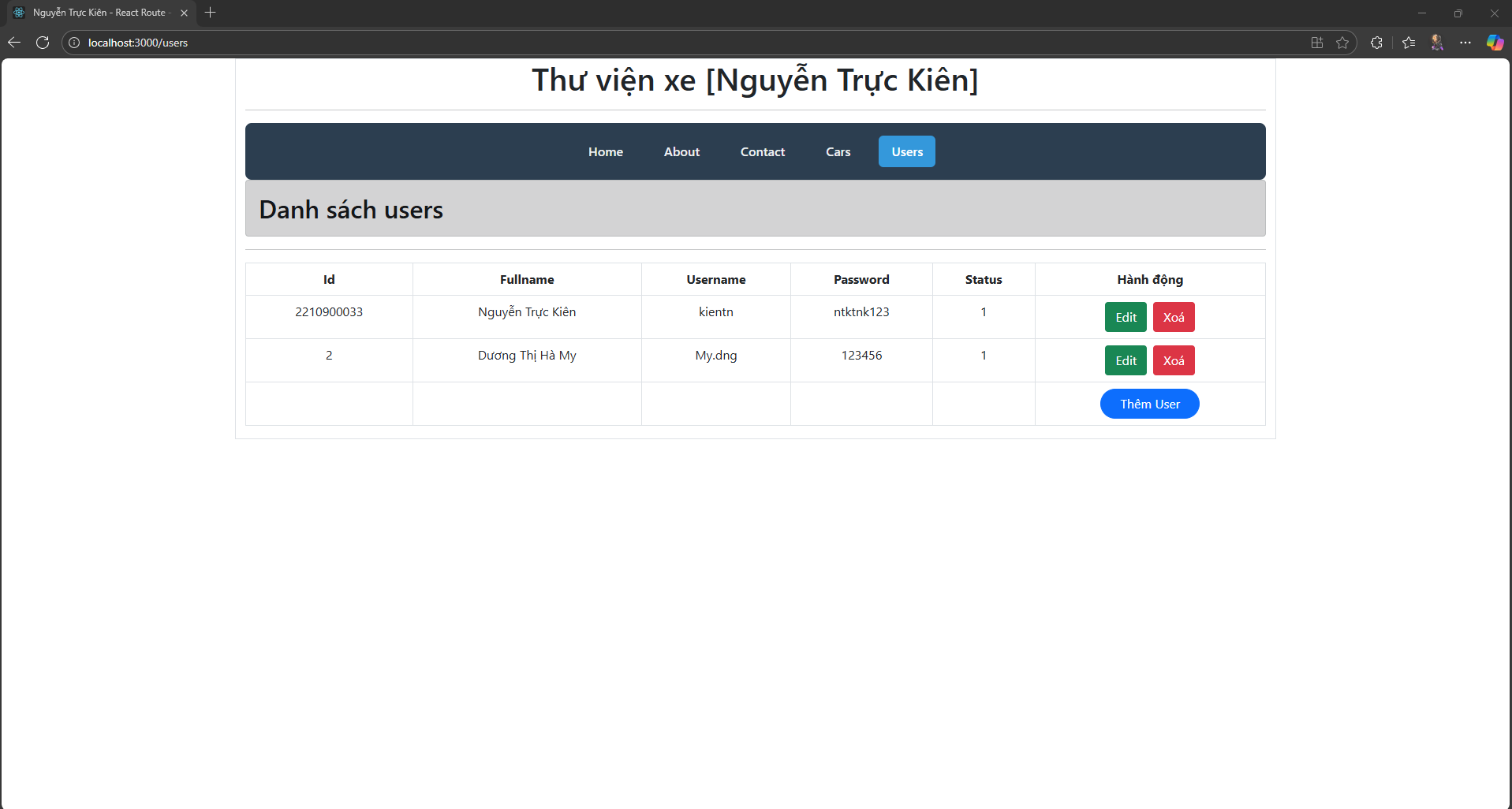
### 2.1. Giao diện Thêm xe



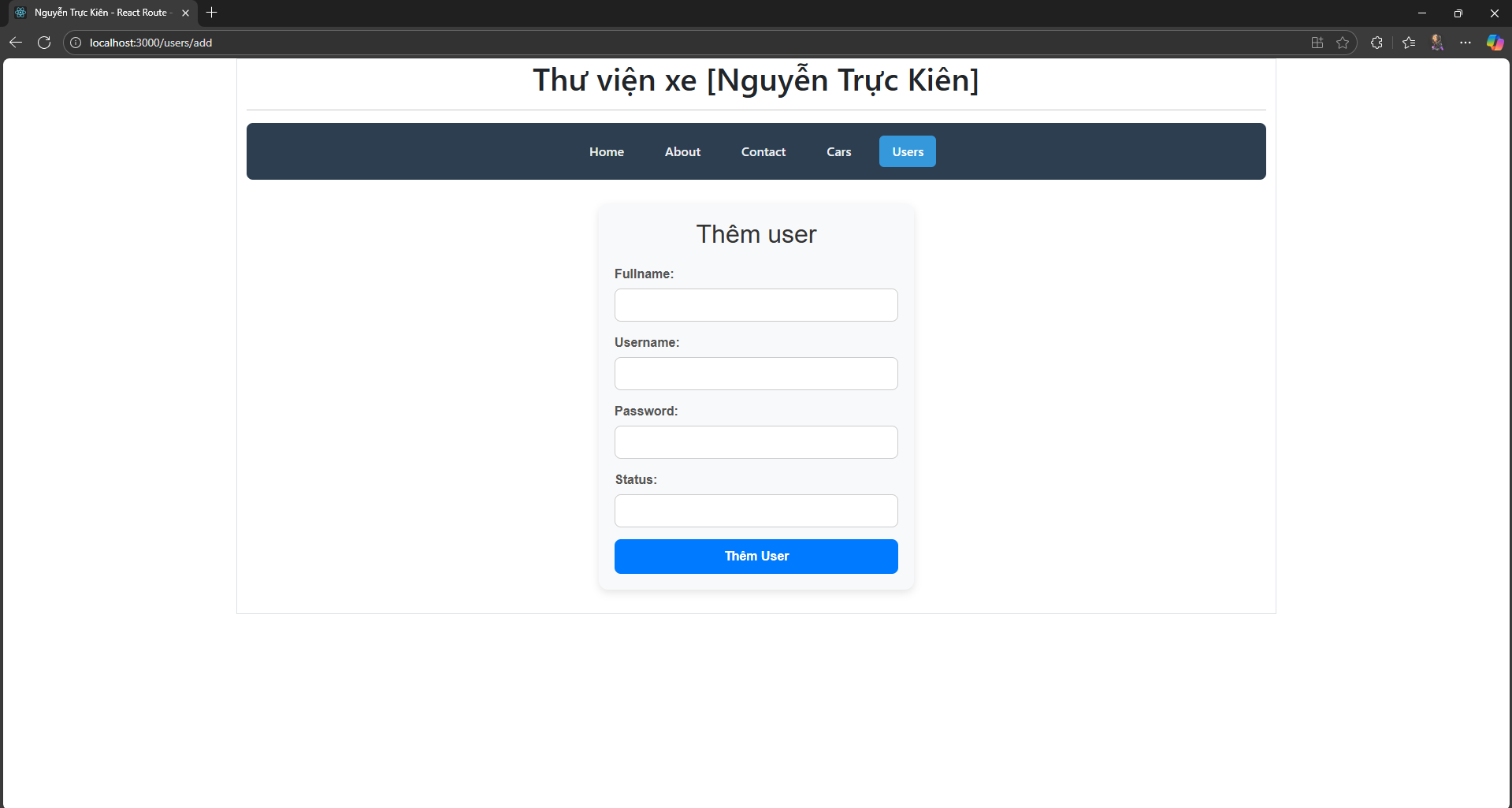
### 2.2. Giao diện Sửa xe



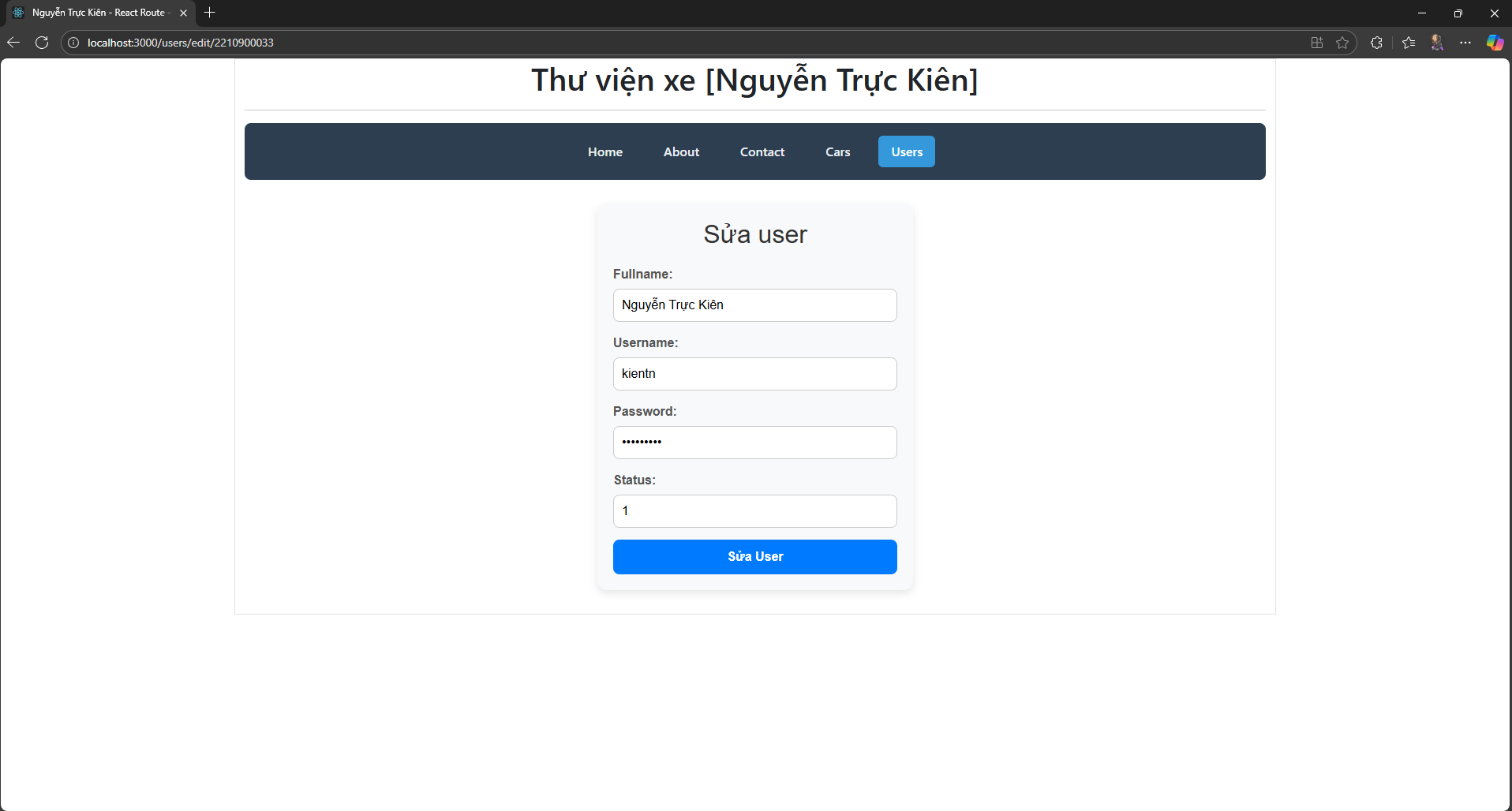
## Giao diện quản lý Users



### 3.1. Giao diện Thêm user



### 3.2. Giao diện Sửa user



# **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## Kết luận

Quá trình làm đồ án chuyên đề với đề tài "……" đã giúp em có cơ hội để trau dồi các kiến thức được học từ thầy cô trên giảng đường, đồng thời cũng tiếp thu và tìm hiểu được thêm nhiều kiến thức, kinh nghiệm mới, cụ thể là:

* + - Nắm được các quy trình nghiệp vụ trong phân tích và thiết kế một sản phẩm phần mềm.
    - Hiểu biết được một số phương pháp phân tích thiết kế hệ thống, từ đó đã áp dụng các hiểu biết của mình để phân tích thiết kế hệ thống bài toán theo mô hình ba lớp.
    - Tiếp thu được các kinh nghiệm thực tế khi thực hành xây dựng một sản phẩm cụ thể, áp dụng được các kiến thức đã học vào thực tiễn.
    - Củng cố các kiến thức về ngôn ngữ lập trình cũng như các thư viện – framework,…

**Ưu điểm:**

Hệ thống cơ bản đã hoàn thiện đầy đủ các tính năng đã được thiết kế và đưa vào sử dụng phục vụ các yêu cầu của người dùng. Quan trọng hơn, hệ thống đảm bảo duy trì được sự ổn định, tính bảo mật và tính tiện dụng cho người sử dụng. Các tính năng đã hoàn thành:

**Nhược điểm:**

Trong những phiên bản tiếp theo, ứng dụng có thể được mở rộng để đáp ứng nhiều hơn những yêu cầu từ người sử dụng như:

* + - Giao diện của hệ thống do bản thân tự thiết kế nên vẫn chưa được hoàn thiện và đẹp mắt nhất.
    - Nhiều chức năng hoạt động chưa được mượt mà tạo hiệu ứng sử dụng không tốt với người dùng.
    - Trong thời gian tới em sẽ tiếp tục hoàn thiện các chức năng hiện thời và bổ sung thêm các chức năng mới nhằm mang lại sự tiện dụng nhất cho người dùng.

## Hướng phát triển:

Trong những phiên bản tiếp theo, ứng dụng có thể được mở rộng để đáp ứng nhiều hơn những yêu cầu từ người sử dụng như:

* + - Cải thiện, bảo trì những lỗi phát sinh ở các tính năng hiện tại.
    - Cải thiện giao diện thân thiện hơn với người dùng.
    - Phát triển thêm tính năng …..

# **DANH MỤC TÀI LIỆU THAM KHẢO**

#### **Tài liệu:**

ReactJS Docs

Bootstrap Docs

#### **Website:**

1. <https://react.dev/learn>
2. <https://getbootstrap.com/docs/5.3/getting-started/introduction/>