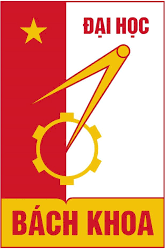
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



**Software design description**

**Bài tập lớn môn học: Project 1- Kỳ 20201**

**Đề tài: Web bán đồng hồ ( Front-end)**

**Giáo viên hướng dẫn: TS. Đào Thành Chung**

**Sinh viên thực hiện: Trần Văn Tuyền**

**MSSV: 20184012**

**Hà Nội, Tháng 1 năm 2020**

# **CHƯƠNG 1: GIỚI THIỆU**

**1.1. Mục đích**

* Tài liệu này là bản thiết kế chi tiết hệ thống, bao gồm các thiết kế: phân tích chi tiết lớp (analyst class), cơ sở dữ liệu (database), giao diện người dùng (graphic userinterface).
* Đưa ra thiết kế cơ sở dữ liệu cụ thể, cơ sở dữ liệu cho phép lưu trữ thông tin về người dùng.
* Từ tài liệu thiết kế này, nhóm phát triển hệ thống có cơ sở để lập trình phát triển, hình thành hệ thống cho người dùng. Thuận tiện cho việc kết nối các module của hệ thống với nhau.

**1.2. Cấu trúc tài liệu**

Bản thiết kế này chia làm 2 phần chính:

* 1. Mô hình hóa cấu trúc hệ thống.
  2. Thiết kế giao diện người dùng

**CHƯƠNG 2: MÔ HÌNH HÓA CẤU TRÚC HỆ THỐNG**

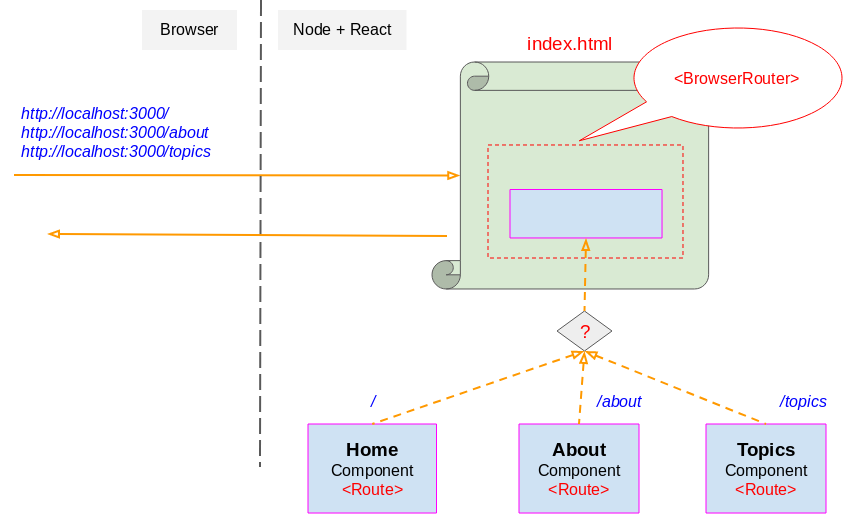
**2.1. React**

React là một thư viện JavaScript được tạo ra bởi Facebook, là một công cụ tuyệt vời để xây dựng giao diện người dùng. React được sử dụng để xây dựng các ứng dụng trang đơn. React cho phép chúng tôi tạo các thành phần UI có thể tái sử dụng.

Một chương trình được phân chia thành các component, các component được đóng gói quản lý trạng thái của riêng chúng, sau đó soạn chúng để tạo giao diện người dùng phức tạp.

Các component giúp chia nhỏ chương trình, mỗi component đảm nhận một chức năng nào đó qua đó giúp ta dễ dàng sửa chữa thay đổi khi cần thiết,tái sử dụng một cách dễ dàng.

**2.2. React Router: định tuyến trong React**

****

React Router là một thư viện định tuyến (routing) tiêu chuẩn trong React. Nó giữ cho giao diện của ứng dụng đồng bộ với URL trên trình duyệt. React Router cho phép bạn định tuyến "luồng dữ liệu"(data flow) trong ứng dụng của bạn một cách rõ ràng. Nó tương đương với sự khẳng định, nếu bạn có URL này, nó sẽ tương đương với Route (tuyến đường) này, và giao diện sẽ thế này!

React Router giúp bạn định nghĩa ra các URL động, và lựa chọn Component phù hợp để hiển thị trên trình duyệt người dùng ứng với từng URL.

React Router cung cấp cho bạn 2 thành phần là <BrowserRouter> & <HashRouter>. Hai thành phần này khác nhau ở kiểu URL mà chúng sẽ tạo ra và đồng bộ.

<BrowserRouter> được sử dụng phổ biến hơn, nó sử dụng History API có trong HTML5 để theo dõi lịch sử bộ định tuyến của bạn. Trong khi đó <HashRouter> sử dụng hash của URL (window.location.hash) để ghi nhớ mọi thứ. Nếu bạn có ý định hỗ trợ các trình duyệt cũ, bạn nên gắn bó với <HashRouter>, hoặc bạn muốn tạo một ứng dụng React sử dụng Router ở phía client thì <HashRouter> là lựa chọn hợp lý.

Thành phần <Route> định nghĩa một ánh xạ (mapping) giữa một URL và một Component. Điều đó có nghĩa là khi người dùng truy cập theo một URL trên trình duyệt, một Component tương ứng sẽ được render trên giao diện.

<*DataProvider*>

            <div *className*="App">

              <*BrowserRouter* >

                <*Navbar1*/>

                <*Search*/>

                <*Switch*>

                    <*Route* *path*='/' *exact* *component*={Home}></*Route*>

                    <*Route* *path*='/search/:find' *component*={SearchProducts}></*Route*>

                    <*Route* *path*='/manWatch' *component*={ManWatch}></*Route*>

                    <*Route* *path*='/womenWatch' *component*={WomenWatch}></*Route*>

                    <*Route* *path*="/login" *component*={Login}></*Route*>

                    <*Route* *path*="/register" *component*={Register}></*Route*>

                    <*Route* *path*="/product/:id" *component*={Detail}></*Route*>

                    <*Route* *path*="/cart" *component*={Cart}></*Route*>

                    <*Route* *path*="/notfound" *component*={NotFound}></*Route*>

                    <*Route* *path*="/order" *component*={Order}></*Route*>

                    <*Route* *path*="/contact" *component*={Contact}></*Route*>

                </*Switch*>

                <*Footer*></*Footer*>

              </*BrowserRouter*>

            </div>

        </*DataProvider*>

**2.3. Context**

Context cung cấp một cách để truyền dữ liệu qua cây thành phần mà không phải chuyển props xuống theo cách thủ công ở mọi cấp độ.

import React, { createContext, useEffect, useState } from 'react'

export const DataContext=createContext();

return (

     <*DataContext.Provider* *value*={value}>

        {*props*.children}

     </*DataContext.Provider*>

)

Tại các component sử dụng context

import React,{useContext} from 'react'

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

import { DataContext} from './DataProvider'

onst value =useContext(DataContext);

**2.4. React hooks**

React Hooks chính thức được giới thiệu trong phiên bản React 16.8. Nó cho phép chúng ta sử dụng state và các tính năng khác của React mà không cần dùng tới class.

API Hooks mới cho phép một số tính năng tuyệt vời được truy cập trên các ứng dụng React, có thể loại bỏ nhu cầu về thư viện quản lý trạng thái, như Redux hoặc Flux.

Hooks cho phép bạn truy cập các phương thức trạng thái và vòng đời, cùng với nhiều thủ thuật tuyệt vời khác để sử dụng trong các functional components, chúng ta hoàn toàn có thể sử dụng state bằng useState và useEffect trong hook.

useState: Cho phép truy cập và kiểm soát trạng thái bên trong các thành phần chức năng. Trước đây, chỉ các thành phần dựa trên lớp mới có quyền truy cập vào trạng thái.

    const [search,setSearch]=useState('');

    const [autoCombobox,setAutoComboBox]=useState([]);

useEffect: Cho phép truy cập trừu tượng vào các phương thức vòng đời của React trong các thành phần chức năng. Bạn sẽ không gọi tên các phương thức vòng đời riêng lẻ, nhưng bạn có được quyền kiểm soát tương tự với nhiều chức năng hơn và mã sạch hơn.

    useEffect(() =>{

        const getTotal = () => {

            const res = cart.reduce((*prev*, *item*) => {

                return *prev* + (*item*.price \* *item*.count\*(1-*item*.discount))

            },0)

            setTotal(res)

        }

        getTotal()

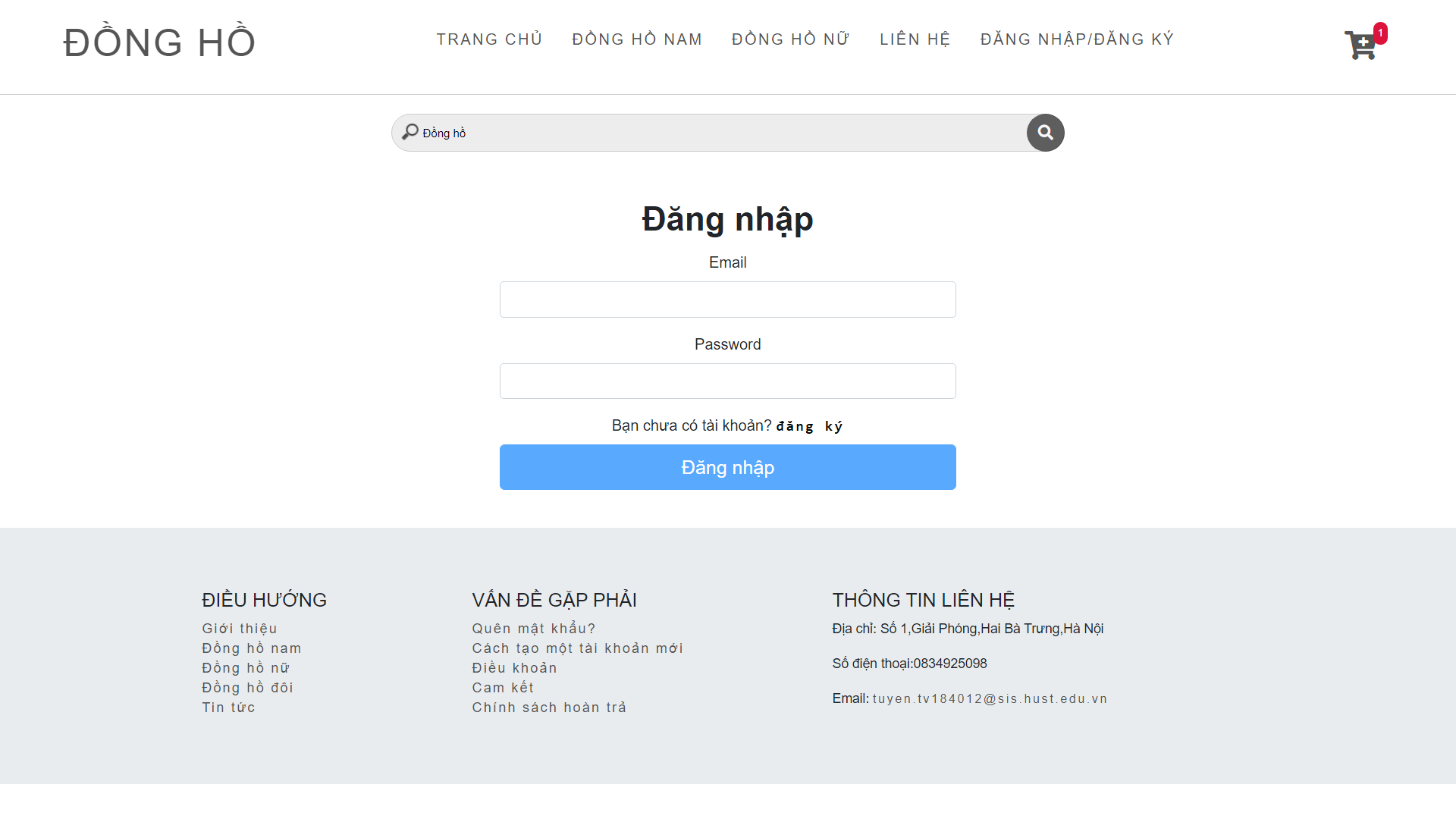
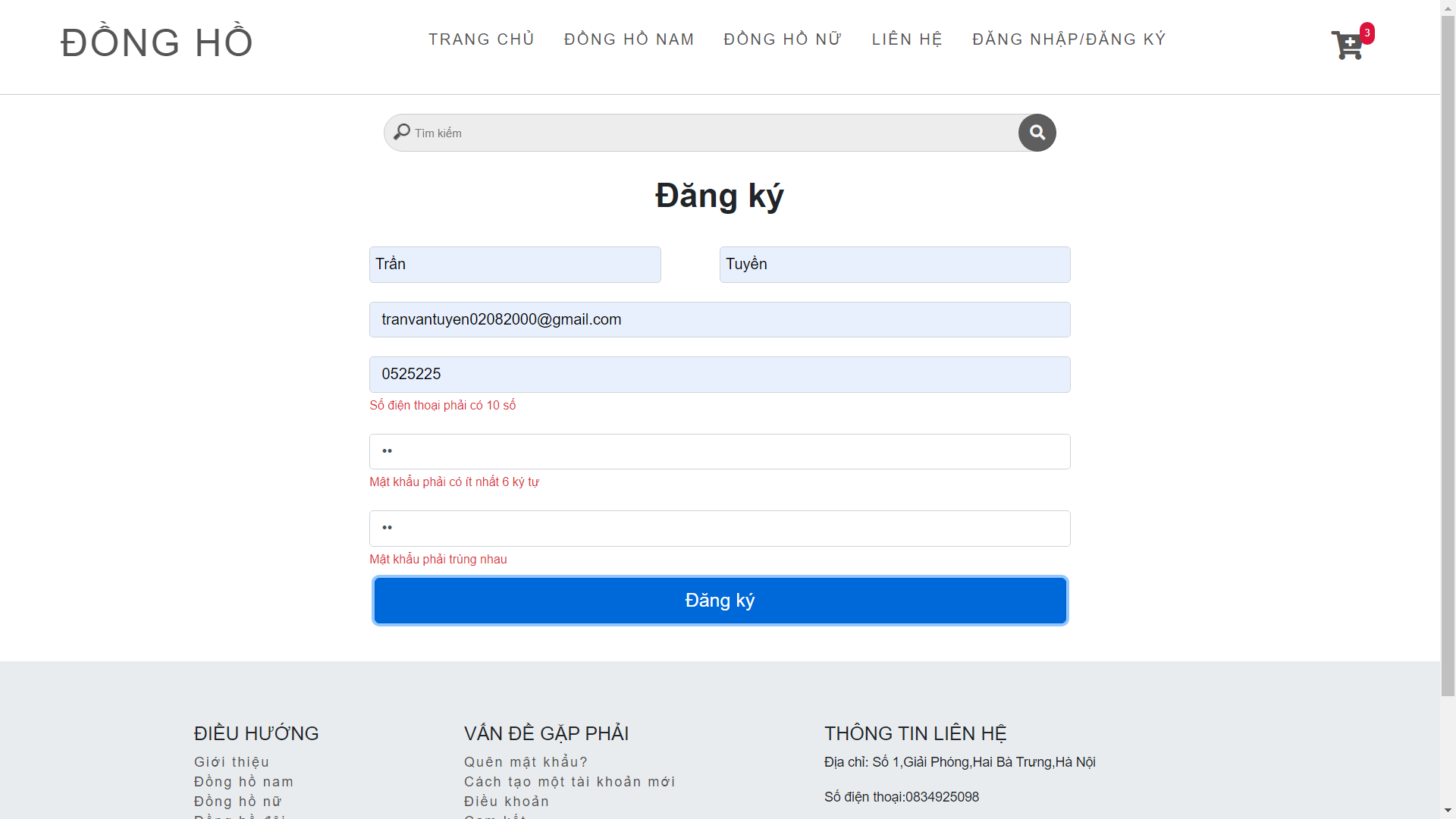
    },[cart])

**2.5. Dữ liệu**

Thông tin sản phẩm được lưu trữ tạm thời trong một mảng các đối tượng sản phẩm và đặt trong component cung cấp context “DataProvider”.

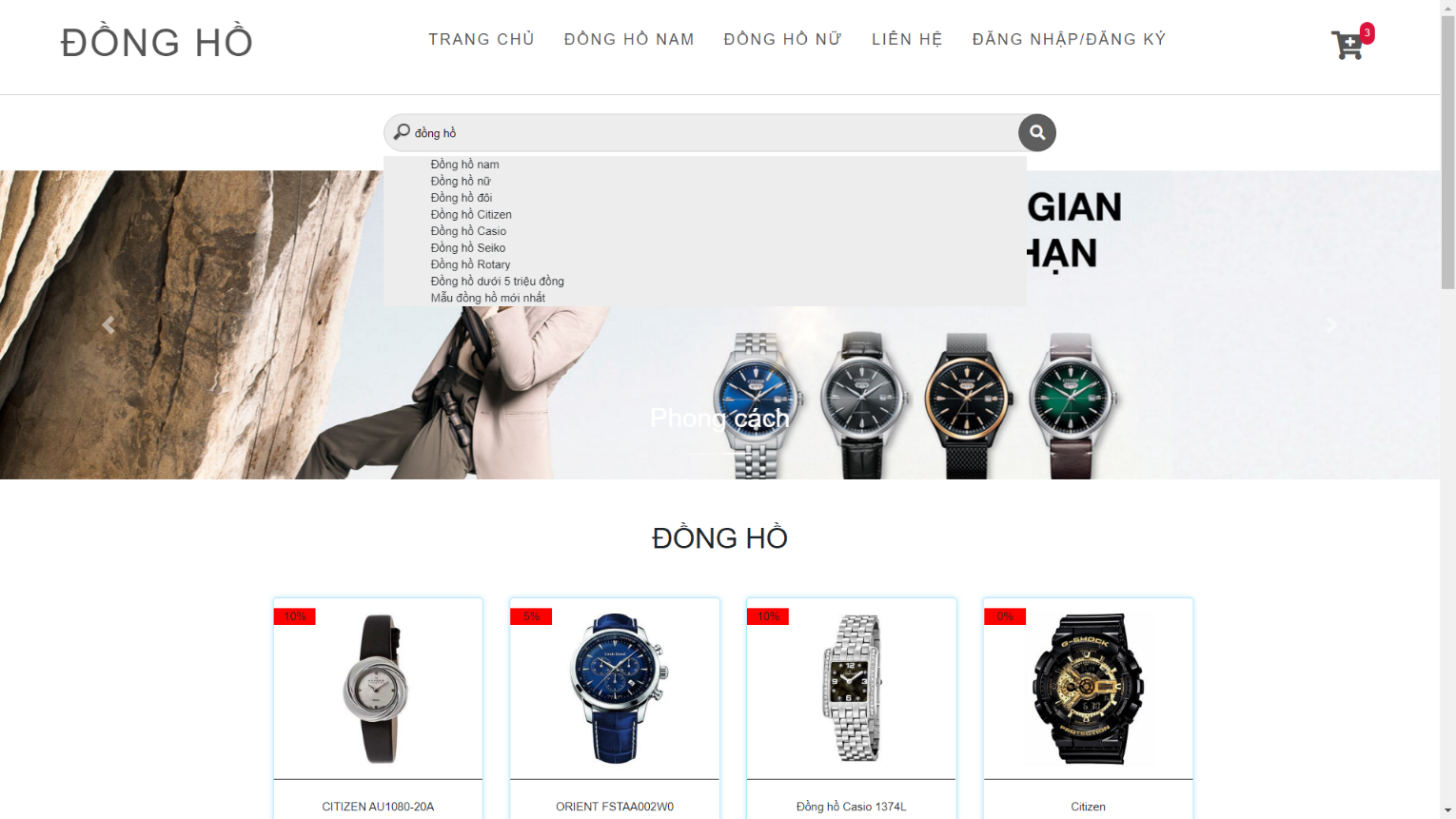
Các sản phẩm trong giỏ hàng được lưu trữ trên localStorage của trình duyệt để đảm bảo khi load lại trang sản phẩm trong giỏ hàng vẫn còn tồn tại.

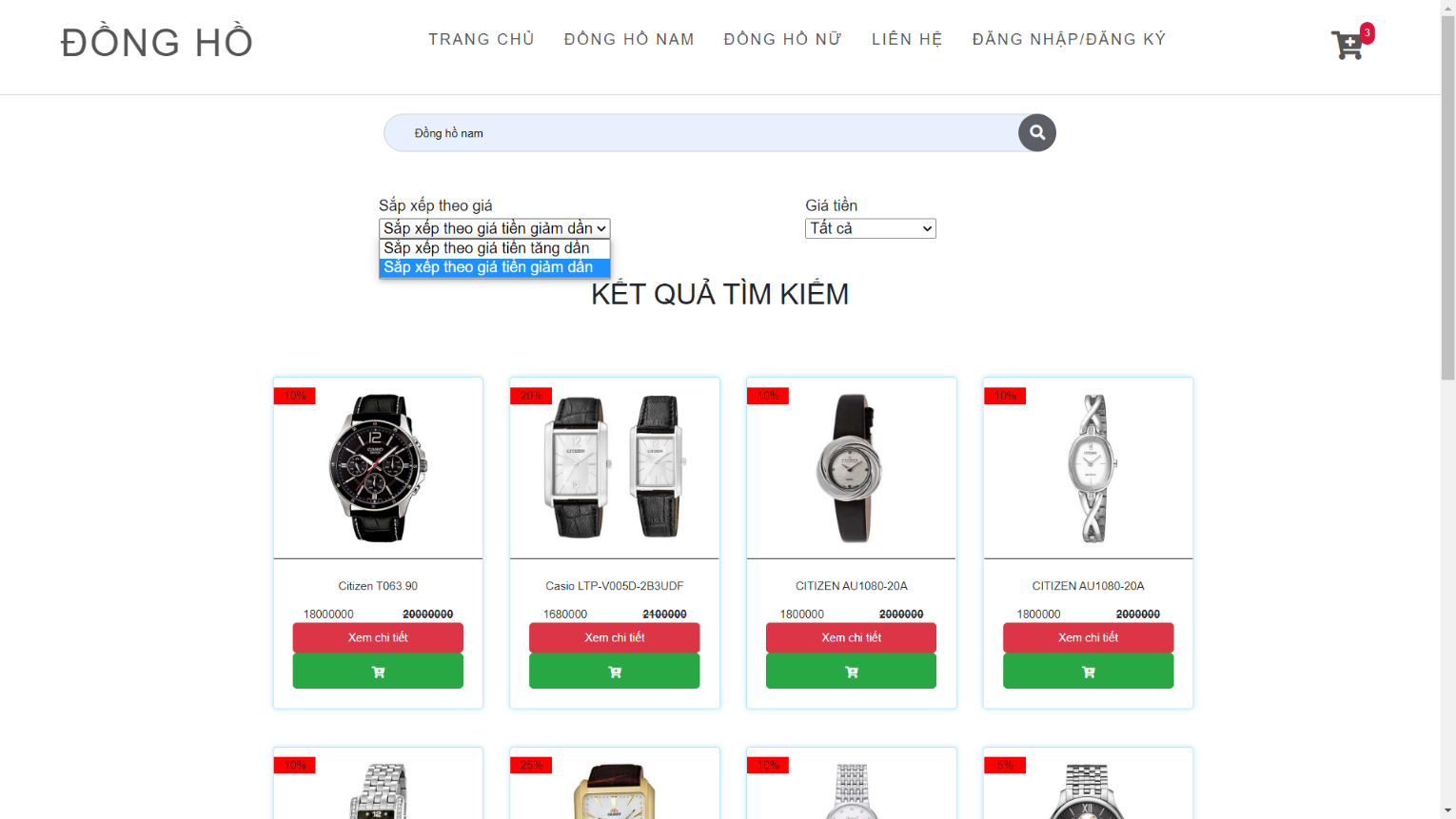
**CHƯƠNG 3: THIẾT KẾ GIAO DIỆN**

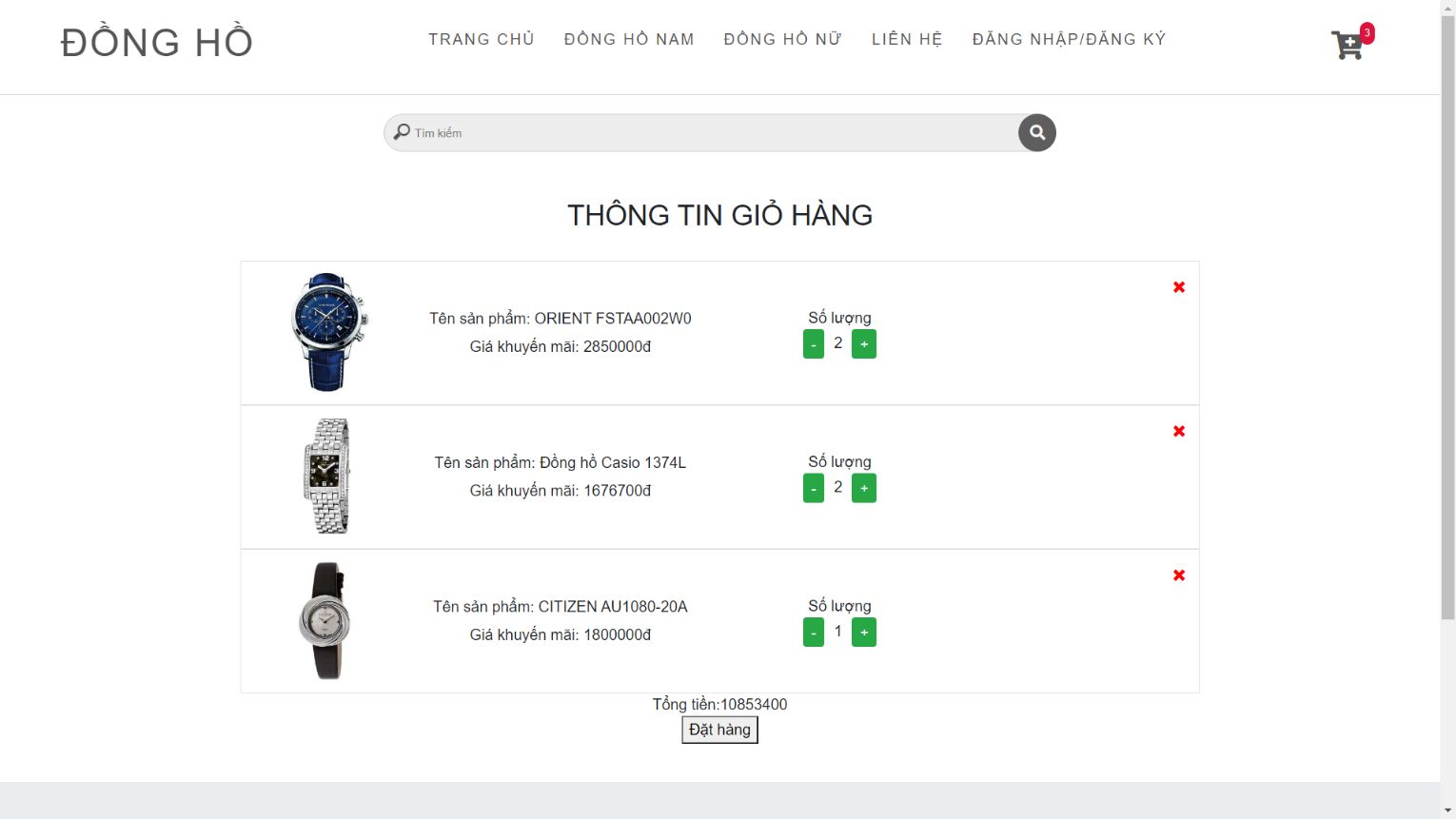
**3.1. Giao diện đăng nhập3.2. Giao diện đăng ký.**

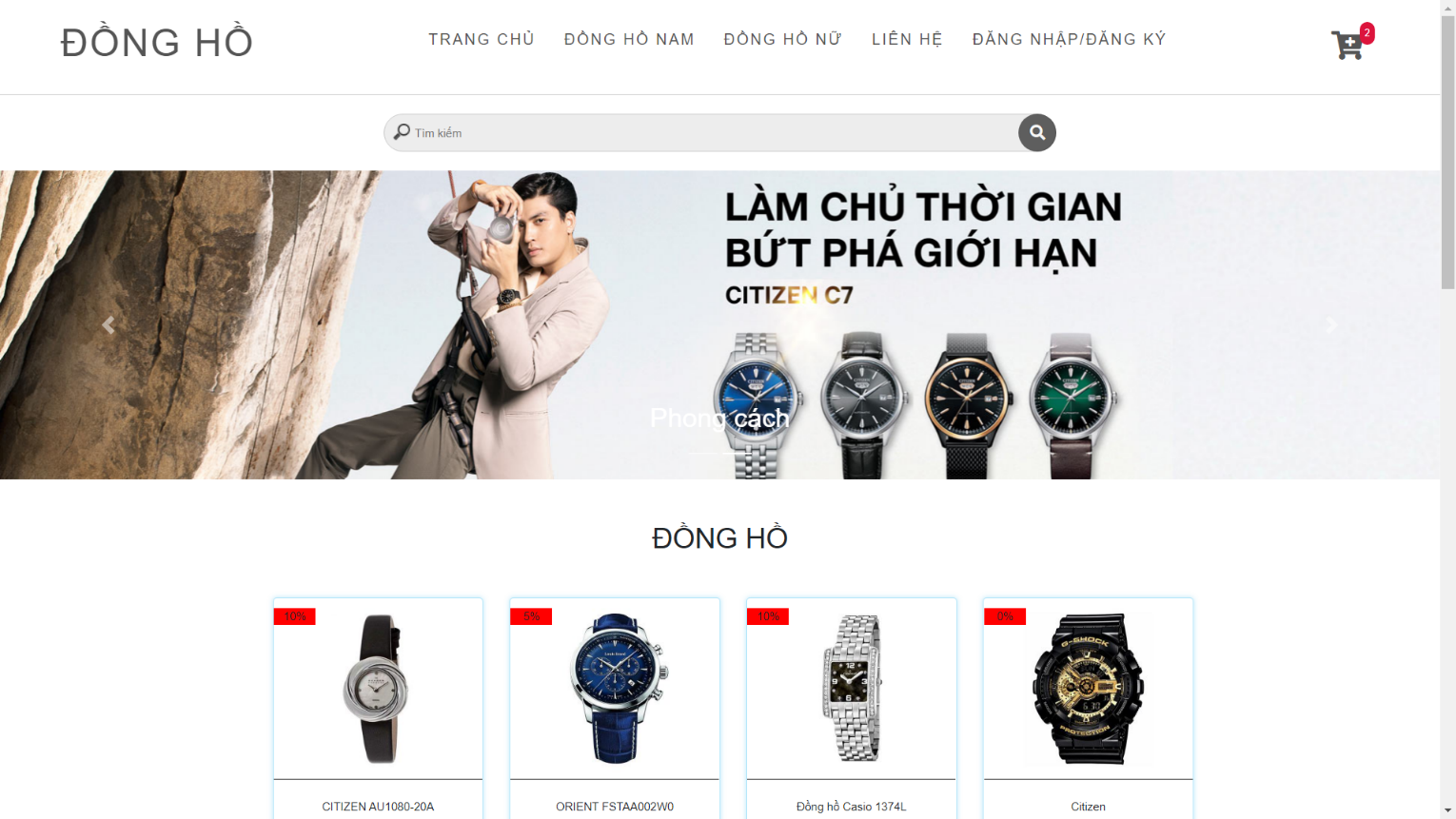
**3.3. Giao diện xem chi tiết sản phẩmẢnh có chứa văn bản

Mô tả được tạo tự động**

**3.4. Giao diện tìm kiếm sản phẩm.**

**3.5.Giao diện sắp xếp sản phẩm**

**3.6. Giao diện giỏ hàng**

**3.7. Giao diện chính.**