**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO CHUYÊN ĐỀ**

**HỌC PHẦN NGÔN NGỮ KỊCH BẢN**

**ĐỀ TÀI: XÂY DỰNG WEBSITE BÁN HÀNG ĐỒ PHƯỢT.**

|  |  |
| --- | --- |
| **Sinh viên thực hiện** | **BÙI MINH ANH** |
|  | **NGUYỄN PHƯƠNG NAM** |
| **Giảng viên hướng dẫn** | **PHẠM HỒNG MINH** |
| **Ngành** | **CÔNG NGHỆ THÔNG TIN** |
| **Chuyên ngành** | **KỸ THUẬT PHẦN MỀM** |
| **Lớp** | **K75CNPM19** |
| **Khóa** | **2018 – 2023** |

***Hà Nội, tháng 11, năm 2021***

**PHIẾU CHẤM ĐIỂM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Chữ kí** | **Điểm** | **Ghi chú** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Họ và tên giảng viên** | **Chữ kí** | **Ghi chú** |
| Giảng viên 1: |  |  |
| Giảng viên 2: |  |  |

**MỤC LỤC**

[CHƯƠNG 1. KHẢO SÁT HỆ THỐNG 6](#_Toc86603315)

[1.1. Đặt vấn đề 6](#_Toc86603316)

[1.3. Mục tiêu đề tài 7](#_Toc86603317)

[2.1. Tổng quan về *Node.js* 8](#_Toc86603318)

[2.1.1. Giới thiệu về *Node.js* 8](#_Toc86603319)

[2.1.2. Giới thiệu về *MongoDB* 14](#_Toc86603320)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG (UML) 18](#_Toc86603321)

[A. Mô hình nghiệp vụ 18](#_Toc86603322)

[B. Sơ đồ phân rã chức năng Website bán đồ Phượt 18](#_Toc86603323)

[C. DANH MỤC CÁCH KÝ HIỆU 18](#_Toc86603324)

[1. Biểu đồ Use Case 22](#_Toc86603325)

[1.1.Use Case Tổng quát 22](#_Toc86603326)

[1.2. Use Case Đăng nhập 23](#_Toc86603327)

[1.3. Use Case quản lý khách hàng 24](#_Toc86603328)

[1.4. Use Case Quản lý sản phẩm 25](#_Toc86603329)

[1.5. Use Case Mua hàng 26](#_Toc86603330)

[1.6. Biểu đồ hoạt động 27](#_Toc86603331)

[2. Biểu đồ tuần tự 35](#_Toc86603332)

[2.1.Biểu đồ tuần tự của đăng ký 36](#_Toc86603333)

[2.2 biểu đồ tuần tự đăng nhập 37](#_Toc86603334)

[2.3.Biểu đồ tuần tự của đăng xuất 38](#_Toc86603335)

[2.4. Biểu đồ tuần tự chức năng Mua hàng 39](#_Toc86603336)

[2.5. Biểu đồ tuần tự chức năng quản lý khách hàng 40](#_Toc86603337)

[3.Biểu đồ lớp (Class) 40](#_Toc86603338)

[3.1. Biểu đồ lớp quản lý nhân viên 40](#_Toc86603339)

[3.2. Biểu đồ lớp quản lý sản phẩm 41](#_Toc86603340)

[3.3. Biểu đồ lớp cho use case bán hàng 42](#_Toc86603341)

[CHƯƠNG 3: CÀI ĐẶT 43](#_Toc86603342)

[3.1 Giao diện người dùng 43](#_Toc86603343)

[3.2. Giao diện quản lý admin 46](#_Toc86603344)

[TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN 49](#_Toc86603345)

[1. Tổng kết 49](#_Toc86603346)

[2. Kết quả đạt được 49](#_Toc86603347)

[3. Hạn chế 49](#_Toc86603348)

[4. Hướng phát triển 49](#_Toc86603349)

[TÀI LIỆU THAM KHẢO 50](#_Toc86603350)

**LỜI NÓI ĐẦU**

Ngày nay *Công Nghệ Thông Tin* là một ngành rất phổ biến trong xã hội. Nó được ứng dụng rộng rãi trong rất nhiều ngành nghề, các lĩnh vực khác nhau và đạt được hiệu quả cao trong công việc. Đặc biệt là trong công tác quản lý, tin học giúp giảm tải sức người, tiết kiệm chi phí lẫn thời gian, gọn nhẹ hơn rất nhiều so với phương thức quản lý bằng giấy tờ như trước kia. Ứng dụng tin học vào công tác quản lý còn giúp thu hẹp không gian lưu trữ dữ liệu, tránh thất lạc dữ liệu, giảm thiểu các sai số, thiếu sót. Hơn thế nó còn giúp việc tra cứu, tìm kiếm thông tin một cách nhanh chóng, chính xác. Để giúp công việc quản lý đạt được những yêu cầu trên, *Công Nghệ Thông Tin* đã được đưa vào sử dụng và dần cho thấy sự hữu ích của nó. Với những thành tựu trong ngành, ta không còn phải đối mặt với những tập hồ sơ dày cộp, những ngăn tủ đựng hồ sơ chiếm nhiều diện tích mà ta có thể phải mất nhiều thời gian để tìm kiếm thông tin, dữ liệu cần thiết. Giờ đây tất cả các bất tiện đó đều được giải quyết bằng phần mềm quản lý – một sản phẩm của ngành *Công Nghệ Thông Tin*.

Chính vì sự hữu ích của ngành *Công Nghệ Thông Tin* mang lại, cùng với mong muốn xây dựng một website quản lý tiện lợi và giúp người dùng dễ dàng tiếp cận. Chúng em đã thực hiện đề tài ***“Xây dựng website bán hàng đồ Phượt”***. Chúng em tin rằng đây sẽ là một đề tài rất hay và có thể phát triển lớn hơn trong tương lai.

Là những sinh viên mới tiếp cận các phương pháp phân tích thiết kế hệ thống nên trong quá trình thực hiện không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý từ thầy cô để đề tài có thể đạt kết quả tốt hơn, sát với nhu cầu thực tiễn và có thể áp dụng trong tương lai.

Chúng em xin chân thành cảm ơn!

# CHƯƠNG 1. KHẢO SÁT HỆ THỐNG

## 1.1. Đặt vấn đề

Cùng với sự phát triển không ngừng về *kỹ thuật máy tính* và *mạng Internet, Công Nghệ Thông Tin* đang ngày càng chinh phục các đỉnh cao. *Mạng Internet* là một trong những sản phẩm có giá trị hết sức lớn lao và ngày càng trở nên là một công cụ không thể thiếu trong mọi lĩnh vực cuộc sống.

Với *Internet*, chúng ta đã thực hiện được nhiều công việc với tốc độ nhanh hơn và chi phí thấp hơn nhiều so với các thức truyền thống. Chính điều này, đã thúc đẩy sự khai sinh và phát triển của các hình thức mua sắm, thanh toán trực tuyến trên khắp thế giới, làm thay đổi bộ mặt văn hóa, nâng cao chất lượng cuộc sống của con người.

Nhu cầu mua sắm từ lâu đã luôn là một trong những nhu cầu thứ yếu trong cuộc sống của mỗi con người, nhất là trong *thời đại 4.0* với sự bùng nổ của thanh toán trực tuyến. Mặt hàng may mặc ngày nay rất phổ biến và mọi người có thể mua sắm cho mình qua thanh toán trực tuyến cũng hết sức dễ dàng tuy nhiên quá trình mua sắm ở các website hiện nay vẫn chưa được tối ưu hóa và còn tương đối khó khăn trong việc quản lý, bảo trì và retarget lại những khách hàng cũ. Xu hướng du lịch bằng xe máy ngày càng nở rộ, cộng đồng hay gọi nhau bằng cái tên phượt thủ cũng ngày càng đông lên, tạo cơ hội cho các cửa hàng bán [đồ du lịch bụi](https://www.kiotviet.vn/) phát triển. Không như một số sản phẩm thời vụ chỉ bán đắt vào đúng mùa du lịch giữa năm, thời trang bụi cho dân phượt lại có thể bội thu nhiều thời điểm trong năm.

Những người ưa du lịch bụi đa phần khá trẻ, sức khỏe tốt, đam mê khám phá, … Độ tuổi có thể tính từ sinh viên cho tới người đã đi làm, đã lập gia đình. Cũng bởi đối tượng đa dạng ngành nghề, nên mức chi tiêu của họ cũng khác nhau. Người sẵn sàng chi hàng triệu bạc cho một đôi giày leo núi, nhưng cũng có người chỉ có thể mua hơn trăm ngàn cho đôi giày thể thao.

Nhận thức được thực trạng đó chúng em mong muốn xây dựng một webste bán đồ Phượt, giờ đây khách hàng vẫn có thể dễ dàng mua sắm trực tuyến tại gia lựa chọn dịch vụ đi kèm, còn đối với các đại lý cửa hàng giờ đây đã có thể quản lý các mặt hàng sản phẩm, khách hàng nhân viên của mình một cách dễ dàng hơn, tất cả đã được gói gọn vào website và dịch vụ đi kèm.

Với đề tài này chúng em xin được trình bày một cách thức quản lý website cho phép người quản trị dễ dàng trong việc quản lý của hàng và các sản phẩm hiệu quả tới khách hàng và các đơn vị hợp tác.

**1.2. Mục đích nghiên cứu**

* Tìm hiểu về hệ thống phần mềm *Node.js.*
* Tìm hiểu các công cụ hỗ trợ như: *HTML, CSS, JS, Web Service.*
* Tìm hiểu về *MongoDB* và tương tác kiến thức *csdl.*
* Cách điều khiển dữ liệu *(Controller).*

## 1.3. Mục tiêu đề tài

* *Khách hàng (User):*
* Tìm kiếm và lựa chọn sản phẩm cần mua.
* Xem chi tiết sản phẩm.
* Thực hiện việc đặt và mua hàng trực tuyến.
* Xem thông tin giới thiệu về trang web.
* Xem thông tin liên hệ người bán.
* *Người quản trị* *(Admin):*
* Quản lý trang web thông qua tài khoản *admin.*
* Quản lý tất cả các sản phẩm.
* Quản lý các danh mục sản phẩm.
* Thêm sửa xóa các sản phẩm của cửa hàng

## 2.1. Tổng quan về *Node.js*

### 2.1.1. Giới thiệu về *Node.js*

***Node.js là gì?***

*Node.js* là một hệ thống phần mềm được thiết kế để viết các ứng dụng *internet* có khả năng mở rộng, đặc biệt là *máy chủ web.* Chương trình được viết bằng *JavaScript,* sử dụng kỹ thuật điều khiển theo sự kiện, nhập/xuất không đồng bộ để tối thiểu tổng chi phí và tối đa khả năng mở rộng. *Node.js* bao gồm có *V8 JavaScript engine* của Google, *libUV,* và vài thư viện khác.

*Node.js* được tạo bởi Ryan Dahl từ năm 2009, và phát triển dưới sự bảo trợ của Joyent.

Mục tiêu ban đầu của Dahl là làm cho trang web có *khả năng push* như trong một số ứng dụng web như *Gmail.* Sau khi thử với vài ngôn ngữ Dahl chọn *JavaScript* vì một *API Nhập/Xuất không đầy đủ.* Điều này cho phép anh có thể định nghĩa một quy ước *Nhập/Xuất* điều khiển theo sự kiện *non-blocking.*

Vài môi trường tương tự được viết trong các ngôn ngữ khác bao gồm *Twisted* cho *Python, Perl Object Environment* cho *Perl, libevent* cho *C* và *EventMachine* cho *Ruby.* Khác với hầu hết các chương trình *JavaScript, Node.js* không chạy trên một trình duyệt mà chạy trên *Server. Node.js* sử dụng nhiều chi tiết kỹ thuật của *CommonJS.* Nó cung cấp một môi trường *REPL* cho kiểm thử tương tác.

*Node.js* được *InfoWorld* bình chọn là *“Công Nghệ của năm”* vào 2012.

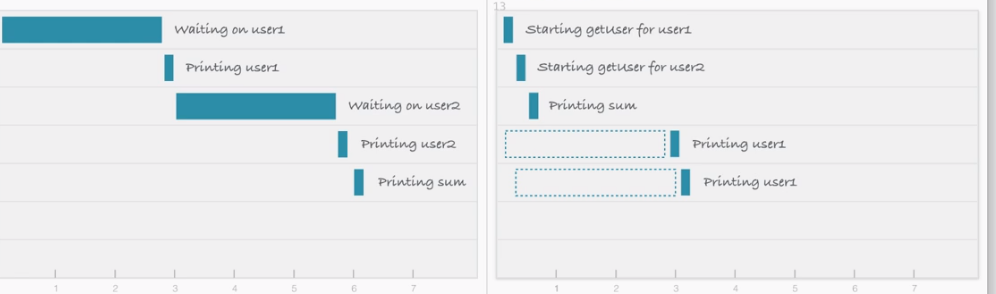
***Tại sao lại là Node.js?***

*Chúng ta sẽ đi tìm hiểu tại sao Node.js lại phổ biến đến vậy nhé!*

*I/O* là *Input/Output.* Nó có thể là bất cứ thứ gì từ đọc/viết các *file* nội bộ cho đến tạo *HTTP* đến *API.*

*I/O* tốn thời gian và do đó nó sẽ chặn các chức năng khác.

Xét một tình huống là chúng ta cần dữ liệu *backend* chi tiết cho *user1* và *user2* sau đó hiển thị chúng lên màn hình hoặc bảng điều khiển. Để phản hồi cho lệnh này sẽ mất thời gian, nhưng cả 2 dữ liệu *user* có thể thực hiện độc lập và cùng lúc.



*Blocking I/O (trái) và Non-Blocking I/O (phải).*

***Blocking I/O***

Trong phương thức *blocking,* dữ liệu của *user2* sẽ không bắt đầu cho đến khi dữ liệu của *user1* xuất trên màn hình.

Nếu đây là một *web server,* chúng ta phải bắt đầu một *thread* mới cho từng *user* mới. Nhưng *JavaScript* là đơn luồng *(tính chất “xong cái này mới đến cái khác”).* Và điều này làm *JavaScript* không phù hợp cho các tác vụ đa luồng.

Đó là lý do *non-blocking* xuất hiện.

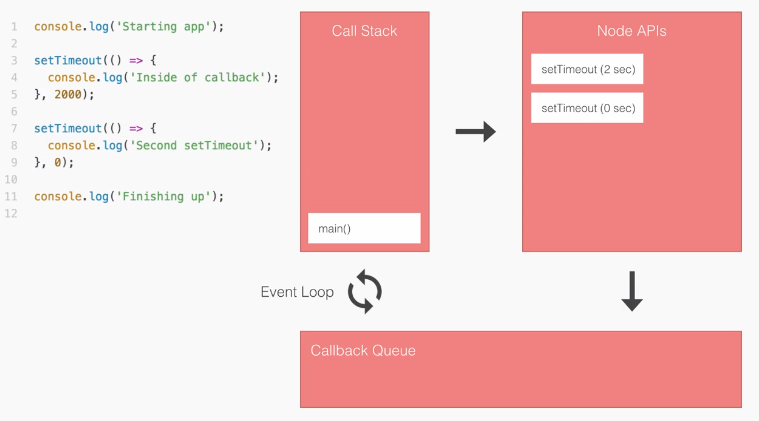
***Non-blocking I/O***

Mặt khác, sử dụng *request non-blocking,* bạn có thể *request* dữ liệu cho *user2* mà không phải đợi phản hồi *request* từ *user1.* Bạn có thể bắt đầu cả 2 *request* song song.

*Non-blocking I/O* loại bỏ nhu cầu đa luồng vì *server* có thể xử lý nhiều *request* cùng một lúc.

***JavaScript Event Loop***

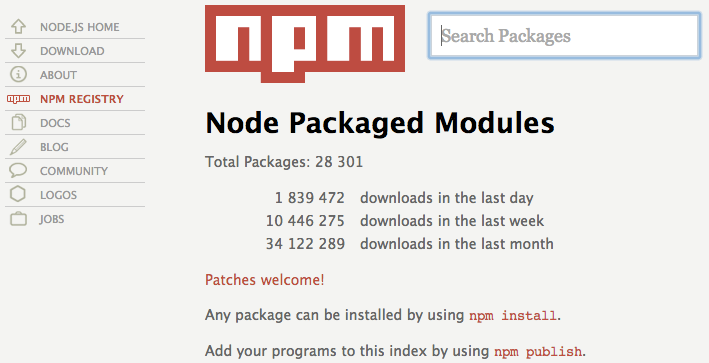
*Event Loop* là cơ chế giúp *JavaScript* có thể thực hiện nhiều thao tác cùng một lúc *(concurrent model),* trước giờ vẫn nghe nói *Node.js* có thể xử lý cả hàng ngàn *request* cùng một lúc mặc dù nó chỉ dùng một *thread* duy nhất *(Single Threaded).* Nếu như ở *PHP* hay *Java* thì với mỗi một *request* sẽ sinh ra một *thread* để xử lý *request* đó, các *thread* hoạt động độc lập, được cấp bộ nhớ, giao tiếp ngoại vi và trả về kết quả.



*Cách thức JavaScript Event Loop hoạt động*

***npm***

*npm –* viết tắt của *Node package manager* là một công cụ tạo và quản lý các thư viện lập trình *JavaScript* cho *Node.js.* Trong cộng đồng *JavaScript,* các lập trình viên chia sẻ hàng trăm nghìn các thư viện với các đoạn *code* đã thực hiện sẵn một chức năng nào đó. Nó giúp cho các dự án mới tránh phải viết lại các thành phần cơ bản, các thư viện lập trình hay thậm chí là cả *framework.*



*npm* có sẵn khi bạn tải *Node.js* về. Để kiểm tra xem trên hệ thống của bạn đã được cài *npm* chưa chúng ta sử dụng lệnh *npm -v,* nếu một phiên bản hiện ra thì hệ thống của bạn đã được cài đặt *npm.*

Vì *npm* là một phần mềm cài đặt trên máy tính của bạn nên bạn có thể sử dụng nó để cài đặt các thư viện *JavaScript* từ trên *Internet.* Để cài đặt một thư viện nào đó, chỉ cần mở cửa sổ *Terminal (hoặc CMD)* và thực thi lệnh giống dưới đây:

*npm install package-name*

*Ví dụ: ta thử tải Vuejs về sử dụng sẽ dùng lệnh:*

*npm install vue*

*Khi ai đó muốn sử dụng Vue.js chúng ta chỉ cần sử dụng lệnh require():*

*var Vue = require(‘vue’);*

***Cài đặt global và local***

Có 2 cách để cài đặt một gói bằng *npm:*

* ***Local:*** sẽ tạo ra thư mục *node\_modules* nếu chưa có trong *project* hoặc nếu có rồi sẽ lấy *code* của gói cần cài đặt đưa vào đây, tức chỉ hiện diện trong thư mục của *project* hiện tại. Khi cần sử dụng bạn có thể sử dụng lệnh *require().*
* ***Global:*** sẽ lưu trữ *code* của gói trong các *file* hệ thống cố định trong máy, chỉ có thể dùng các *package* này thông qua các hàm *CLI (Command Line Interface)* ví dụ như *glub.* Không thể dùng *package* thông qua *require().*

Mặc định thì các *package* khi cài đặt đều sẽ là cài trên *project* của bạn.

*npm update package\_name*

Các câu lệnh này có thể sử dụng *flag -g* để thực hiện cập nhật cho các gói được cài đặt *global.*

***Kiểm tra các gói cài đặt***

Để kiểm tra các gói đã được thêm thông qua *npm* sử dụng câu lệnh *npm ls,* nếu kiểm tra các cài đặt *global* thêm tham số *-g.*

*npm ls*

*npm ls -g*

***Package.json***

Để quản lý các gói cài đặt cục bộ bằng *npm* thì cách tốt nhất là thông qua *file package.json,* chính là *file* nằm trong thư mục gốc của *project. File JSON* này chứa các nội dung:

* Các gói thư viện lập trình mà *project* sử dụng.
* Cho phép xác định phiên bản chính xác của các gói thư viện lập trình được sử dụng.
* Các gói bạn xây dựng có thể chia sẻ dễ dàng với các lập trình viên khác trên toàn cầu thông qua *npm.*

Lệnh *npm init -yes* sẽ tạo ra *file package.json* mẫu.

Có một số các thuộc tính trong *package.json,* chúng ta cùng điểm qua:

* *name: tên gói thư viện*
* *version: phiên bản gói*
* *description: phần mô tả về gói thư viện*
* *homepage: trang chủ của gói*
* *author: tác giả*
* *contributors: tên người đóng góp cho package*
* *dependencies: danh sách các gói phụ thuộc, tự động được cài theo*
* *repository: loại repository và url của package, thông thường là git*
* *main: index.js*
* *keywords: các từ khóa*

***Require***

*Require* làm 3 thứ:

* Tải *module* đi kèm với *Node.js* như hệ thống *file* và *HTTP* từ *Node.js API.*
* Tải thư viện thứ 3 như *Express* và *Mongoose* mà bạn cần cài đặt từ *npm.*
* Giúp *require file* của bạn và *mô-đun hóa project.*

*Require* là một chức năng, và nó nhận tham số *path* tinh chỉnh và trả về *module.export.*

***Node Modules***

*Module Node* là 1 *block code* có thể tái sử dụng mà sự tồn tại của nó không vô tình ảnh hưởng đến các *code* khác.

Bạn có thể tạo *Module* của riêng mình và sử dụng nó với nhiều ứng dụng khác nhau.

*Node.js* có 1 bộ *built-in module* bạn có thể sử dụng mà không cần cài đặt thêm gì nữa.

***V8 turbo-charges JavaScript bằng cách tận dụng C++***

*V8* là một công cụ *runtime* nguồn mở được viết bằng *C++.*

*JavaScript ­­-> V8 (C++) -> Machine Code.*

*V8* thực hiện một tệp lệnh gọi là *ECMAScript* như được chỉ định trong *ECMA-262.*

*ECMAScript* được tạo ra bởi *ECMA International* để chuẩn hóa *JavaScript.*

Về cơ bản viêcj này cho phép bạn thêm các tính năng vào *JavaScript* ngoài *V8* của *Chrome* như *SpiderMonkey* của *Mozilla, Chakra* của *Microsoft, v.v…*

***Events***

Có 2 loại *event* trong *Node:*

* *Event System: C++* có lõi từ một thư viện gọi là *libuv. (Vd: đọc xong một file)*
* *Event Custom:* Lõi *JavaScript.*

### 2.1.2. Giới thiệu về *MongoDB*

***MongoDB là gì?***

*MongoDB* là một *database* hướng tài liệu *(document),* một dạng *NoSQL database.* Vì thế, *MongoDB* sẽ tránh cấu trúc *table-based* của *relational database* để thích ứng với các tài liệu như *JSON* có một *schema* rất linh hoạt gọi là *BSON. MongoDB* sử dụng lưu trữ dữ liệu dưới dạng *Document JSON* nên mỗi một *collection* sẽ có các kích cỡ và các *document* khác nhau. Các dữ liệu được lưu trữ trong *document* kiểu *JSON* nên truy vấn sẽ rất nhanh.

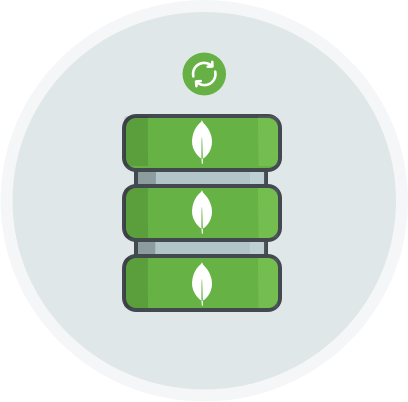
***Định nghĩa thêm về MongoDB***

*MongoDB* lần đầu ra đời bởi *MongoDB Inc.,* tại thời điểm đó là thế hệ 10, vào tháng Mười năm 2007 nó là một phần của sản phẩm *PaaS (Platform as a Service)* tương tự như *Windows Azure* và *Google App Engine.* Sau đó nó đã được chuyển thành nguồn mở từ năm 2009.

*MongoDB* đã trở thành một trong nhưng *NoSQL database* nổi trội nhất bấy giờ, được dùng làm *backend* cho rất nhiều *website* như *eBay, SourceForge* và *The New York Times.*

***Các feature của MongoDB gồm có:***

* ***Các ad hoc query:*** hỗ trợ *search* bằng *field,* các phép *search* thông thường, *regular expression searches* và *range queries.*
* ***Indexing:*** bất kì *field* nào trong *BSON document* cũng có thể được *index.*
* ***Replication:*** có ý nghĩa là *“nhân bản”,* là có một phiên bản giống hệt phiên bản đang tồn tại, đang sử dụng. Với cơ sở dữ liệu, nhu cầu lưu trữ lớn, đòi hỏi *cơ sở dữ liệu* toàn vẹn, không bị mất mát trước những sự cố ngoài dự đoán là rất cao. Vì vậy, người ta nghĩ ra khái niệm *“nhân bản”,* tạo một phiên bản *cơ sở dữ liệu* giống hệt *cơ sở dữ liệu* đang tồn tại, và lưu trữ ở một nơi khác, đề phòng có sự cố.
* ***Aggregation:*** các *Aggregation operation* xử lý các bản ghi dữ liệu và trả về kết quả đã được tính toán. Các phép toán tập hợp các nhóm giá trị từ nhiều *Document* lại với nhau, và có thể thực hiện nhiều phép toán đa dạng trên dữ liệu đã được nhóm đó để trả về một kết quả duy nhất. Trong đó *SQL, count (\*)* và *GROUP BY* là tương đương với *Aggregation* trong *MongoDB.*
* ***Lưu trữ file:*** *MongoDB* được dùng như một hệ thống *file* tận dụng những *function* trên và hoạt động như một cách phân phối qua *sharding.*



***Khi nào sử dụng MongoDB?***

* ***Quản lý và truyền tải content:*** Quản lý đa dạng nhiều *product* của *content* chỉ trong một kho lưu trữ *data* cho phép thay đổi và phản hồi nhanh chóng mà không chịu thêm phức tạp từ hệ thống *content.*
* ***Cấu truc Mobile và Social:*** *MongoDB* cung cấp một *platform* có sẵn, phản xạ nhanh, và dễ mở rộng cho phép rất nhiều khả năng đột phá, phân tích *real-time,* và hỗ trợ toàn cầu.
* ***Quản lý data khách hàng:*** Tận dụng khả năng *query* nhanh chóng cho phân tích *real-time* trên *cơ sở dữ liệu* người dùng cực lớn với các mô hình *data* phức tạp bằng các *schema* linh hoạt và tự động *sharding* cho mở rộng chiều ngang.

***Ưu điểm của MongoDB***

* Dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ.
* Dữ liệu được *caching (ghi đệm)* lên *RAM,* hạn chế truy cập vào *ổ cứng* nên tốc độ đọc và ghi cao.

***Nhược điểm của MongoDB***

* Không ứng dụng được cho các mô hình giao dịch nào yêu cầu dộ chính xác cao do không ràng buộc.
* Không có cơ chế *transaction (giao dịch)* để phục vụ các ứng dụng ngân hàng.
* Dữ liệu lấy *RAM* làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ *RAM* lớn*.*
* Mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống *ổ cứng* ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao.

# CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG (UML)

## A. Mô hình nghiệp vụ

\*Sơ đồ nghiệp vụ



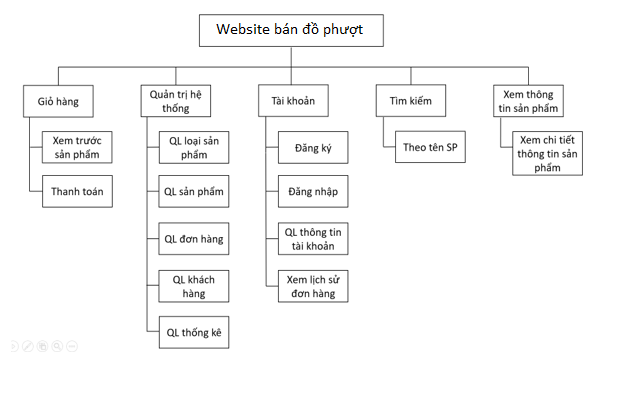
\*Yêu cầu phi chức năng

-Giao diện thân thiện, dễ dùng, hấp dẫn, dễ tìm kiếm thông tin các sản phẩm.

-Hệ thống chạy đúng, ổn định.

-Đáp ứng hầu hết các yêu cầu của người dùng.

## B. Sơ đồ phân rã chức năng Website bán đồ Phượt



## C. DANH MỤC CÁCH KÝ HIỆU

|  |  |  |
| --- | --- | --- |
| **PHẦN TỬ MÔ HÌNH** | **KÝ HIỆU** | **Ỹ NGHĨA** |
| **Biểu đồ USE CASE** | | |
| Tác nhân (Actor) |  | Một người/nhóm người hoặc một thiết bị hoặc hệ thống tác động hoặc thao tác đến chương trình. |
| Use-Case  (Ca sử dụng ) |  | Biểu diễn một chức năng xác định của hệ thống. |
| Mỗi quan hệ giữa các Use case |  | Use Case này sử dụng lại chức năng của use case khách. |
|  | Use case này mở rộng từ use case kia bằng cách thêm chức năng cụ thể. |
|  | Use case này kế thừa các chức năng từ use case kia. |
| **Biểu đồ LỚP** | | |
| Lớp (class) |  | Biểu diễn tên lớp, thuộc tính, và phương thức của lớp đó. |
| Quan hệ kiểu kết hợp |  | Biểu diễn quan hệ giữa hai lớp độc lập , có liên quan đến nhau. |
| Quan hệ hợp thành |  | Biểu diễn quan hệ bộ phận tổng thể |
| Quan hệ phụ thuộc |  | Các lợp phụ thuộc lẫn nhau trong hoạt động của hệ thống. |
| **Biểu đồ TRẠNG THÁI** | | |
| Trạng thái |  | Biểu diễn tạng thái của đối tượng trong vòng đời của đối tượng đó. |
| Trạng thái khởi đầu |  | Khởi đầu vòng đời của đối tượng đó |
| Trạng thái kết thúc |  | Kết thúc vòng đời của đối tượng đó |
| Chuyển tiếp  (Transition) |  | Chuyển từ trạng thái này sang trạng thái khác. |
| **Biểu đồ TUẦN TỰ** | | |
| Procedure  (Phương thức) |  | Là một phương thức của B mà đối tượng A gọi thực hiện. |
| Message  (Thông điệp) |  | Là một htoong báo cảu B gửi cho A |
| **Biểu đồ HOẠT ĐỘNG** | | |
| Hoạt động |  | Mô tả hoạt động gồm tên hoạt động |
| Trạng thái khởi đầu |  | Khởi đầu vòng đời của đối tượng đó |
| Trạng thái kết thúc |  | Kết thúc vòng đời của đối tượng đó |
| Thanh đồng bộ ngang |  | Mô tả thanh đồng bộ ngang |
| Chuyển tiếp |  |  |
| Quyết định |  | Mô tả một lựa chọn điều kiện |
| Các luồng | Phân tách các lớp đối tượng khách nhau trong biểu đồ hoạt động | Phân cách nhau bởi một đường kẻ dọc từ trên xuống dưới biểu đồ. |
| **Biểu đồ THÀNH PHẦN** | | |
| Thành phần |  | Mô tả thành phần của biểu đồ, mỗi thành phần có thể chứa nhiều lớp hoặc nhiều chương trình con. |
| Mỗi quan hệ phụ thuộc giữa các thành phần |  | Mỗi quan hệ giữa các thành phần (nếu có) |
| **Biểu đồ TRIỂN KHAI** | | |
| Các node  (các thiết bị) |  | Biểu diễn các thành phần không có bộ vi sử lý. |
| Các bộ sử lý |  | Biểu diễn các thành phần của bộ vi sử lý. |
| Liên kết truyền thông TCP/IP |  | Giao thức truyền thông TCP/IP thông qua kết nói mạng LAN |

## 1. Biểu đồ Use Case

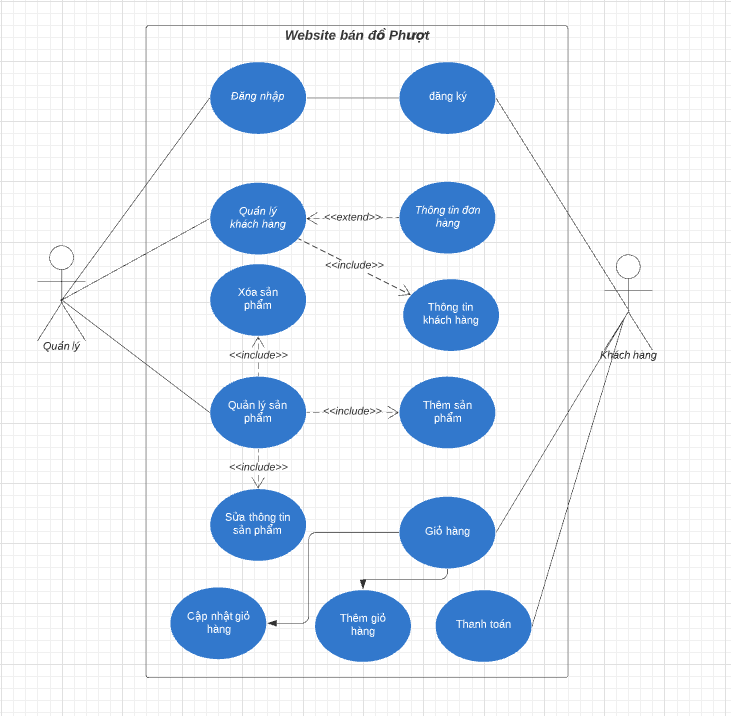
### 1.1.Use Case Tổng quát

Quản lý, vào mỗi đầu giờ khi bắt đầu làm, quản lý đăng nhập vào hệ thống và tích điểm danh ca làm.

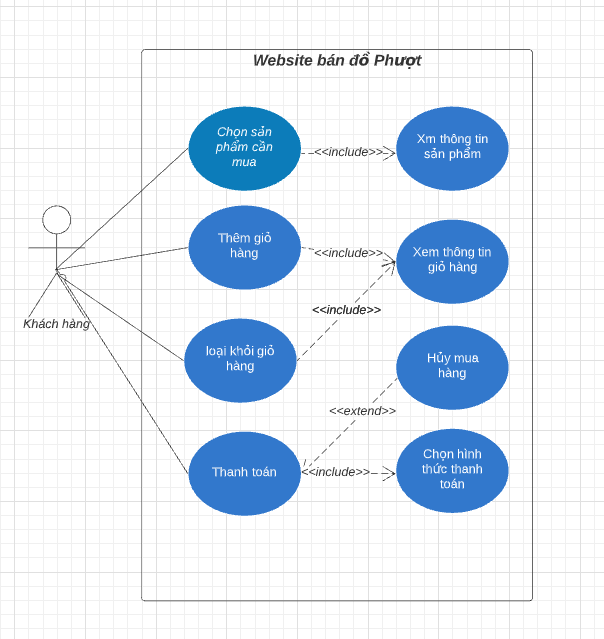
Quản lý có thể tìm sản phẩm theo tên sản phẩm hoặc theo mã sản phẩm. Nhập thông tin về mã sản phẩm hoặc sản phẩm, hệ thống sẽ tìm kiếm và gửi lại kết quả.

Quản lý có thể thêm, sửa xóa thông tin sản phẩm trên hệ thống quản trị web.

Quản lý có thể tìm khách hàng theo mã khách hàng sau đó có thể tùy chọn các chức năng tương ứng như nhập thông tin khách hàng, xóa thông tin khách hàng, sửa thông tin khách hàng…

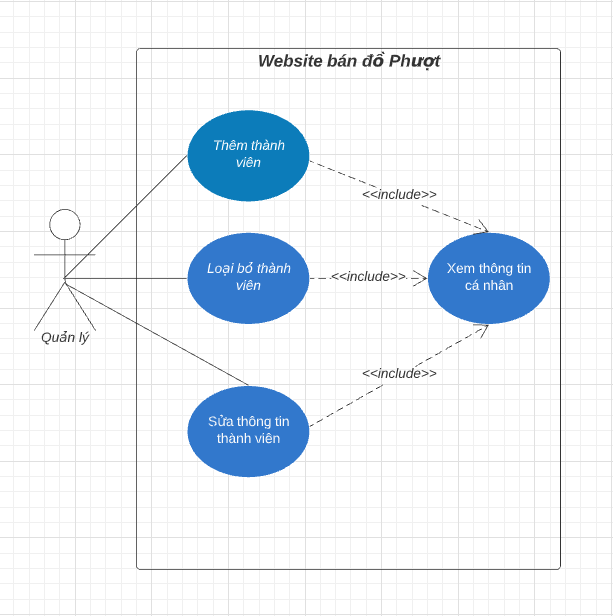


### 1.2. Use Case Đăng nhập

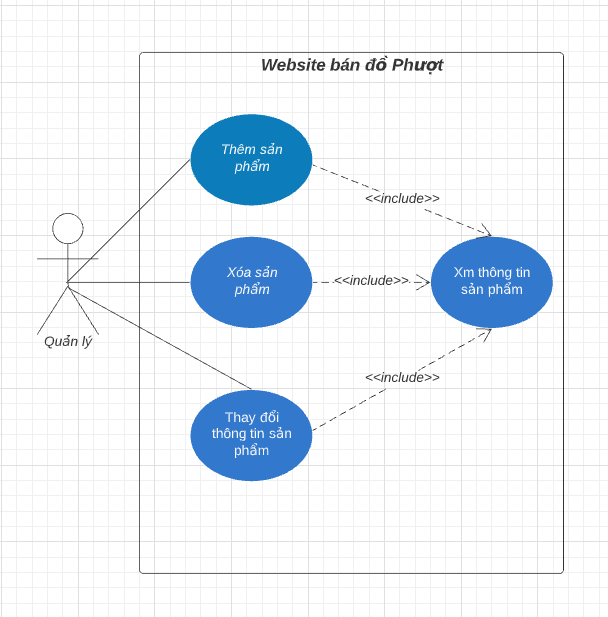


### 1.3. Use Case quản lý khách hàng

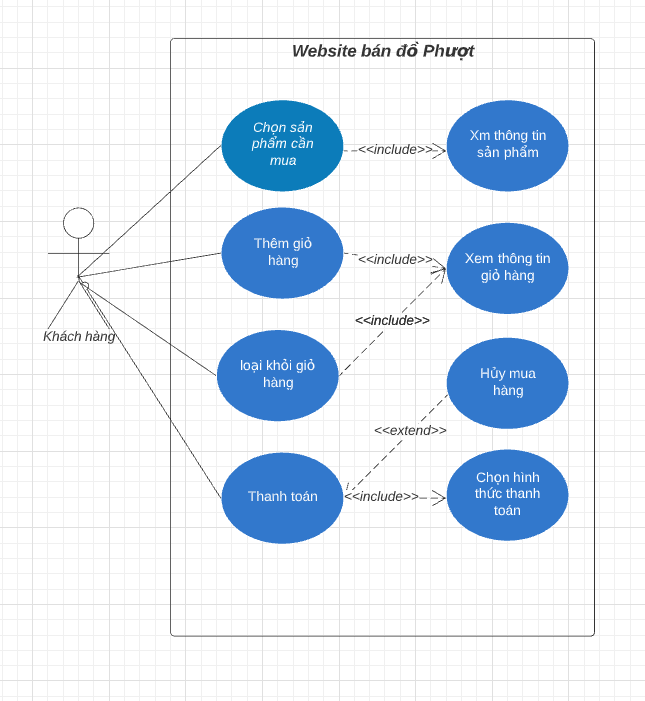
Khách hàng vào trang web hoặc tới trực tiếp cửa hàng lựa chọn sản phẩm và xem mã hoặc tên sản phẩm trực tiếp trên sản phẩm hoặc nếu mua online thì xem trên trang chủ cửa hàng sau đó đặt hàng theo mã sản phẩm. Sau khi đặt mua sản phẩm khách hàng có thể thanh toán ngay tại quầy theo hai hình thức là thẻ ngân hàng hoặc tiền mặt. Khách hàng có thể lập thẻ thành viên để tích điểm giảm giá cho những lần mua sau.



### 1.4. Use Case Quản lý sản phẩm



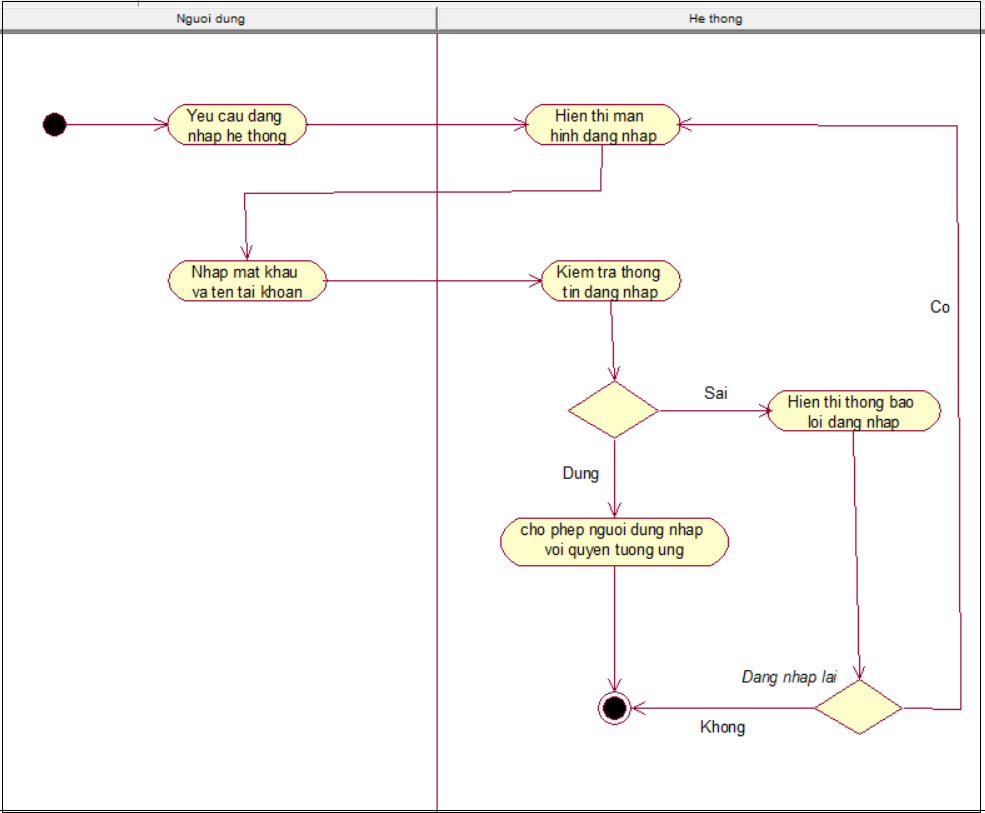
### 1.5. Use Case Mua hàng



### 1.6. Biểu đồ hoạt động

***Biểu đồ hoạt động đăng nhập***

* Mô tả: Use case cho phép người dùng đang nhập vào hệ thống bằng tài khoản được cấp của mình để sử dụng chương trình.
* Dòng sự kiện chính:

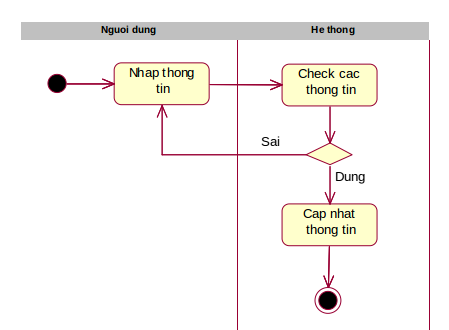


* Dòng sự kiện khác: Không có
* Yêu cầu đặc biệt: Không có
* Tiền điều kiện:
* Người dùng phải được cấp tài khoản.
* Người dùng chưa đăng nhập tài khoản của mình vào hệ thống.
* Hậu điều kiện: Nếu use case thành công thì sẽ được sử dụng hệ thông tương ứng. Ngược lại, trạng thái hệ thống sẽ không thay đổi.
* Điểm mở rộng: Không có

***Biểu đồ hoạt động của chức năng Đăng ký***

- Mô tả: User case cho phép người dùng đăng nhập vào hệ thống bằng tài khoản được cấp của mình để sử dụng hệ thống.

- Dòng sự kiện chính:



- Dòng sự kiện khác: Không có.

- Yêu cầu đặc biệt: Không có.

- Tiền điều kiện: Tài khoản đã đăng nhập và chưa đăng cuất khỏi hệ thống.

- Hậu điều kiện: Nếu use case thành công thì người dùng sẽ có quyền sử dụng hệ thống để nhập và tìm kiếm thông tin. Còn ngược lại, thì trạng thái của hệ thống không đổi.

- Luồng sự kiện:

+Người dùng truy cập hệ thống đăng ký tài khoản.

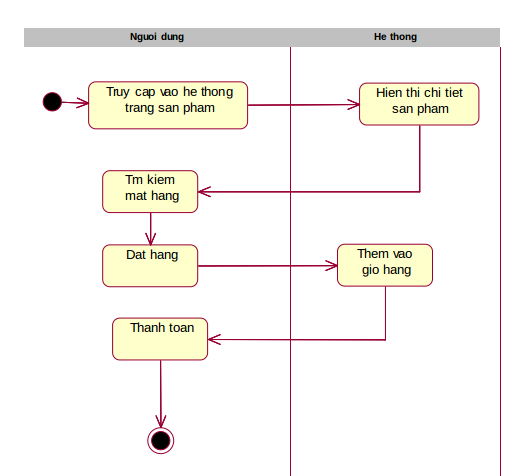
+Nhập các thông tin yêu cầu

+Thành công hệ thống sẽ lưu thông tin vào hệ thống

***Biểu đồ hoạt động của chức năng Mua hàng***

- Mô tả: Use case cho phép người dùng đăng nhập vào hệ thống bằng tài khoản được cấp của mình để sử dụng hệ thống, người dùng có thể vào xem sản phẩm để thực hiện mua hàng

- Dòng sự kiện chính:



- Dòng sự kiện khác: Không có.

- Yêu cầu đặc biệt: Không có.

- Tiền điều kiện: Tài khoản đã đăng nhập và chưa đăng cuất khỏi hệ thống.

- Hậu điều kiện: Nếu use case thành công thì người dùng sẽ có quyền sử dụng hệ thống để nhập và tìm kiếm thông tin. Còn ngược lại, thì trạng thái của hệ thống không đổi.

- Luồng sự kiện:

+Sau khi xác nhận đăng nhập, hệ thống sẽ hiện ra các mục quản lý và người dùng sẽ chọn vào sản phẩm mình muốn.

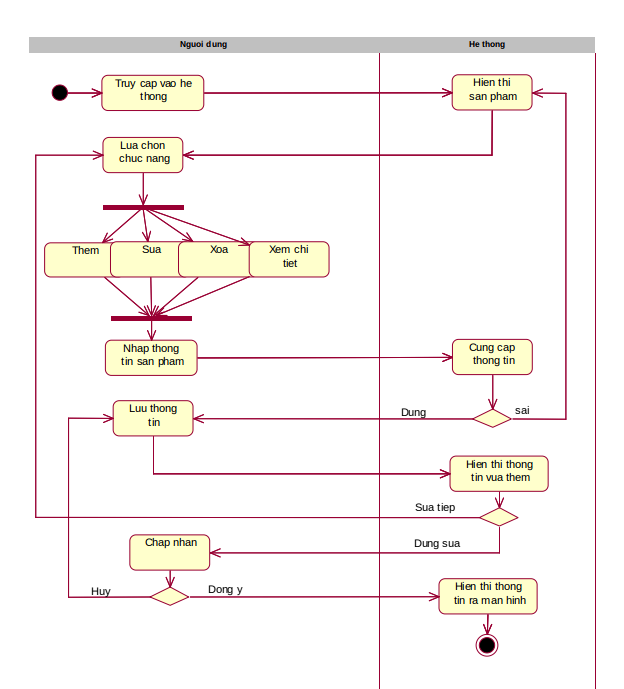
+Sau khi chọn và xem xét sản phẩm người dùng click thêm vào giỏ hàng.

+Người dùng click vào giỏ hàng và xác nhận thanh toán.

***Biểu đồ hoạt động của chức năng Quản lý sản phẩm***

-Mô tả: User case cho phép người dùng đăng nhập vào hệ thống bằng tài khoản được cấp của mình để sử dụng hệ thống, người dùng có thể vào thực hiện thao tác thêm sửa, xóa sản phẩm

-Dòng sự kiện chính:



- Dòng sự kiện khác: Không có.

- Yêu cầu đặc biệt: Không có.

- Tiền điều kiện: Tài khoản đã đăng nhập và chưa đăng cuất khỏi hệ thống.

- Hậu điều kiện: Nếu use case thành công thì người dùng sẽ có quyền sử dụng hệ thống để nhập và tìm kiếm thông tin. Còn ngược lại, thì trạng thái của hệ thống không đổi.

- Luồng sự kiện:

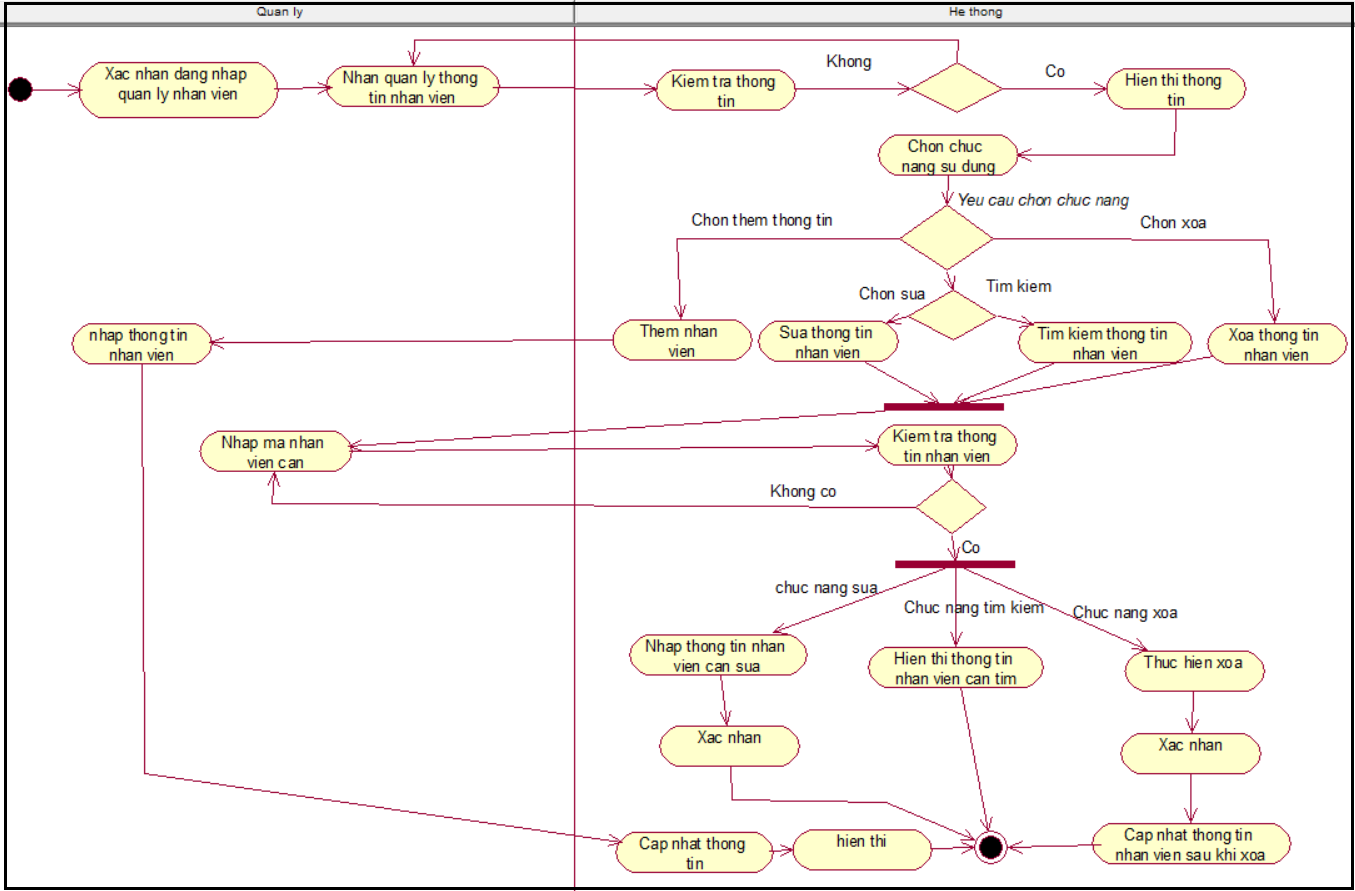
+Sau khi xác nhận đăng nhập, hệ thống sẽ hiện ra các mục quản lý và người dùng sẽ chọn vào sản phẩm mình muốn.

+Người dùng có thể vào thêm, sửa xóa sản phẩm với quyền được cấp

***Biểu đồ hoạt động Quản lý nhân viên***

- Mô tả: Use case quản lý nhân viên cho phép quản lý có thể lưu trữ và xử lí dữ liệu của nhân viên để tạo thành những thông tin hữu ích giúp nhà quản lý quản lý được nhân viên khi vào làm, trong khi làm, nghỉ làm.

- Dòng sự kiện chính:



- Dòng sự kiện khác: Không có

- Yêu cầu đặc biệt: Không có

- Tiền điều kiện:

* Quản lý phải đăng nhập vào hệ thống.
* Thông tin của nhân viên phải được lưu trữ đầy đủ trong cơ sở dữ liệu.
* Quản lý muốn thực hiện các thao tác quản lý đối với những thông tin về nhân viên như thêm, sửa, xóa hoặc tìm kiếm thông tin nhân viên.

- Hậu điều kiện:

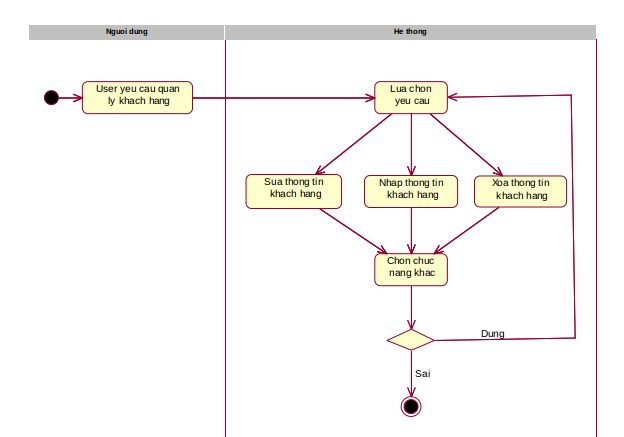
* Thành công: Thông tin nhân viên được cập nhật.
* Lỗi: Không thành công khi xuất hiện lỗi trong quá trình xử lí thông tin nhân viên

- Điểm mở rộng: Không có

***Biểu đồ hoạt động quản lý khách hàng***

- Mô tả : Quản lý khách hàng là quản lý thông tin cá nhân khách hàng đăng ký tại cơ sở dữ liệu của trang web, người quản lý có thể có những thay đổi đối với thông tin khách hàng trên hệ thống.

- Dòng sự kiện chính:



-Dòng sự kiện khác: Không có

-Yêu cầu đặc biệt: Không có

-Tiền điều kiện:

* Quản lý phải đăng nhập vào hệ thống.
* Thông tin của khách hàng phải được lưu trữ đầy đủ trong cơ sở dữ liệu.
* Quản lý muốn thực hiện các thao tác quản lý đối với những thông tin về khách hàng như thêm, sửa, xóa hoặc tìm kiếm thông tin.

-Hậu điều kiện:

* Thành công: Thông tin khách hàng được cập nhật.
* Lỗi: Không thành công khi xuất hiện lỗi trong quá trình xử lí thông tin khách hàng.

-Điểm mở rộng: Không có

## 2. Biểu đồ tuần tự

Biểu đồ tuần tự biểu diễn tương tác giữa những người dùng và những đối tượng bên trong hệ thống. Biểu đồ này cho biết các thông điệp được truyền tuần tự như thế nào theo thời gian. Thứ tự các sự kiện trong biểu đồ tuần tự hoàn toàn tương tự như trong scenario mô tả use case tương ứng.

Biểu đồ tuần tự được biểu diễn bởi các đối tượng và đường truyền đi giữa các đối tượng đó.

Trong hệ thống quản lý bán hàng, chúng ta lựa chọn biểu đồ tương tác dạng tuần tự để biểu diễn các tương tác giữa các đối tượng. Để xác định rõ các thành phần cần bổ sung trong biểu đồ lớp, trong mỗi biểu đồ tuần tự của hệ thống quản lý bán hàng sẽ thực hiện:

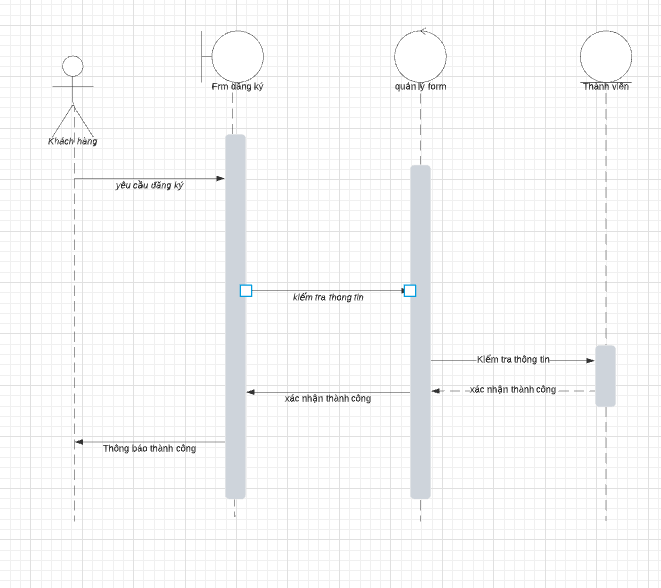
- Xác định rõ kiểu của đối tượng tham gia trong tương tác (ví dụ giao diện, điều khiển hay thực thể).

- Mỗi biểu đồ tuần tự có thể có ít nhất một lớp giao diện (Form) tương ứng

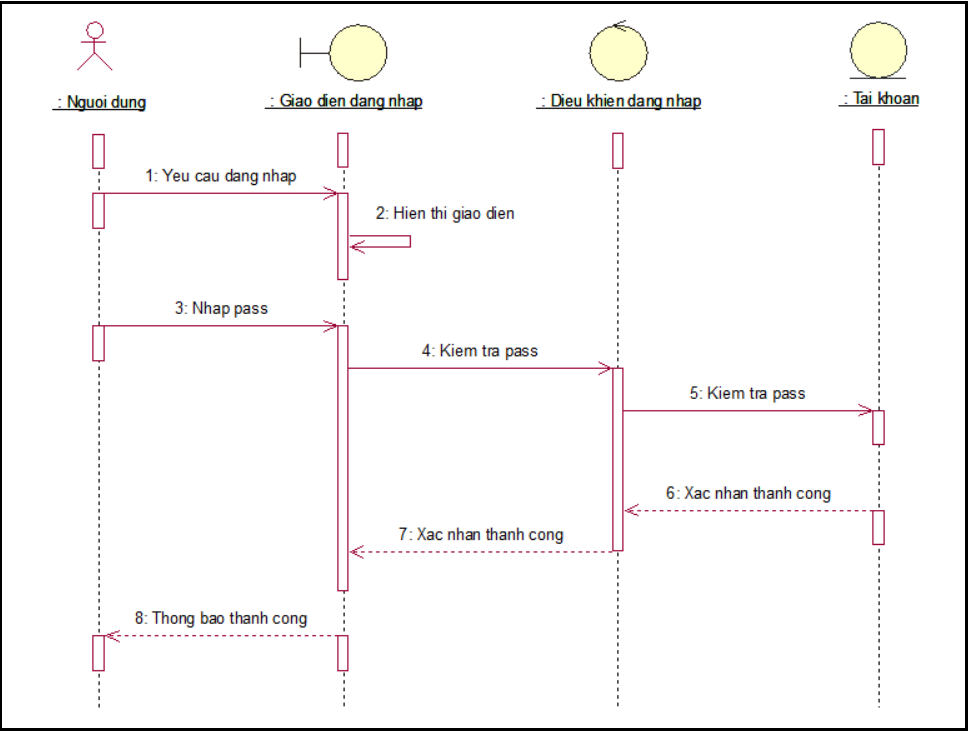
với chức năng (use case) mà biểu đồ đó mô tả

- Mỗi biểu đồ tuần tự có thể liên quan đến một hoặc nhiều đối tượng thực thể. Các đối tượng thực thể chính là các đối tượng của các lớp đã được xây dựng trong biểu đồ thiết kế chi tiết. Dưới đây là một số biểu đồ tuần tự cho các chức năng của hệ thống quản lý bán đồ Phượt:

### 2.1.Biểu đồ tuần tự của đăng ký

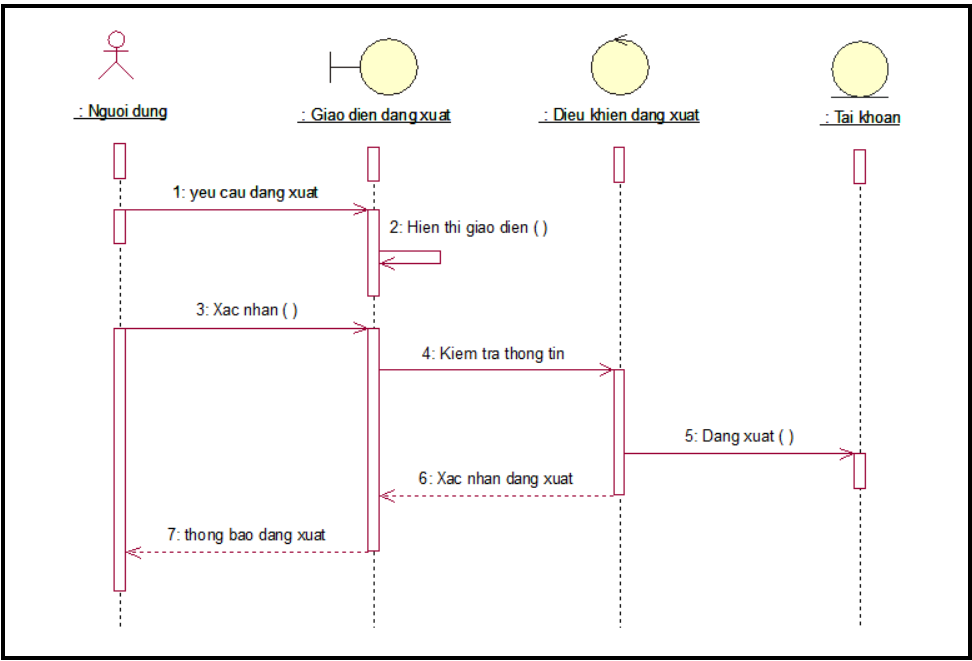


### 2.2 biểu đồ tuần tự đăng nhập



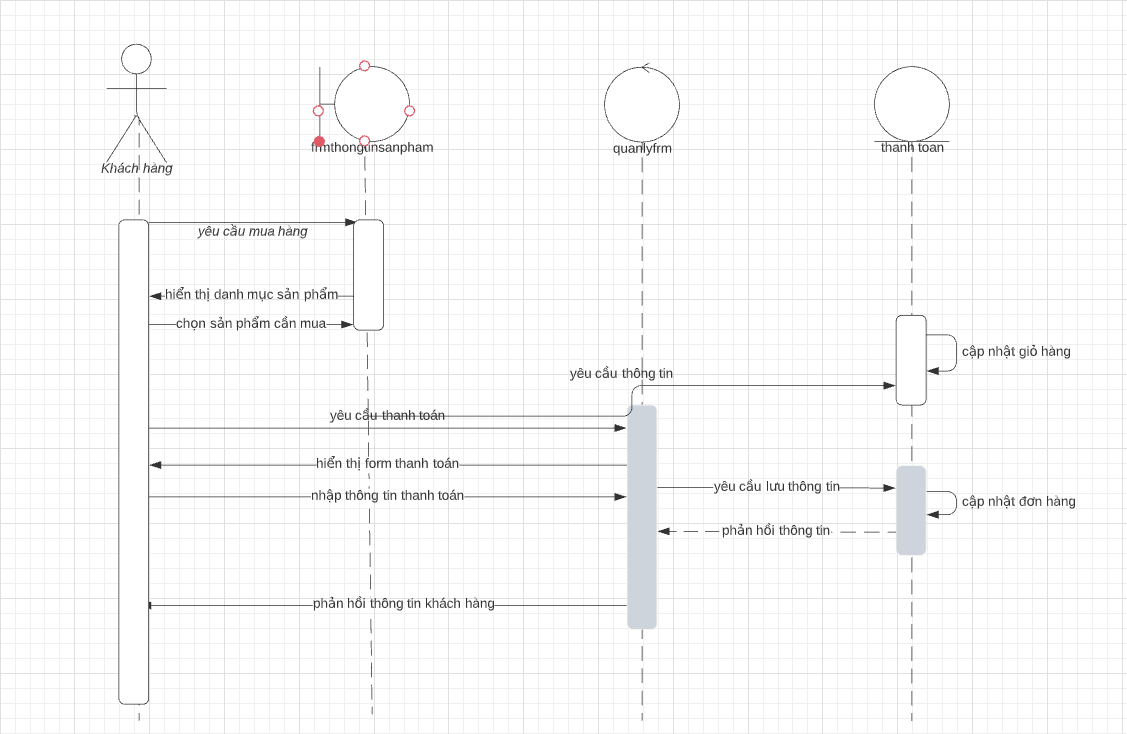
*Hình 2.2:Biểu đồ tuần tự của đăng nhập*

### 2.3.Biểu đồ tuần tự của đăng xuất

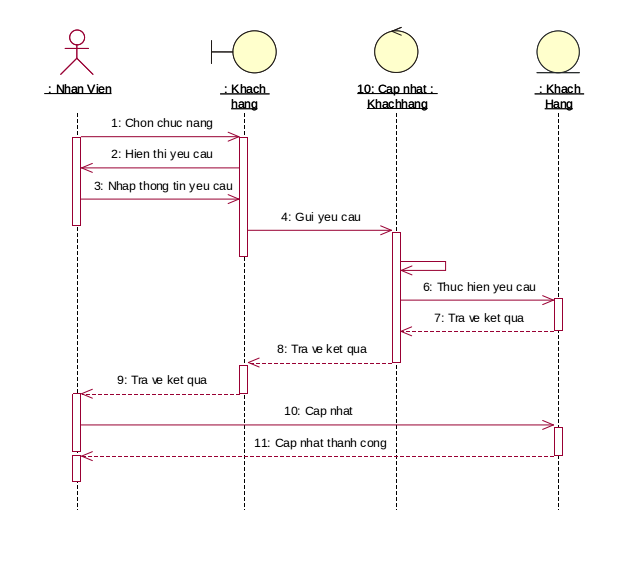


*Hình 2.3:Biểu đồ tuần tự của đăng xuất*

### 2.4. Biểu đồ tuần tự chức năng Mua hàng

*Hình 2.4. Biểu đồ tuần tự chức năng Mua hàng*

### 2.5. Biểu đồ tuần tự chức năng quản lý khách hàng



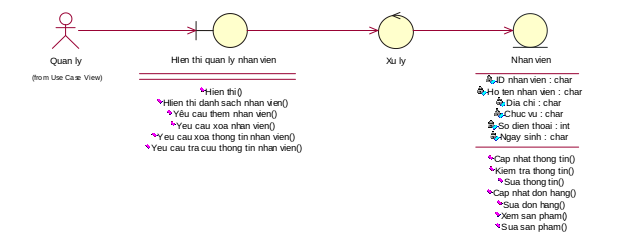
*Hình 2.5. Biểu đồ tuần tự chức năng quản lý Khách hàng*

## 3.Biểu đồ lớp (Class)

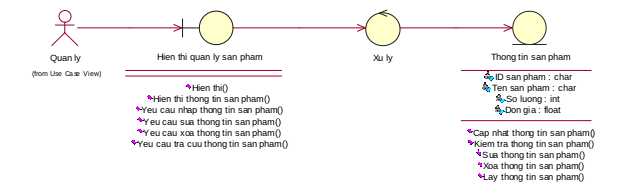
- Là một tập hợp các đối tượng chia sẻ chung một cấu trúc và hành vi (cùng thuộc tính, hoạt động, mối quan hệ và ngũ nghĩa). Cấu trúc được mô tả bởi các thuộc tính và các mối quan hệ, còn hành vi được mô tả bằng các hoạt động. Mỗi lớp là 1 sự trừu tượng hoá của các đối tượng thế giới thực, và các đối tượng tồn tại trong thế giới thực được xem như là các thể hiện của lớp

- Ký hiệu : lớp được trình bày bởi 3 phần : Tên lớp, danh sách các thuộc tính (attribute), danh sách các họat động (operation), trong đó phần thuộc tính và phần họat động có thể được che dấu đi trong mức độ trình bày tổng quan

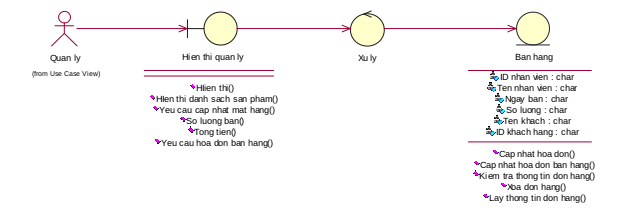
### 3.1. Biểu đồ lớp quản lý nhân viên



### 3.2. Biểu đồ lớp quản lý sản phẩm

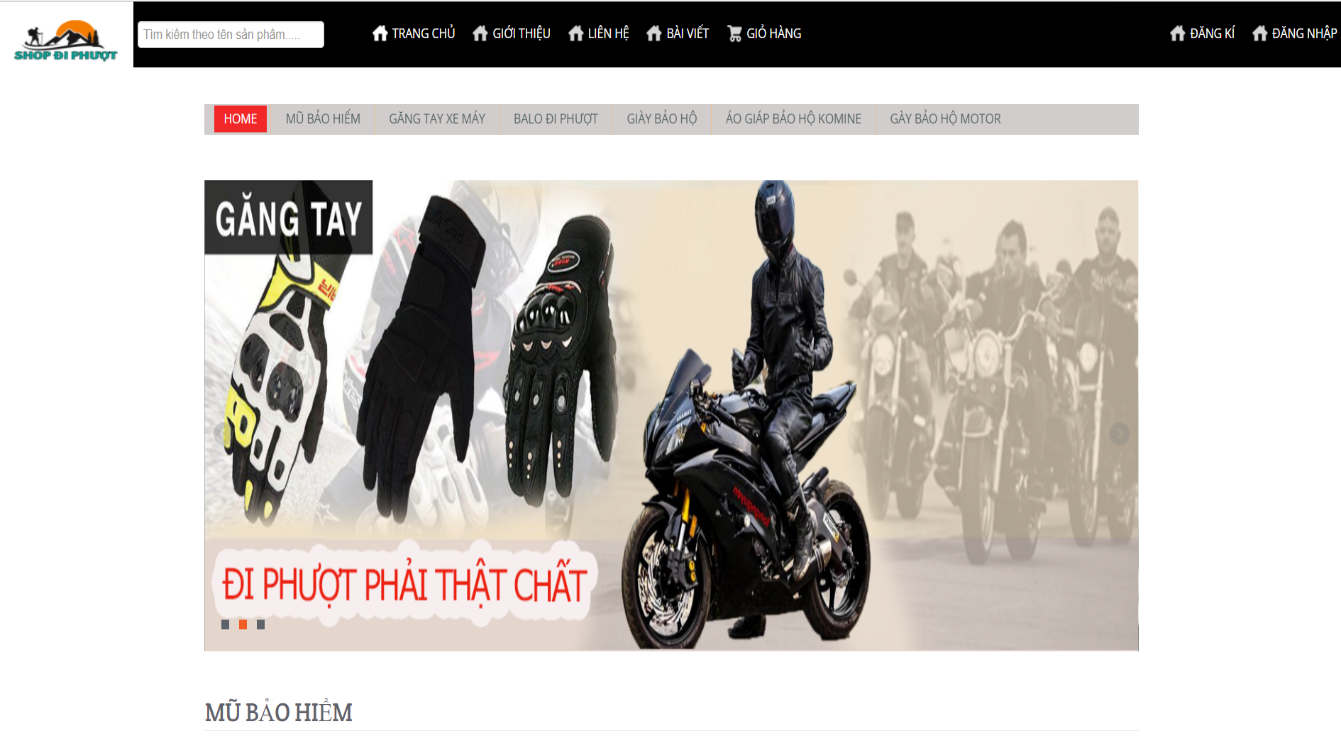


### 3.3. Biểu đồ lớp cho use case bán hàng

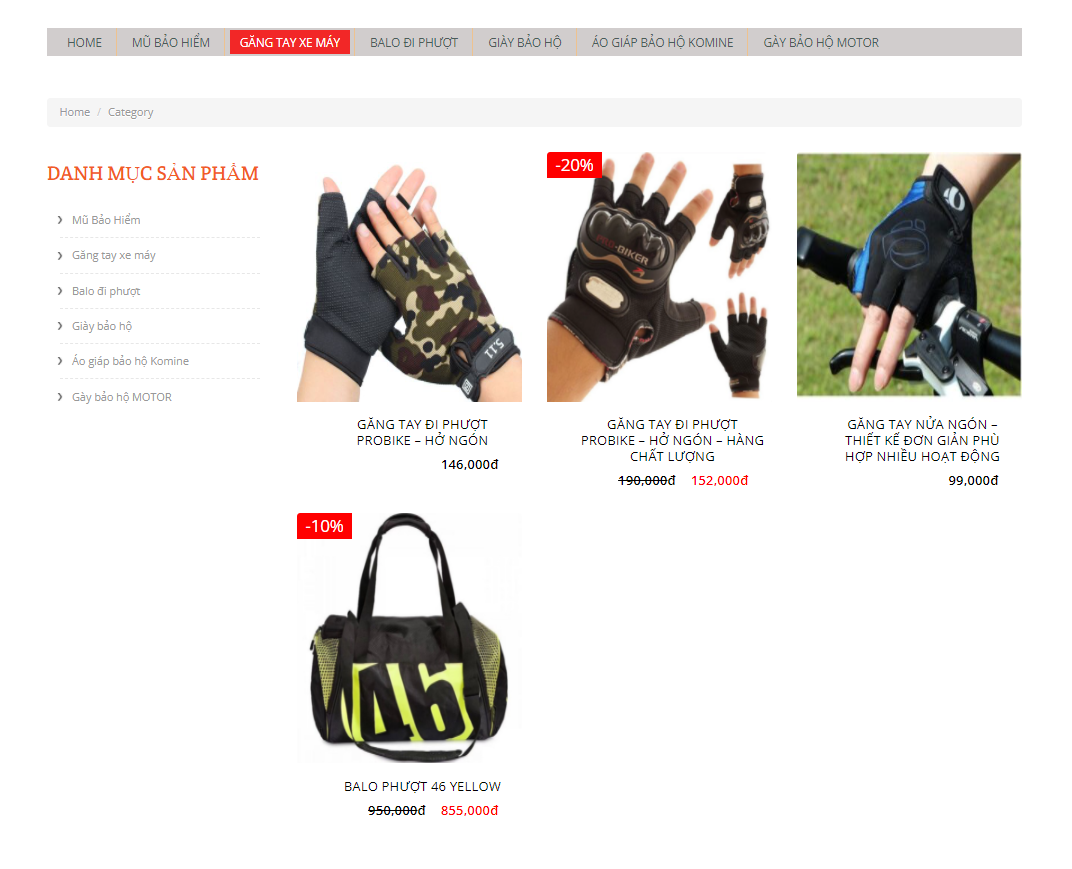


# CHƯƠNG 3: CÀI ĐẶT

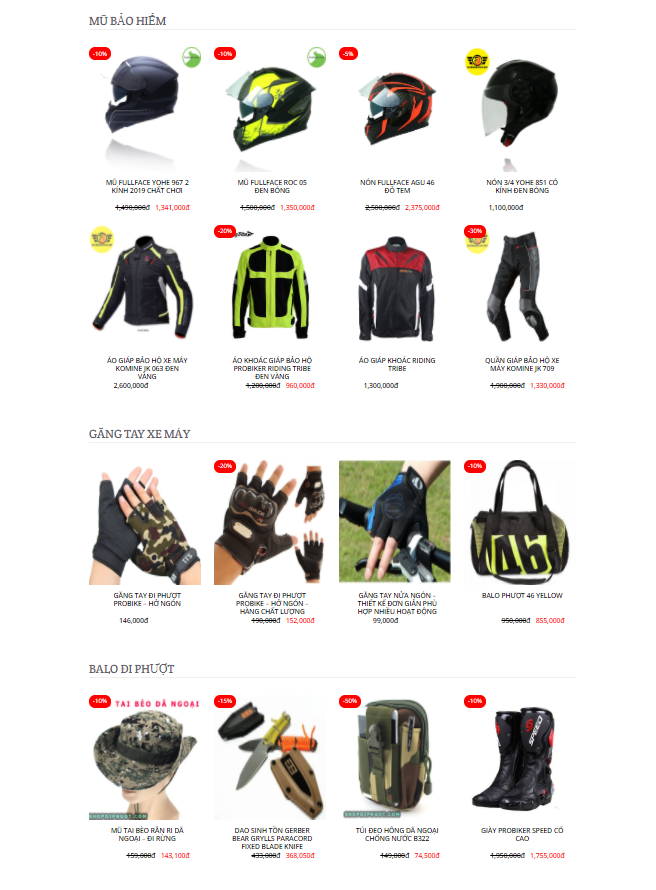
## 3.1 Giao diện người dùng



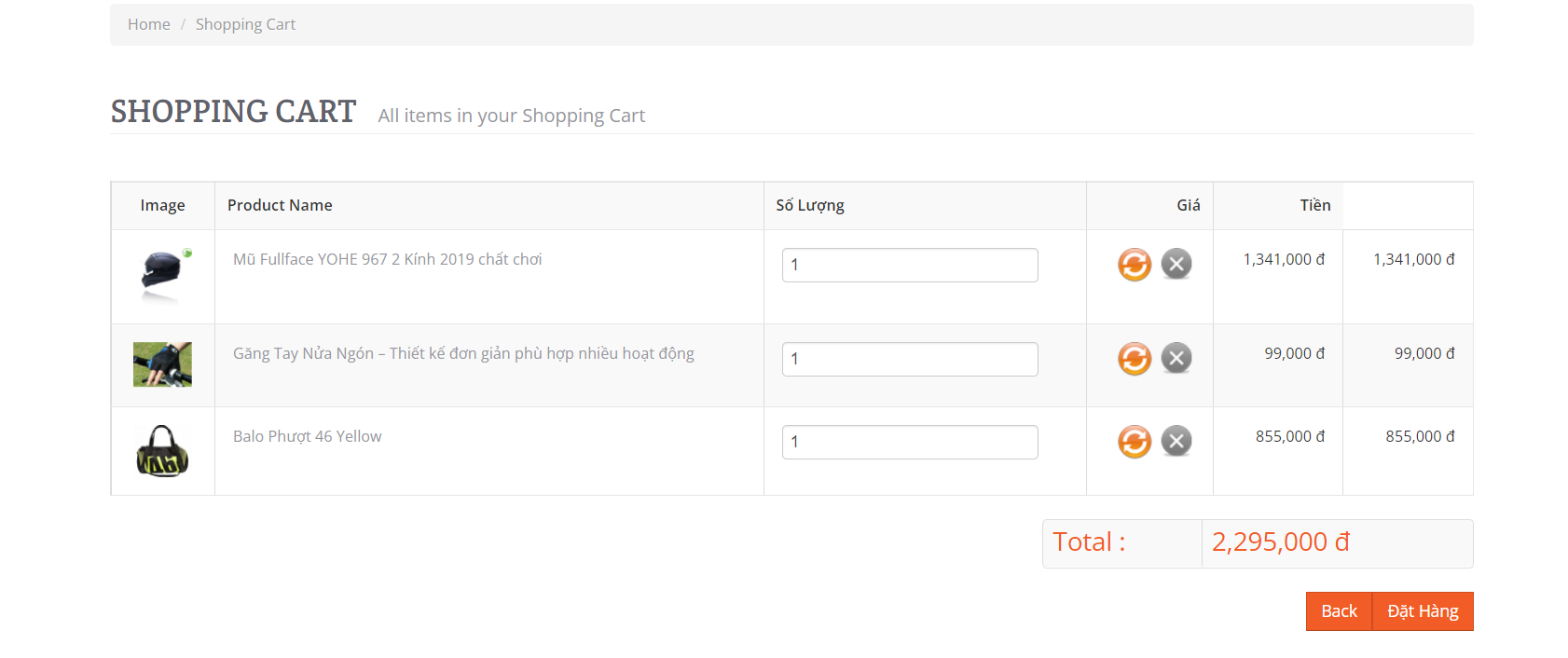
*Hình 3.1.1 Giao diện trang chủ*

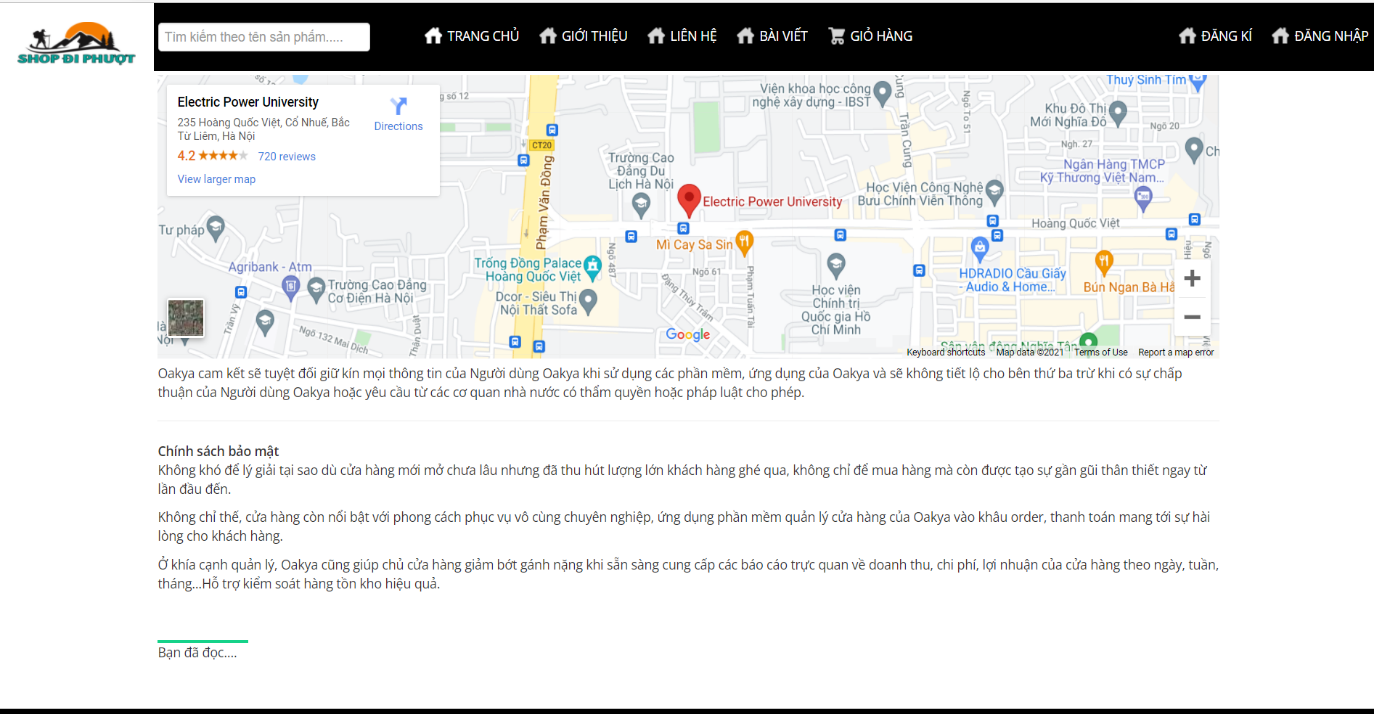


*Hình 3.1.2 Giao diện danh mục sản phẩm.*

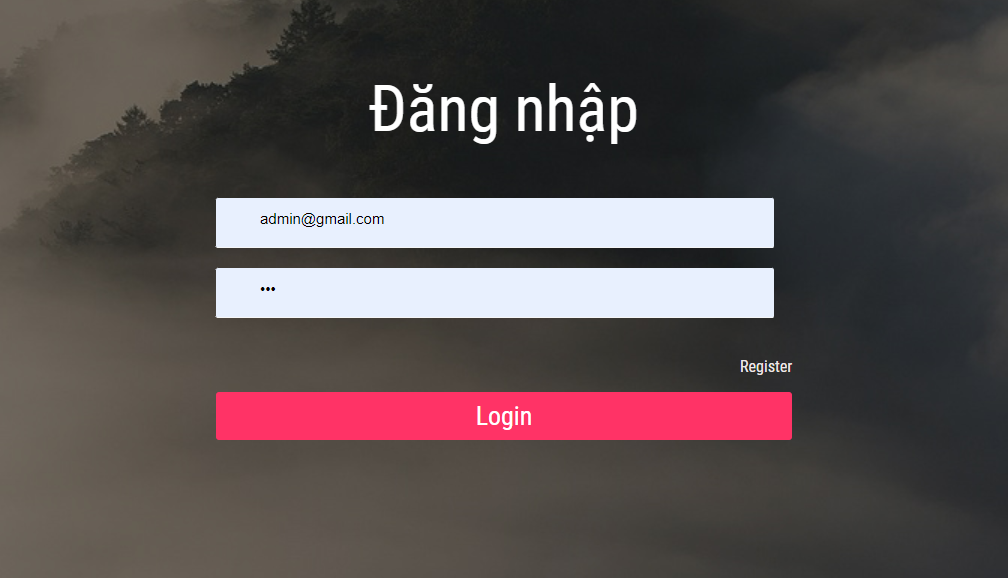


*Hình 3.1.3 Giao diện sản phẩm.*

*Hình 3.1.4. Giao diện giỏ hàng.*

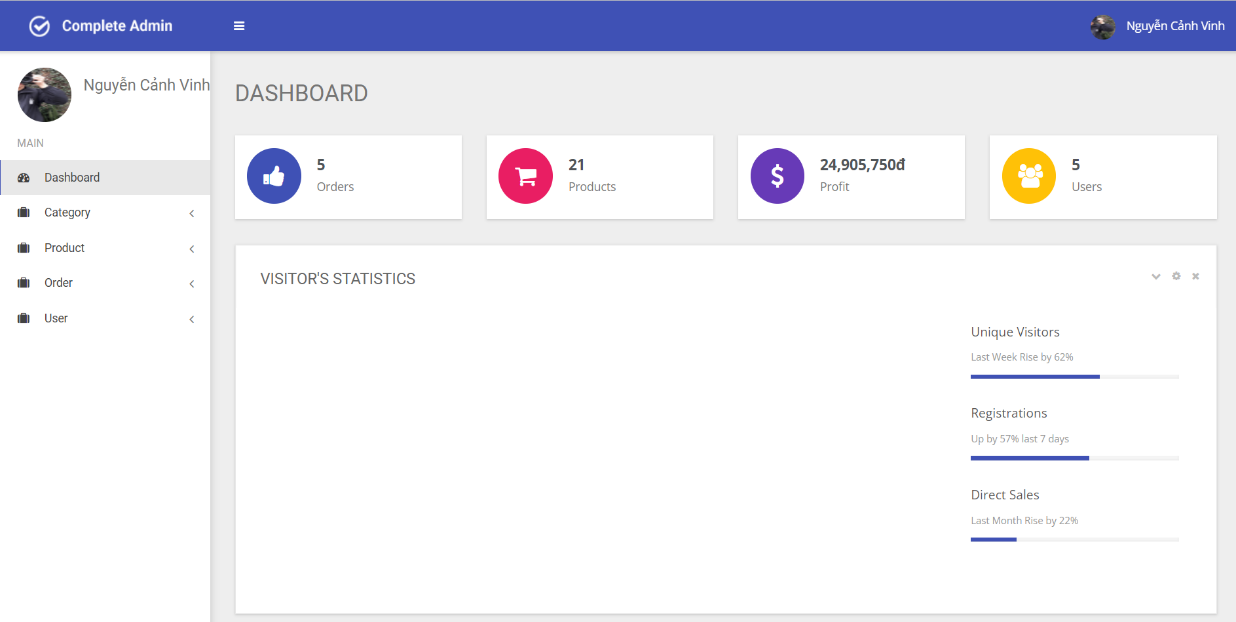


*Hình 3.1.5. Giao diện chi tiết liên hệ*

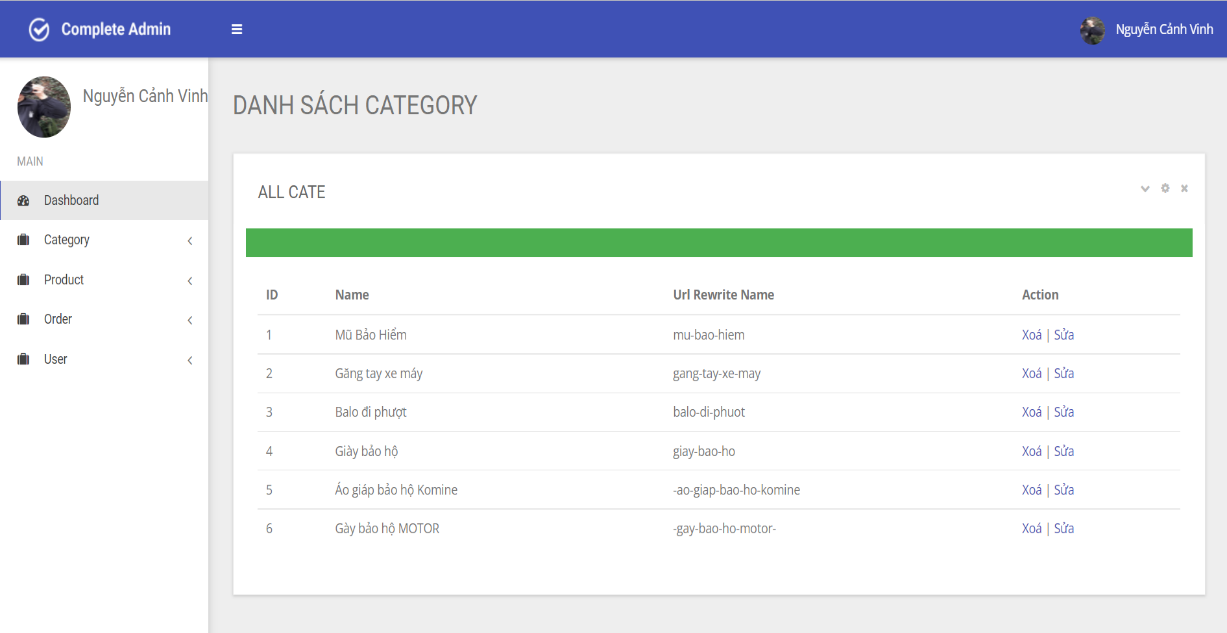
**

*Hình 3.1.6. Giao diện Đăng nhập/ Đăng ký*

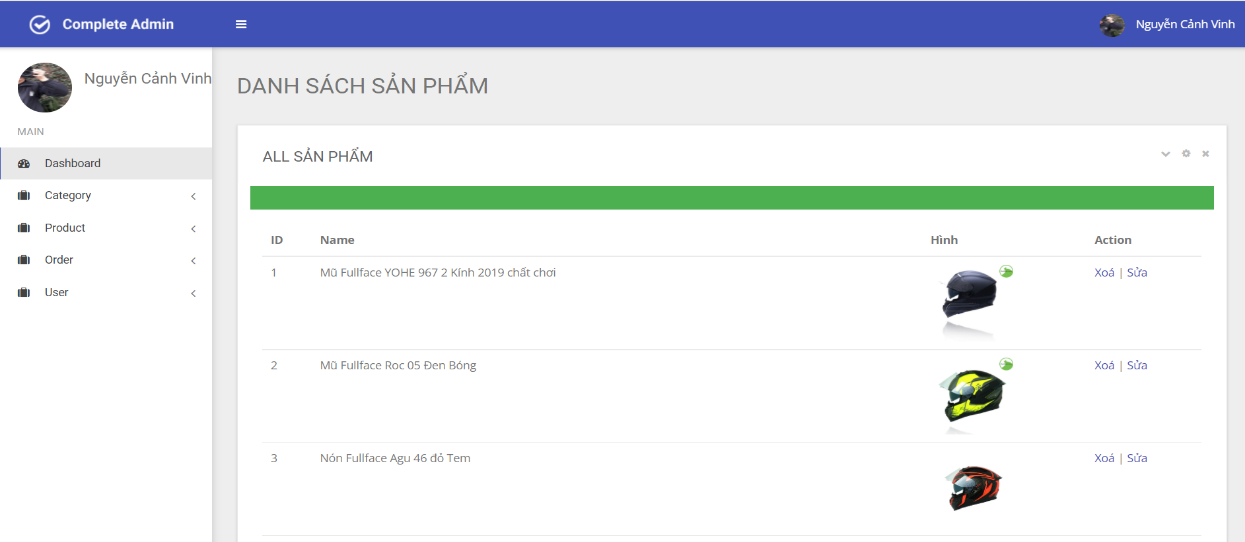
## 3.2. Giao diện quản lý admin

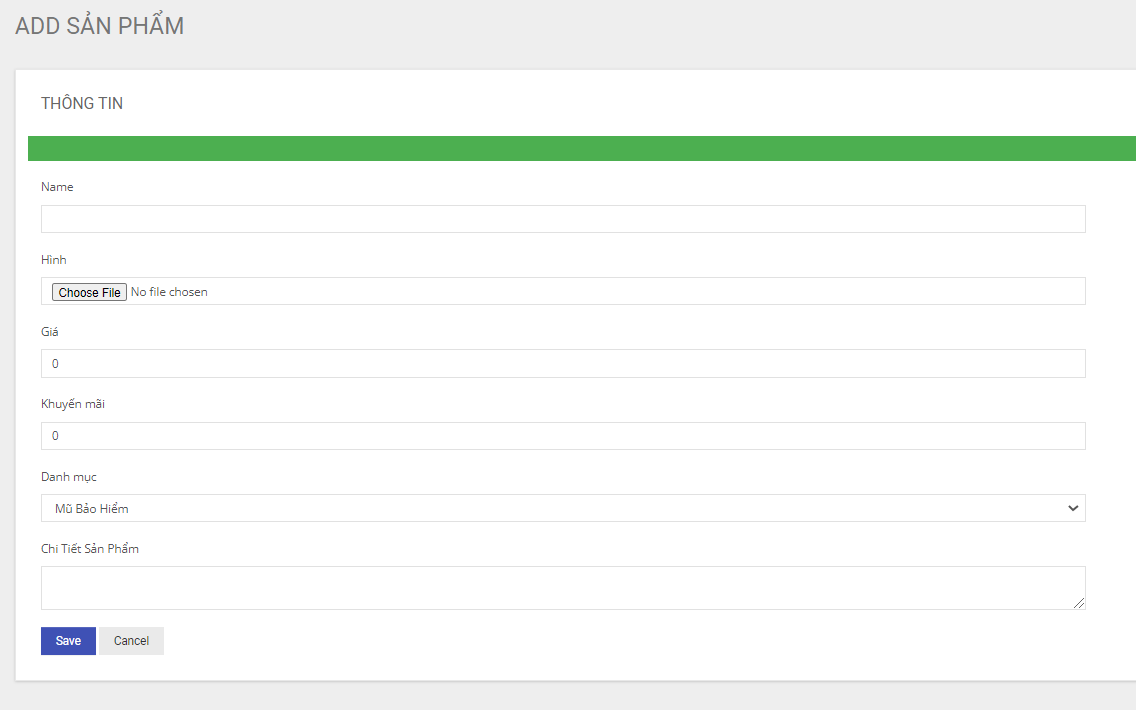


*Hình 3.2.1. Giao diện admin*

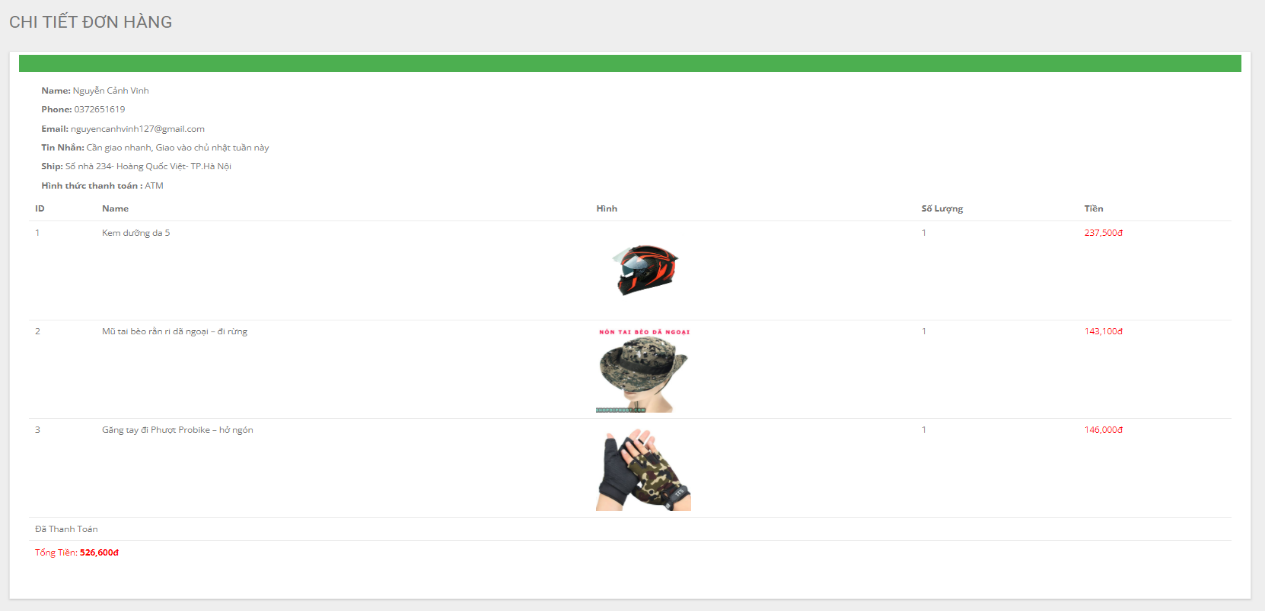
**

*Hình 3.2.2. Giao diện danh mục sản phẩm.*

**

**

*Hình 3.2.3. Giao diện sản phẩm*

**

*Hình 3.2.4. Giao diện order*

# TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN

## 1. Tổng kết

Hệ thống Website bán đồ Phượt được xây dựng nhằm mục đích giúp doanh nghiệp vừa và nhỏ có thể: Quản lý, vận hành và thống kê hoạt động kinh doanh các mặt hàng thời trang dành riêng cho đồ Phượt. Tạo ra nơi cung cấp các mặt hàng mới nhất trên thị trường dành cho dân Phượt chính thống, đồng thời việc tìm kiếm, thanh toán nhanh gọn.

Trong quá trình nghiên cứu và hoàn thiện dự án Website bán đồ Phượt thì nhóm đã nỗ lực hoàn thiện hầu hết những chức năng cơ bản như: đẩy dữ liệu từ database, tìm kiếm, đăng ký, đăng nhập, giỏ hàng, quản lý sản phẩm...

Thông qua đề tài, nhóm em đạt được nhiều mục tiêu đặt ra và bổ sung rất nhiều kiến thức mới.

## 2. Kết quả đạt được

- Phân tích và thiết kế Database.

- Nắm được căn bản một ngôn ngữ lập trình.

- Xây dựng được cơ sở dữ liệu và giao diện cho chương trình bằng hệ quản trị cơ sở dữ liệu Mongo DB

- Quản lý được số lượng sản phẩm, chi tiết sản phẩm

- Các chức năng đăng nhập, đăng ký, gior hàng, quản lý sản phẩm

## 3. Hạn chế

Chương trình không hoạt động được trong trình trạng không có internet.

Thiếu kinh nghiệm trong việc phát triển dự án nên gặp nhiều vấn đề về cách tổ chức dự án và thời gian thực hiện.

## 4. Hướng phát triển

Chỉnh sửa giao diện đẹp mắt, trực quan, tối ưu hóa hiển thị và độ mượt mà của các trang người dùng.

Nâng cấp thêm chức năng thanh toán thông qua ngân hàng, ví điện tử.

Đăng nhập Website bằng tài khoản Google, Facebook…

Xây dựng và hoàn thiện chức năng trao đổi trực tiếp với khách hàng bằng chatbox.

# TÀI LIỆU THAM KHẢO

[1]. Website tài liệu về NodeJS: <https://www.nodebeginner.org/index-vi.html> .

[2]. Website [W3Schools Online Web Tutorials](https://www.w3schools.com/).