**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**TRUNG TÂM ĐÀO TẠO SAU ĐẠI HỌC**

****

**TIỂU LUẬN**

**HỌC PHẦN: CƠ SỞ DỮ LIỆU NÂNG CAO**

**ĐỀ TÀI: ỨNG DỤNG CSDL MONGODB VÀO XÂY DỰNG TRANG WEB BÁN NỘI THẤT**

**Giảng viên hướng dẫn:** TS. Phạm Văn Hà

**Mã lớp độc lập:** 20231IT6077001.

**Nhóm:** 5.

**Sinh viên thực hiện:** 1. Nguyễn Viết An-2024700086

2. Bùi Văn Bắc-2024700074

3. Phan Tiến Phương-2024700146

4. Trịnh Sỹ Tới-2024700127

***Hà Nội - 2024***

**Lời cảm ơn**

Đầu tiên, chúng em xin gửi lời cảm ơn đến quý thầy cô, nhà trường và bạn bè đã quan tâm và động viên chúng em trong suốt thời gian qua. Sự quan tâm và sự hỗ trợ của quý thầy cô không chỉ là nguồn động lực mà còn là một phần quan trọng trong việc xác định hướng đi và hoàn thiện đề tài của chúng em.

Chúng em cũng muốn bày tỏ lòng biết ơn đặc biệt đến thầy TS. Phạm Văn Hà vì sự tận tâm và sự chỉ bảo tận tình trong quá trình hướng dẫn chúng em. Những kiến thức mà thầy đã truyền đạt và những gợi ý quý báu đã giúp chúng em hiểu sâu hơn về môn học. Sự kiên nhẫn và sự đồng hành của thầy đã giúp chúng em vượt qua những khó khăn và hoàn thành đề tài một cách tốt nhất.

Đồng thời chúng em cũng xin gửi lời cảm ơn đến Trung Tâm Đào Tạo Sau Đại Học của trường Đại Học Công Nghiệp Hà Nội đã tạo điều kiện cho chúng em có được một môi trường học tập, rèn luyện và nghiên cứu những kiến thức về chuyên môn.

Cuối cùng, chúng em xin kết thúc lời cảm ơn này bằng việc cam kết sẽ tiếp tục nỗ lực, phấn đấu và trưởng thành hơn trong con đường nghiên cứu và học tập. Chúng em sẽ luôn đặt lòng biết ơn và trân trọng những sự giúp đỡ mà chúng em đã nhận được và sẽ cố gắng chia sẻ kiến thức và kinh nghiệm của mình để đóng góp vào sự phát triển của cộng đồng.

Chúng em xin chân thành cảm ơn!

***Học Viên***

Nguyễn Viết An.

Bùi Văn Bắc.

Trịnh Sỹ Tới.

Phan Tiến Phương.

**Mục lục**

[GIỚI THIỆU 2](#_Toc175244767)

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 3](#_Toc175244768)

[1.1. Giới thiệu về đề tài 3](#_Toc175244769)

[1.2. Lý do chọn đề tài 3](#_Toc175244770)

[1.3. NodeJS 3](#_Toc175244771)

[1.3.1. Giới thiệu 3](#_Toc175244772)

[1.3.2. Những ứng dụng nên viết bằng NodeJS 4](#_Toc175244773)

[1.3.3. Cài đặt NodeJS 4](#_Toc175244774)

[1.3.4. Khởi tạo dự án 5](#_Toc175244775)

[1.4. MongoDB 5](#_Toc175244776)

[1.4.1. Giới thiệu MongoDB 5](#_Toc175244777)

[1.4.2. Một số câu lệnh cơ bản trên MongoDB 6](#_Toc175244778)

[1.4.3. Ưu điểm của MongoDB 7](#_Toc175244779)

[1.4.4. Nhược điểm của MongoDB 8](#_Toc175244780)

[1.4.5. Các ứng dụng cần MongoDB 8](#_Toc175244781)

[1.4.6. Cài đặt MongoDB 8](#_Toc175244782)

[1.5. ExpressJS 11](#_Toc175244783)

[1.5.1. Giới thiệu ExpressJS 11](#_Toc175244784)

[1.5.2. Cài đặt ExpressJS 11](#_Toc175244785)

[1.6. Restful API 12](#_Toc175244786)

[1.6.1. Giới thiệu Restful API 12](#_Toc175244787)

[1.6.2. Đặc điểm của Restful API 12](#_Toc175244788)

[1.7. Mô hình MVC 12](#_Toc175244789)

[1.8. GitHub 13](#_Toc175244790)

[1.8.1. Giới thiệu GitHub 13](#_Toc175244791)

[1.8.2. Tính năng của GitHub 14](#_Toc175244792)

[CHƯƠNG 2: KHẢO SÁT HỆ THỐNG 15](#_Toc175244793)

[2.1. Phát biểu bài toán 15](#_Toc175244794)

[2.2. Xác định yêu cầu hệ thống 15](#_Toc175244795)

[2.2.1. Yêu cầu phi chức năng: 15](#_Toc175244796)

[2.2.2. Yêu cầu chức năng: 15](#_Toc175244797)

[2.3. Xác định các tác nhân, các usecase sử dụng và biểu đồ usecase 15](#_Toc175244798)

[2.3.1. Các tác nhân 15](#_Toc175244799)

[2.3.2. Các usecase sử dụng 15](#_Toc175244800)

[2.4. Biểu đồ các usecase 17](#_Toc175244801)

[2.4.1. Biểu đồ usecase tổng quát 17](#_Toc175244802)

[2.4.2. Biểu đồ usecase đăng nhập 18](#_Toc175244803)

[2.4.3. Biểu đồ usecase đăng ký 19](#_Toc175244804)

[2.4.4. Biểu đồ usecase xem nội thất 21](#_Toc175244805)

[2.4.5. Biểu đồ usecase đặt nội thất 22](#_Toc175244806)

[2.4.6. Biểu đồ usecase quản lý danh mục 25](#_Toc175244807)

[2.4.7. Biểu đồ usecase quản lý loại 27](#_Toc175244808)

[2.4.8. Biểu đồ usecase quản lý nội thất 29](#_Toc175244809)

[2.4.9. Biểu đồ usecase quản lý đơn hàng 31](#_Toc175244810)

[2.5. Xây dựng cơ sở dữ liệu 32](#_Toc175244811)

[2.5.1. Bảng “Admins” 32](#_Toc175244812)

[2.5.2. Bảng “Customers” 33](#_Toc175244813)

[2.5.3. Bảng “Types” 33](#_Toc175244814)

[2.5.4. Bảng “Suppliers” 33](#_Toc175244815)

[2.5.5. Bảng “Products” 33](#_Toc175244816)

[2.5.6. Bảng “Bills” 34](#_Toc175244817)

[CHƯƠNG 3: XÂY DỰNG SẢN PHẨM 35](#_Toc175244818)

[3.1. Cấu hình dự án 35](#_Toc175244819)

[3.2. Cấu hình MongoDB 36](#_Toc175244820)

[3.3. Code backend theo chuẩn cấu trúc MVC 37](#_Toc175244821)

[3.3.1. Models 37](#_Toc175244822)

[3.3.2. Controllers 38](#_Toc175244823)

[3.3.3. Routers 39](#_Toc175244824)

[3.4. Code frontend 40](#_Toc175244825)

[3.5. Code file root server 42](#_Toc175244826)

[3.6. Giao diện sản phẩm 44](#_Toc175244827)

[KẾT LUẬN 52](#_Toc175244828)

[1. Kết quả đạt được 52](#_Toc175244829)

[2. Mặt hạn chế 52](#_Toc175244830)

[TÀI LIỆU THAM KHẢO 53](#_Toc175244831)

# GIỚI THIỆU

Hiện nay, công nghệ thông tin được xem là một ngành mũi nhọn của các quốc gia, đặc biệt là các quốc gia đang phát triển, tiến hành công nghiệp hóa và hiện đại hoá như nước ta. Sự bùng nổ thông tin và sự phát triển mạnh mẽ của công nghệ kỹ thuật số, yêu cầu muốn phát triển thì phải tin học hoá tất cả các ngành, các lĩnh vực.

Cùng với sự phát triển nhanh chóng về phần cứng máy tính, các phần mềm ngày càng trở nên đa dạng, phong phú, hoàn thiện hơn và hỗ trợ hiệu quả cho con người. Các phần mềm hiện nay ngày càng mô phỏng được rất nhiều nghiệp vụ khó khăn, hỗ trợ cho người dùng thuận tiện sử dụng, thời gian xử lý nhanh chóng, và một số nghiệp vụ được tự động hoá cao.

Do vậy mà trong việc phát triển phần mềm, sự đòi hỏi không chỉ là sự chính xác, xử lý được nhiều nghiệp vụ thực tế mà còn phải đáp ứng các yêu cầu khác như về tốc độ, giao diện thân thiện, mô hình hoá được thực tế vào máy tính để người sử dụng tiện lợi, quen thuộc, tính tương thích cao, bảo mật cao (đối với các dữ liệu nhạy cảm),… Các phần mềm, trang web giúp tiết kiệm một lượng lớn thời gian, công sức của con người, và tăng độ chính xác và hiệu quả trong công việc (nhất là việc sửa lỗi và tự động đồng bộ hoá).

Kết hợp với sự nâng cấp của engine JavaScript phiên bản v8, và platform NodeJS do Google phát triển đã tạo ra bước đột phá cho phép tạo ra nhiều cách cải tiến hiệu năng hệ thống phần mềm.

Vì vậy, nhóm chúng em đã chọn thực hiện đề tài **“Ứng dụng CSDL MongoDB vào xây dựng trang web bán nội thất”** nhằm nâng cao kiến thức và tầm hiểu biết của mình về lĩnh vực thiết kế phần mềm, website nói riêng và cũng như công nghệ thông tin nói chung.

# CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

## Giới thiệu về đề tài

Chủ đề của đề tài là **“Ứng dụng CSDL MongoDB vào xây dựng trang web bán nội thất”**. Với mục tiêu áp dụng những kiến thức về thiết kế web, phân tích và đặc tả yêu cầu phần mềm để thực hành xây dựng một trang web bán nội thất. Qua đó biết được cách ứng dụng các framework vào sản phẩm thực tế, đồng thời rèn luyện kĩ năng làm việc nhóm. Ngoài ra do tình hình Covid-19 xảy ra làm thay đổi xu hướng tiêu dùng khiến cho việc sử dụng Internet để mua sắm online bùng nổ. Website bán nội thất cho phép khách hàng truy cập để tiến hành xem các loại nội thất với những thông tin đầy đủ và tiến hành đặt hàng một cách thuận tiện.

## Lý do chọn đề tài

Trong vài năm trở lại đây, với sự “lên ngôi” của thương mại điện tử, thị trường bán lẻ tiêu dùng đã chứng kiến cuộc đua của các doanh nghiệp trong việc đầu tư vào công nghệ và phát triển kênh bán hàng online. Các web bán hàng xuất hiện rất nhiều trên thị trường và ngày càng phát triển mạnh mẽ. Khi lựa chọn để tài này, chúng em muốn tiếp xúc sâu hơn về lập trình web và cụ thể là các web bán hàng - một nền tảng đang cực kỳ lớn mạnh, giúp người dùng có thể mua sắm trực tuyến thông qua internet. Lựa chọn một sản phẩm dựa trên nền tảng NodeJS, chúng em sẽ có thể thử sức tạo nên một website thực tế giúp nâng cao một phần kinh nghiệm của bản thân về kiến thức cũng như kỹ năng lập trình. Có cơ hội làm việc với một số ứng dụng hữu ích.

## NodeJS

### Giới thiệu

**NodeJS** là một platform xây dựng trên Chrome Javascript Runtime (V8) nhằm phát triển các ứng dụng phía máy chủ nhanh chóng và dễ mở rộng. Nền tảng này được phát triển bởi Ryan Dahl vào năm 2009, được xem là một giải pháp hoàn hảo cho các ứng dụng sử dụng nhiều dữ liệu nhờ vào mô hình hướng sự kiện (event-driven) không đồng bộ.

### Những ứng dụng nên viết bằng NodeJS

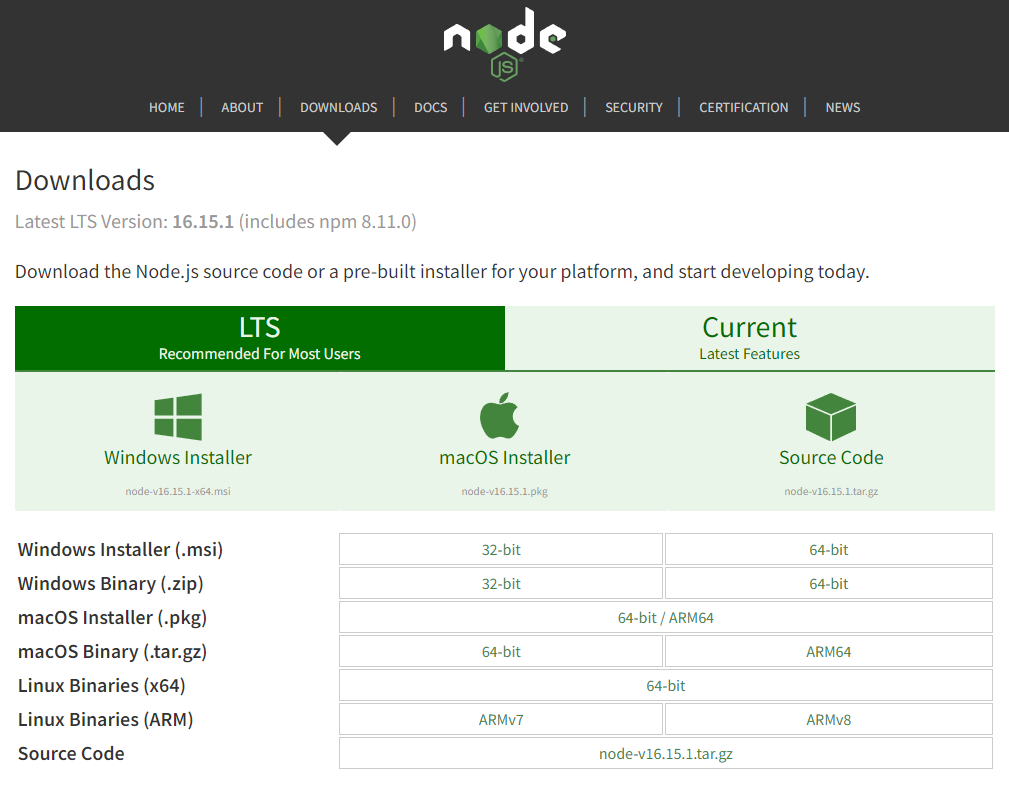
Rõ ràng, không phải cứ hot và mới là NodeJS làm gì cũng tốt, ví dụ như một ứng dụng cần tính ổn định cao, logic phức tạp thì các ngôn ngữ PHP hay Ruby… vẫn là sự lựa chọn tốt hơn. Còn dưới đây là những ứng dụng có thể và nên viết bằng NodeJS:

* **Websocket server:** Các máy chủ web socket như là Online Chat, Game Server…
* **Fast File Upload Client:** Các chương trình upload file tốc độ cao.
* **Ad Server:** Các máy chủ quảng cáo.
* **Cloud Services:** Các dịch vụ đám mây.
* **RESTful API:** Những ứng dụng mà được sử dụng cho các ứng dụng khác thông qua API.
* **Any Real-time Data Application:** Bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực.
* **Micro Services:** Ý tưởng của micro services là chia nhỏ một ứng dụng lớn thành các dịch vụ nhỏ và kết nối chúng lại với nhau. NodeJS có thể làm tốt điều này.

### Cài đặt NodeJS

Để download NodeJS, cần truy cập vào địa chỉ dưới đây:

[https://NodeJS.org/en/download/](https://nodejs.org/en/download/)



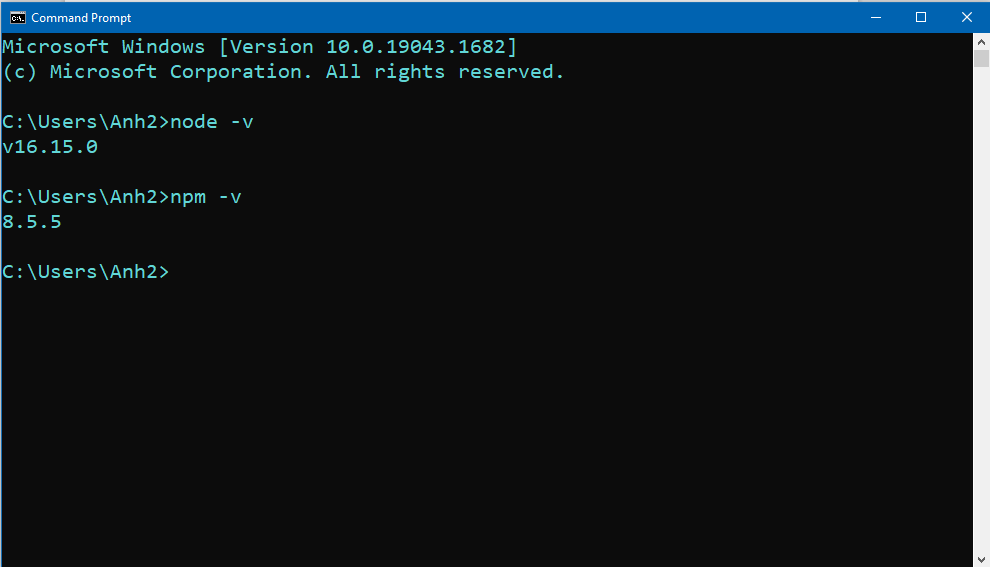
*Hình 1.1. Trang chủ NodeJS*

Sau khi download thành công, mở file vừa download về và tiến hành cài đặt:

Cài đặt NodeJS trên Windows rất đơn giản, chấp nhận các tùy chọn mặc định và nhấn "Next … Next" cho tới bước cuối cùng.

Theo mặc định, phần mềm NPM cũng được cài đặt vào hệ thống của bạn. Đây là một phần mềm quản lý các thư viện Javascript.

Kiểm tra và cấu hình: (Command Prompt với Windows)



*Hình 1.2. Kiểm tra cấu hình*

### Khởi tạo dự án

Khởi tạo bằng câu lệnh: *$* *npm init*

NodeJS sẽ tạo ra một file package.json. Tệp tin này ghi cấu hình cần thiết cho ứng dụng để có thể chạy được trên server, các thông tin tối thiểu như phiên bản, tên, description… Mọi thông tin trong package.json được viết dưới định dạng JSON.

## MongoDB

### Giới thiệu MongoDB

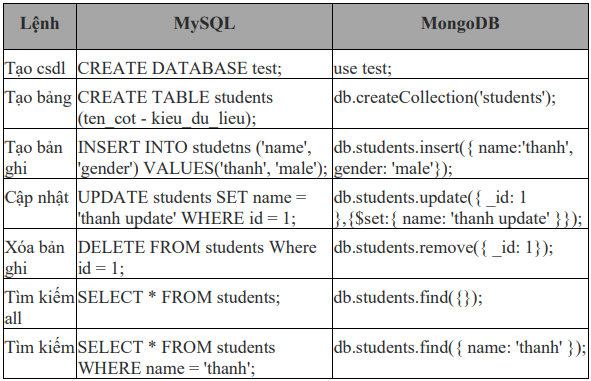
MongoDB là hệ CSDL mã nguồn mở, là CSDL phi quan hệ hay còn gọi là NoSQL (None-Relationship SQL hay còn gọi là Not only SQL). NoSQL được phát triển trên Javascript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value. NoSQL ra đời như là sự bổ sung cho những khuyết điểm và thiếu xót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS (Relational Database Management System - Hệ quản trị cơ sở dữ liệu quan hệ) về tốc độ, tính năng, khả năng mở rộng. Với NoSQL bạn có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc. NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng. NoSQL được sử dụng ở rất nhiều công ty, tập đoàn lớn. Ví dụ như FaceBook sử dụng Cassandra do FaceBook phát triển, Google phát triển và sử dụng BigTable. MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.

Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là collection thay vì bảng.

Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định. Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB.

### Một số câu lệnh cơ bản trên MongoDB

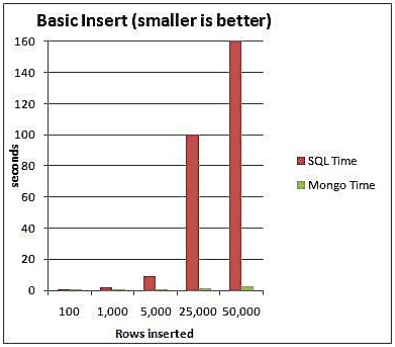
Mục này so sánh một số câu lệnh giữa MongoDB và MySQL để làm nổi bật cách xử lý dữ liệu của MongoDB:



*Hình 1.3. Một số câu lệnh cơ bản MongoDB*

### Ưu điểm của MongoDB

* 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, linh hoạt trong việc lưu trữ dữ liệu, do đó việc chèn thêm dữ liệu vào không bị hạn chế.
* Dữ liệu trong MongoDB không bị ràng buộc như cơ sở dữ liệu quan hệ, không kết nối. Do đó, khi bổ sung, xóa hay cập nhật sẽ không bị mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.
* MongoDB rất dễ mở rộng. Trong MongoDB có một khái niệm cụm dữ liệu (cluster), là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster.
* Trường dữ liệu “\_id” luôn được tự động đánh chỉ mục để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
* Khi có một truy vấn dữ liệu, các bản ghi được ghi tạm thời lên bộ nhớ Ram, để phục vụ các truy vấn tiếp theo, vì vậy sự xử lý cũng diễn ra nhanh hơn mà không cần phải đọc lại từ ổ cứng.
* Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn khi thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.



*Hình 1.4. So sánh thời gian chèn dữ liệu giữa MongoDB và SQL*

### Nhược điểm của MongoDB

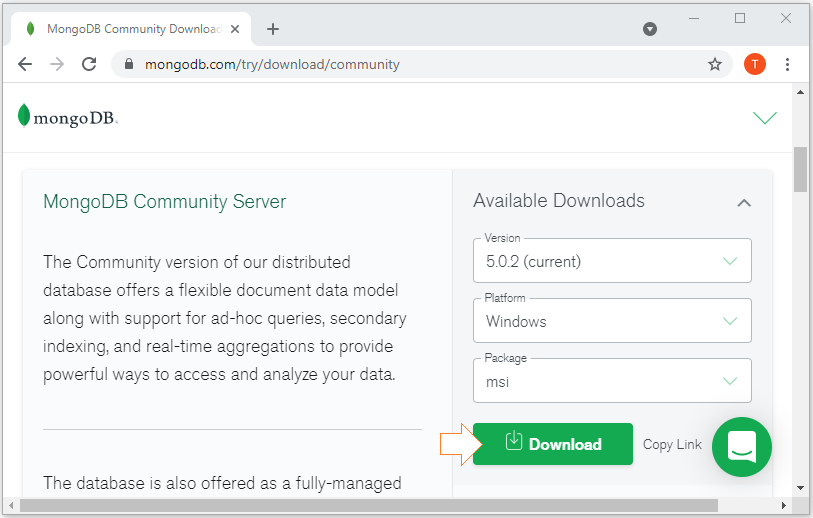
Hệ quản trị cơ sở dữ liệu MongoDB có nhiều ưu điểm, tuy nhiên cũng có một số nhược điểm. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với MongoDB thì chuyên gia phải tự xử lý các mối quan hệ giữa các dữ liệu. Bộ nhớ lưu trữ bị tăng, do dữ liệu lưu dưới dạng key-value, các bộ dữ liệu chỉ khác về giá trị do đó các Khóa sẽ bị lặp lại. MongoDB cũng không hỗ trợ liên kết nên dữ liệu bị dư thừa. Một điều nữa, khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau một khoảng thời gian 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống thiết bị lưu trữ, điều này sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện.

### Các ứng dụng cần MongoDB

Dù MongoDB có nhiều ưu điểm, nhưng tùy vào quy mô hệ thống để lựa chọn. Ví dụ như các hệ thống yêu cầu phản hồi nhanh, Các hệ thống dữ liệu lớn với yêu cầu truy vấn nhanh hay các hệ thống có lượng request lớn thì MongoDB sẽ là sự lựa chọn ưu tiên hơn CSDL quan hệ. Tùy theo dự án và trường hợp cụ thể để sử dụng CSDL quan hệ hay sử dụng MongoDB đem lại hiệu quả cao.

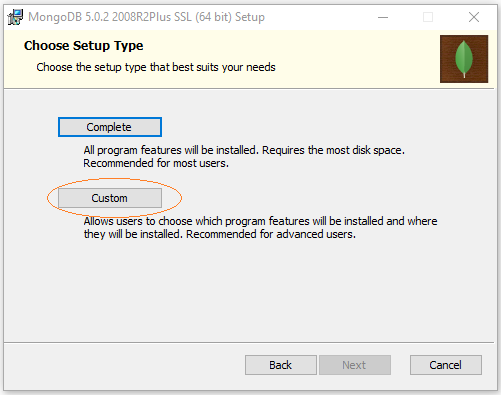
### Cài đặt MongoDB

Truy cập: <https://www.mongodb.com/try/download/community>

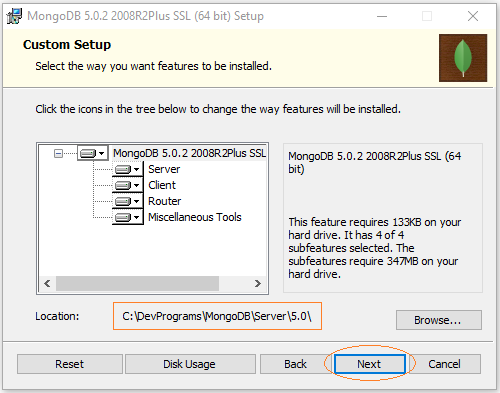


*Hình 1.5. Trang download MongoDB*

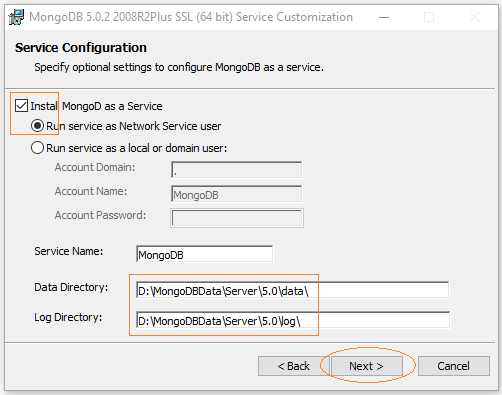
Có thể download file cài đặt (msi hoặc zip), trong trường hợp download file zip, chỉ cần giải nén ra không cần cài đặt gì, sau đó bắt đầu cấu hình theo các bước sau:



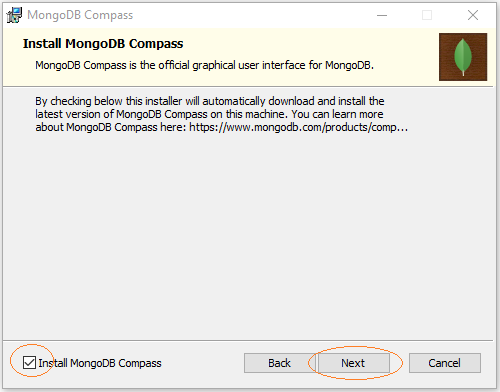
*Hình 1.6. Lựa chọn Custom*



*Hình 1.7. Lựa chọn thư mục cài đặt*

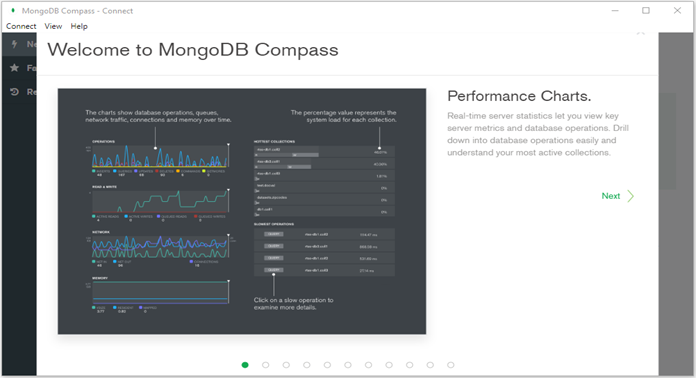


*Hình 1.8. Lựa chọn thư mục để lưu trữ dữ liệu và log cho MongoDB*



*Hình 1.8. Lựa chọn cài đặt MongoDB Compass*

Nhấn “Next...”, sau đó khi Finish ở bước cuối cùng



*Hình 1.9. Giao diện MongoDB Compass*

Khởi động lại máy đề hoàn thành việc cài đặt

## ExpressJS

### Giới thiệu ExpressJS

**ExpressJS** là một framework dành cho NodeJS. Nó cung cấp cho chúng ta rất nhiều tính năng mạnh mẽ trên nền tảng web cũng như trên các ứng dụng di động. ExpressJS hỗ trợ các phương thức HTTP và middleware tạo ra một API vô cùng mạnh mẽ và dễ sử dụng. Có thể tổng hợp một số chức năng chính của ExpressJS như sau:

* Thiết lập các lớp trung gian để trả về các HTTP request
* Định nghĩa router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL
* Cho phép trả về các trang HTML dựa vào các tham số.

### Cài đặt ExpressJS

Để cài đặt ExpressJS, vào Terminal và gõ lệnh sau:

*$ mkdir express*

*$ cd express*

*$ npm install express --save*

Với câu lệnh trên sẽ lưu phần cài đặt trong thư mục **node\_modules** và tạo thư mục **ExpressJS** trong đó. Trong đề tài này, chúng em sẽ cài đặt thêm một số **module** đi cùng **ExpressJS** hỗ trợ việc xây dựng website như sau:

* **connect-flash:** Hỗ trợ mở flash message
* **express-session:** Thao tác với session, như lưu lại thông tin phiên đăng nhập của người dùng, hoặc một số thông tin khác...
* **ejs:** Embedded JavaScript Template Engine, sử dụng JS cơ bản để nhúng vào trong code HTML
* **mkdir:** Hỗ trợ tạo thư mục
* **multer:** Hỗ trợ upload file
* **passport-local:** Hỗ trợ authentication
* **nodemon:** Hỗ trợ tự động reset server trong quá trình dev
* **mongoose:** Hỗ trợ kết nối, thao tác với MongoDB

Để cài đặt các module trên, chạy lệnh sau vào terminal:

*$ npm i connect-flash express-session ejs mkdir multer*

*$ npm i passport-local nodemon mongoose*

## Restful API

### Giới thiệu Restful API

API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.

REST: REST là viết tắt của REpresentational State Transfer. REST là tập hợp các hướng dẫn và kiến trúc sử dụng cho việc truyền dữ liệu. REST áp dụng phổ biến cho các Web app, nhưng cũng hoàn toàn có thể sử dụng cho các phần mềm nói chung. Vì thế RESTful API là các API mà tuân theo các kiến trúc và quy tắc của REST.

### Đặc điểm của Restful API

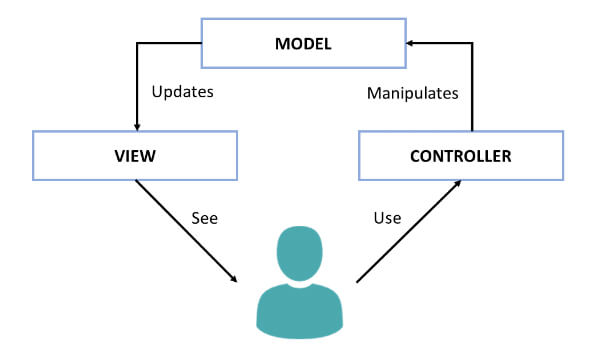
* Nhất quán xuyên suốt các API
* Stateless existence
* Sử dụng HTTP status code khi có thể
* Sử dụng URL Endpoint có phân tầng logic
* Đánh version trong URL thay vì trong HTTP Headers

Những phương thức truy cập phổ biến:

* **GET** để lấy về 1 đối tượng
* **POST** để tạo ra đối tượng mới
* **PUT** để sửa đổi hoặc thay thế đối tượng
* **DELETE** để xoá đi đối tượng
* Tương tự như CRUD: Create, read, update, delete

## Mô hình MVC

**MVC** là viết tắt của cụm từ **“Model-View-Controller”**. Đây là mô hình thiết kế sử dụng trong kỹ thuật phần mềm. MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau như tên gọi: Model (dữ liệu), View (giao diện) và Controller (bộ điều khiển).



*Hình 1.10. Mô hình MVC*

* **Model:** Bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Một model là dữ liệu được sử dụng bởi chương trình, là cơ sở dữ liệu hoặc file XML bình thường hay một đối tượng đơn giản.
* **View:** Đây là phần giao diện dành cho người sử dụng, hiển thị các đối tượng trong một ứng dụng. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.
* **Controller:**Bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View. Một controller bao gồm cả **Model** lẫn **View**. Nó nhận input và thực hiện các update tương ứng.

## GitHub

### Giới thiệu GitHub

**GitHub** là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn Git cho các dự án phần mềm. GitHub có đầy đủ những tính năng của Git, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

* Là công cụ giúp quản lý source code tổ chức theo dạng dữ liệu phân tán.
* Giúp đồng bộ source code của team lên 1 server.
* Hỗ trợ các thao tác kiểm tra source code trong quá trình làm việc (diff, check modifications, show history, merge source, …)

GitHub trở thành một yếu tố có sức ảnh hưởng lớn trong cộng động nguồn mở. Cùng với Linkedin, GitHub được coi là một sự thay thế cho CV của ứng viên. Các nhà tuyển dụng cũng rất hay tham khảo GitHub profile để hiểu về năng lực coding của ứng viên.

Giờ đây, kỹ năng sử dụng git và GitHub từ chỗ ưu thích sang bắt buộc phải có đối với các ứng viên đi xin việc.

### Tính năng của GitHub

**GitHub** được coi là một mạng xã hội dành cho lập trình viên lớn nhất và dễ dùng nhất với các tính năng cốt lõi như:

1. Wiki, issue, thống kê, đổi tên project, project được đặt vào namespace là user.
2. Watch project: theo dõi hoạt động của project của người khác. Xem quá trình người ta phát triển phần mềm thế nào, project phát triển ra sao.
3. Follow user: theo dõi hoạt động của người khác.

Có 2 cách tiếp cận GitHub: Tạo project của riêng mình Contribute cho project có sẵn: fork project có sẵn của người khác, sửa đổi, sau đó đề nghị họ cập nhật sửa đổi của mình (tạo pull request).

# CHƯƠNG 2: KHẢO SÁT HỆ THỐNG

## 2.1. Phát biểu bài toán

Một cửa hàng bán nội thất muốn tăng thêm nguồn thu từ việc bán thêm nội thất đóng hộp cho khách hàng qua website trực tuyến. Người quản trị trang web có thể tùy biến danh mục nội thất, thêm sửa xóa nội thất, chọn nguồn cung cấp. Khách hàng sẽ đăng kí tài khoản và truy cập trang web để tiến hành đặt hàng. Khi nhận được đơn hàng, người quản trị có thể xác nhận đơn hàng và tiến hành giao hàng hoặc hủy đơn hàng.

## 2.2. Xác định yêu cầu hệ thống

Từ dữ liệu thu thập được trong quá trình khảo sát, có thể xác định được hệ thống gồm những chức năng chính sau đây:

### 2.2.1. Yêu cầu phi chức năng:

* Giao diện hài hòa, dễ sử dụng, thân thiện với người dùng.
* Dễ bảo trì.

### 2.2.2. Yêu cầu chức năng:

* Yêu cầu hệ thống quản lý tài khoản, đăng nhập.
* Yêu cầu nghiệp vụ đặt hàng, quản lý danh mục, nội thất, loại, duyệt đơn hàng.

## 2.3. Xác định các tác nhân, các usecase sử dụng và biểu đồ usecase

### 2.3.1. Các tác nhân

* Người quản lý
* Khách hàng

### 2.3.2. Các usecase sử dụng

**Đăng nhập**

|  |  |
| --- | --- |
| UC1 | Đăng nhập |
| 1 | Đăng nhập hệ thống |
| 2 | Thoát khỏi hệ thống |

**Đăng ký**

|  |  |
| --- | --- |
| UC2 | Đăng ký |
| 1 | Đăng ký hệ thống |
| 2 | Thoát khỏi hệ thống |

**Xem nội thất**

|  |  |
| --- | --- |
| UC3 | Xem nội thất |
| 1 | Chọn nội thất |
| 2 | Đặt nội thất |

**Đặt nội thất**

|  |  |
| --- | --- |
| UC4 | Đặt nội thất |
| 1 | Chọn nội thất |
| 2 | Đặt nội thất |

**Quản lý danh mục**

|  |  |
| --- | --- |
| UC5 | Quản lý danh mục |
| 1 | Tạo danh mục |
| 2 | Sửa danh mục |
| 3 | Xóa danh mục |

**Quản lý loại**

|  |  |
| --- | --- |
| UC6 | Quản lý loại |
| 1 | Tạo loại |
| 2 | Sửa loại |
| 3 | Xóa loại |

**Quản lý nội thất**

|  |  |
| --- | --- |
| UC7 | Quản lý nội thất |
| 1 | Tạo nội thất |
| 2 | Sửa nội thất |
| 3 | Xóa nội thất |

**Quản lý đơn hàng**

|  |  |
| --- | --- |
| UC8 | Quản lý đơn hàng |
| 1 | Xác nhận đơn hàng |
| 2 | Thay đổi trạng thái đơn hàng |
| 3 | Hủy đơn hàng |

## 2.4. Biểu đồ các usecase

### 2.4.1. Biểu đồ usecase tổng quát



*Hình 2.1. Biểu đồ usecase tổng quát*

Tên usecase: Xây dựng website bán nội thất

Tên tác nhân: Người quản lý, khách hàng

Chức năng usecase: Biểu đồ này cho thấy các use case sẽ tương tác với nhau thông qua website bán nội thất.

Dòng sự kiện chính:

* Người Quản lý được lựa chọn tất cả các chức năng được phân trên hệ thống
* Khách hàng xem, chọn và đặt những thông tin được phân
* Hệ thống sẽ kiểm tra tính hợp lệ và xác nhận
* Thông tin sẽ được lưu vào cơ sở dữ liệu
* Hệ thống sẽ báo thành công

Dòng sự kiện phụ:

* Nếu thông tin nhập vào không đúng thì hệ thống sẽ báo là không hợp lệ
* Hệ thống yêu cầu nhập lại thông tin

### 2.4.2. Biểu đồ usecase đăng nhập

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn chức năng đăng nhập |  |
|  | 2. Hiển thị giao diện đăng nhập |
| 3. Nhập thông tin đăng nhập |  |
|  | 4. Kiểm tra thông tin và xác nhận |

Tên usecase: Đăng nhập

Tác nhân: Khách hàng, người quản lý

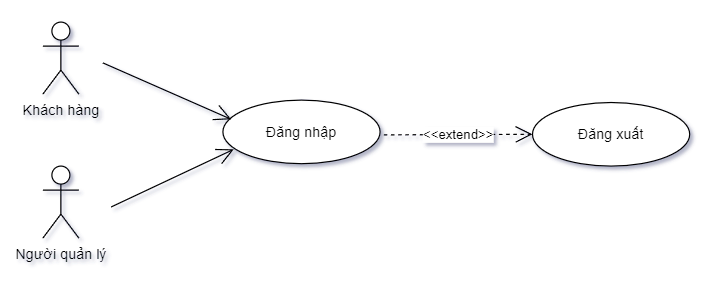
Chức năng usecase: Cho phép tác nhân đăng nhập vào hệ thống

Dòng sự kiện chính:

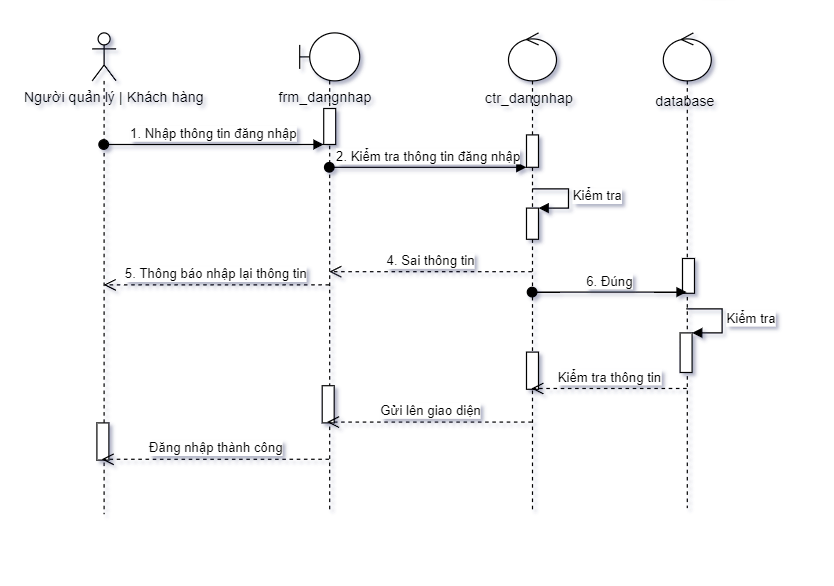
* Tác nhân đăng nhập vào website
* Hệ thống yêu cầu tác nhân đăng nhập
* Tác nhân nhập “Tên đăng nhập” và “Mật khẩu”
* Hệ thống kiểm tra tính hợp lệ thông tin đăng nhập của tác nhân và cho phép tác nhân đăng nhập vào hệ thống
* Nếu tác nhân chưa nhập thông tin đăng nhập mà ấn vào nút đăng nhập, hệ thống sẽ phản hồi lại không hợp lệ

Dòng sự kiện phụ:

* Tác nhân nhập sai tài khoản hoặc mật khẩu thì website sẽ báo lỗi và yêu cầu tác nhân đăng nhập lại, nếu tác nhân không đăng nhập được thì không sử dụng được các chức năng cần đăng nhập trong website



*Hình 2.2. Biểu đồ usecase đăng nhập*



*Hình 2.3. Biểu đồ tuần tự chức năng đăng nhập*

### 2.4.3. Biểu đồ usecase đăng ký

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn chức năng đăng ký |  |
|  | 2. Hiển thị giao diện đăng ký |
| 3. Nhập thông tin đăng ký |  |
|  | 4. Kiểm tra thông tin và xác nhận |

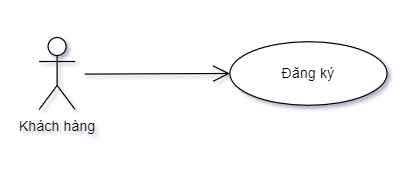
Tên usecase: Đăng ký

Tác nhân: Khách hàng

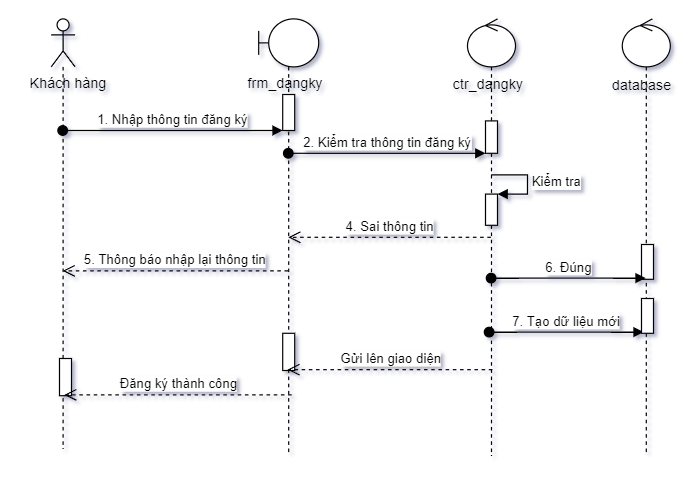
Chức năng usecase: Cho phép khách hàng đăng ký vào hệ thống

Dòng sự kiện chính:

* Khách hàng đăng ký vào website
* Hệ thống yêu cầu khách hàng đăng ký
* Khách hàng nhập “Họ”, “Tên”, “CMND”, “Số điện thoại”, “Email”, “Tên đăng nhập”, “Mật khẩu” và “Nhập lại mật khẩu”
* Hệ thống kiểm tra tính hợp lệ thông tin đăng ký của khách hàng và cho phép khách hàng đăng ký vào hệ thống, thông báo đăng ký thành công.
* Nếu khách hàng chưa nhập thông tin đăng ký mà ấn vào nút đăng ký, hệ thống sẽ phản hồi lại không hợp lệ



*Hình 2.4. Biểu đồ usecase đăng ký*



*Hình 2.5. Biểu đồ tuần tự chức năng đăng ký*

### 2.4.4. Biểu đồ usecase xem nội thất

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn danh mục nội thất |  |
|  | 2. Hiển thị giao diện danh sách nội thất |
| 3. Chọn một nội thất |  |
|  | 4. Hiển thị giao diện chi tiết nội thất được chọn |

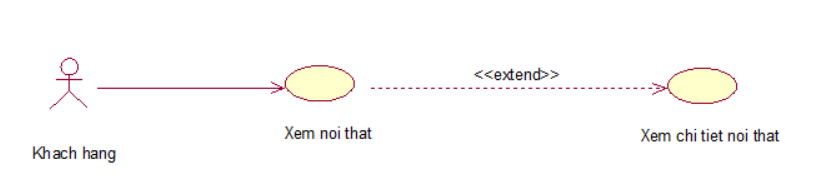
Tên usecase: Xem nội thất.

Tác nhân: Khách hàng.

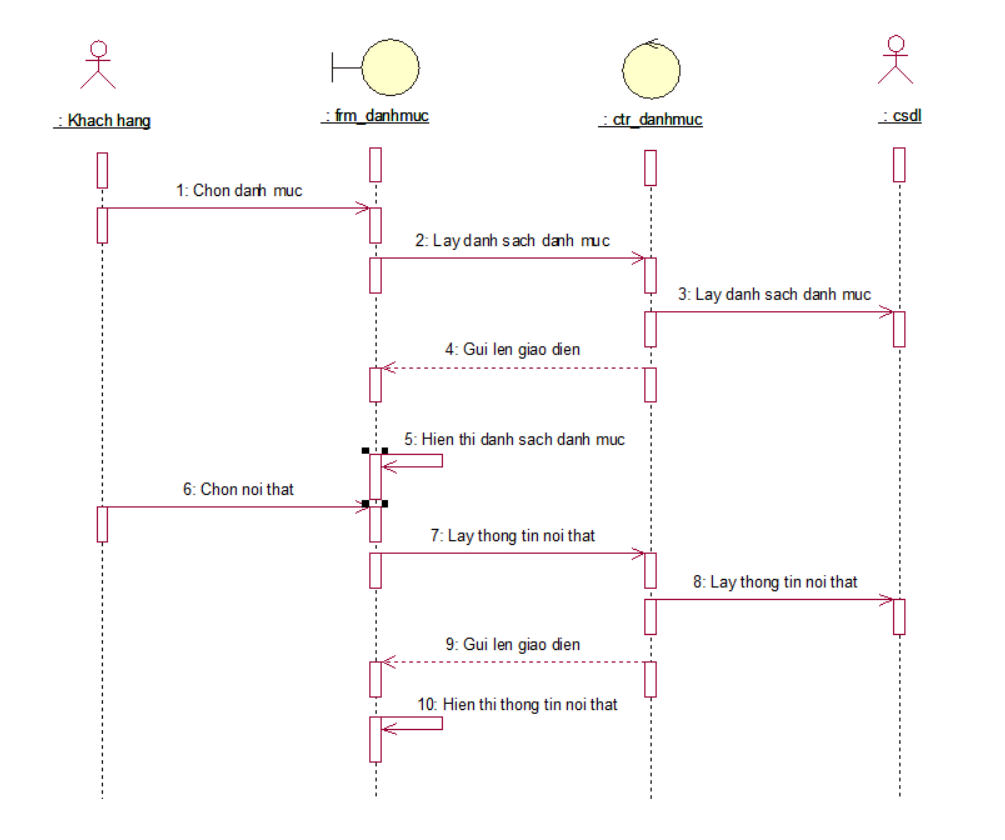
Chức năng usecase: Cho phép khách hàng xem nội thất.

Dòng sự kiện chính:

* Khách hàng chọn danh mục nội thất.
* Hệ thống hiển thị danh sách nội thất.
* Khách hàng chọn một nội thất.
* Hệ thống hiển thị giao diện chi tiết nội thất được chọn.



*Hình 2.6. Biểu đồ usecase xem nội thất*

**

*Hình 2.7. Biểu đồ tuần tự usecase xem nội thất*

### 2.4.5. Biểu đồ usecase đặt nội thất

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn giỏ hàng |  |
|  | 2. Hiển thị giao diện giỏ hàng |
| 3. Chọn đặt nội thất |  |
|  | 4. Hiển thị giao diện nhập thông tin giao hàng |
| 5. Nhập thông tin giao hàng |  |
|  | 6. Kiểm tra thông tin và xác nhận |

Tên usecase: Đặt nội thất.

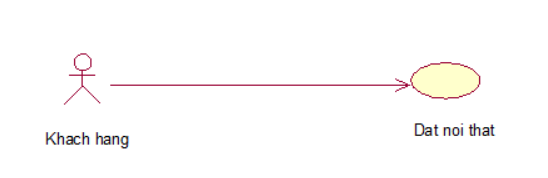
Tác nhân: Khách hàng.

Chức năng usecase: Cho phép khách hàng đặt những nội thất đã chọn.

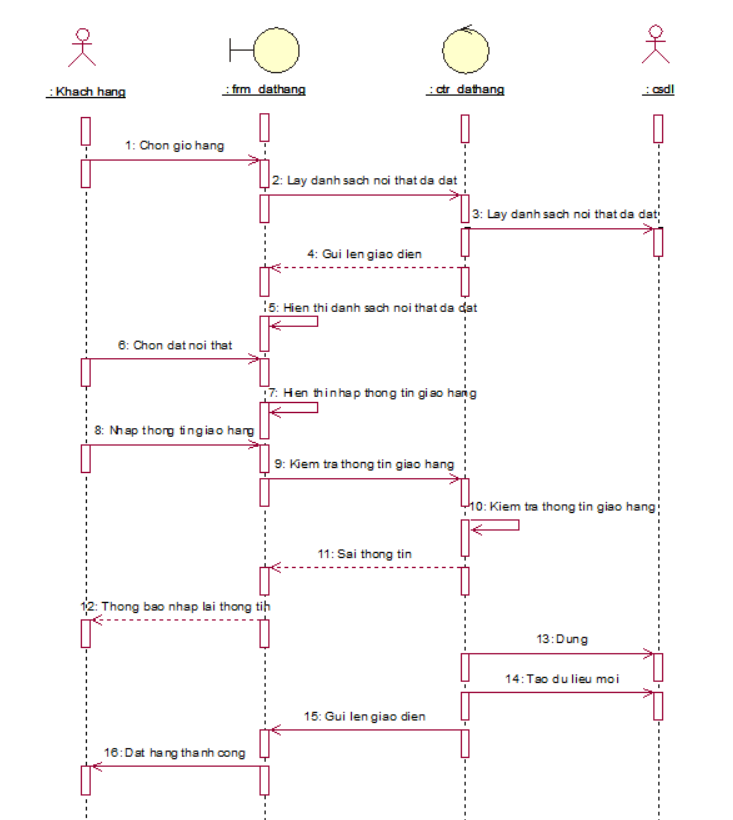
Tiền điều kiện: Khách hàng đã đăng nhập vào hệ thống.

Dòng sự kiện chính:

* Khách hàng chọn giỏ hàng
* Hệ thống hiển thị giao diện giỏ hàng
* Khách hàng chọn đặt nội thất
* Hệ thống yêu cầu nhập thông tin giao hàng
* Khách hàng nhập địa chỉ giao hàng, phương thức thanh toán
* Hệ thống kiểm tra tính hợp lệ thông tin giao hàng của khách hàng và thông báo đăng ký thành công.
* Nếu khách hàng chưa nhập đủ thông tin giao hàng mà ấn vào nút đặt nội thất, hệ thống sẽ phản hồi lại không hợp lệ



*Hình 2.8. Biểu đồ usecase đặt nội thất*



*Hình 2.9. Biểu đồ tuần tự usecase đặt nội thất*

### 2.4.6. Biểu đồ usecase quản lý danh mục

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn quản lý danh mục |  |
|  | 2. Hiển thị danh sách danh mục |
| 3. Thêm mới danh mục |  |
| 4. Sửa danh mục |  |
| 5. Xóa danh mục |  |
|  | 6. Kiểm tra thông tin và xác nhận |

Tên usecase: Quản lý danh mục

Tác nhân: Người quản lý

Chức năng usecase: Cho phép người quản lý thêm, sửa, xóa danh mục

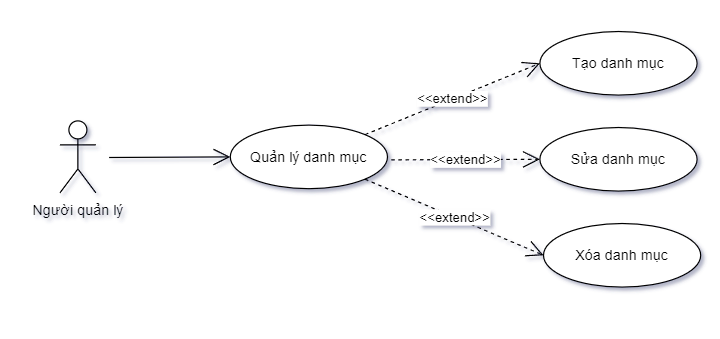
Tiền điều kiện: Người quản lý đã đăng nhập vào trang quản trị website

Dòng sự kiện chính:

* Người quản lý chọn quản lý danh mục
* Hệ thống hiển thị danh sách danh mục
* Người quản lý thêm mới danh mục
* Người quản lý sửa danh mục
* Người quản lý xóa danh mục

Dòng sự kiện thay thế:

* Xóa danh mục đồng nghĩa với việc xóa tất cả thông tin danh mục



*Hình 2.10. Biểu đồ usecase quản lý danh mục*

**

*Hình 2.11. Biểu đồ tuần tự usecase quản lý danh mục*

### 2.4.7. Biểu đồ usecase quản lý loại

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn quản lý loại |  |
|  | 2. Hiển thị danh sách loại |
| 3. Thêm mới loại |  |
| 4. Sửa loại |  |
| 5. Xóa loại |  |
|  | 6. Kiểm tra thông tin và xác nhận |

Tên usecase: Quản lý loại

Tác nhân: Người quản lý

Chức năng usecase: Cho phép người quản lý thêm, sửa, xóa loại

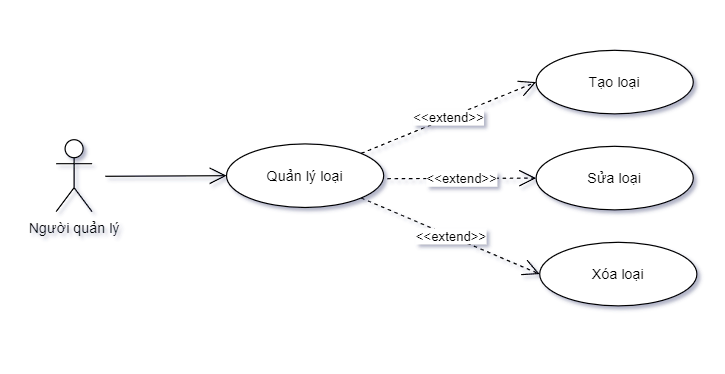
Tiền điều kiện: Người quản lý đã đăng nhập vào trang quản trị website

Dòng sự kiện chính:

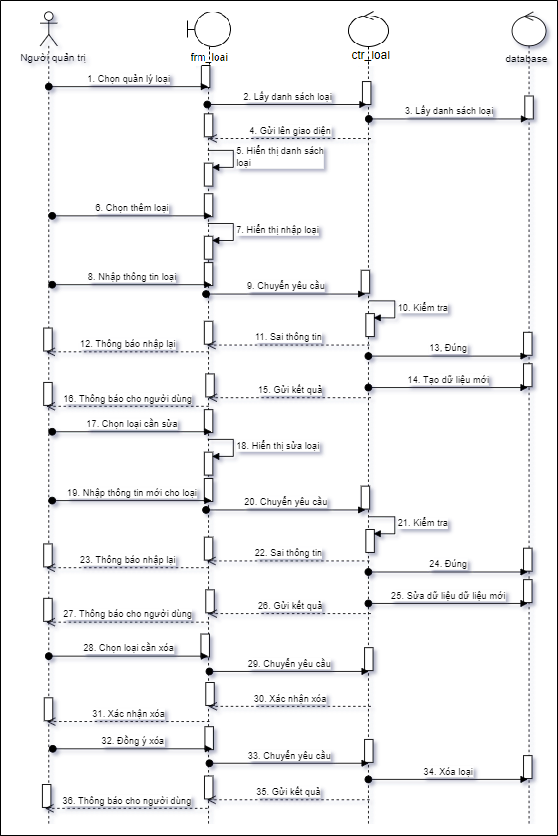
* Người quản lý chọn quản lý loại
* Hệ thống hiển thị danh sách loại
* Người quản lý thêm mới loại
* Người quản lý sửa loại
* Người quản lý xóa loại

Dòng sự kiện thay thế:

* Xóa loại đồng nghĩa với việc xóa tất cả thông tin loại



*Hình 2.12. Biểu đồ usecase quản lý loại*

**

*Hình 2.13. Biểu đồ tuần tự usecase quản lý loại*

### 2.4.8. Biểu đồ usecase quản lý nội thất

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn quản lý nội thất |  |
|  | 2. Hiển thị danh sách nội thất |
| 3. Thêm mới nội thất |  |
| 4. Sửa loại nội thất |  |
| 5. Xóa loại nội thất |  |
|  | 6. Kiểm tra thông tin và xác nhận |

Tên usecase: Quản lý nội thất

Tác nhân: Người quản lý

Chức năng usecase: Cho phép người quản lý thêm, sửa, xóa nội thất

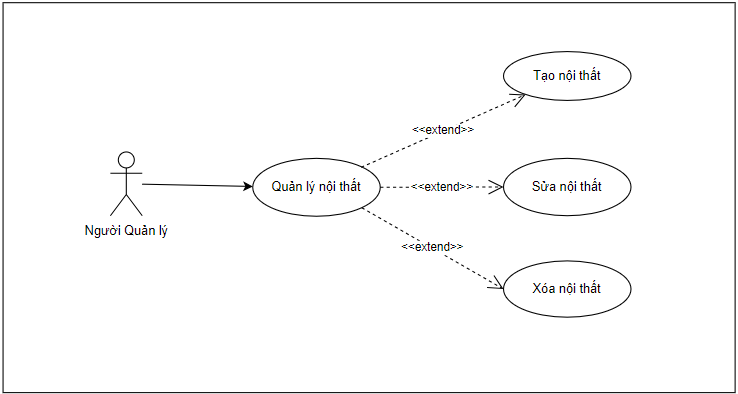
Tiền điều kiện: Người quản lý đã đăng nhập vào trang quản trị website

Dòng sự kiện chính:

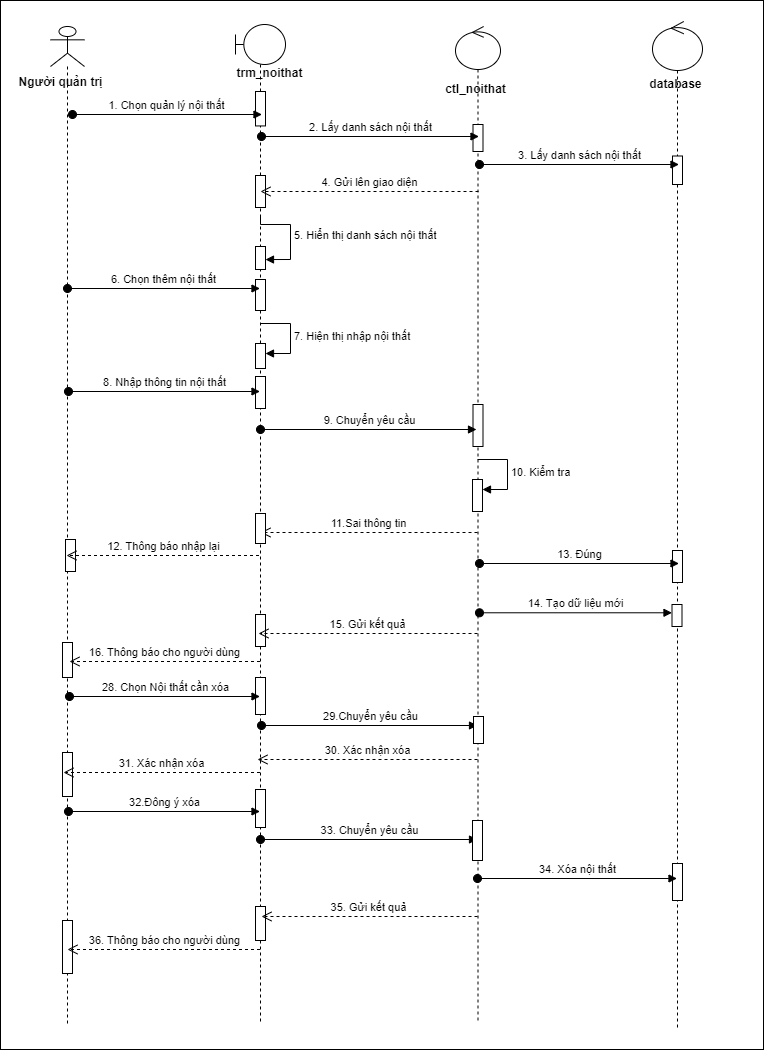
* Người quản lý chọn quản lý nội thất
* Hệ thống hiển thị danh sách nội thất
* Người quản lý thêm mới nội thất
* Người quản lý sửa nội thất
* Người quản lý xóa nội thất

Dòng sự kiện thay thế:

* Xóa nội thất đồng nghĩa với việc xóa tất cả thông tin nội thất



*Hình 2.14. Biểu đồ usecase quản lý nội thất*

**

*Hình 2.15. Biểu đồ tuần tự usecase quản lý nội thất*

### 2.4.9. Biểu đồ usecase quản lý đơn hàng

Đặc tả usecase:

|  |  |
| --- | --- |
| Hoạt động của tác nhân | Phản ứng của hệ thống |
| 1. Chọn quản lý đơn hàng |  |
|  | 2. Hiển thị danh sách đơn hàng |
| 3. Xác nhận đơn hàng |  |
| 4. Thay đổi tình trạng đơn hàng |  |
| 5. Hủy đơn hàng |  |
|  | 6. Kiểm tra thông tin và xác nhận |

Tên usecase: Quản lý đơn hàng

Tác nhân: Người quản lý

Chức năng usecase: Cho phép người quản lý xác nhận, thay đổi tình trạng đơn hàng, hủy đơn hàng

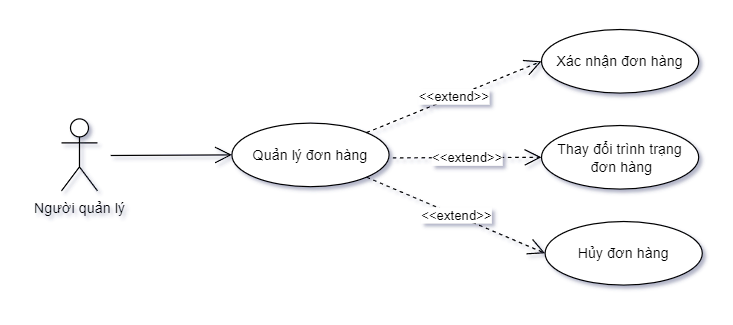
Tiền điều kiện: Người quản lý đã đăng nhập vào trang quản trị website

Dòng sự kiện chính:

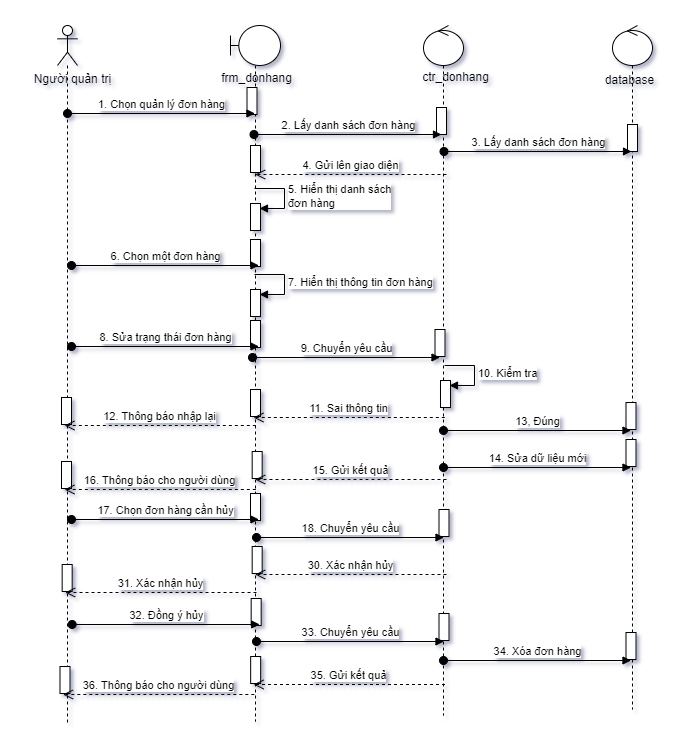
* Người quản lý chọn quản lý đơn hàng
* Hệ thống hiển thị danh sách đơn hàng
* Người quản lý xác nhận đơn hàng
* Người quản lý thay đổi tình trạng đơn hàng
* Người quản lý hủy đơn hàng

Dòng sự kiện thay thế:

* Xóa nội thất đồng nghĩa với việc xóa tất cả thông tin nội thất



*Hình 2.16. Biểu đồ usecase quản lý đơn hàng*

**

*Hình 2.17. Biểu đồ tuần tự usecase quản lý đơn hàng*

## 2.5. Xây dựng cơ sở dữ liệu

### 2.5.1. Bảng “Admins”

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** |
| 1 | ObjectId | ObjectId |
| 2 | fỉrstname | String |
| 3 | lastname | String |
| 4 | identityCardNumber | String |
| 5 | phoneNumber | String |
| 6 | email | String |
| 7 | userName | String |
| 8 | password | String |
| 9 | type | String |

### 2.5.2. Bảng “Customers”

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** |
| 1 | ObjectId | ObjectId |
| 2 | fỉrstname | String |
| 3 | lastname | String |
| 4 | identityCardNumber | String |
| 5 | phoneNumber | String |
| 6 | email | String |
| 7 | userName | String |
| 8 | password | String |
| 9 | type | String |

### 2.5.3. Bảng “Types”

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** |
| 1 | ObjectId | ObjectId |
| 2 | typeName | String |
| 3 | thumbnail | String |

### 2.5.4. Bảng “Suppliers”

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** |
| 1 | ObjectId | ObjectId |
| 2 | supplierName | String |
| 3 | status | Boolean |

### 2.5.5. Bảng “Products”

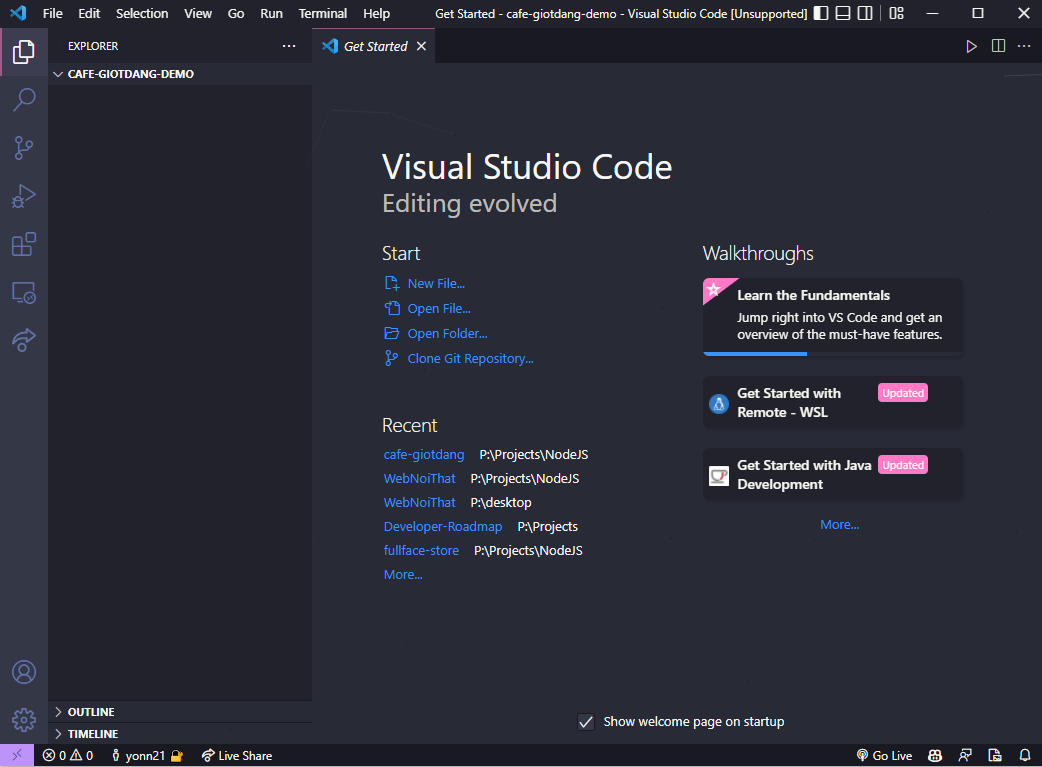
|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** |
| 1 | ObjectId | ObjectId |
| 2 | productName | String |
| 3 | imageList | Arrays[String] |
| 4 | productDescription | String |
| 5 | price | String |
| 6 | unit | String |
| 7 | supplierCode | String |
| 8 | typeCode | String |
| 9 | status | Boolean |

### 2.5.6. Bảng “Bills”

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** |
| 1 | ObjectId | ObjectId |
| 2 | listProduct | Arrays[String] |
| 3 | productID | String |
| 4 | productName | String |
| 5 | productPrice | String |
| 6 | productImage | String |
| 7 | amount | int |
| 8 | userID | String |
| 9 | firstName | String |
| 10 | lastName | String |
| 11 | address | String |
| 12 | paymentMethod | String |
| 13 | resquest | String |
| 14 | status | String |

# CHƯƠNG 3: XÂY DỰNG SẢN PHẨM

## 3.1. Cấu hình dự án



*Hình 3.1. Chọn thư mục dự án trong Visual Studio Code*

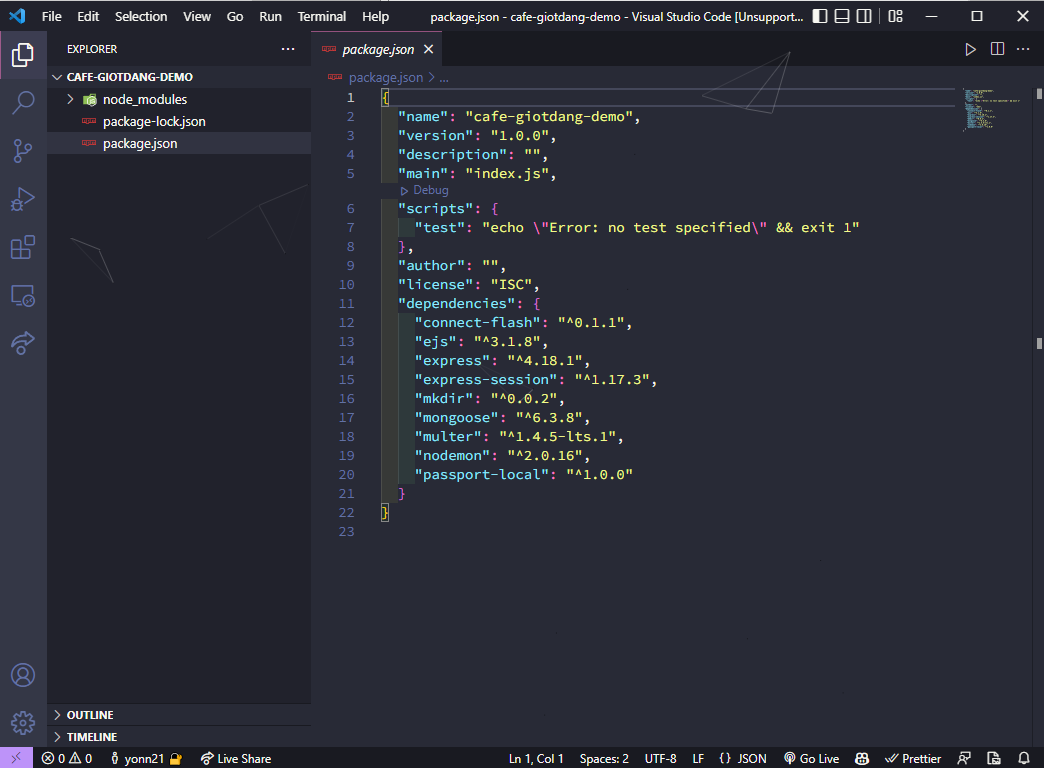
* Mở terminal và nhập lệnh: *$ npm init*
* Nhập những thông tin của dự án như tên dự án, phiên bản, mô tả,...
* Sau khi tạo xong ta có file ***package.json*** lưu những thông tin của dự án
* Tiến hành cài **ExpressJS** cùng các **module** đi cùng:

*$ npm install express*

*$ npm i connect-flash express-session ejs mkdir multer*

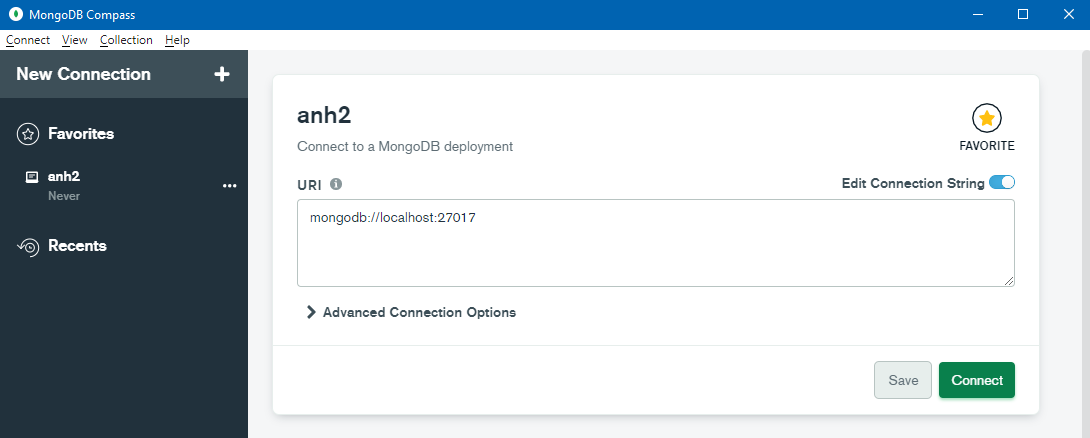
*$ npm i passport-local nodemon mongoose*

* Kết quả

**

*Hình 3.2. Kết quả*

## 3.2. Cấu hình MongoDB



*Hình 3.3. Giao diện MongoDB Compass*

* Connect URI: mongodb://localhost:27017

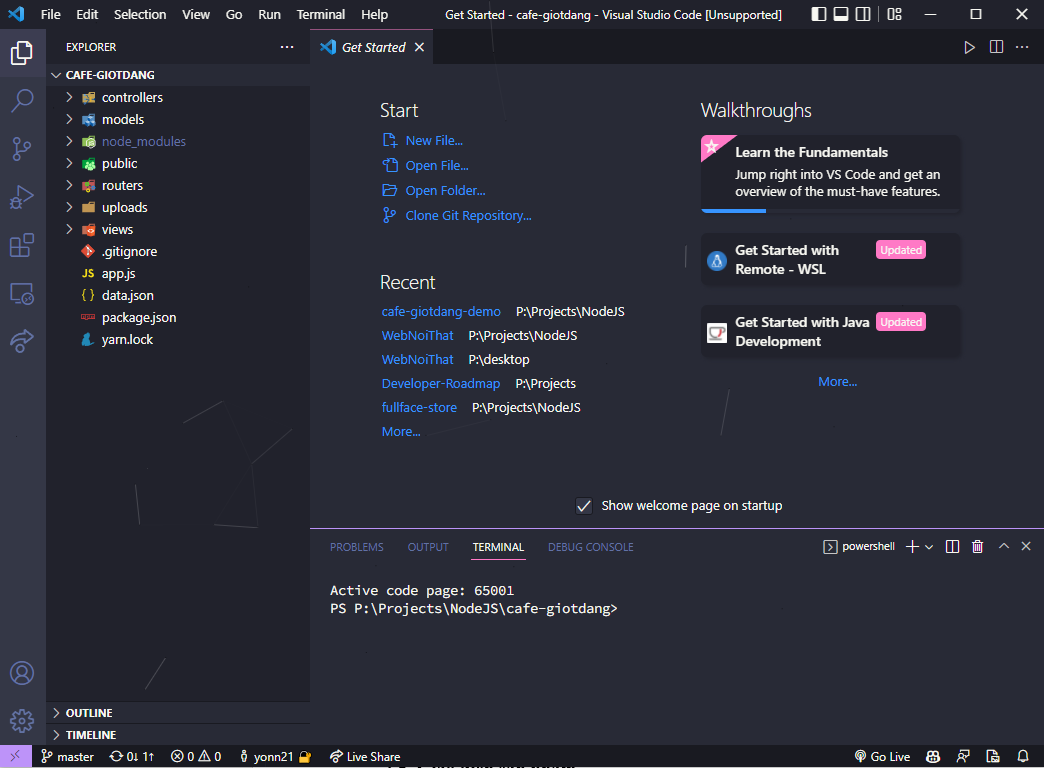


*Hình 3.4. Tạo database mới*

* Tạo Database **ecommerce**
* Trong database **ecommerce**, lần lượt tạo các Collection (Table) sau:
  + Admins
  + Bills
  + Customers
  + Products
  + Regions
  + Suppliers
  + Types

## 3.3. Code backend theo chuẩn cấu trúc MVC

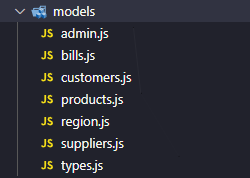
* Trong thư mục dự án, lần lượt tạo các thư mục sau:
  + controllers
  + models
  + public
  + routers
  + uploads
  + views
* Và tạo thêm file **app.js** là nơi khởi tạo server



*Hình 3.5. Cấu hình thư mục*

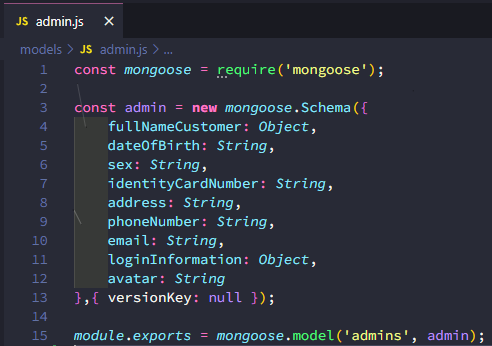
### 3.3.1. Models

* Tạo các file ***js*** theo tên table của database đã thiết kế:



*Hình 3.6. Models*

* Tiến hành code theo cấu trúc đã thiết kế với thư viện *mongoose*

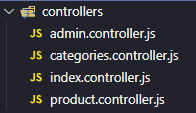


*Hình 3.7. Ví dụ file* ***admin.js***

* Code tương tự với các file còn lại.

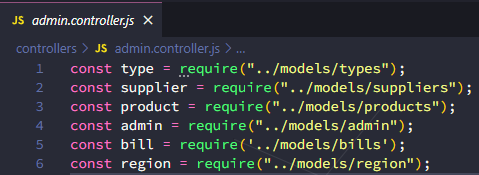
### 3.3.2. Controllers

* Tạo các file ***js*** dưới đây để xử lý các nghiệp vụ của website



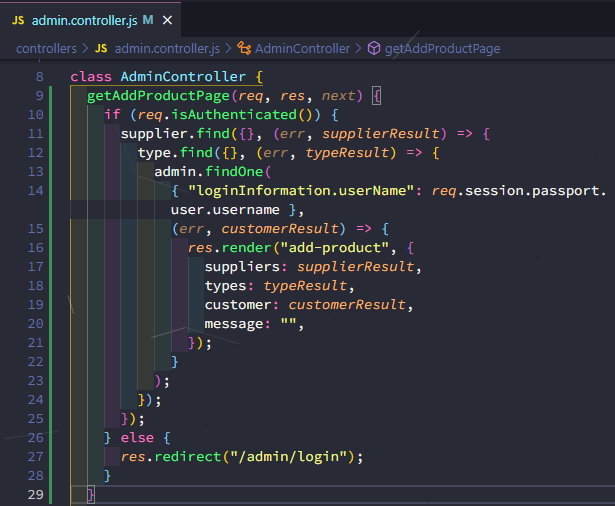
*Hình 3.8. Controllers*

* Import các model cần tương tác



*Hình 3.9. Ví dụ file* ***admin.controller.js***

* Tạo 1 lớp trong đó chứa các function xử lý các nghiệp vụ của website



*Hình 3.10.* ***admin.controller.js*** *với lớp* ***AdminController*** *và nghiệp vụ* ***getAddProductPage***

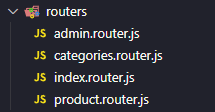
* Dưới cùng mỗi file luôn luôn export **lớp** ra để phía Router gọi đến:

*module.exports = new AdminController();*

* Tương tự với các file khác với những nghiệp vụ tương ứng

### 3.3.3. Routers

* Tạo các file ***js*** tương ứng với Controllers



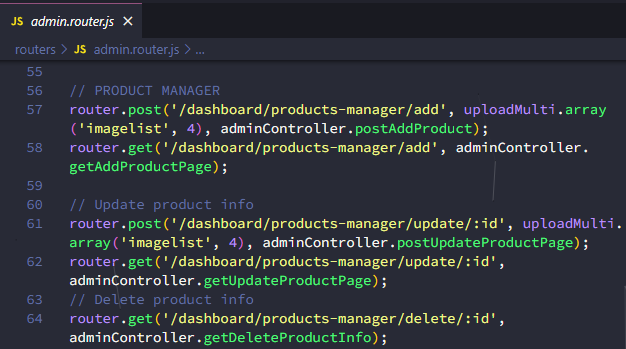
*Hình 3.11. Routers*

* Các file này sẽ định nghĩa đường đi của trang web.
* Tiếp theo, import các **module** cần thiết và Controller tương ứng, ví dụ với file **admin.router.js**



*Hình 3.12. admin.router.js - import*

* Định nghĩa đường đi cho các nghiệp vụ

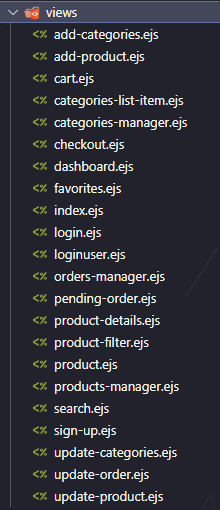


*Hình 3.13. admin.router.js*

* Tại dòng *57*, ***router.post*** định nghĩa giao thức **POST** với đường dẫn là “http://{domain}/dashboard/products-manager/add”, tiếp theo sẽ chạy đến ***uploadImage*** (multer) rồi mới gọi tới nghiệp vụ ***postAddProduct*** của ***adminController.***
* Code tương tự với các file còn lại. Cuối mỗi file tiếp tục export **router** để file server ***app.js*** gọi tới.

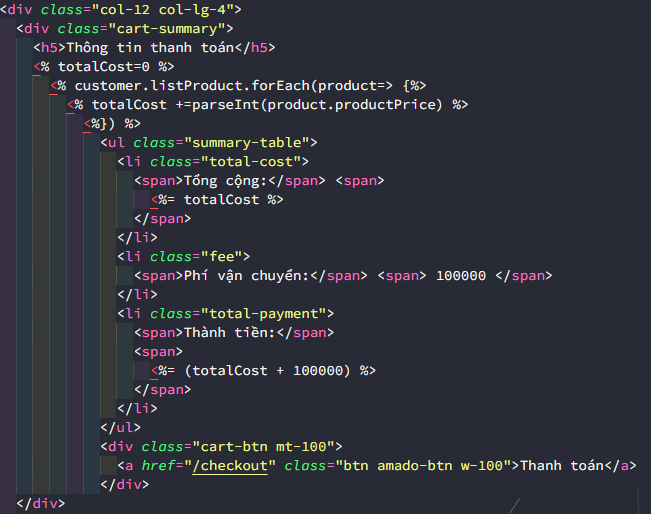
## 3.4. Code frontend

* Thiết kế các trang giao diện và code thuần HTML - CSS
* Sau khi code xong, tiến hành nhúng vào thư mục **views** bằng template engine ***ejs***



*Hình 3.14. Views*

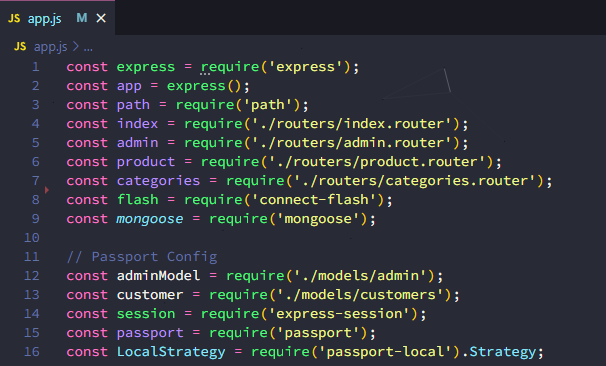
* Các file ***css*** sẽ được lưu trong “public/css”
* Thư mục **public** cũng là nơi lưu hình ảnh giao diện của website
* Trong file ***ejs*** này, tiến hành nhúng những mục cần tương tác với database bằng các cú pháp của ***ejs***, ví dụ nghiệp vụ tính tổng tiền của giỏ hàng trong trang giỏ hàng:



*Hình 3.15. Ví dụ tính* ***totalCost***

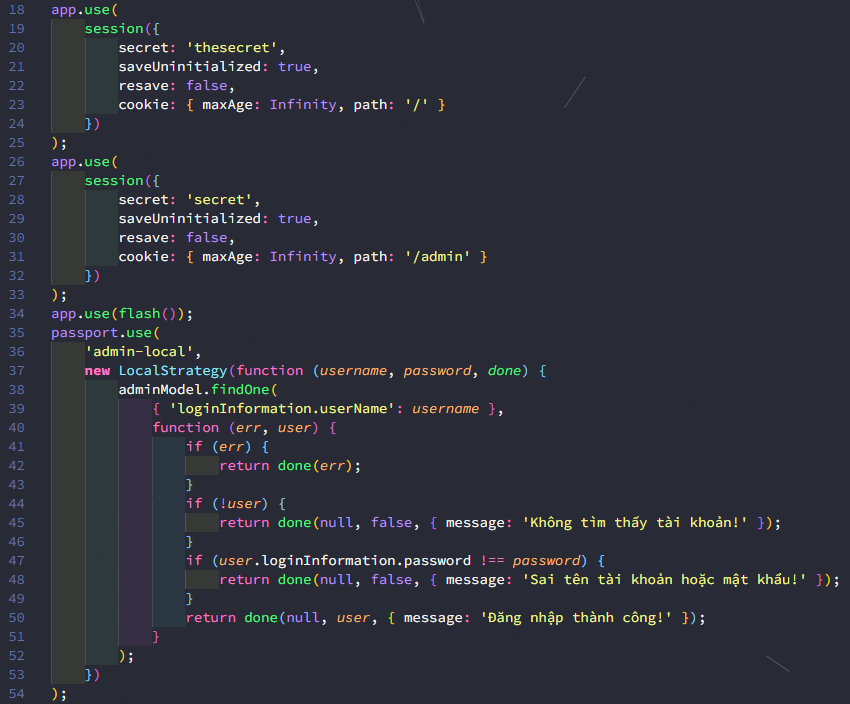
## 3.5. Code file root server

* Import module



*Hình 3.16.* ***app.js***

* Kết hợp sử dụng *passport, session* và *flash* để thông báo và lưu thông tin đăng nhập



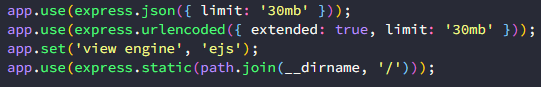
*Hình 3.17. passport, session và flash*

* Kết nối với MongoDB bằng *mongoose*



*Hình 3.18. mongoose setup*

* Set template engine *ejs*



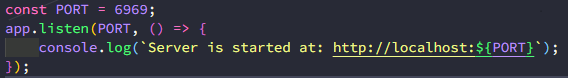
*Hình 3.19. ejs setup*

* Gọi các **module** từ *Routers*

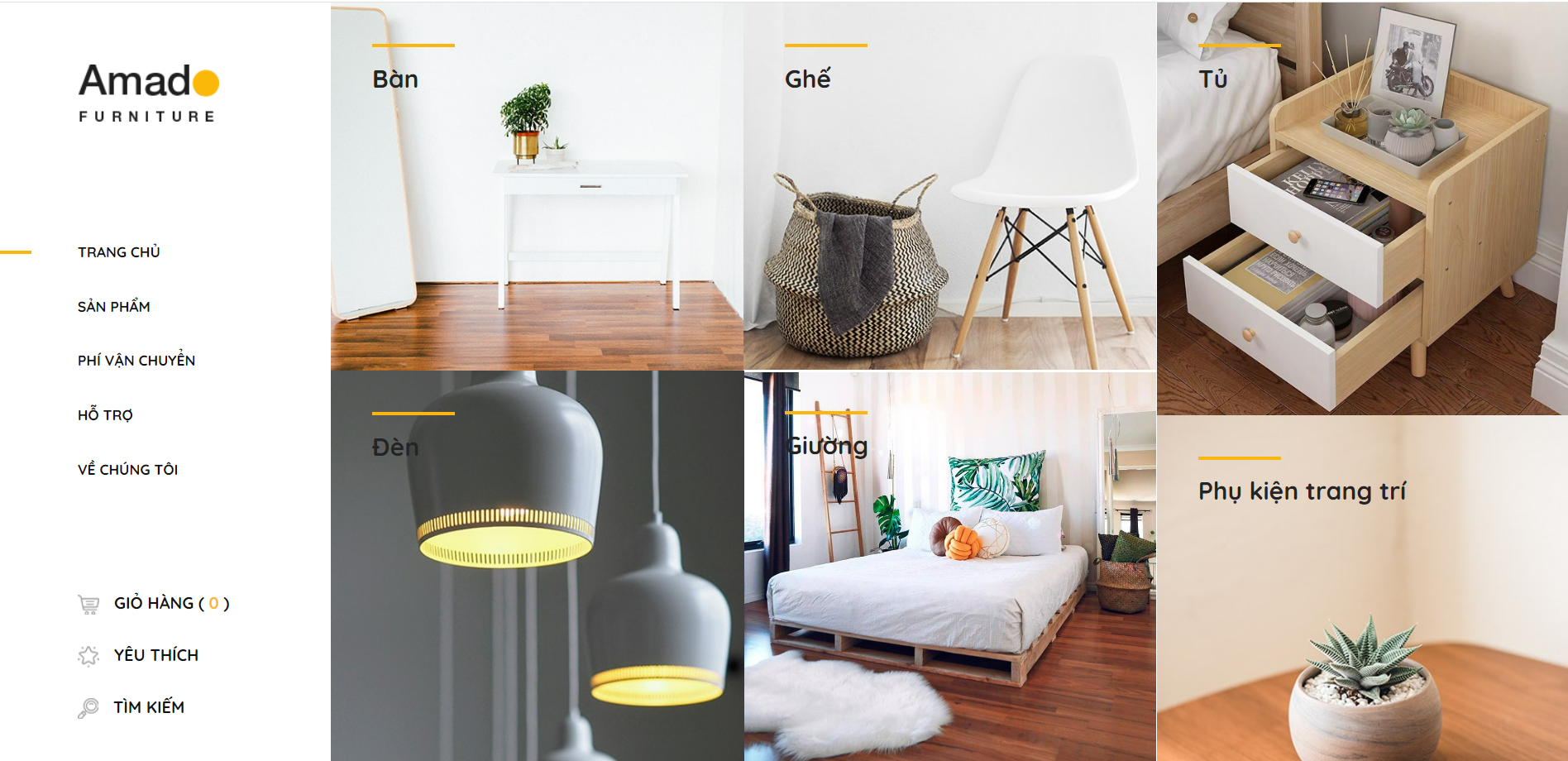


*Hình 3.20. route setup*

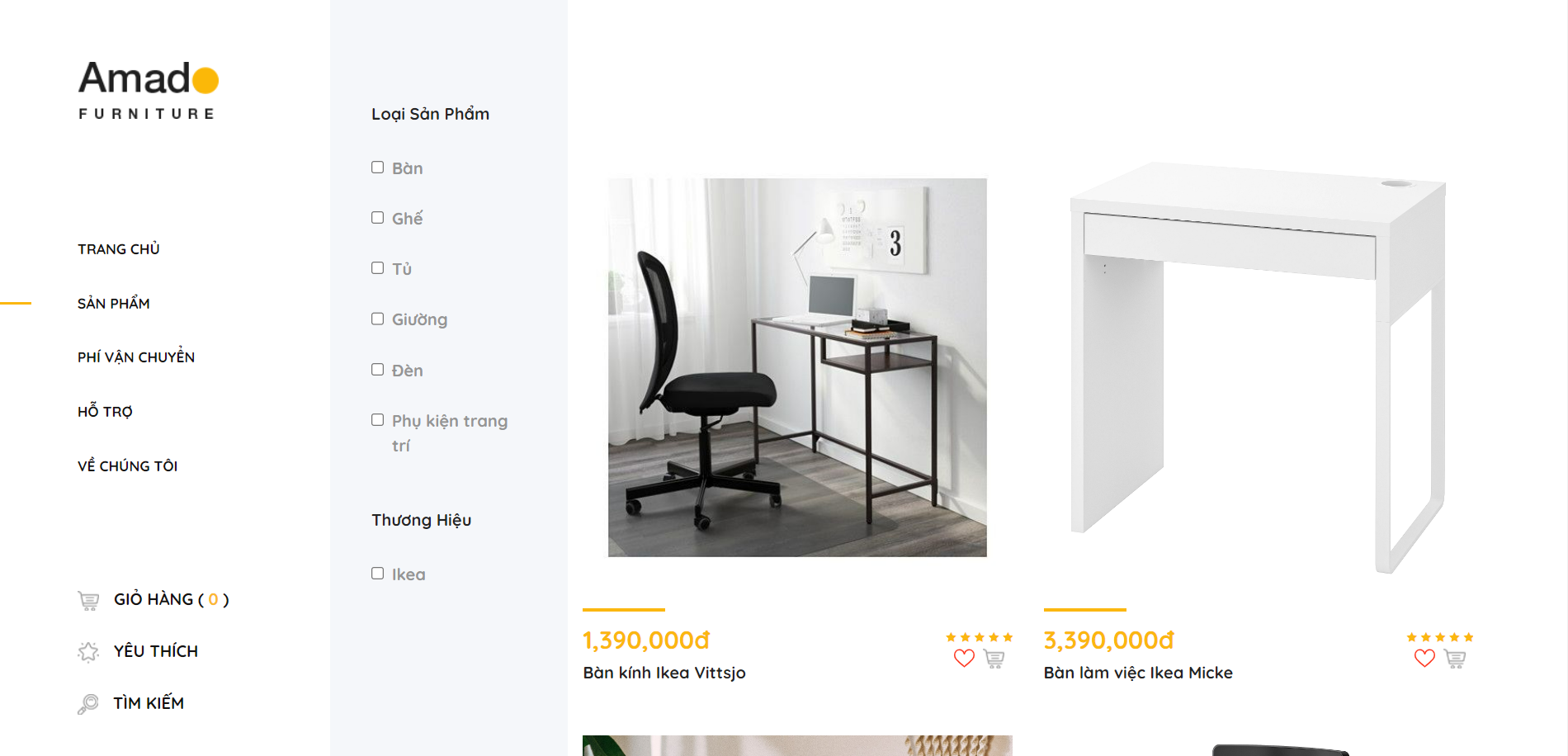
* Set ***port*** và để server lắng nghe từ địa chỉ: ***http://localhost:${port}***

**

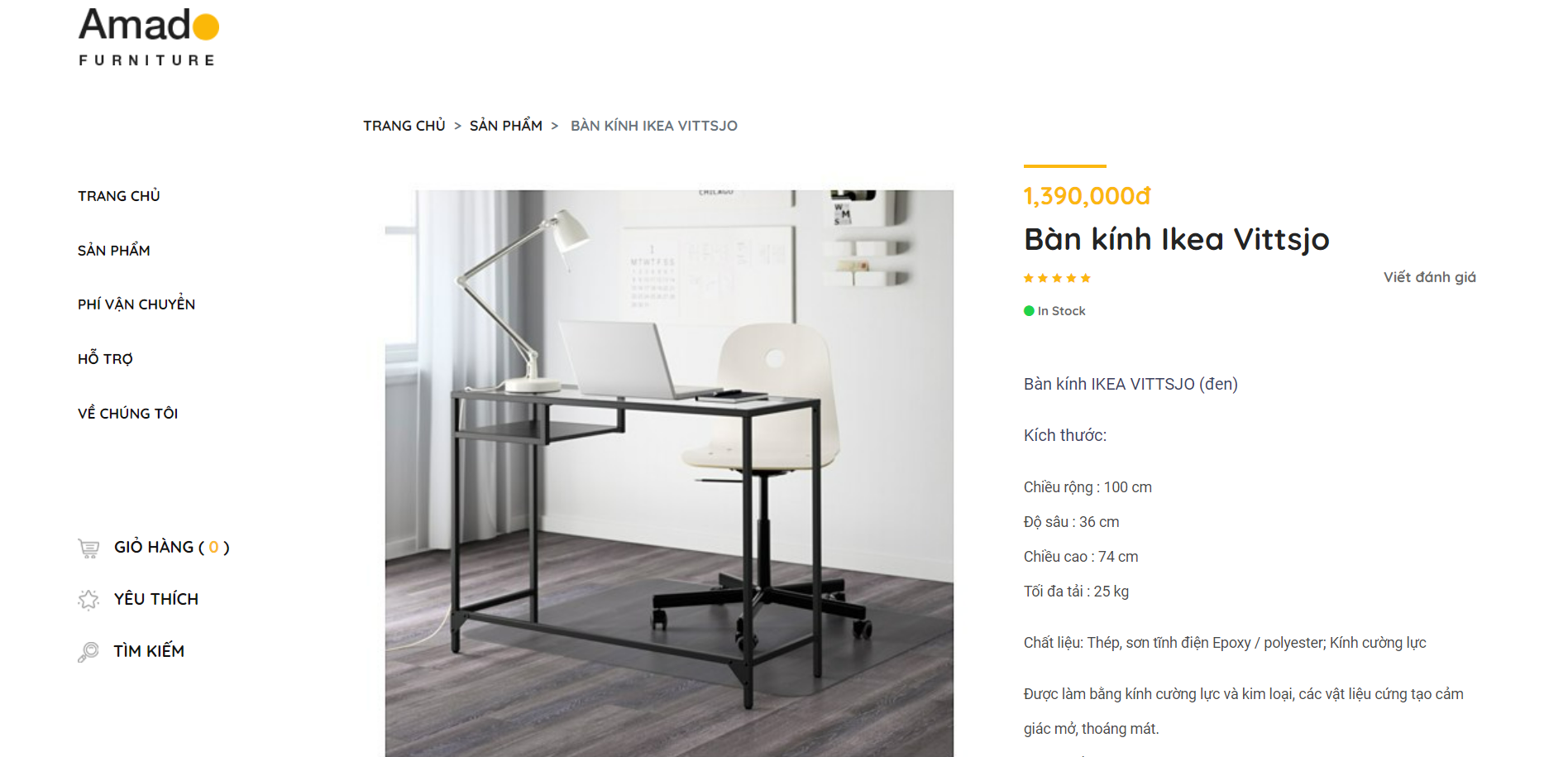
## 3.6. Giao diện sản phẩm



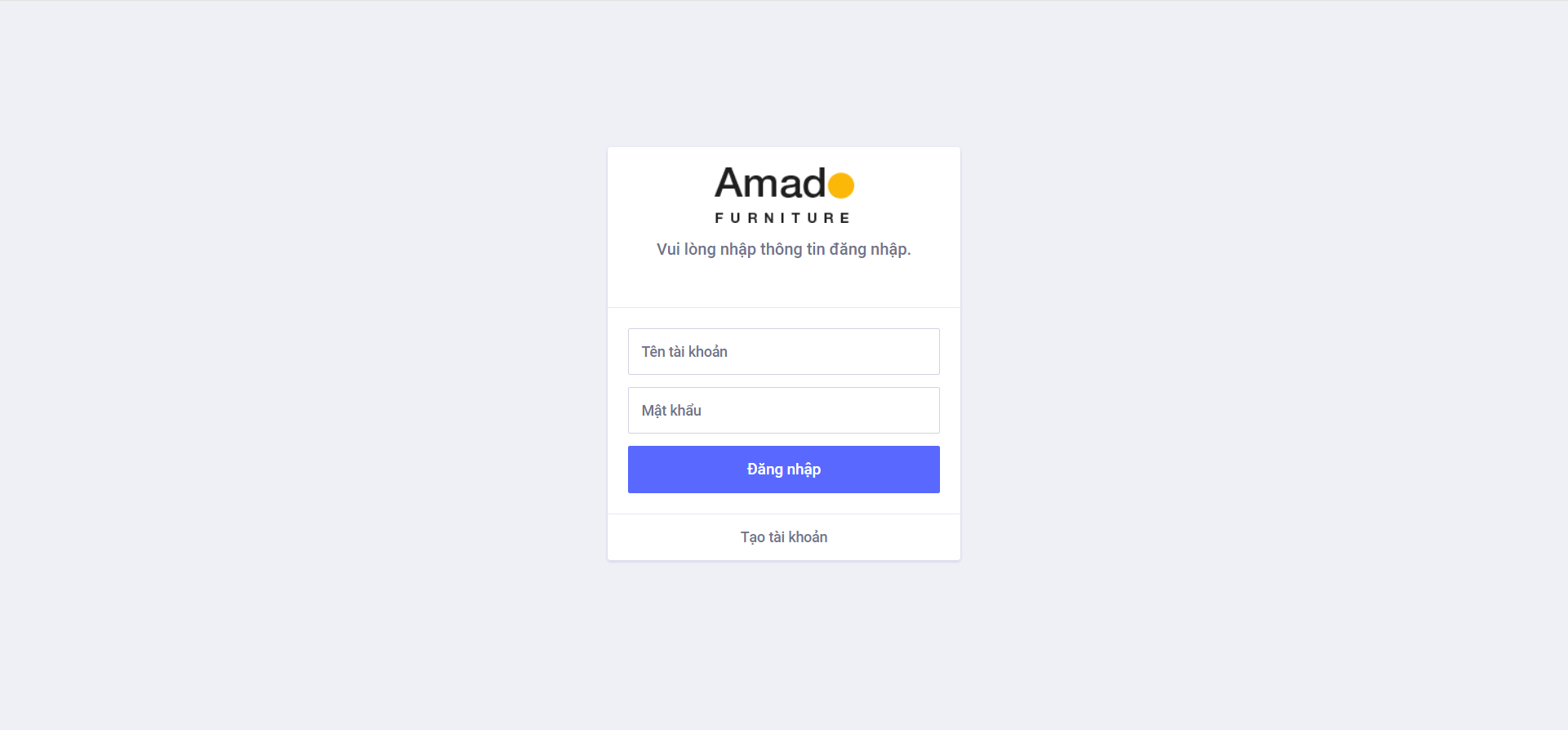
*Hình 3.21. Trang chủ*



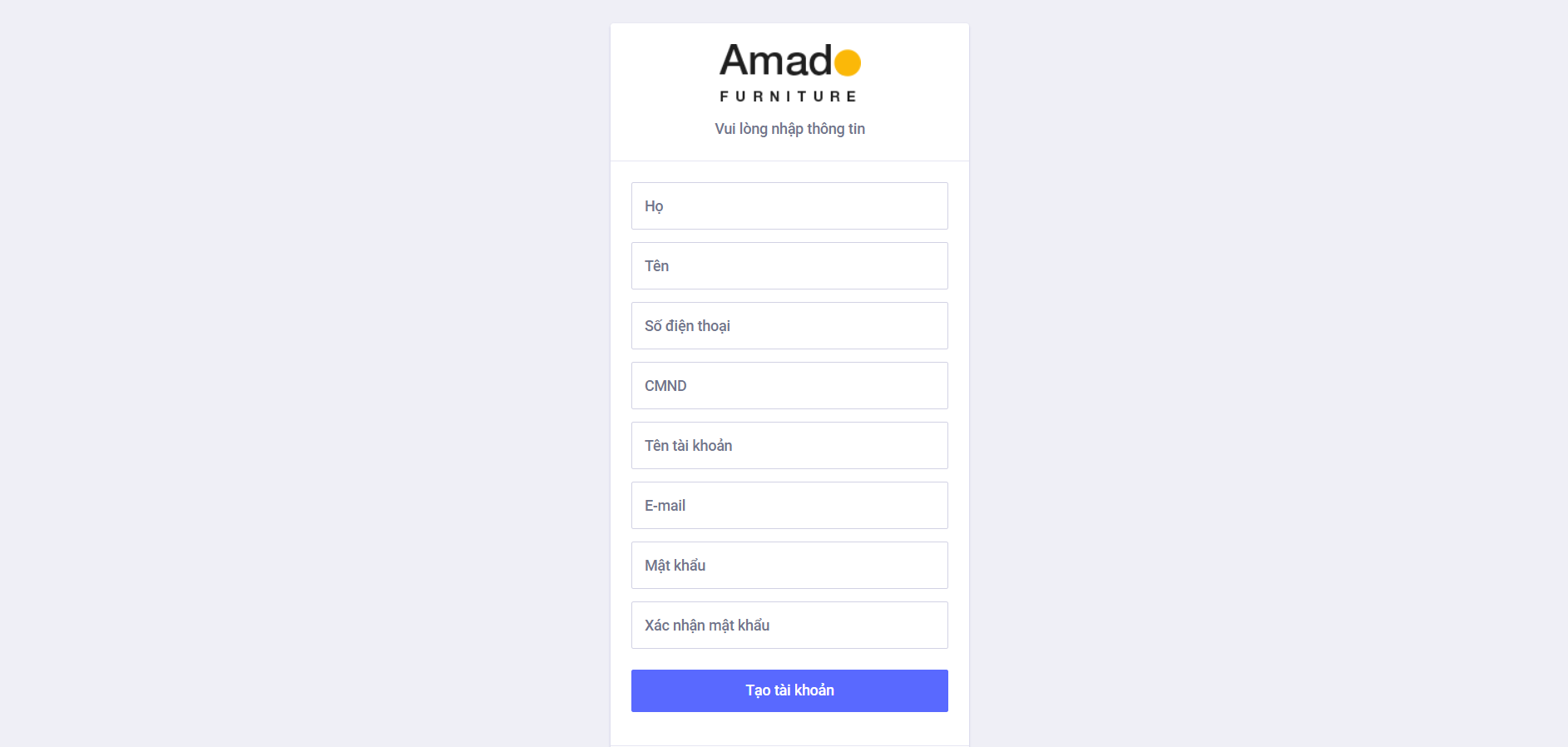
*Hình 3.22. Nội thất*



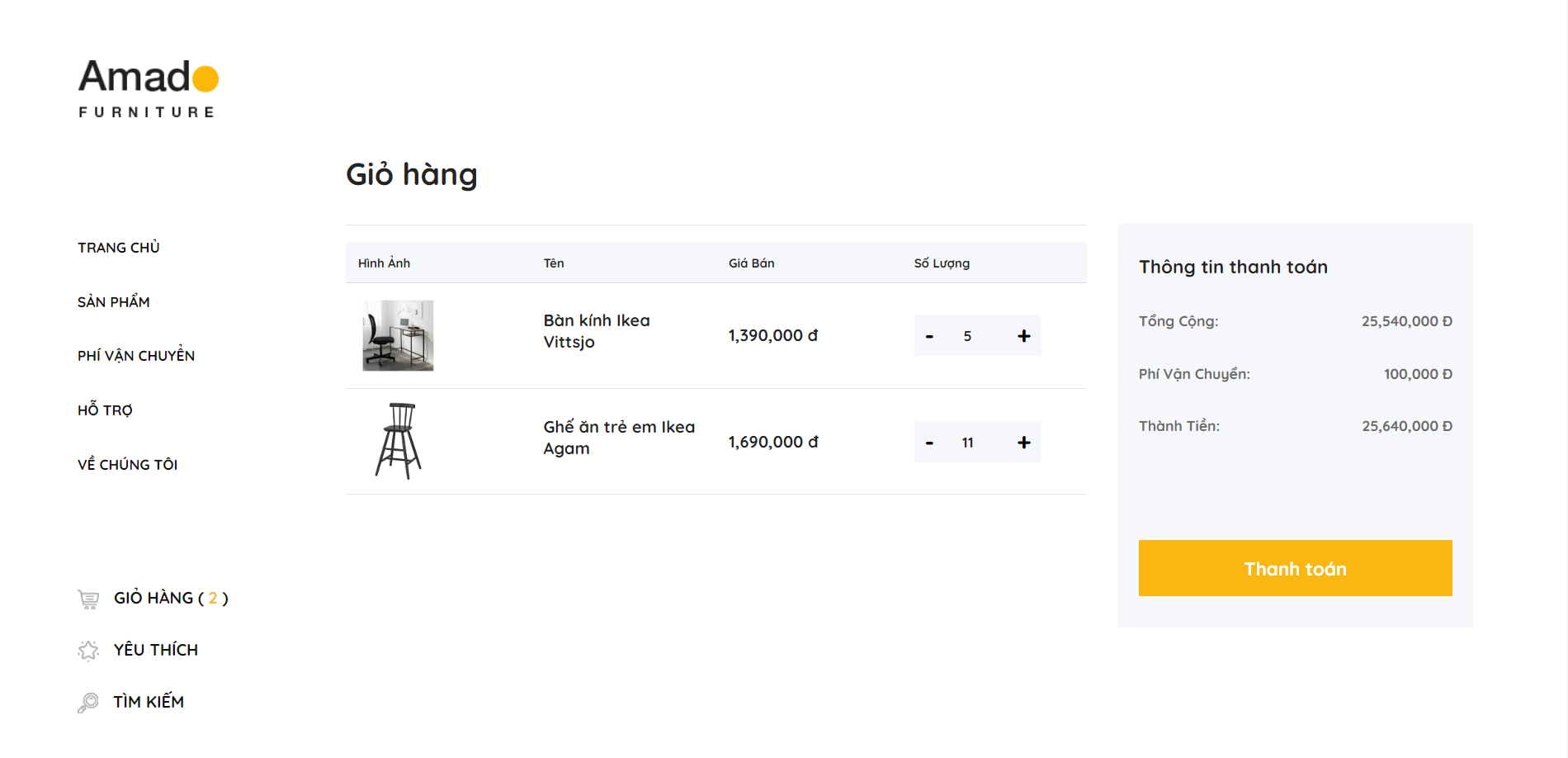
*Hình 3.23. Chi tiết nội thất*



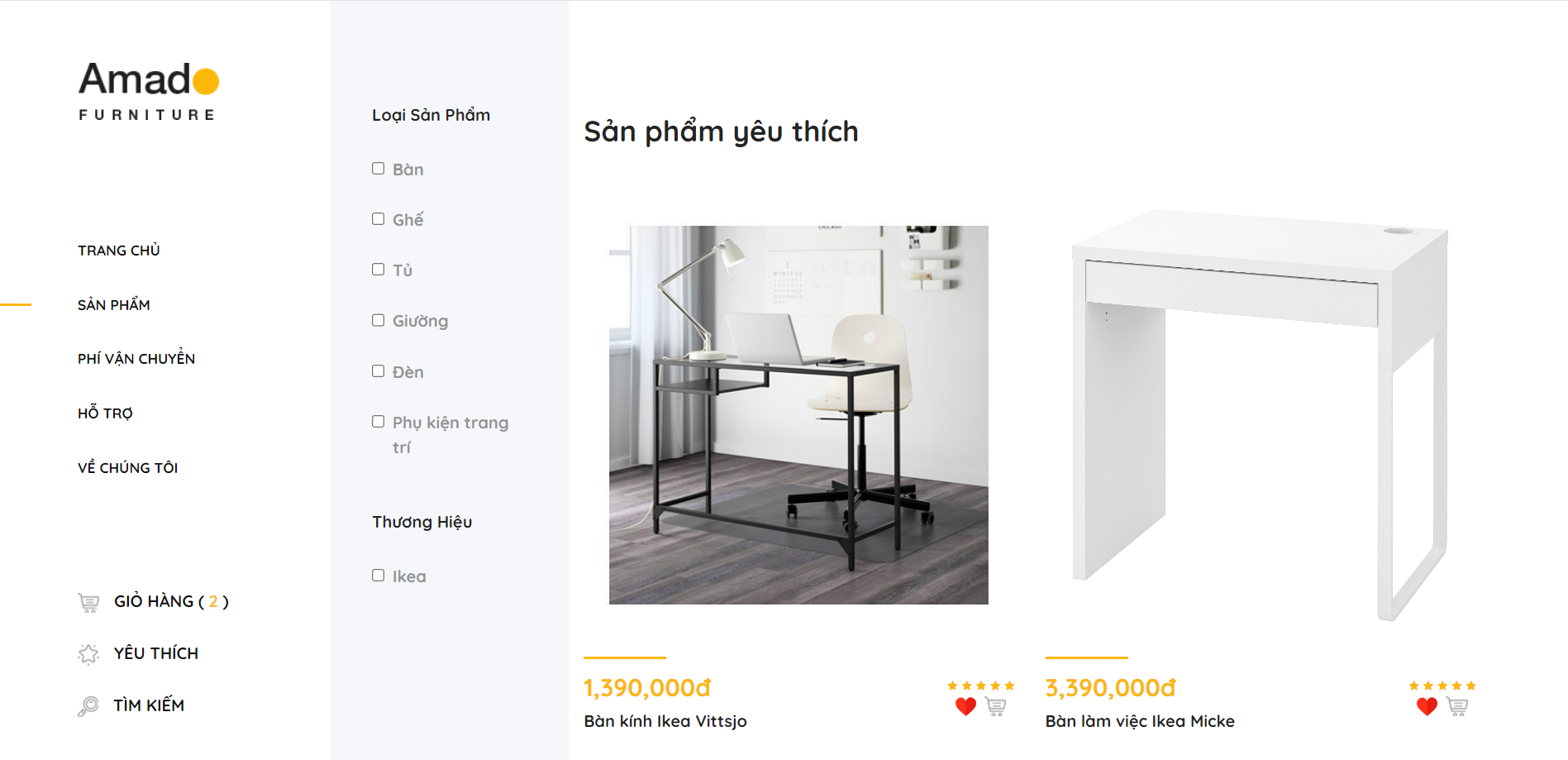
*Hình 3.24. Đăng nhập*



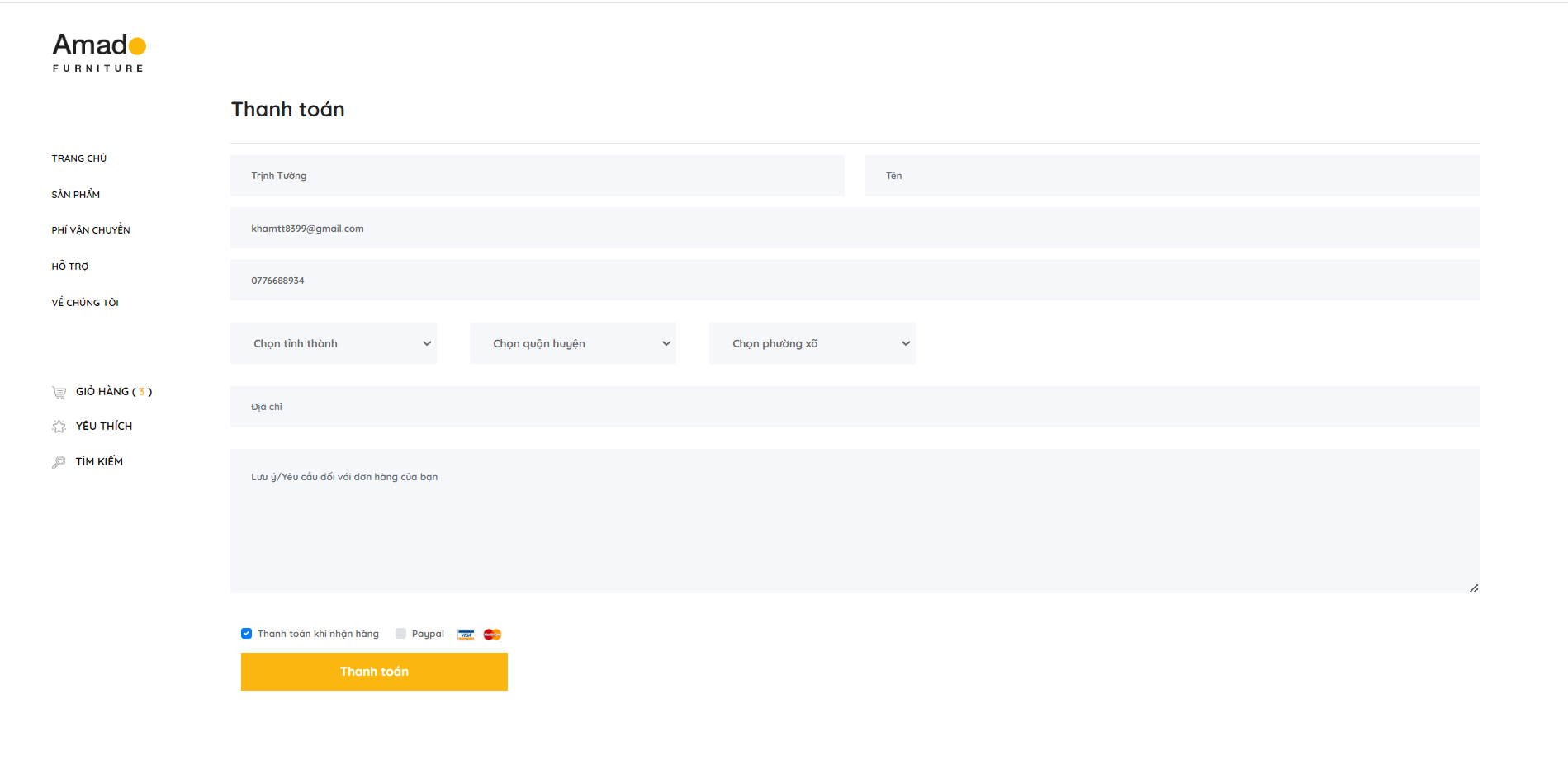
*Hình 3.25. Đăng ký*



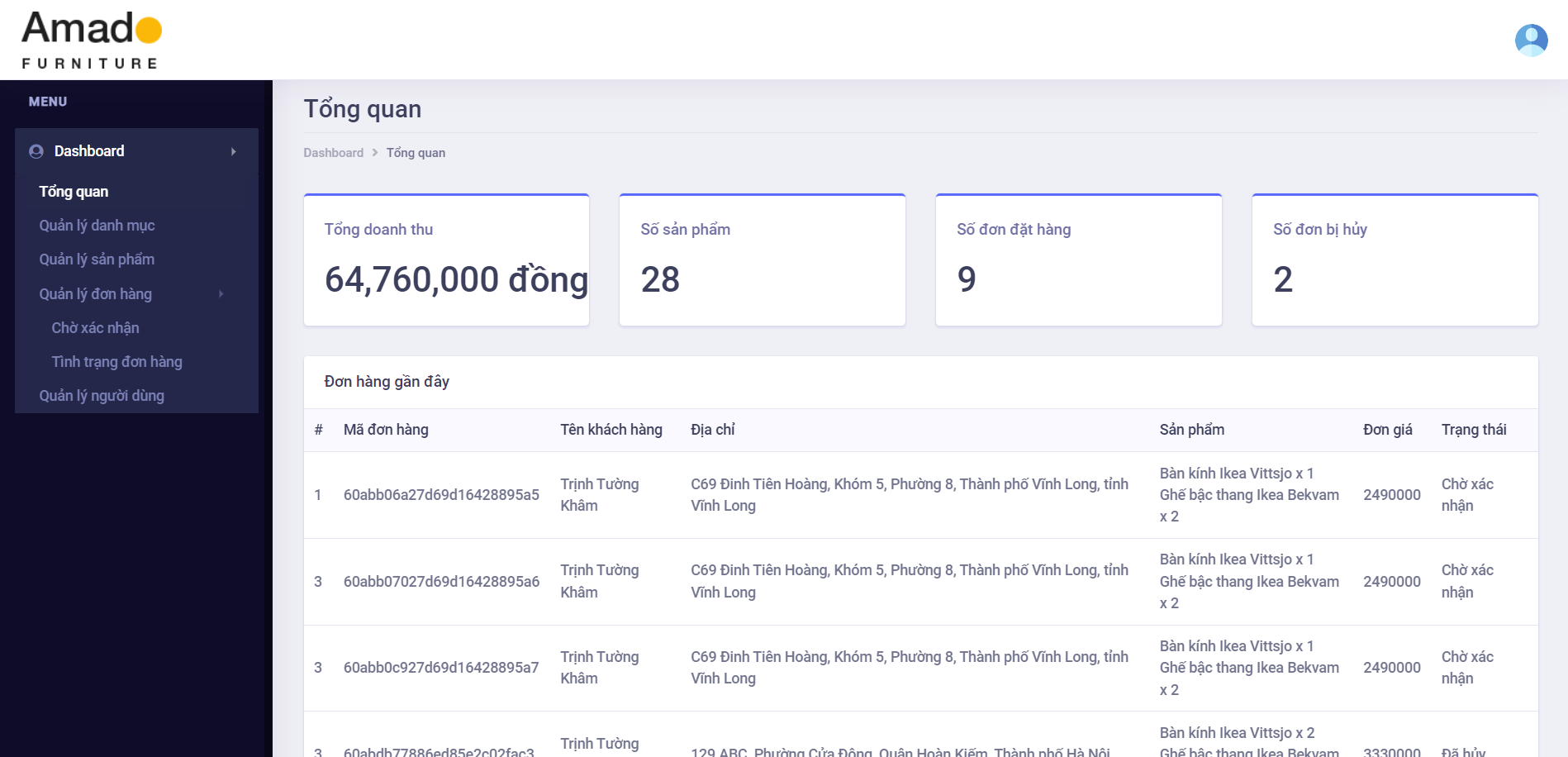
*Hình 3.26. Giỏ hàng*



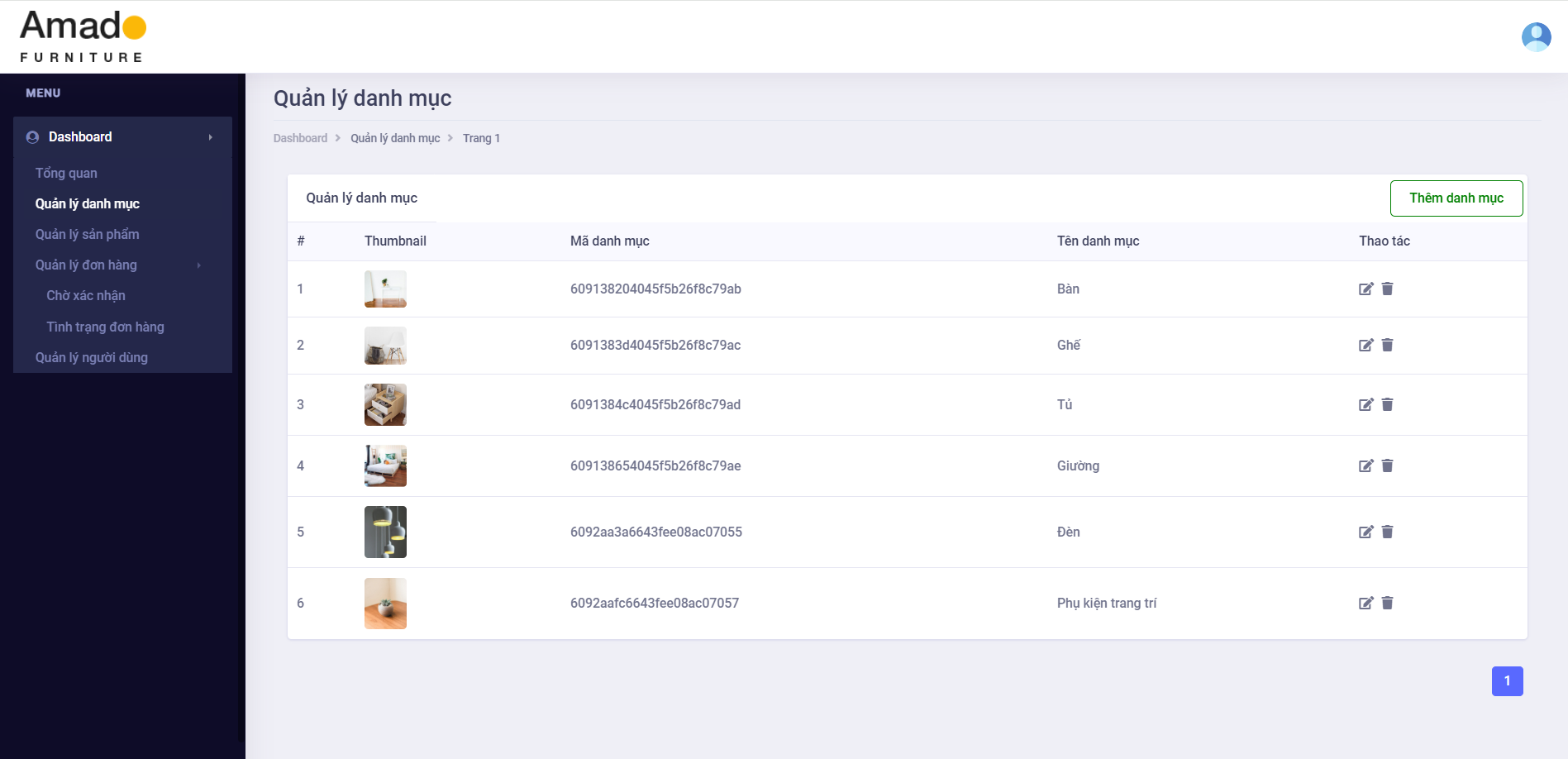
*Hình 3.27. Yêu thích*



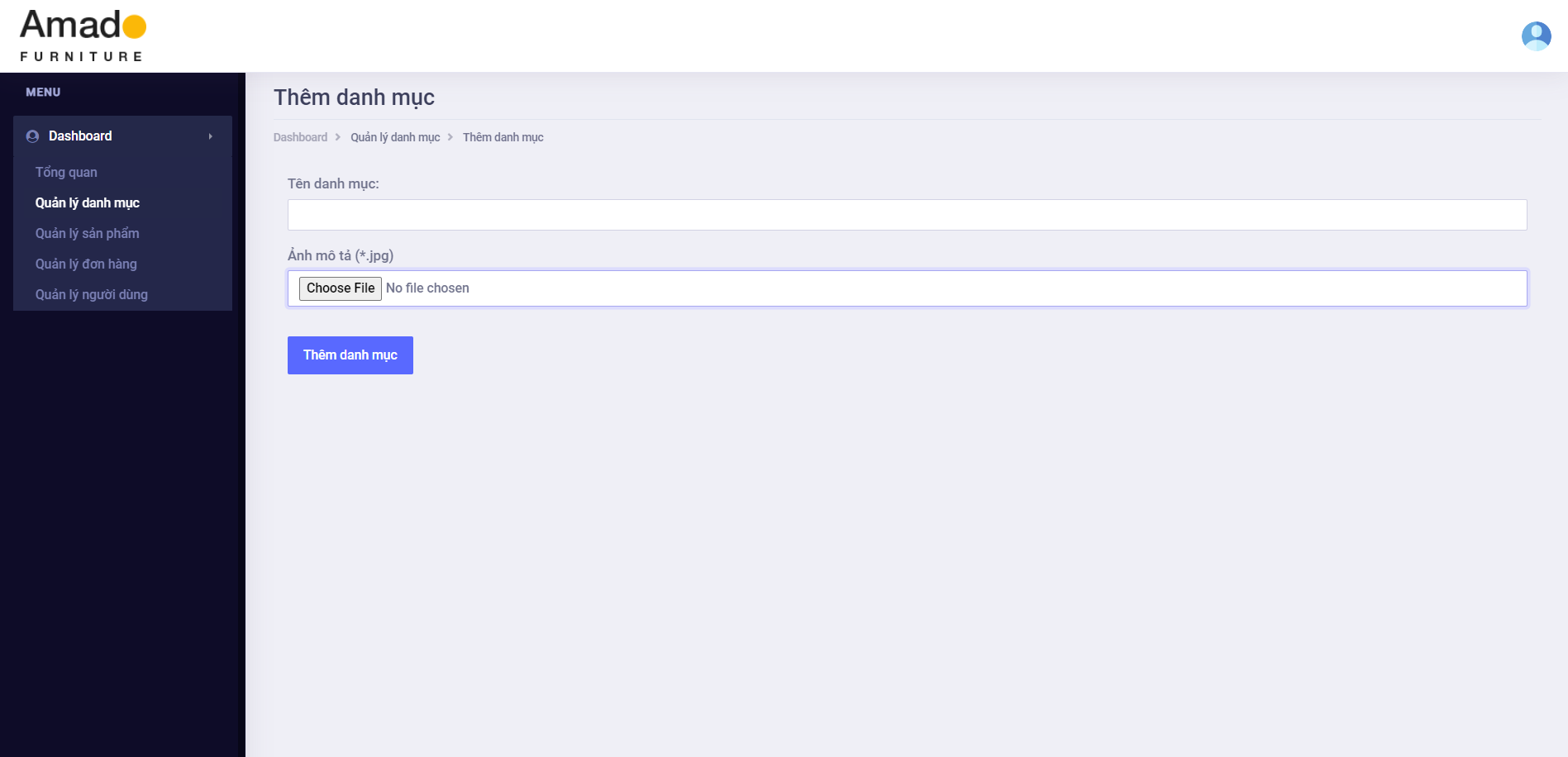
*Hình 3.28. Thanh toán*

**

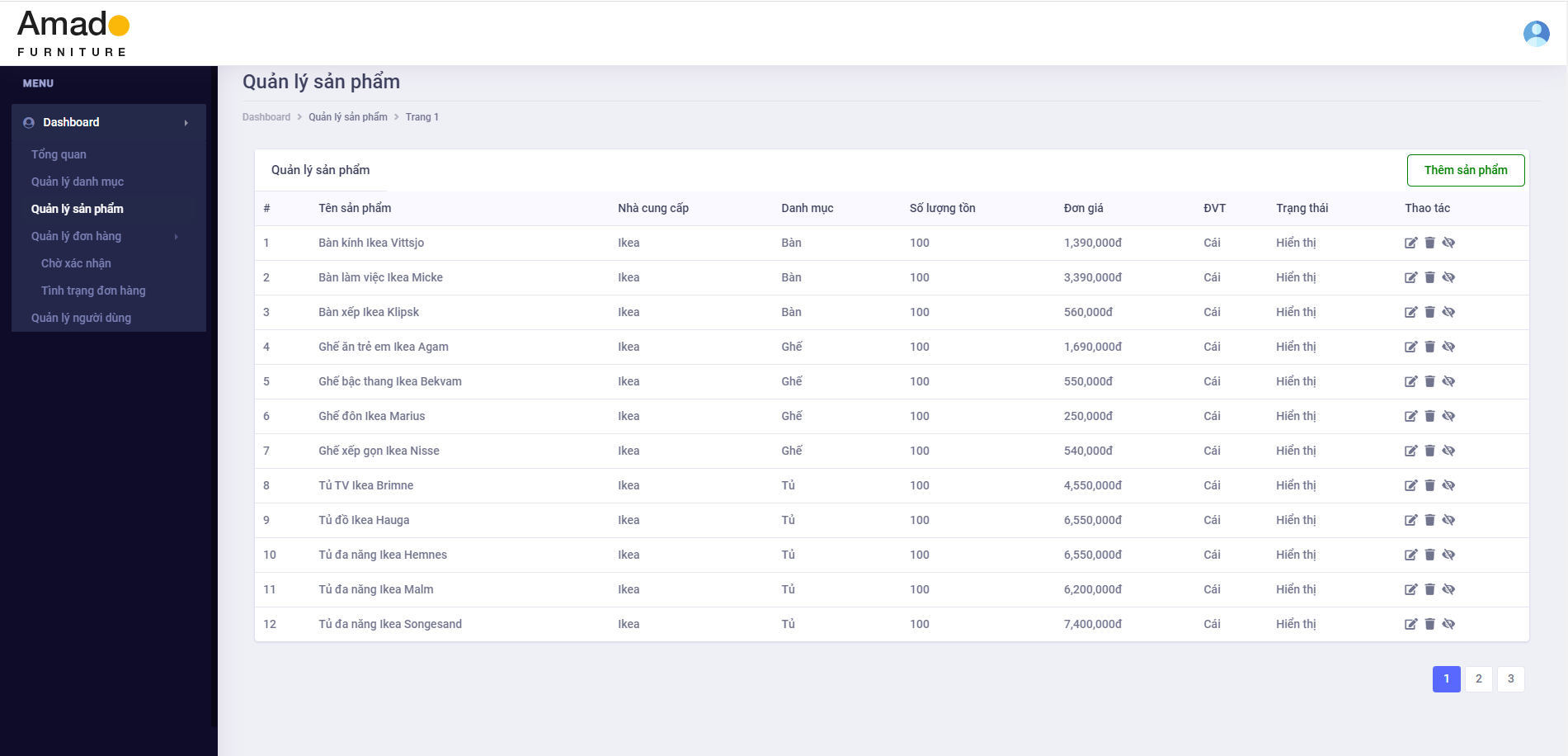
*Hình 3.29. Trang quản lý*

**

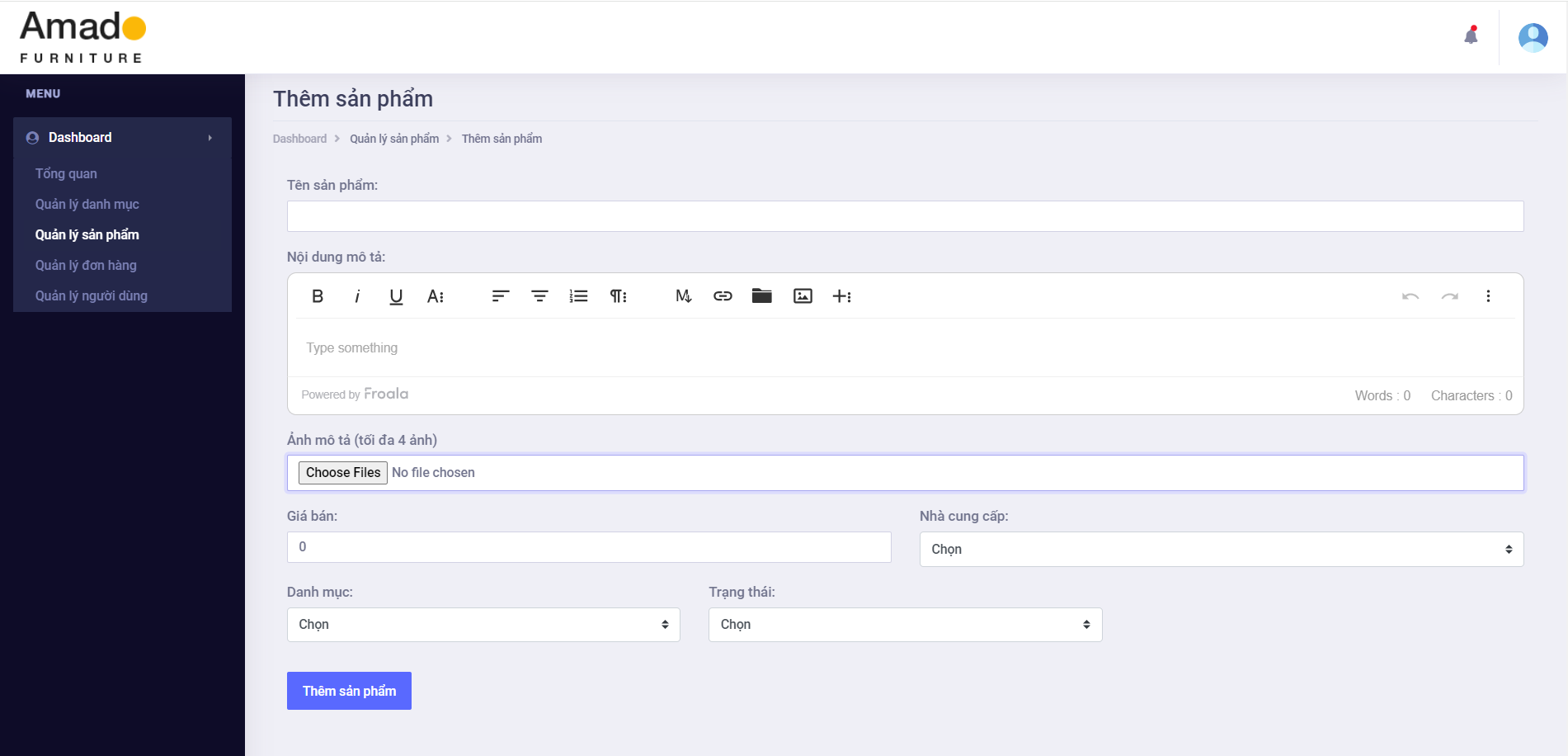
*Hình 3.30. Quản lý danh mục*

**

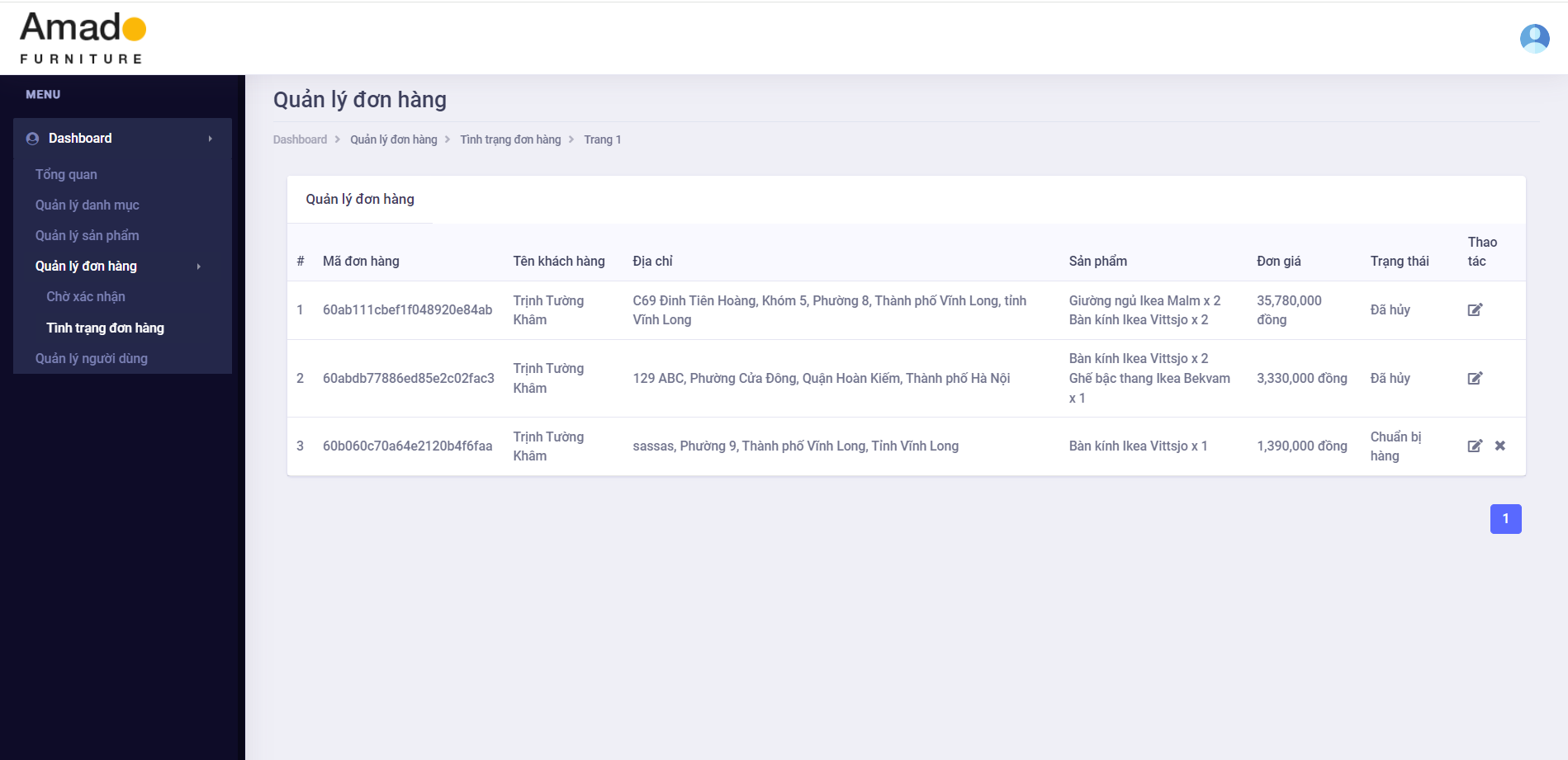
*Hình 3.31. Thêm danh mục*

**

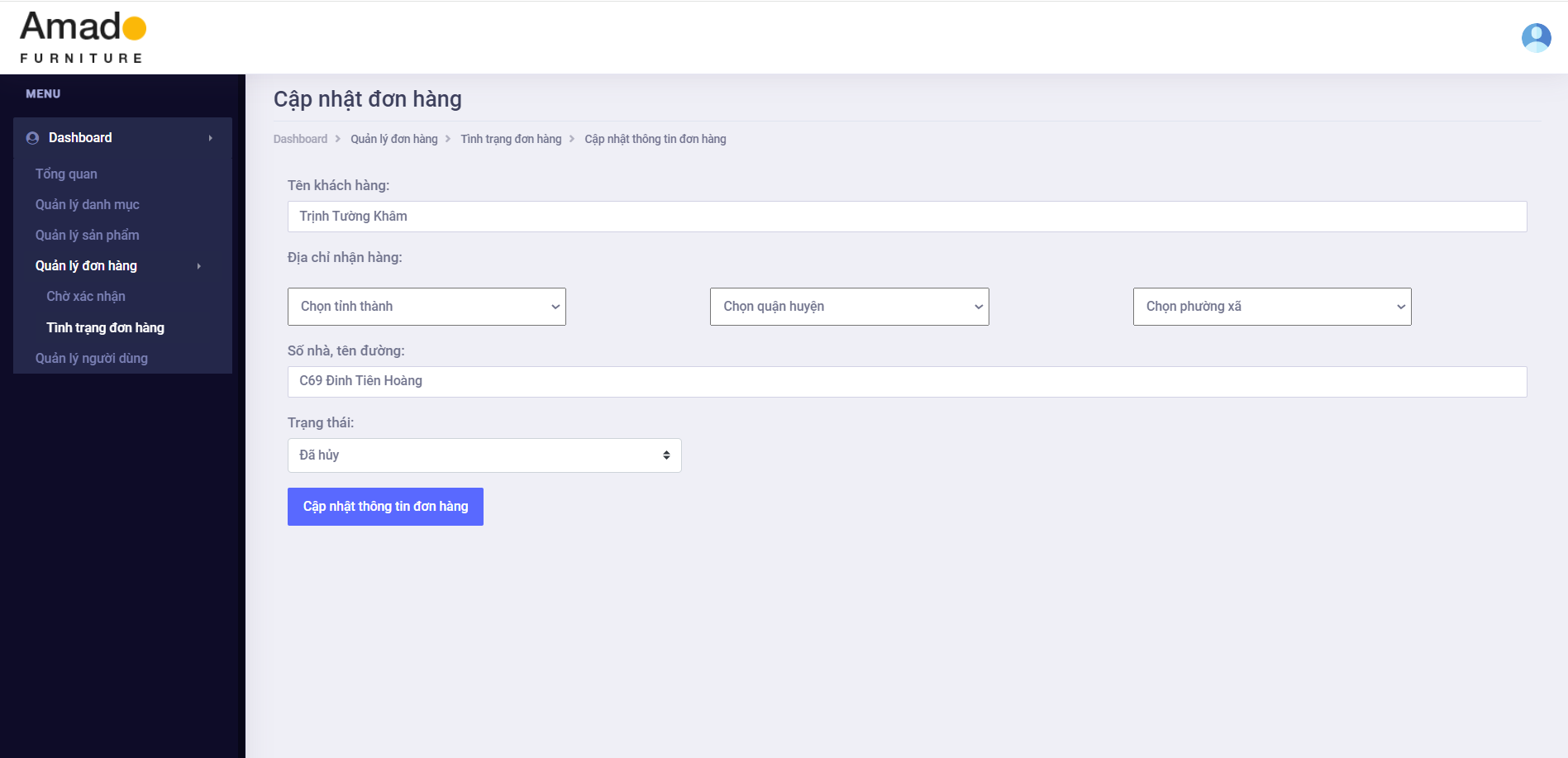
*Hình 3.32. Quản lý sản phẩm*

**

*Hình 3.33. Thêm sản phẩm*

**

*Hình 3.34. Quản lý đơn hàng*

**

*Hình 3.36. Cập nhập trạng thái đơn hàng*

# 

# KẾT LUẬN

Sau quá trình thực hiện và nghiên cứu đề tài, dưới sự hướng dẫn của thầy, Tiến sĩ Phạm Văn Hà, đề tài của cả chúng em đã đạt được những kết quả sau

## Kết quả đạt được

* + Tìm kiểu, nắm bắt được kĩ thuật sử dụng NodeJS kết nối với cơ sở dữ liệu MongoDB qua mô hình MVC.
  + Nắm được nguyên lý hoạt động của Server phía Backend.
  + Hiểu được các cơ sở dữ liệu MongoDB.
  + Biết cách viết API để phía Frontend sử dụng
  + Xây dựng được website với các chức năng như tạo, cập nhật, xóa, với các chức năng như trên phân tích

## Mặt hạn chế

Trong thời gian qua, chúng em đã cố gắng hết sức để tìm hiểu thực hiện đề tài. Tuy nhiên với kinh nghiệm và thời gian hạn chế nên không thể tránh khỏi những thiếu sót trong đề tài. Cụ thể:

* + Phần giao diện vẫn còn chưa được đẹp mắt
  + Chưa có chức năng, bình luận, đánh giá
  + Thời gian và trình độ còn hạn chế nên website chỉ dừng ở mức thử nghiệm. Do đó nếu muốn áp dụng vào thực tiễn cần phải có thời gian và công sức để hoàn thiện

# 

# TÀI LIỆU THAM KHẢO

[1] Tài liệu tìm hiểu Node Js: <https://nodejs.org/en/>

[2] Tài liệu tìm hiểu Express Js: <https://expressjs.com/>

[3] Tài liệu tìm hiểu MongoDB: <https://mongoosejs.com/>

[4] Tài liệu tìm hiểu EJS: <https://ejs.co/>

[5] Tài liệu tìm hiểu NPM: <https://www.npmjs.com/>