# Báo Cáo Phân Tích và Đề Xuất Kiến Trúc Cơ Sở Dữ Liệu cho Hệ Thống Quản Lý Nhà Thuốc Trực Tuyến Tích Hợp Chatbot

## Giới thiệu

### Mục đích của Báo cáo

Báo cáo này được biên soạn nhằm cung cấp một bản phân tích chuyên sâu về mô hình cơ sở dữ liệu (CSDL) quản lý sản phẩm nhà thuốc hiện hữu và đề xuất một kiến trúc CSDL hiện đại, tối ưu cho đề tài luận văn "Xây dựng hệ thống quản lý bán thuốc trực tuyến tích hợp chatbot tư vấn sản phẩm" trên nền tảng công nghệ MERN (MongoDB, Express.js, React, Node.js).

### Bối cảnh và Thách thức

Thách thức cốt lõi trong việc xây dựng một hệ thống nhà thuốc trực tuyến là quản lý một danh mục sản phẩm đa dạng và phức tạp. Các mặt hàng không chỉ bao gồm thuốc (kê đơn và không kê đơn) mà còn có thực phẩm chức năng, dược mỹ phẩm, thiết bị y tế, và các sản phẩm chăm sóc cá nhân khác.1 Mỗi loại sản phẩm này có những thuộc tính đặc thù riêng biệt. Ví dụ, thuốc có hoạt chất, hàm lượng, chỉ định, chống chỉ định; trong khi mỹ phẩm có loại da phù hợp, chỉ số chống nắng (SPF), thành phần.

Một hệ thống trực tuyến, đặc biệt là hệ thống có tích hợp chatbot tư vấn, đòi hỏi một CSDL có cấu trúc chặt chẽ, linh hoạt và hiệu năng cao để đáp ứng các truy vấn phức tạp từ người dùng, chẳng hạn như tìm kiếm theo triệu chứng, lọc sản phẩm theo thuộc tính chuyên biệt, và cung cấp thông tin tư vấn chính xác. Việc lựa chọn kiến trúc CSDL không chỉ là một quyết định kỹ thuật mà còn là nền tảng quyết định sự thành công, khả năng mở rộng và tính chính xác của toàn bộ hệ thống.

### Bối cảnh Luận văn

Trong khuôn khổ một đề tài luận văn công nghệ thông tin, việc lựa chọn và biện luận cho một kiến trúc CSDL hiện đại, tuân thủ các nguyên tắc thiết kế phần mềm tiên tiến là một yêu cầu quan trọng. Kiến trúc được đề xuất không chỉ cần giải quyết bài toán nghiệp vụ mà còn phải thể hiện được sự am hiểu sâu sắc về các mẫu thiết kế (design patterns) phù hợp với công nghệ NoSQL (cụ thể là MongoDB), chứng minh được tính ưu việt so với các giải pháp truyền thống hoặc đơn giản hóa.

### Cấu trúc Báo cáo

Báo cáo được cấu trúc thành năm phần chính:

1. **Phân tích Mô hình Dữ liệu "Đơn Bảng" Quan sát được:** Đánh giá sâu về ưu và nhược điểm của mô hình CSDL hiện tại mà bạn đã tham khảo.
2. **Các Mẫu Thiết kế CSDL Nâng cao cho Dữ liệu Đa hình thái trong MongoDB:** Giới thiệu nền tảng lý thuyết về các mẫu thiết kế hiện đại để giải quyết bài toán quản lý sản phẩm đa dạng.
3. **Kiến trúc CSDL Đề xuất cho Hệ thống Nhà thuốc Trực tuyến:** Trình bày chi tiết một mô hình CSDL mới, tối ưu và có khả năng mở rộng.
4. **Tích hợp Kiến trúc và Khả năng Đáp ứng Tính năng:** Chứng minh cách kiến trúc mới hỗ trợ hiệu quả các tính năng chính của hệ thống, đặc biệt là giao diện bán hàng và chatbot tư vấn.
5. **Chiến lược Triển khai và Các Thực hành Tốt nhất trong MERN Stack:** Cung cấp hướng dẫn cụ thể để hiện thực hóa kiến trúc đề xuất bằng Mongoose và các công cụ liên quan.

## Phân tích Mô hình Dữ liệu "Đơn Collection" Quan sát được

Mô hình CSDL mà bạn quan sát được tại nhà thuốc thực tế, nơi tất cả các mặt hàng được lưu trữ trong một bảng (hoặc một collection trong MongoDB) duy nhất, là một cách tiếp cận phổ biến trong các hệ thống quản lý nội bộ đời đầu. Để đưa ra quyết định có nên áp dụng mô hình này hay không, cần phải phân tích một cách có hệ thống các đặc điểm, ưu điểm trong ngữ cảnh hẹp và các nhược điểm nghiêm trọng khi áp dụng cho hệ thống của bạn.

### Đặc điểm của Mô hình Hiện hữu

Dựa trên các hình ảnh giao diện (image\_cfcf25.png, image\_cfd6e4.png, image\_cfd9af.png) và tệp dữ liệu Drugs\_11846\_10082025214352.xlsx 1, mô hình CSDL này có các đặc điểm sau:

* **Cấu trúc Phẳng, Rộng (Flat, Wide-Table Structure):** Tất cả các loại sản phẩm, từ thuốc "GOURCUFF 5MG" đến dược mỹ phẩm "ABENA ZINC OINTMENT" và dụng cụ y tế "Dolphin Mask Khẩu trang", đều được lưu trữ trong cùng một cấu trúc với một tập hợp lớn các trường (cột).
* **Phi chuẩn hóa Nặng (Heavy Denormalization):** Các thông tin như "Nhóm Thuốc" (ví dụ: "Dạ dày - Đường ruột - Tiêu hóa") và "Hãng SX" được lưu trữ dưới dạng chuỗi ký tự trực tiếp trong mỗi bản ghi sản phẩm. Điều này dẫn đến sự lặp lại dữ liệu và tiềm ẩn nguy cơ không nhất quán khi có sự thay đổi (ví dụ: đổi tên một nhóm thuốc).
* **Phân loại Ngầm định (Implicit Typing):** Loại hình của một sản phẩm được xác định hoàn toàn dựa trên giá trị chuỗi của trường "Nhóm Thuốc". Không có sự khác biệt về cấu trúc dữ liệu ở cấp độ CSDL để phân biệt một viên thuốc với một tuýp kem chống nắng.

### Ưu điểm Thực dụng trong một Hệ thống Quản lý Nội bộ

Việc mô hình này được đánh giá là "khá hiệu quả" trong môi trường vận hành của nhà thuốc thực tế là có cơ sở, xuất phát từ các yếu tố sau:

* **Đơn giản cho Tác vụ Nhập liệu và Báo cáo Cơ bản:** Đối với một dược sĩ hoặc nhân viên bán hàng sử dụng giao diện phần mềm được thiết kế riêng (như trong image\_cfd6e4.png và image\_cfd9af.png), mô hình này rất trực quan. Giao diện người dùng hiển thị tất cả các trường có thể có, và người dùng chỉ cần điền thông tin vào các trường liên quan. Việc tạo một danh sách kiểm kê kho đơn giản (như trong image\_cfcf25.png) hoặc một báo cáo bán hàng cuối ngày trở nên dễ dàng vì tất cả dữ liệu đều nằm ở một nơi.
* **Chi phí Phát triển Ban đầu Thấp:** Việc tạo một Schema Mongoose duy nhất cho một collection ban đầu sẽ nhanh hơn so với việc thiết kế một kiến trúc phức tạp hơn gồm nhiều collection có liên kết hoặc áp dụng các mẫu thiết kế nâng cao. Đây là lý do tại sao nó phổ biến trong các ứng dụng nội bộ, có phạm vi chức năng giới hạn.

### Các Nhược điểm Trầm trọng đối với Hệ thống Trực tuyến Tích hợp Chatbot

Mặc dù hiệu quả trong ngữ cảnh hẹp, mô hình này sẽ trở thành một trở ngại kỹ thuật lớn khi được áp dụng cho một hệ thống bán hàng trực tuyến hiệu năng cao và đặc biệt là khi tích hợp chatbot tư vấn.

* Dữ liệu Thưa thớt và Kém hiệu quả (Data Sparsity and Inefficiency):  
  Hầu hết các bản ghi (document) sẽ có một tỷ lệ lớn các trường bị bỏ trống (null) hoặc không có giá trị. Ví dụ, sản phẩm TH7787 ("Dolphin Mask Khẩu trang 3D Trẻ Em 4 Lớp") là một dụng cụ y tế. Khi lưu trữ theo mô hình này, nó sẽ có các trường như Hoạt Chất (Active Ingredient), Hàm Lượng (Contents), Số ĐK (Registered Number), Chỉ định (Indications), Chống chỉ định (Contraindications) hoàn toàn trống.1 Những trường này chỉ có ý nghĩa đối với thuốc. Việc này không chỉ lãng phí không gian lưu trữ mà còn làm tăng gánh nặng xử lý và truyền tải dữ liệu không cần thiết.
* **Độ phức tạp Truy vấn và Hiệu năng Thảm họa:**
  + **Bất khả thi trong việc Xây dựng Bộ lọc Nâng cao (Faceted Search):** Một trang web thương mại điện tử hiện đại phải cho phép người dùng lọc sản phẩm theo các thuộc tính đặc thù. Ví dụ, một truy vấn như "Tìm tất cả kem chống nắng cho da dầu với chỉ số SPF từ 50 trở lên". Trong mô hình hiện tại, thông tin về "loại da" hay "chỉ số SPF" có thể không tồn tại dưới dạng trường riêng biệt, hoặc bị trộn lẫn trong chuỗi văn bản của trường "Tên Thuốc" hoặc "Thông Tin". Để thực hiện truy vấn này, hệ thống sẽ phải dùng đến các phép toán tìm kiếm văn bản toàn phần (LIKE trong SQL hoặc $regex trong MongoDB) trên toàn bộ collection. Các truy vấn này cực kỳ chậm, không thể tối ưu bằng chỉ mục (index) một cách hiệu quả, và cho kết quả không đáng tin cậy.
  + **Vô hiệu hóa Khả năng của Chatbot:** Chức năng cốt lõi của chatbot là cung cấp tư vấn dựa trên ngữ cảnh và thuộc tính. Một câu hỏi từ người dùng như "Tìm giúp tôi thuốc trị đau đầu có chứa Paracetamol 500mg" là không thể thực hiện một cách chính xác. Chatbot không thể phân biệt một cách đáng tin cậy đâu là hoạt chất, đâu là tên thương mại, đâu là hàm lượng nếu tất cả chỉ là một phần của chuỗi văn bản Tên Thuốc hoặc Hoạt Chất. Sự thành bại của tính năng chatbot phụ thuộc trực tiếp vào khả năng thực hiện các truy vấn có cấu trúc lên CSDL, một điều mà mô hình này không thể đáp ứng.
* Thất bại trong Đảm bảo Toàn vẹn và Xác thực Dữ liệu:  
  Không thể áp đặt các quy tắc nghiệp vụ riêng cho từng loại sản phẩm ở tầng CSDL. Ví dụ, không thể thiết lập một quy tắc rằng mọi sản phẩm có "Nhóm Thuốc" là "Hạ nhiệt giảm đau - kháng viêm" thì bắt buộc phải có trường Hoạt Chất khác rỗng. Mọi logic xác thực đều phải được xử lý ở tầng ứng dụng, dẫn đến mã nguồn phức tạp, khó bảo trì và dữ liệu không nhất quán. Trong lĩnh vực dược phẩm, sự thiếu nhất quán này có thể gây ra hậu quả nghiêm trọng.
* **Cơn ác mộng về Bảo trì và Mở rộng:**
  + Khi cần thêm một loại sản phẩm mới với các thuộc tính đặc thù (ví dụ: "Sữa công thức" với các trường "độ tuổi", "thông tin dinh dưỡng"), nhà phát triển sẽ phải thêm các cột mới vào bảng vốn đã cồng kềnh, làm gia tăng mức độ thưa thớt của dữ liệu.
  + Việc sửa đổi cấu trúc (schema) trở thành một hoạt động rủi ro cao. Logic của ứng dụng sẽ trở thành một mớ hỗn độn các câu lệnh if/else để xử lý các loại sản phẩm khác nhau, khiến mã nguồn trở nên "giòn" (brittle) và cực kỳ khó để bảo trì hay nâng cấp.

**Bảng 1: So sánh Phân tích các Mô hình Cơ sở dữ liệu**

| Tiêu chí | Mô hình "Đơn Collection" Quan sát được | Mô hình "Đa hình thái" Đề xuất |
| --- | --- | --- |
| **Toàn vẹn Dữ liệu** | Thấp. Khó khăn trong việc áp đặt các quy tắc xác thực riêng cho từng loại sản phẩm. Dễ dẫn đến dữ liệu thiếu nhất quán. | Cao. Cho phép định nghĩa các quy tắc xác thực nghiêm ngặt cho từng loại sản phẩm thông qua các schema con. |
| **Linh hoạt Truy vấn** | Rất thấp. Hầu như không thể thực hiện tìm kiếm/lọc nâng cao (faceted search) một cách hiệu quả. Phụ thuộc vào tìm kiếm văn bản chậm chạp. | Rất cao. Hỗ trợ hoàn hảo cho các truy vấn phức tạp, lọc sản phẩm theo nhiều thuộc tính đặc thù, tối ưu cho giao diện e-commerce. |
| **Khả năng Hỗ trợ Chatbot** | Gần như bằng không. Chatbot không thể thực hiện các truy vấn tư vấn có cấu trúc, làm mất đi tính "thông minh" của nó. | Rất cao. Cung cấp nền tảng dữ liệu có cấu trúc để chatbot thực hiện các truy vấn chính xác, hiểu ngữ cảnh và đưa ra tư vấn đáng tin cậy. |
| **Hiệu năng khi Mở rộng** | Kém. Hiệu năng truy vấn suy giảm nhanh chóng khi dữ liệu tăng lên do không thể sử dụng index hiệu quả cho các truy vấn phức tạp. | Tốt. Có thể tối ưu hóa hiệu năng bằng cách tạo các chỉ mục (index) chuyên biệt trên các trường chung và các trường đặc thù. |
| **Khả năng Bảo trì** | Khó khăn. Việc thêm/sửa thuộc tính sản phẩm ảnh hưởng đến toàn bộ collection. Logic ứng dụng trở nên phức tạp và rối rắm. | Dễ dàng. Dễ dàng thêm loại sản phẩm mới mà không ảnh hưởng đến các loại hiện có. Logic ứng dụng rõ ràng, mỗi loại sản phẩm được xử lý riêng. |
| **Tính Thưa thớt Dữ liệu** | Rất cao. Nhiều trường bị bỏ trống trong mỗi bản ghi, gây lãng phí không gian lưu trữ và tăng gánh nặng xử lý. | Thấp. Mỗi bản ghi chỉ chứa các trường liên quan đến loại sản phẩm của nó, giúp cấu trúc dữ liệu gọn gàng và hiệu quả. |

## Các Mẫu Thiết kế CSDL Nâng cao: Mẫu Đa hình thái (Polymorphic Pattern)

Để giải quyết các nhược điểm cố hữu của mô hình "đơn collection" phẳng, chúng ta cần áp dụng một mẫu thiết kế đã được chứng minh hiệu quả trong thế giới NoSQL, đặc biệt là MongoDB. Mẫu thiết kế phù hợp nhất cho bài toán này là **Mẫu Đa hình thái (Polymorphic Pattern)**.

### Thách thức Cốt lõi: Mô hình hóa "Những thứ vừa giống, vừa khác"

Bản chất của danh mục sản phẩm nhà thuốc là sự đa dạng trong một thể thống nhất.

* **Điểm chung (Giống nhau):** Mọi mặt hàng đều là "sản phẩm". Chúng đều có mã (sku), tên, giá bán, số lượng tồn kho, nhà sản xuất, danh mục.
* **Điểm riêng (Khác nhau):** Một viên thuốc khác biệt về bản chất so với một tuýp kem chống nắng. Thuốc có hoạt chất, số đăng ký, chỉ định. Mỹ phẩm có loại da phù hợp, chỉ số SPF. CSDL phải có khả năng mô hình hóa cả hai khía cạnh này một cách tự nhiên và hiệu quả.

### Mẫu Đa hình thái: Giải pháp Nền tảng của NoSQL

* **Định nghĩa:** Mẫu Đa hình thái là một mẫu thiết kế CSDL trong đó các tài liệu (document) trong cùng một collection chia sẻ một tập hợp các trường chung, nhưng đồng thời cũng chứa các trường hoặc các tài liệu con (sub-document) riêng biệt, đặc thù cho từng loại.
* **Khóa Phân định (Discriminator Key):** Đây là yếu tố then chốt của mẫu này. Nó là một trường đặc biệt (ví dụ, chúng ta sẽ đặt tên là productType) có giá trị xác định "hình dạng" cấu trúc của phần còn lại của tài liệu. Ví dụ:
  + Nếu productType có giá trị là 'DRUG', ứng dụng sẽ biết và kỳ vọng rằng tài liệu này sẽ chứa một tài liệu con tên là drugDetails.
  + Nếu productType có giá trị là 'COSMETIC', ứng dụng sẽ tìm kiếm tài liệu con cosmeticDetails.
* **Lợi ích của Mẫu Đa hình thái:**
  + **Dữ liệu được Đồng vị trí (Co-location):** Tất cả sản phẩm vẫn nằm trong một collection duy nhất (products). Điều này giúp đơn giản hóa rất nhiều truy vấn phổ biến, ví dụ: "tìm tất cả sản phẩm có giá dưới 50,000 VND" hoặc "liệt kê tất cả sản phẩm của hãng X".
  + **Linh hoạt về Cấu trúc:** Cho phép lưu trữ dữ liệu đặc thù, phong phú cho từng loại sản phẩm mà không gặp phải vấn đề các trường bị bỏ trống (null) như trong mô hình SQL cứng nhắc hoặc mô hình NoSQL phẳng.
  + **Thực thi ở Tầng Ứng dụng:** Logic ứng dụng (ví dụ, sử dụng Mongoose trong MERN stack) có thể đọc giá trị của khóa phân định và tự động áp dụng đúng mô hình (model) và các quy tắc xác thực (validation rules) tương ứng cho tài liệu đó. Điều này giúp mã nguồn trở nên sạch sẽ, có tổ chức và dễ bảo trì.

## Kiến trúc CSDL Đề xuất cho Hệ thống Nhà thuốc Trực tuyến

Dựa trên nguyên tắc của Mẫu Đa hình thái, kiến trúc CSDL được đề xuất bao gồm một collection trung tâm là products và một số collection tham chiếu để đảm bảo tính toàn vẹn và nhất quán của dữ liệu.

### Collection Trung tâm: products

Collection này sẽ lưu trữ thông tin về tất cả các mặt hàng có thể bán. Mỗi tài liệu trong collection này sẽ bao gồm các trường chung và một tài liệu con đặc thù tùy thuộc vào loại sản phẩm.

#### Cấu trúc các Trường chung

Đây là những trường tồn tại trong mọi tài liệu sản phẩm, bất kể loại hình.

* sku (String, required, unique): Mã định danh sản phẩm duy nhất, tương ứng với "Mã Thuốc".1 Ví dụ: "TH7807".
* name (String, required, text-indexed): Tên đầy đủ của sản phẩm. Ví dụ: "GOURCUFF 5MG ALFUZOSIN 10X10".1
* slug (String, required, unique): Tên sản phẩm được chuẩn hóa để sử dụng trên URL, thân thiện với SEO. Ví dụ: "gourcuff-5mg-alfuzosin-10x10".
* productType (String, required, indexed): **Khóa phân định**. Giá trị của trường này sẽ là một trong các chuỗi định sẵn (enum): DRUG, COSMETIC, MEDICAL\_DEVICE, SUPPLEMENT, PERSONAL\_CARE.
* description (Object): Chứa các mô tả chi tiết.
  + short (String): Mô tả ngắn gọn, hiển thị trên trang danh sách sản phẩm.
  + long (String): Mô tả đầy đủ, hiển thị trên trang chi tiết sản phẩm.
  + usageInstructions (String): Hướng dẫn sử dụng.
* images (Array of String): Một mảng chứa các URL của hình ảnh sản phẩm.
* pricing (Object): Chứa thông tin về giá.
  + importPrice (Number): Giá nhập.
  + retailPrice (Number): Giá bán lẻ.
  + wholesalePrice (Number): Giá bán sỉ/buôn (nếu có).
* inventory (Object): Quản lý tồn kho.
  + quantity (Number): Số lượng tồn kho hiện tại.
  + warningLevel (Number): Ngưỡng cảnh báo tồn kho thấp ("Số lượng cảnh báo" 1).
* brand (ObjectId, ref: 'Brand'): Một tham chiếu (reference) đến một tài liệu trong collection brands.
* categories (Array of ObjectId, ref: 'Category'): Một mảng các tham chiếu đến một hoặc nhiều tài liệu trong collection categories.
* unit (ObjectId, ref: 'UnitOfMeasure'): Tham chiếu đến đơn vị bán chính (ví dụ: Hộp, Chai, Vỉ).
* status (String): Trạng thái của sản phẩm (ví dụ: ACTIVE, INACTIVE, DISCONTINUED).
* barcode (String, unique, sparse): Mã vạch của sản phẩm.1

### Hiện thực Mẫu Đa hình thái: Các Tài liệu con Đặc thù

Dựa vào giá trị của productType, một trong các tài liệu con sau sẽ được nhúng vào tài liệu product.

#### Khi productType: 'DRUG'

Tài liệu sẽ chứa một object drugDetails với các trường:

* activeIngredients (Array of Object): Đây là một thiết kế quan trọng. Thay vì lưu chuỗi, chúng ta lưu một mảng các đối tượng, mỗi đối tượng chứa tham chiếu đến hoạt chất và hàm lượng của nó.
  + ingredient (ObjectId, ref: 'ActiveIngredient'): Tham chiếu đến collection activeIngredients.
  + content (String): Hàm lượng, ví dụ: "500mg".
* registrationNumber (String): Số đăng ký lưu hành (Số ĐK 1).
* dosageForm (String): Dạng bào chế (Dạng Bào Chế 1).
* packingWay (String): Quy cách đóng gói (QC Đóng Gói 1).
* indications (String): Chỉ định điều trị (Chỉ định).
* contraindications (String): Chống chỉ định (Chống chỉ định).
* prescriptionType (String): Loại thuốc, ví dụ: OTC (không kê đơn), PRESCRIPTION (kê đơn), CONTROLLED (kiểm soát đặc biệt).

#### Khi productType: 'COSMETIC'

Tài liệu sẽ chứa một object cosmeticDetails với các trường:

* skinTypes (Array of String): Loại da phù hợp, ví dụ: ['oily', 'combination', 'dry'].
* concerns (Array of String): Vấn đề da cần giải quyết, ví dụ: ['acne', 'aging', 'dark\_spots'].
* spf (Number): Chỉ số chống nắng SPF.
* paIndex (String): Chỉ số chống tia UVA, ví dụ: "PA+++".
* volume (String): Dung tích/Trọng lượng, ví dụ: "50ml", "100g".
* ingredients (String): Danh sách đầy đủ thành phần.

#### Khi productType: 'SUPPLEMENT'

Tài liệu sẽ chứa một object supplementDetails với các trường:

* mainIngredients (Array of String): Các thành phần chính, ví dụ: ``.
* targetAudience (String): Đối tượng sử dụng, ví dụ: "Trẻ em", "Phụ nữ mang thai", "Người cao tuổi".
* form (String): Dạng sản phẩm, ví dụ: "Viên nang mềm", "Siro", "Viên sủi".

*(Các schema tương tự có thể được định nghĩa cho MEDICAL\_DEVICE, PERSONAL\_CARE...)*

### Các Collection Tham chiếu (Referenced Collections)

Để đảm bảo tính nhất quán, tránh lặp lại dữ liệu và tăng cường khả năng truy vấn, các thông tin phụ nên được tách ra thành các collection riêng và được tham chiếu từ collection products.

* **categories**: Lưu trữ thông tin danh mục sản phẩm.
  + Schema: { name: String, slug: String, parentCategory: ObjectId }.
  + Ví dụ: { name: 'Dạ dày - Đường ruột - Tiêu hóa', slug: 'da-day-duong-ruot-tieu-hoa', parentCategory: null }.
* **brands**: Lưu trữ thông tin nhà sản xuất/thương hiệu.
  + Schema: { name: String, country: String, description: String }.
  + Ví dụ: { name: 'Mekophar', country: 'Việt Nam' }.
* **unitsOfMeasure**: Lưu trữ các đơn vị tính.
  + Schema: { name: String, abbreviation: String }.
  + Ví dụ: { name: 'Viên', abbreviation: 'viên' }, { name: 'Hộp', abbreviation: 'hộp' }.
* **activeIngredients**: Đây là một collection cực kỳ quan trọng cho tính năng tư vấn dược phẩm.
  + Schema: { name: String, description: String, therapeuticClass: String }.
  + Ví dụ: { name: 'Paracetamol', description: 'Là một loại thuốc giảm đau và hạ sốt.', therapeuticClass: 'Analgesics' }.

**Bảng 2: Cấu trúc Chi tiết cho Collection products và các Collection Hỗ trợ**

| Collection | Tên Trường | Kiểu Dữ liệu Mongoose | Mô tả / Ví dụ | Yêu cầu / Chỉ mục |
| --- | --- | --- | --- | --- |
| **products** | \_id | ObjectId | Khóa chính tự động tạo bởi MongoDB | Mặc định |
|  | sku | String | Mã sản phẩm duy nhất. Ví dụ: "TH7807" | required, unique |
|  | name | String | Tên sản phẩm. Ví dụ: "GOURCUFF 5MG" | required, text |
|  | slug | String | Chuỗi định danh trên URL | required, unique |
|  | productType | String | Khóa phân định. Ví dụ: 'DRUG' | required, indexed |
|  | description | Object | { short: String, long: String,... } |  |
|  | images |  | Mảng các URL hình ảnh |  |
|  | pricing | Object | { importPrice: Number, retailPrice: Number } | required |
|  | inventory | Object | { quantity: Number, warningLevel: Number } | required |
|  | brand | ObjectId | Tham chiếu đến brands.\_id | ref: 'Brand' |
|  | categories | [ObjectId] | Mảng tham chiếu đến categories.\_id | ref: 'Category' |
|  | unit | ObjectId | Tham chiếu đến unitsOfMeasure.\_id | ref: 'UnitOfMeasure' |
|  | status | String | Ví dụ: 'ACTIVE', 'INACTIVE' | required |
|  | barcode | String | Mã vạch sản phẩm | unique, sparse |
|  | drugDetails | Object | (Chỉ tồn tại nếu productType là 'DRUG') |  |
|  | cosmeticDetails | Object | (Chỉ tồn tại nếu productType là 'COSMETIC') |  |
| **drugDetails** | activeIngredients | [Object] |  | ref: 'ActiveIngredient' |
|  | registrationNumber | String | Số đăng ký |  |
|  | prescriptionType | String | Ví dụ: 'OTC', 'PRESCRIPTION' | indexed |
| **cosmeticDetails** | skinTypes |  | Ví dụ: ['oily', 'dry'] | indexed |
|  | spf | Number | Chỉ số chống nắng | indexed |
| **categories** | name | String | Tên danh mục | required, unique |
|  | parentCategory | ObjectId | Tham chiếu đến danh mục cha (nếu có) | ref: 'Category' |
| **brands** | name | String | Tên thương hiệu/nhà sản xuất | required, unique |
| **activeIngredients** | name | String | Tên hoạt chất | required, unique |

## Tích hợp Kiến trúc và Khả năng Đáp ứng Tính năng

Kiến trúc CSDL được đề xuất không chỉ là một sự thay đổi về mặt kỹ thuật, mà nó còn trực tiếp mở ra những khả năng mạnh mẽ cho các tính năng cốt lõi của hệ thống, điều mà mô hình cũ không thể làm được.

### Cung cấp Năng lượng cho Giao diện Bán hàng (E-commerce Frontend)

* **Tìm kiếm và Lọc Sản phẩm Nâng cao (Faceted Search):** Với cấu trúc mới, việc xây dựng một thanh công cụ lọc sản phẩm (sidebar) mạnh mẽ trở nên khả thi. Người dùng có thể dễ dàng lọc sản phẩm theo nhiều tiêu chí cùng lúc. Ví dụ, để tìm "kem chống nắng cho da dầu với SPF 50+":
  1. Frontend gửi yêu cầu đến backend với các tham số: productType=COSMETIC, skinTypes=oily, minSpf=50.
  2. Backend sử dụng MongoDB Aggregation Framework để xây dựng một truy vấn hiệu quả:  
     JavaScript  
     db.products.find({  
      productType: 'COSMETIC',  
      'cosmeticDetails.skinTypes': 'oily',  
      'cosmeticDetails.spf': { $gte: 50 }  
     })
  3. Truy vấn này cực kỳ nhanh vì nó tận dụng được các chỉ mục đã tạo trên các trường productType, cosmeticDetails.skinTypes, và cosmeticDetails.spf.
* **Trang Chi tiết Sản phẩm Động (Dynamic Product Detail Pages):** Giao diện người dùng có thể đọc trường productType của sản phẩm và tự động hiển thị các thành phần (component) giao diện phù hợp.
  + Nếu productType là DRUG, trang chi tiết sẽ hiển thị các mục "Hoạt chất", "Chỉ định", "Chống chỉ định".
  + Nếu productType là COSMETIC, trang chi tiết sẽ hiển thị "Loại da phù hợp", "Chỉ số SPF/PA", "Thành phần".  
    Điều này tạo ra một trải nghiệm người dùng chuyên nghiệp và cung cấp thông tin rõ ràng, mạch lạc.

### Kích hoạt một Chatbot Tư vấn Thông minh

Đây là nơi kiến trúc CSDL mới thực sự tỏa sáng và là yếu tố quyết định cho thành công của luận văn. Chatbot giờ đây có thể thực hiện các truy vấn có ngữ cảnh và cung cấp giá trị tư vấn thực sự.

* **Truy vấn Dựa trên Ngữ cảnh:**
  + **Người dùng:** "Tôi bị đau đầu, sổ mũi."
  + **Chatbot (Logic):**
    1. Sử dụng một dịch vụ Xử lý Ngôn ngữ Tự nhiên (NLU) để nhận diện các thực thể (entities): symptom: 'đau đầu', symptom: 'sổ mũi'.
    2. Thực hiện truy vấn CSDL: db.products.find({ productType: 'DRUG', 'drugDetails.indications': { $all: [/đau đầu/i, /sổ mũi/i] } }).
    3. Trả về danh sách các loại thuốc phù hợp.
  + **Người dùng:** "Thuốc GOURCUFF có dùng được cho phụ nữ có thai không?"
  + **Chatbot (Logic):**
    1. Tìm sản phẩm có name là "GOURCUFF".
    2. Kiểm tra trường drugDetails.contraindications của sản phẩm đó.
    3. Nếu trường này chứa các từ khóa như "phụ nữ có thai", "mang thai", chatbot sẽ đưa ra cảnh báo.
  + **Người dùng:** "Tìm các thuốc khác cũng có hoạt chất Alfuzosin."
  + **Chatbot (Logic):**
    1. Đầu tiên, tìm ObjectId của hoạt chất "Alfuzosin" trong collection activeIngredients.
    2. Sau đó, thực hiện một truy vấn cực kỳ hiệu quả: db.products.find({ 'drugDetails.activeIngredients.ingredient': ObjectId('...') }).
    3. Hành động này không chỉ nhanh mà còn cho phép chatbot cung cấp thông tin bổ sung về chính hoạt chất đó bằng cách tra cứu trong collection activeIngredients.

Việc tách activeIngredients thành một collection riêng là một quyết định kiến trúc mang lại lợi ích to lớn. Nó biến chatbot từ một công cụ tìm kiếm văn bản đơn thuần thành một trợ lý dược phẩm có khả năng hiểu và liên kết các khái niệm y khoa.

## Chiến lược Triển khai và Các Thực hành Tốt nhất trong MERN Stack

Để hiện thực hóa kiến trúc này trong MERN stack, cần sử dụng các công cụ và kỹ thuật phù hợp.

### Định nghĩa và Xác thực Schema với Mongoose

Mongoose cung cấp một tính năng mạnh mẽ gọi là **Discriminators**, được thiết kế đặc biệt để hiện thực hóa Mẫu Đa hình thái.

* **Cách tiếp cận:**
  1. Định nghĩa một BaseProductSchema chứa tất cả các trường chung.
  2. Sử dụng BaseProductSchema để tạo một model Product.
  3. Định nghĩa các schema riêng cho từng loại sản phẩm (ví dụ: DrugSchema, CosmeticSchema).
  4. Sử dụng phương thức Product.discriminator('Drug', DrugSchema) để báo cho Mongoose biết rằng khi một tài liệu có productType là 'DRUG', nó nên sử dụng DrugSchema để xác thực và xử lý.
* **Ví dụ mã nguồn (minh họa):**  
  JavaScript  
  const mongoose = require('mongoose');  
  const { Schema } = mongoose;  
    
  // 1. Định nghĩa Schema cơ sở  
  const baseOptions = {  
   discriminatorKey: 'productType', // Khóa phân định  
   collection: 'products',  
  };  
  const BaseProductSchema = new Schema({  
   name: { type: String, required: true },  
   sku: { type: String, required: true, unique: true },  
   //... các trường chung khác  
  }, baseOptions);  
    
  // 2. Tạo Model cơ sở  
  const Product = mongoose.model('Product', BaseProductSchema);  
    
  // 3. Định nghĩa Schema cho loại 'DRUG'  
  const DrugSchema = new Schema({  
   drugDetails: {  
   registrationNumber: String,  
   //... các trường thuốc khác  
   }  
  });  
    
  // 4. Áp dụng Discriminator  
  const Drug = Product.discriminator('DRUG', DrugSchema);  
    
  // Tương tự cho Cosmetic, MedicalDevice...  
  const CosmeticSchema = new Schema({  
   cosmeticDetails: {  
   spf: Number,  
   skinTypes:  
   }  
  });  
  const Cosmetic = Product.discriminator('COSMETIC', CosmeticSchema);  
    
  module.exports = { Product, Drug, Cosmetic };

### Tối ưu Hiệu năng qua Chỉ mục (Indexing)

Để đảm bảo hệ thống phản hồi nhanh chóng, việc tạo chỉ mục trên các trường thường xuyên được truy vấn là bắt buộc.

* **Các chỉ mục đề xuất:**
  + Trên sku: { sku: 1 } (unique).
  + Trên slug: { slug: 1 } (unique).
  + Trên productType: { productType: 1 } để lọc nhanh theo loại sản phẩm.
  + **Chỉ mục phức hợp (Compound Index) cho các truy vấn phổ biến:**
    - { productType: 1, 'cosmeticDetails.skinTypes': 1, 'cosmeticDetails.spf': -1 }: Tối ưu cho việc lọc mỹ phẩm.
    - { productType: 1, 'drugDetails.activeIngredients.ingredient': 1 }: Tối ưu cho việc tìm kiếm thuốc theo hoạt chất.
  + **Chỉ mục văn bản (Text Index):**
    - { name: 'text', 'description.short': 'text' }: Hỗ trợ tìm kiếm văn bản tự do cơ bản.

### Kế hoạch Di chuyển Dữ liệu (Data Migration)

Để chuyển đổi dữ liệu từ tệp Excel phẳng 1 sang cấu trúc CSDL mới, cần viết một kịch bản (script) di chuyển một lần, ví dụ bằng Node.js.

1. **Đọc Dữ liệu:** Sử dụng một thư viện như xlsx để đọc từng dòng trong tệp Excel.
2. **Xử lý và Ánh xạ:** Lặp qua mỗi dòng:
   * Dựa vào giá trị của cột "Nhóm Thuốc", xác định giá trị cho productType.
   * Tạo các tài liệu cho các collection tham chiếu (categories, brands, unitsOfMeasure) một cách linh hoạt. Sử dụng một đối tượng Map để lưu trữ các giá trị đã tạo và ID của chúng, tránh tạo bản ghi trùng lặp.
   * Xây dựng đối tượng JSON hoàn chỉnh cho sản phẩm, bao gồm các trường chung và tài liệu con đặc thù (drugDetails, cosmeticDetails,...).
3. **Ghi vào CSDL:** Sử dụng các model Mongoose đã định nghĩa để chèn (insert) tài liệu sản phẩm đã được cấu trúc lại vào MongoDB.

## Kết luận và Khuyến nghị

Việc phân tích mô hình CSDL của nhà thuốc hiện hữu cho thấy đó là một giải pháp thực dụng nhưng mang tính kế thừa, không còn phù hợp với yêu cầu của một ứng dụng web hiện đại, đặc biệt là một ứng dụng có tham vọng tích hợp trí tuệ nhân tạo như chatbot tư vấn. Mô hình "đơn collection" phẳng gây ra các vấn đề nghiêm trọng về hiệu năng, tính toàn vẹn dữ liệu và khả năng bảo trì, đồng thời là rào cản kỹ thuật không thể vượt qua đối với các tính năng truy vấn thông minh.

**Khuyến nghị chính thức cho đề tài luận văn này là hoàn toàn không sao chép mô hình CSDL đã quan sát được.**

Thay vào đó, kiến nghị được đưa ra là áp dụng một kiến trúc CSDL hiện đại dựa trên **Mẫu Thiết kế Đa hình thái (Polymorphic Pattern)** kết hợp với việc chuẩn hóa dữ liệu tham chiếu. Kiến trúc này mang lại các lợi ích vượt trội:

1. **Tính toàn vẹn và Cấu trúc:** Đảm bảo dữ liệu được lưu trữ một cách nhất quán, có cấu trúc rõ ràng và dễ dàng xác thực.
2. **Hiệu năng Vượt trội:** Cho phép tạo các chỉ mục chuyên biệt, giúp các truy vấn phức tạp, đặc biệt là cho tính năng lọc sản phẩm và chatbot, được thực thi với tốc độ cao.
3. **Nền tảng cho Tính năng Thông minh:** Cung cấp một nền tảng dữ liệu vững chắc để chatbot có thể thực hiện các phân tích và truy vấn có ngữ cảnh, mang lại giá trị tư vấn thực sự cho người dùng.
4. **Khả năng Bảo trì và Mở rộng:** Giúp mã nguồn phía backend trở nên sạch sẽ, có tổ chức, và dễ dàng mở rộng hệ thống trong tương lai bằng cách thêm các loại sản phẩm mới mà không phá vỡ cấu trúc hiện có.

Việc lựa chọn và triển khai kiến trúc được đề xuất không chỉ giúp xây dựng một sản phẩm chất lượng cao mà còn thể hiện một trình độ kỹ thuật và tư duy thiết kế hệ thống vững chắc, là một điểm cộng lớn cho giá trị học thuật của một đề tài luận văn tốt nghiệp ngành Công nghệ thông tin.

#### Nguồn trích dẫn

1. Drugs\_11846\_10082025214352.xlsx