# Báo cáo Phân tích và Tái thiết kế Cơ sở dữ liệu cho Hệ thống Quản lý Nhà thuốc Trực tuyến theo Chuẩn Mực Hiện đại

## Phần 1: Phân tích Chuyên sâu về Mô hình Cơ sở dữ liệu Hiện tại

### 1.1. Đặc điểm của Mô hình "Bảng Phẳng" (Flat-Table Model)

Mô hình cơ sở dữ liệu (CSDL) hiện tại, được minh họa qua tệp dữ liệu Drugs\_11846\_10082025214352.xlsx và các giao diện quản trị được cung cấp (Image 1, Image 2, Image 3), là một ví dụ điển hình của thiết kế "bảng phẳng" hay "siêu bảng" (mega-table). Đặc điểm cốt lõi của mô hình này là tập trung toàn bộ các loại mặt hàng—từ thuốc kê đơn (TH7807), dược mỹ phẩm (TH7800), thực phẩm chức năng (TH7799) đến thiết bị y tế (TH7787)—vào một cấu trúc bảng duy nhất với hơn 40 thuộc tính (cột).1

Một phân tích sâu hơn cho thấy thiết kế này phản ánh một tư duy kỹ thuật lấy nghiệp vụ quản lý kho làm trung tâm (inventory-centric). Các trường dữ liệu như Vị trí tủ, kho, Số dư đầu kỳ, Giá Nhập, Số Lô, Hạn Dùng là những yếu tố nền tảng cho một hệ thống bán hàng tại điểm (Point-of-Sale - POS) hoặc một phần mềm quản lý kho vận hành ở quy mô nhỏ.1

### 1.2. Ưu điểm (Bề nổi và Ngắn hạn)

Mô hình này sở hữu một số ưu điểm dễ nhận thấy trong giai đoạn triển khai ban đầu và vận hành cơ bản:

* **Đơn giản trong triển khai ban đầu:** Với một bảng dữ liệu duy nhất, việc xây dựng các chức năng cơ bản như tạo, đọc, cập nhật, xóa (CRUD) trở nên nhanh chóng. Giao diện "Thêm mới thuốc" (Image 2, Image 3) chỉ cần một biểu mẫu (form) duy nhất để nhập liệu cho mọi loại sản phẩm, giúp giảm thiểu thời gian phát triển ban đầu.
* **Truy vấn toàn bộ sản phẩm dễ dàng:** Việc hiển thị một danh sách tổng hợp tất cả sản phẩm như trong giao diện "DANH SÁCH THUỐC" (Image 1) có thể được thực hiện bằng một câu lệnh truy vấn đơn giản, không yêu cầu các phép nối (JOIN) phức tạp thường thấy trong CSDL quan hệ.

Sự đơn giản này chính là lý do tại sao hệ thống hiện tại được đánh giá là "làm việc khá hiệu quả" trong bối cảnh vận hành của một nhà thuốc vật lý. Khi các yêu cầu nghiệp vụ chủ yếu xoay quanh việc kiểm kê, nhập/xuất hàng và bán hàng cơ bản, mô hình này đáp ứng đủ. Tuy nhiên, chính sự đơn giản này lại là một "món nợ kỹ thuật" (technical debt) tiềm ẩn. Nó sẽ trở thành rào cản lớn khi hệ thống cần phát triển các tính năng phức tạp hơn, đặc biệt là các tính năng hướng tới người dùng cuối trên một nền tảng trực tuyến hiện đại. Góc nhìn "hiệu quả" của người quản lý kho ("tôi có thể xem tất cả hàng hóa, giá cả, tồn kho ở một nơi") sẽ xung đột trực tiếp với nhu cầu của khách hàng trực tuyến ("làm sao tôi tìm được kem chống nắng cho da dầu, không chứa cồn, xuất xứ từ Pháp?"). Mô hình hiện tại không thể trả lời câu hỏi thứ hai một cách hiệu quả, do đó, sự "hiệu quả" trong quá khứ sẽ là nguyên nhân gây ra sự "kém hiệu quả" trong tương lai khi mục tiêu của hệ thống thay đổi.

### 1.3. Phân tích Nhược điểm Cốt lõi và Rủi ro Kỹ thuật Dài hạn

Mặc dù có vẻ hiệu quả cho các tác vụ đơn giản, mô hình bảng phẳng bộc lộ nhiều điểm yếu nghiêm trọng khi áp dụng cho một hệ thống trực tuyến phức tạp, có khả năng mở rộng.

* **Dữ liệu thưa thớt (Sparse Data) và Lãng phí Tài nguyên:** Phân tích tệp dữ liệu cho thấy sự lãng phí rõ rệt.1 Một sản phẩm là thiết bị y tế như TH7787 (Khẩu trang Dolphin Mask) sẽ có hàng loạt cột như Số ĐK, Hoạt Chất, Hàm Lượng, Dạng Bào Chế bị bỏ trống (null). Ngược lại, một loại thuốc như TH7788 (ROSUVASTATIN 20MG) sẽ không sử dụng các thuộc tính dành cho mỹ phẩm. Trong một CSDL NoSQL như MongoDB, mặc dù giá trị null không chiếm nhiều dung lượng đĩa, việc có quá nhiều trường không cần thiết trong mỗi tài liệu (document) vẫn làm tăng kích thước metadata, tiêu tốn bộ nhớ RAM khi tải dữ liệu và làm phức tạp hóa logic xử lý ở tầng ứng dụng.
* **Phình to Chỉ mục (Index Bloat) và Suy giảm Hiệu năng:** Để hỗ trợ tìm kiếm nhanh, hệ thống sẽ cần đánh chỉ mục (index) trên các cột quan trọng. Ví dụ, để tìm thuốc theo hoạt chất, một chỉ mục trên cột Hoạt Chất là cần thiết. Do tất cả sản phẩm nằm chung một bảng, chỉ mục này sẽ chứa cả hàng ngàn giá trị null từ các sản phẩm không phải là thuốc. Điều này làm chỉ mục phình to, chiếm dụng bộ nhớ và làm chậm tất cả các thao tác ghi (insert, update, delete) vì mọi thay đổi đều phải cập nhật nhiều chỉ mục lớn, gây suy giảm hiệu năng toàn hệ thống.
* **Thiếu Toàn vẹn và Ràng buộc Dữ liệu (Lack of Data Integrity):** Mô hình phẳng không cho phép áp đặt các quy tắc nghiệp vụ riêng cho từng loại sản phẩm ở cấp độ CSDL. Ví dụ, không thể bắt buộc trường Số ĐK (Số đăng ký) phải có giá trị đối với "Thuốc" nhưng không bắt buộc đối với "Dược mỹ phẩm". Mọi logic kiểm tra đều phải được xử lý ở tầng ứng dụng, dẫn đến nguy cơ cao về dữ liệu không nhất quán và sai sót khi nhập liệu.
* **Khó khăn trong Bảo trì và Mở rộng (Low Maintainability & Scalability):** Khi có nhu cầu thêm một loại sản phẩm mới, ví dụ "Sữa công thức" với các thuộc tính đặc thù như "Độ tuổi sử dụng", "Thành phần dinh dưỡng", lập trình viên buộc phải thêm các cột mới vào cấu trúc chung. Điều này làm cho lược đồ (schema) ngày càng cồng kềnh, khó quản lý, và mọi thay đổi đều có nguy cơ ảnh hưởng đến toàn bộ hệ thống, đi ngược lại các nguyên tắc phát triển phần mềm linh hoạt.
* **Hạn chế Khả năng Truy vấn Phức tạp và Trải nghiệm Người dùng:** Đây là điểm yếu chí mạng khi đối chiếu với mục tiêu xây dựng một hệ thống "hiện đại như Long Châu". Hệ thống của Long Châu cho phép người dùng thực hiện tìm kiếm và lọc đa chiều (faceted search): lọc đồng thời theo danh mục sản phẩm, thương hiệu, khoảng giá, nơi sản xuất, đối tượng sử dụng, và nhiều tiêu chí khác.2 Việc thực hiện một truy vấn phức tạp như vậy trên mô hình phẳng sẽ cực kỳ chậm, yêu cầu quét toàn bộ bảng hoặc sử dụng các chỉ mục phức hợp không hiệu quả. Điều này dẫn đến thời gian phản hồi của trang web và chatbot kéo dài, phá hỏng trải nghiệm người dùng và không đáp ứng được kỳ vọng về một hệ thống hiện đại, phản hồi tức thì.

## Phần 2: Kiến trúc Cơ sở dữ liệu Hiện đại: Mô hình Đa hình (Polymorphic Pattern) và Phân tách Thực thể

Để khắc phục những hạn chế của mô hình hiện tại và xây dựng một nền tảng vững chắc cho các tính năng nâng cao, một kiến trúc CSDL hiện đại dựa trên các nguyên tắc thiết kế cho dữ liệu không đồng nhất là cần thiết.

### 2.1. Nguyên tắc Thiết kế Hiện đại

* **Mô hình Đa hình (Polymorphic Pattern):** Đây là một giải pháp kiến trúc tiêu chuẩn trong MongoDB để quản lý các thực thể có cấu trúc khác nhau nhưng cùng chung một "bản chất" (đều là sản phẩm) trong cùng một collection. Cách tiếp cận này sử dụng một "trường phân loại" (discriminator key) để xác định loại sản phẩm (productType) và từ đó áp dụng một lược đồ (schema) riêng cho từng loại. Điều này giúp giữ được sự linh hoạt của NoSQL trong khi vẫn đảm bảo cấu trúc và tính toàn vẹn cho từng loại dữ liệu cụ thể.
* **Phân tách Thực thể (Entity Separation / Normalization by Reference):** Các dữ liệu có tính lặp lại cao và có thể quản lý độc lập (như thương hiệu, danh mục, hoạt chất, nhà sản xuất) cần được tách ra các collection riêng. Thay vì lưu trữ chuỗi "GSK" trong hàng trăm sản phẩm, hệ thống sẽ lưu một ObjectId tham chiếu đến tài liệu "GSK" trong collection brands. Điều này giúp chuẩn hóa dữ liệu, dễ dàng cập nhật (chỉ cần sửa thông tin ở một nơi duy nhất), giảm thiểu sai sót nhập liệu và tạo nền tảng vững chắc cho các tính năng như trang danh sách sản phẩm theo thương hiệu hoặc nhà sản xuất.

### 2.2. Thiết kế Lược đồ Chi tiết cho MongoDB (Tối ưu cho MERN Stack)

Dựa trên các nguyên tắc trên, cấu trúc CSDL mới được đề xuất như sau:

#### Collection products - Áp dụng Polymorphic Pattern

Đây là collection trung tâm, chứa thông tin của tất cả sản phẩm.

* **Lược đồ Nền tảng (ProductBase):** Bao gồm các trường chung cho MỌI sản phẩm.
  + sku: String, unique, indexed (Mã sản phẩm, ví dụ: TH7807).
  + name: String, text-indexed (Tên sản phẩm, ví dụ: GOURCUFF 5MG ALFUZOSIN 10X10).
  + slug: String, unique, indexed (Tên sản phẩm thân thiện với URL, ví dụ: gourcuff-5mg-alfuzosin-10x10).
  + images: Mảng các chuỗi (URL của hình ảnh sản phẩm).
  + brand: ObjectId, ref: 'Brand', indexed (Tham chiếu đến collection brands).
  + categories: Mảng các ObjectId [ObjectId], ref: 'Category', indexed (Cho phép sản phẩm thuộc nhiều danh mục).
  + productType: String, indexed (Trường phân loại, ví dụ: 'Drug', 'Cosmeceutical', 'MedicalDevice', 'FunctionalFood'). Đây chính là "discriminator key".
  + shortDescription: String (Mô tả ngắn gọn).
  + fullDescription: String (Mô tả chi tiết, có thể chứa định dạng HTML).
  + status: String (Trạng thái sản phẩm, ví dụ: 'active', 'inactive').
  + variants: Mảng các đối tượng con. Thay vì mỗi quy cách đóng gói là một sản phẩm riêng, hệ thống sẽ quản lý một sản phẩm gốc với nhiều biến thể.
    - **VariantSchema** sẽ chứa: variantSku (mã SKU riêng cho biến thể), packingWay (ví dụ: "Hộp 10 vỉ x 10 viên"), barcode, prices (đối tượng chứa giá nhập, giá bán lẻ, giá sỉ), và inventory (đối tượng chứa số lượng tồn kho và vị trí).
* **Lược đồ Mở rộng (Discriminators) dựa trên productType:**
  + **Khi productType là 'Drug':** Kế thừa ProductBase và thêm các trường đặc thù cho thuốc.
    - activeSubstances: Mảng đối tượng `` (Hoạt chất và hàm lượng).
    - dosageForm: ObjectId, ref: 'Attribute' (Dạng bào chế, ví dụ: "Viên nén").
    - registrationNumber: String (Số đăng ký).
    - isPrescriptionRequired: Boolean (Thuốc cần kê toa hay không).
    - indications, contraindications, dosage, sideEffects: Các trường văn bản mô tả chỉ định, chống chỉ định, liều dùng, tác dụng phụ.
  + **Khi productType là 'Cosmeceutical':** Kế thừa ProductBase và thêm các trường cho dược mỹ phẩm.3
    - skinSolutions: Mảng [ObjectId], ref: 'Attribute' (Tham chiếu đến các giải pháp như "Da mụn", "Chống lão hóa").
    - skinTypes: Mảng [ObjectId], ref: 'Attribute' (Tham chiếu đến các loại da như "Da dầu", "Da nhạy cảm").
    - spfIndex: String (Chỉ số chống nắng SPF).
    - ingredients: String (Thành phần).
  + **Khi productType là 'MedicalDevice':** Kế thừa ProductBase và thêm các trường cho thiết bị y tế.
    - technicalSpecs: Object (Thông số kỹ thuật).
    - warrantyPeriod: String (Thời gian bảo hành).
    - accessories: Mảng (Phụ kiện đi kèm).
  + **Khi productType là 'FunctionalFood':** Kế thừa ProductBase và thêm các trường cho thực phẩm chức năng.2
    - mainBenefits: Mảng [ObjectId], ref: 'Attribute' (Tham chiếu đến các công dụng chính như "Bổ mắt", "Tăng đề kháng").
    - userGroup: Mảng [ObjectId], ref: 'Attribute' (Tham chiếu đến đối tượng sử dụng như "Phụ nữ có thai", "Người cao tuổi").

### 2.3. Thiết kế các Collection Tham chiếu (Supporting Collections)

* **Collection categories:**
  + name, slug, parent (ObjectId, tham chiếu đến chính nó), ancestors ([ObjectId]). Cấu trúc này, sử dụng mẫu Materialized Paths, cho phép truy vấn toàn bộ cây danh mục hoặc một nhánh con một cách cực kỳ hiệu quả, giúp tái tạo lại menu đa cấp phức tạp như của Long Châu.2
* **Collection brands:**
  + name, slug, logo, countryOfOrigin.
* **Collection attributes:**
  + name: String (ví dụ: "Viên nén", "Da dầu", "Bổ mắt").
  + type: String, indexed (ví dụ: "dosageForm", "skinType", "mainBenefit").
  + Thiết kế này mang lại sự linh hoạt tối đa. Nó cho phép người quản trị hệ thống có thể thêm các tùy chọn lọc mới (ví dụ, thêm một "Dạng bào chế" mới) mà không cần thay đổi mã nguồn. Giao diện người dùng có thể tự động tạo ra các bộ lọc bằng cách truy vấn db.attributes.find({ type: 'dosageForm' }).
* **Collection activeSubstances:**
  + name, description.

#### Bảng 1: So sánh Cấu trúc Dữ liệu Hiện tại và Đề xuất

| Thuộc tính nghiệp vụ | Mô hình Hiện tại (Flat-Table) | Mô hình Đề xuất (Polymorphic & Relational) | Lợi ích Kỹ thuật và Nghiệp vụ |
| --- | --- | --- | --- |
| **Hoạt chất** | Cột Hoạt Chất (kiểu String) | activeSubstances (Mảng tham chiếu đến collection ActiveSubstance) | Chuẩn hóa dữ liệu, tránh sai sót chính tả, cho phép truy vấn "tìm tất cả thuốc chứa Paracetamol".6 |
| **Danh mục** | Cột Nhóm Thuốc (kiểu String) | categories (Mảng tham chiếu đến collection Category với cấu trúc cây) | Hỗ trợ danh mục đa cấp, gán sản phẩm vào nhiều danh mục, tối ưu cho menu và breadcrumbs.3 |
| **Quy cách đóng gói** | Mỗi quy cách là một dòng sản phẩm riêng | variants (Mảng các đối tượng con trong cùng một sản phẩm) | Giảm trùng lặp dữ liệu, quản lý tập trung, cải thiện trải nghiệm người dùng trên trang chi tiết sản phẩm. |
| **Dạng bào chế** | Cột Dạng Bào Chế (kiểu String) | dosageForm (Tham chiếu đến collection Attribute) | Dữ liệu nhất quán, dễ dàng tạo bộ lọc động trên giao diện, quản lý tập trung các dạng bào chế. |

## Phần 3: Triển khai và Tối ưu hóa trong Hệ sinh thái MERN

### 3.1. Hiện thực hóa bằng Mongoose và Discriminators

Để triển khai kiến trúc này trong MERN stack, thư viện Mongoose cung cấp một tính năng mạnh mẽ gọi là "Discriminators", hoàn toàn phù hợp với mô hình đa hình đã đề xuất.

**Đoạn mã 1: Định nghĩa ProductBaseSchema và VariantSchema**

JavaScript

const mongoose = require('mongoose');  
const { Schema } = mongoose;  
  
// Lược đồ cho các biến thể sản phẩm (quy cách đóng gói)  
const variantSchema = new Schema({  
 variantSku: { type: String, required: true, unique: true },  
 packingWay: { type: String, required: true }, // VD: "Hộp 3 vỉ x 10 viên"  
 barcode: { type: String, index: true },  
 prices: {  
 import: { type: Number, default: 0 },  
 retail: { type: Number, default: 0 },  
 wholesale: { type: Number, default: 0 }  
 },  
 inventory: {  
 quantity: { type: Number, default: 0 },  
 location: String // VD: "Kệ A1, Tầng 2"  
 }  
}, { \_id: false });  
  
// Lược đồ cơ sở cho tất cả sản phẩm  
const baseOptions = {  
 discriminatorKey: 'productType', // Trường dùng để phân loại  
 collection: 'products'  
};  
  
const ProductBaseSchema = new Schema({  
 sku: { type: String, required: true, unique: true, index: true },  
 name: { type: String, required: true, trim: true, text: true },  
 slug: { type: String, required: true, unique: true, index: true },  
 images:,  
 brand: { type: Schema.Types.ObjectId, ref: 'Brand', index: true },  
 categories:,  
 shortDescription: String,  
 fullDescription: String,  
 status: { type: String, enum: ['active', 'inactive'], default: 'active' },  
 variants:  
}, baseOptions);  
  
const Product = mongoose.model('Product', ProductBaseSchema);

**Đoạn mã 2: Định nghĩa DrugSchema và CosmeceuticalSchema (Sử dụng Discriminators)**

JavaScript

// Lược đồ cho sản phẩm là Thuốc  
const Drug = Product.discriminator('Drug', new Schema({  
 activeSubstances:,  
 dosageForm: { type: Schema.Types.ObjectId, ref: 'Attribute' },  
 registrationNumber: String,  
 isPrescriptionRequired: { type: Boolean, default: false },  
 indications: String,  
 contraindications: String,  
 dosage: String  
}));  
  
// Lược đồ cho sản phẩm là Dược mỹ phẩm  
const Cosmeceutical = Product.discriminator('Cosmeceutical', new Schema({  
 skinSolutions:,  
 skinTypes:,  
 spfIndex: String,  
 ingredients: String  
}));  
  
// Tương tự cho MedicalDevice và FunctionalFood...

Các đoạn mã này cung cấp một bộ khung hoàn chỉnh và sẵn sàng để tích hợp vào một dự án MERN, minh họa rõ ràng cách triển khai kỹ thuật của mô hình đã đề xuất, phù hợp với yêu cầu của một luận văn công nghệ thông tin.

### 3.2. Chiến lược Đánh chỉ mục (Indexing Strategy) cho Hiệu năng Tối đa

Một chiến lược đánh chỉ mục được thiết kế tốt là yếu tố then chốt để đảm bảo hiệu năng của ứng dụng khi dữ liệu tăng lên.

* **Chỉ mục đơn (Single Field Indexes):** Cần thiết cho các trường thường xuyên được dùng để truy vấn độc lập: sku, slug, brand, categories, productType, variants.barcode.
* **Chỉ mục phức hợp (Compound Indexes):** Tối ưu cho các truy vấn lọc và sắp xếp phức tạp.
  + { "categories": 1, "brand": 1, "variants.prices.retail": 1 }: Tối ưu hóa cho trang danh mục sản phẩm khi người dùng lọc theo thương hiệu và sắp xếp theo giá.
  + { "productType": 1, "skinTypes": 1, "variants.prices.retail": -1 }: Tối ưu cho việc tìm kiếm dược mỹ phẩm theo loại da và sắp xếp theo giá giảm dần.
* **Chỉ mục văn bản (Text Index):**
  + { name: 'text', 'brand.name': 'text', 'activeSubstances.name': 'text' }: Phục vụ chức năng tìm kiếm toàn văn, cho phép người dùng tìm theo tên sản phẩm, tên thương hiệu, và cả tên hoạt chất.

Chiến lược này trực tiếp giải quyết vấn đề hiệu năng yếu kém của mô hình phẳng, đảm bảo hệ thống có thể mở rộng quy mô và cung cấp trải nghiệm tìm kiếm nhanh chóng cho người dùng.

### 3.3. Tích hợp với Chatbot Tư vấn Sản phẩm

Cấu trúc dữ liệu có tổ chức và chatbot thông minh có mối quan hệ cộng sinh. Dữ liệu tốt cho phép chatbot thực hiện các truy vấn phức tạp, và một chatbot thông minh sẽ khai thác tối đa tiềm năng của cấu trúc CSDL. Thiết kế này là nền tảng cốt lõi cho tính năng AI trong đề tài.

* **Kịch bản 1: "Tìm thuốc trị đau đầu cho người bị dạ dày."**
  1. **Logic Chatbot (Phân tích):** Chatbot phân tích ý định là "tìm thuốc". Nó trích xuất triệu chứng "đau đầu" và điều kiện "người bị dạ dày".
  2. **Logic Chatbot (Ánh xạ):** "Đau đầu" được ánh xạ vào trường indications. "Bị dạ dày" được xem là một yếu tố cần tránh, ánh xạ vào trường contraindications.
  3. **Truy vấn CSDL:** Chatbot tạo ra một truy vấn MongoDB chính xác: Product.find({ productType: 'Drug', indications: /đau đầu/i, contraindications: { $not: /dạ dày/i } }).
* **Kịch bản 2: "Tôi cần bổ sung canxi cho bà bầu."**
  1. **Logic Chatbot (Phân tích):** Chatbot xác định ý định "tìm thực phẩm chức năng". Nó trích xuất công dụng "bổ sung canxi" và đối tượng sử dụng "bà bầu".
  2. **Logic Chatbot (Ánh xạ):** "Bổ sung canxi" được ánh xạ tới một ObjectId cụ thể trong collection attributes (với type: 'mainBenefit'). Tương tự, "bà bầu" được ánh xạ tới một ObjectId khác (với type: 'userGroup').
  3. **Truy vấn CSDL:** Product.find({ productType: 'FunctionalFood', mainBenefits: 'objectId\_cua\_bo\_sung\_canxi', userGroup: 'objectId\_cua\_ba\_bau' }).

Các kịch bản này cho thấy rằng một CSDL được cấu trúc tốt không chỉ là một phần kỹ thuật phụ trợ mà là nền tảng cho phép chatbot thực hiện các tư vấn ngữ nghĩa phức tạp, nâng cao giá trị học thuật và thực tiễn của luận văn.

## Phần 4: Lộ trình Triển khai và Kết luận

### 4.1. Lộ trình Triển khai

1. **Bước 1: Định nghĩa Schema:** Dựa trên các đoạn mã ở mục 3.1, hoàn thiện các file schema Mongoose cho Product (với đầy đủ discriminators), Category, Brand, Attribute, và ActiveSubstance.
2. **Bước 2: Di chuyển Dữ liệu (Data Migration):** Viết một script Node.js để đọc tệp Excel.1 Script này sẽ duyệt qua từng dòng, xử lý và chuẩn hóa dữ liệu. Với các thực thể như thương hiệu, danh mục, hoạt chất, script sẽ sử dụng lệnh findOneAndUpdate với tùy chọn upsert: true để tạo mới nếu chưa tồn tại hoặc lấy \_id nếu đã có. Cuối cùng, tạo tài liệu sản phẩm tương ứng trong collection products với các ObjectId tham chiếu chính xác.
3. **Bước 3: Xây dựng API:** Sử dụng Express.js để tạo các API endpoints cho các chức năng chính:
   * GET /products: API tìm kiếm và lọc sản phẩm đa năng, nhận các tham số query như category, brand, price\_min, price\_max, q (cho tìm kiếm text), và các thuộc tính đặc thù khác.
   * GET /products/:slug: API lấy chi tiết một sản phẩm, sử dụng $lookup để "populate" thông tin từ các collection tham chiếu (thương hiệu, danh mục).
   * GET /categories: API lấy toàn bộ cây danh mục để hiển thị menu.
   * GET /attributes?type=...: API lấy các tùy chọn cho một bộ lọc cụ thể trên giao diện (ví dụ: ?type=skinType để lấy danh sách các loại da).
4. **Bước 4: Tích hợp Frontend và Chatbot:** Kết nối giao diện React và logic của chatbot với các API đã xây dựng để hoàn thiện hệ thống.

### 4.2. Kết luận

Mô hình CSDL "bảng phẳng" hiện tại, dù tỏ ra hiệu quả cho các nghiệp vụ quản lý kho cơ bản, lại là một rào cản kỹ thuật không thể vượt qua đối với việc xây dựng một hệ thống bán thuốc trực tuyến hiện đại, tích hợp chatbot thông minh. Các nhược điểm cố hữu về hiệu năng, khả năng mở rộng, tính toàn vẹn dữ liệu và đặc biệt là khả năng hỗ trợ truy vấn phức tạp khiến nó hoàn toàn không phù hợp với mục tiêu của dự án.

Kiến trúc được đề xuất, dựa trên **Mô hình Đa hình** và **Phân tách Thực thể**, không chỉ giải quyết triệt để các vấn đề trên mà còn cung cấp một nền tảng vững chắc, linh hoạt và có khả năng mở rộng. Nó cho phép quản lý các loại sản phẩm đa dạng một cách có cấu trúc, tối ưu hóa hiệu năng truy vấn cho các tính năng lọc đa chiều, và quan trọng nhất, tạo điều kiện để chatbot thực hiện các tư vấn ngữ nghĩa phức tạp, mang lại giá trị thực tiễn cao.

Việc đầu tư thời gian và công sức vào một thiết kế CSDL vững chắc ngay từ đầu là yếu tố quyết định sự thành công của một dự án phức tạp như "hệ thống quản lý nhà thuốc trực tuyến hiện đại". Đây sẽ là một điểm nhấn quan trọng cả về mặt kỹ thuật lẫn học thuật trong luận văn tốt nghiệp.

#### Nguồn trích dẫn

1. Drugs\_11846\_10082025214352.xlsx
2. Nhà thuốc FPT Long Châu - Số 1 thuốc kê đơn, truy cập vào tháng 10 8, 2025, <https://nhathuoclongchau.com.vn/>
3. Dược mỹ phẩm chính hãng từ các thương hiệu lớn - Nhà thuốc ..., truy cập vào tháng 10 8, 2025, <https://nhathuoclongchau.com.vn/duoc-my-pham>
4. Sản phẩm chăm sóc cá nhân giá tốt Tại Nhà Thuốc Long Châu, truy cập vào tháng 10 8, 2025, <https://nhathuoclongchau.com.vn/cham-soc-ca-nhan>
5. Thực phẩm chức năng tăng cường sức khỏe giá tốt, giao nhanh, truy cập vào tháng 10 8, 2025, <https://nhathuoclongchau.com.vn/thuc-pham-chuc-nang>
6. Thuốc Panadol xanh 500mg hạ sốt, giảm đau (10 vỉ x 12 viên), truy cập vào tháng 10 8, 2025, <https://nhathuoclongchau.com.vn/thuoc/panadol-giam-dau-ha-sot-2716.html>
7. Tổng hợp thuốc, thuốc biệt dược chính hãng, giá tốt - Long Châu, truy cập vào tháng 10 8, 2025, <https://nhathuoclongchau.com.vn/thuoc>