Lê Văn Bảo Quốc TÊN ĐỀ TÀI : ỨNG DỤNG AI TRONG MUA SẮM THỜI TRANG THÔNG MINH 2025

**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

## NGÀNH: CÔNG NGHỆ THÔNG TIN

## CHUYÊN NGÀNH: AN TOÀN THÔNG TIN

**ĐỀ TÀI:**

# ỨNG DỤNG AI TRONG MUA SẮM THỜI TRANG THÔNG MINH

Người hướng dẫn: **TS. TRỊNH CÔNG DUY**
Sinh viên thực hiện: **LÊ VĂN BẢO QUỐC**
Số thẻ sinh viên: **102210227**
Lớp**:** **21TCLC_DT3**

**Đà Nẵng, 06/2025**

**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

## NGÀNH: CÔNG NGHỆ THÔNG TIN

## CHUYÊN NGÀNH: AN TOÀN THÔNG TIN

**ĐỀ TÀI:**

# ỨNG DỤNG AI TRONG MUA SẮM THỜI TRANG THÔNG MINH

Người hướng dẫn: **TS. TRỊNH CÔNG DUY**
Sinh viên thực hiện: **LÊ VĂN BẢO QUỐC**
Số thẻ sinh viên: **102210227**
Lớp**:** **21TCLC_DT3**

**Đà Nẵng, 06/2025**

# TÓM TẮT

Tên đề tài: Ứng dụng AI trong mua sắm thời trang thông minh.

Sinh viên thực hiện: Lê Văn Bảo Quốc

Số thẻ SV: 102210227 Lớp: 21TCLC_DT3

Người hướng dẫn: TS. Trịnh Công Duy

Thương mại điện tử ngày càng phát triển kéo theo nhu cầu nâng cao trải nghiệm người dùng và cá nhân hóa hành vi mua sắm. Tuy nhiên, việc lựa chọn sản phẩm qua màn hình còn nhiều bất tiện như mất thời gian, thiếu gợi ý phù hợp và hạn chế tương tác. Đề tài "Ứng dụng AI trong mua sắm thời trang thông minh" nhằm xây dựng một nền tảng tích hợp công nghệ trí tuệ nhân tạo để hỗ trợ người dùng hiệu quả hơn trong quá trình ra quyết định.

Hệ thống tập trung vào tính tiện lợi, cá nhân hóa và khả năng mở rộng. Người dùng có thể đăng nhập, tìm kiếm và xem sản phẩm nhanh chóng. Chatbot AI, phát triển dựa trên mô hình ngôn ngữ Gemini, tư vấn sản phẩm thông qua phân tích dữ liệu khách hàng và xử lý ngôn ngữ tự nhiên. Bên cạnh đó, hệ thống tích hợp tìm kiếm thông minh, cho phép mô tả sản phẩm bằng lời nói hoặc văn bản tự nhiên thay vì dùng từ khóa.

Về công nghệ, giao diện được xây dựng bằng Next.js giúp tối ưu tốc độ và trải nghiệm người dùng. Backend sử dụng Python kết hợp Django, cơ sở dữ liệu PostgreSQL đảm bảo hiệu quả lưu trữ và truy vấn. Mô hình Gemini đóng vai trò cốt lõi trong việc xử lý và tương tác ngôn ngữ tự nhiên.

Kết quả mong đợi là một nền tảng thương mại điện tử thông minh, hoạt động ổn định, đáp ứng tốt nhu cầu người dùng và dễ mở rộng. Hệ thống giúp rút ngắn thời gian tìm kiếm, tăng mức độ hài lòng và giảm chi phí vận hành nhờ tự động hóa. Đồ án đồng thời có giá trị thực tiễn và khoa học, mở ra hướng ứng dụng AI trong nhiều lĩnh vực khác như y tế, giáo dục và du lịch.

ĐẠI HỌC ĐÀ NẴNG
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
KHOA CÔNG NGHỆ THÔNG TIN

**CỘNG HÒA XÃ HÔI CHỦ NGHĨA VIỆT NAM**
Độc lập - Tự do - Hạnh phúc

# NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Lê Văn Bảo Quốc Số thẻ sinh viên: 102210227

Lớp: 21TCLC_DT3 Khoa: Công nghệ thông tin Ngành: Công nghệ thông tin

1. *Tên đề tài đồ án:*
   Ứng dụng AI trong mua sắm thời trang thông minh
2. *Đề tài thuộc diện:* ☐ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*
   ……..……..……...
   ...……
   …..……..……..……
4. *Nội dung các phần thuyết minh và tính toán:*
   ***Mở đầu:***
   *Trình bày tổng quan về đề tài "Ứng dụng AI trong mua sắm thời trang thông minh", nêu bật bối cảnh, vấn đề thực tiễn, mục tiêu hướng đến và phương pháp triển khai đồ án. Giới thiệu ngắn gọn bố cục của báo cáo.*
   ***Chương 1: Cơ sở lý thuyết:***
   *Trình bày các lý thuyết trọng tâm như trí tuệ nhân tạo, xử lý ngôn ngữ tự nhiên (NLP), vector hóa văn bản, mô hình Gemini API, cơ sở dữ liệu vector (ChromaDB), cũng như khái niệm RAG (Retrieval-Augmented Generation). Những nền tảng này đóng vai trò làm công nghệ cốt lõi để xây dựng hệ thống chatbot và tìm kiếm thông minh trong đề tài.*
   ***Chương 2: Phân tích và thiết kế hệ thống:***
   *Trình bày chi tiết quá trình phân tích yêu cầu người dùng và nghiệp vụ trong hệ thống thương mại điện tử thông minh. Bao gồm đặc tả chức năng, tác nhân (actor), sơ đồ use-case, sơ đồ hoạt động, thiết kế cơ sở dữ liệu và logic hệ thống AI.*
   *Chương này cũng phân tích hành vi người tiêu dùng, so sánh hệ thống với nền tảng hiện có và trình bày quá trình kiểm thử hệ thống.*
   ***Chương 3: Triển khai và đánh giá kết quả:***
   *Trình bày công nghệ sử dụng (Next.js, Django, Gemini API, Docker, PostgreSQL, ChromaDB), cấu trúc hệ thống và cách tích hợp các mô đun AI. Chương cũng minh họa quá trình triển khai thực tế hệ thống và hiển thị giao diện chức năng, đồng thời đưa ra các đánh giá về hiệu quả hoạt động, phản hồi AI và trải nghiệm người dùng.*

***Kết luận và hướng phát triển:***

*Tổng kết các kết quả đạt được trong đề tài, đánh giá mức độ hoàn thành so với mục tiêu ban đầu. Nêu ra một số hạn chế còn tồn tại và đề xuất các hướng mở rộng như tích hợp voice search, recommendation engine hoặc phân tích cảm xúc để phát triển hệ thống thành nền tảng thương mại điện tử AI hoàn chỉnh hơn.*

5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ ):*

…...……

…...……

…...……

…...……

6. *Họ tên người hướng dẫn: ……...……*
7. *Ngày giao nhiệm vụ đồ án: ……/……/202…..*
8. *Ngày hoàn thành đồ án: ……/……/202…..*

*Đà Nẵng, ngày tháng năm 202*

**Trưởng Bộ môn** ……..  **Người hướng dẫn**

# LỜI NÓI ĐẦU

Để bắt đầu với bài báo cáo, trước tiên em xin phép được gửi những lời cảm ơn chân thành nhất đến với tất cả thầy, cô đã giúp đỡ em trong suốt quá trình thực hiện đồ án tốt nghiệp nói riêng và trong suốt quá trình học tập nói chung.

Em xin chân thành cảm ơn thầy **TS. Trịnh Công Duy**, người đã trực tiếp hướng dẫn, nhận xét, góp ý cho em trong suốt quá trình thực hiện đồ án tốt nghiệp. Em cũng xin gửi lời cảm ơn sâu sắc đến Ban giám hiệu nhà trường, các thầy cô trong khoa Công nghệ thông tin và các cán bộ tại nhà trường đã tạo điều kiện tốt nhất cho em trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp.

Với điều kiện thời gian thực hiện có hạn, kiến thức còn nhiều hạn chế nên đồ án chắc chắn sẽ không tránh khỏi những thiếu sót nhất định. Em rất mong nhận được những lời nhận xét, góp ý của quý thầy cô giáo để em có thể hoàn thiện và phát triển đề tài đồ án trong tương lai.

Một lần nữa em xin gửi lời cảm ơn chân thành đến tất cả mọi người đã giúp đỡ em hoàn thiện đồ án này!

Sinh viên thực hiện

# CAM ĐOAN

Tôi xin cam đoan:

1. Nội dung trong đồ án này là do tôi thực hiện dưới sự hướng dẫn trực tiếp của TS. Trịnh Công Duy.

2. Các tham khảo dùng trong đồ án đều được trích dẫn rõ ràng tên tác giả, tên công trình, thời gian, địa điểm công bố.

3. Nếu có những sao chép không hợp lệ, vi phạm, tôi xin chịu hoàn toàn trách nhiệm.

Sinh viên thực hiện

# MỤC LỤC

# DANH SÁCH CÁC BẢNG

# DANH SÁCH CÁC HÌNH

# DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

CHỮ VIẾT TẮT:

| STT | Viết tắt | Diễn giải |
|---|---|---|
| 1 | API | Application Programming Interface |
| 2 | RESTful API | Representational State Transfer API |
| 3 | SSR | Relational Database Management System |
| 4 | SSG | Decision Support System |
| 5 | AI | Artificial Intelligence |
| 6 | URL | Uniform Resource Locator |
| 7 | NLP | Natural Language Processing |
| 8 | CSR | Client-Side Rendering |
| 9 | SEO | Search Engine Optimization |
| 10 | RAG | Retrieval-Augmented Generation |

# MỞ ĐẦU

## 1. Tổng quan về đề tài

Trong bối cảnh thương mại điện tử phát triển mạnh mẽ, người tiêu dùng ngày càng kỳ vọng vào những trải nghiệm mua sắm tiện lợi, nhanh chóng và mang tính cá nhân hóa cao. Tuy nhiên, việc lựa chọn sản phẩm trực tuyến vẫn gặp nhiều trở ngại do thiếu tư vấn phù hợp và tốn thời gian tìm kiếm.

Trí tuệ nhân tạo (AI) đang mở ra những hướng đi mới trong việc cải thiện trải nghiệm người dùng trên các nền tảng mua sắm. Bằng cách tích hợp các mô hình xử lý ngôn ngữ tự nhiên và tìm kiếm thông minh, AI có thể giúp người tiêu dùng tìm được sản phẩm phù hợp hơn chỉ thông qua mô tả đơn giản bằng lời nói hoặc văn bản.

Xuất phát từ thực tiễn đó, đề tài "Ứng dụng AI trong mua sắm thời trang thông minh" được lựa chọn với mục tiêu xây dựng một hệ thống thương mại điện tử tích hợp các công nghệ AI hiện đại nhằm hỗ trợ người dùng trong quá trình tìm kiếm và lựa chọn sản phẩm một cách hiệu quả và tự nhiên hơn.

## 2. Mục đích

Mục tiêu chính của đề tài là xây dựng một nền tảng mua sắm trực tuyến thông minh, trong đó AI đóng vai trò trung tâm trong việc:

- Tư vấn sản phẩm dựa trên dữ liệu khách hàng và truy vấn ngôn ngữ tự nhiên.
- Hỗ trợ tìm kiếm sản phẩm thông minh, cho phép người dùng mô tả bằng lời để nhận được gợi ý phù hợp.
- Tăng cường khả năng tương tác giữa hệ thống và người dùng bằng chatbot AI.
- Cung cấp một hệ thống có tính mở rộng, thân thiện với người dùng và dễ bảo trì.

## 3. Phương pháp nghiên cứu

Đồ án được triển khai theo phương pháp tiếp cận hệ thống kết hợp với nghiên cứu ứng dụng công nghệ AI hiện đại. Các bước nghiên cứu bao gồm:

- Phân tích hành vi người tiêu dùng trực tuyến để xác định nhu cầu cốt lõi trong việc tìm kiếm và lựa chọn sản phẩm.
- Ứng dụng xử lý ngôn ngữ tự nhiên (NLP) thông qua Gemini API nhằm xây dựng chatbot và tìm kiếm thông minh.

- Thiết kế hệ thống module hóa, trong đó các thành phần chính như giao diện, API và AI được tách biệt rõ ràng, giúp dễ quản lý và mở rộng.
- Sử dụng PostgreSQL làm hệ quản trị cơ sở dữ liệu nhằm tối ưu hiệu năng truy vấn và khả năng lưu trữ dữ liệu phức tạp.
- Kiểm thử thực nghiệm để đánh giá độ chính xác của AI trong tư vấn và tìm kiếm, cũng như hiệu suất vận hành của toàn hệ thống.

**4. Lợi ích và ứng dụng thực tiễn**

Hệ thống được xây dựng mang lại nhiều lợi ích thiết thực trong môi trường thương mại điện tử hiện nay:

- Với người tiêu dùng: hỗ trợ tìm kiếm sản phẩm nhanh chóng, chính xác và thuận tiện hơn, ngay cả khi không nhớ tên sản phẩm.
- Với doanh nghiệp: giảm tải cho bộ phận tư vấn, nâng cao khả năng tiếp cận khách hàng và tối ưu hóa quy trình bán hàng nhờ tự động hóa.
- Về mặt công nghệ: là minh chứng cho việc ứng dụng thành công AI (NLP + tìm kiếm thông minh) trong thực tiễn, có thể mở rộng sang các lĩnh vực khác như y tế, giáo dục, bất động sản.
- Về lâu dài: hệ thống có thể tích hợp thêm các mô-đun AI nâng cao như phân tích hành vi người dùng hoặc gợi ý theo thời gian thực.

**5. Bố cục của đồ án**

Bản báo cáo gồm: Phần mở đầu, 3 chương và phần kết luận, cụ thể như sau:

- *Mở đầu:*

*Trình bày tổng quan về đề tài "Ứng dụng AI trong mua sắm thời trang thông minh", nêu bật bối cảnh, vấn đề thực tiễn, mục tiêu hướng đến và phương pháp triển khai đồ án. Giới thiệu ngắn gọn bố cục của báo cáo.*

- *Chương 1: Cơ sở lý thuyết:*

*Trình bày các lý thuyết trọng tâm như trí tuệ nhân tạo, xử lý ngôn ngữ tự nhiên (NLP), vector hóa văn bản, mô hình Gemini API, cơ sở dữ liệu vector (ChromaDB), cũng như khái niệm RAG (Retrieval-Augmented Generation). Những nền tảng này đóng vai trò làm công nghệ cốt lõi để xây dựng hệ thống chatbot và tìm kiếm thông minh trong đề tài.*

- *Chương 2: Phân tích và thiết kế hệ thống:*

*Trình bày chi tiết quá trình phân tích yêu cầu người dùng và nghiệp vụ trong hệ thống thương mại điện tử thông minh. Bao gồm đặc tả chức năng, tác nhân (actor), sơ đồ use-case, sơ đồ hoạt động, thiết kế cơ sở dữ liệu và logic hệ thống AI.*

*Chương này cũng phân tích hành vi người tiêu dùng, so sánh hệ thống với nền tảng hiện có và trình bày quá trình kiểm thử hệ thống.*

- *Chương 3: Triển khai và đánh giá kết quả:*

*Trình bày công nghệ sử dụng (Next.js, Django, Gemini API, Docker, PostgreSQL, ChromaDB), cấu trúc hệ thống và cách tích hợp các mô đun AI. Chương cũng minh họa quá trình triển khai thực tế hệ thống và hiển thị giao diện chức năng, đồng thời đưa ra các đánh giá về hiệu quả hoạt động, phản hồi AI và trải nghiệm người dùng.*

- *Kết luận và hướng phát triển:*

*Tổng kết các kết quả đạt được trong đề tài, đánh giá mức độ hoàn thành so với mục tiêu ban đầu. Nêu ra một số hạn chế còn tồn tại và đề xuất các hướng mở rộng như tích hợp voice search, recommendation engine hoặc phân tích cảm xúc để phát triển hệ thống thành nền tảng thương mại điện tử AI hoàn chỉnh hơn.*

# Chương 1: CƠ SỞ LÝ THUYẾT

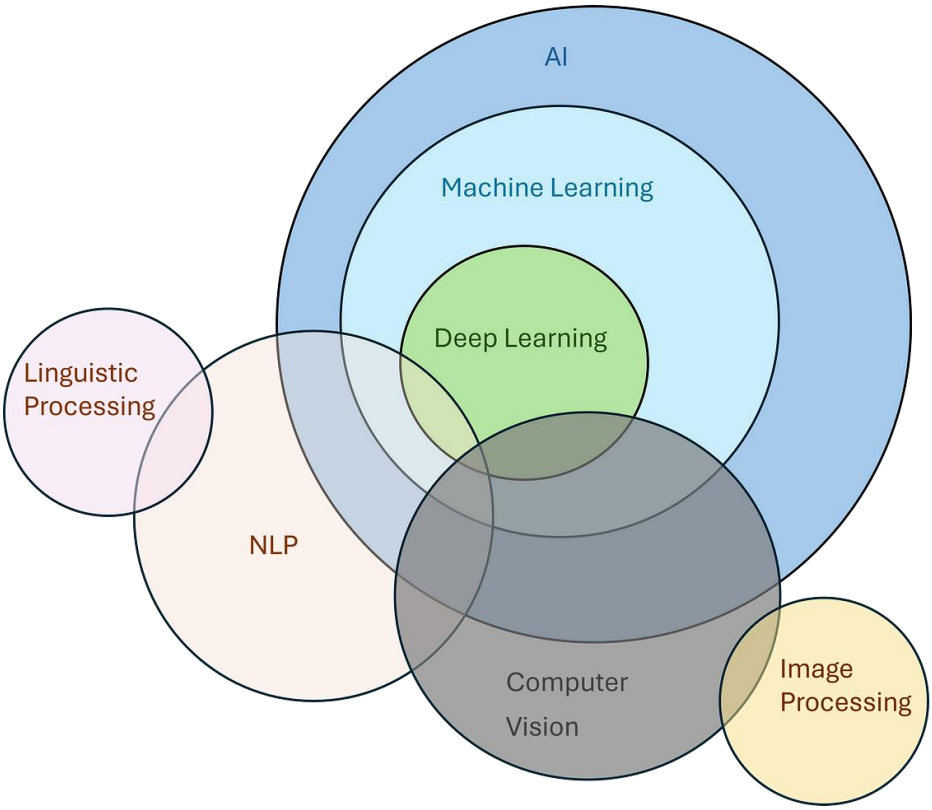
## 1.1. Tổng quan về trí tuệ nhân tạo (AI)

### *1.1.1. Khái niệm về trí tuệ nhân tạo (AI)*

Trí tuệ nhân tạo (Artificial Intelligence - AI) là lĩnh vực nghiên cứu trong khoa học máy tính nhằm tạo ra các hệ thống có khả năng mô phỏng tư duy, học tập và hành động của con người. AI cho phép máy móc thực hiện các nhiệm vụ như phân tích dữ liệu, ra quyết định, hiểu ngôn ngữ và nhận diện hình ảnh.

Các lĩnh vực chính của AI bao gồm:

- **Machine Learning (ML):** Máy học – cho phép hệ thống tự học từ dữ liệu và cải thiện hiệu suất mà không cần lập trình rõ ràng.
- **Deep Learning (DL):** Học sâu – nhánh con của ML, sử dụng mạng nơ-ron nhiều lớp để xử lý dữ liệu phức tạp như hình ảnh và âm thanh.
- **Natural Language Processing (NLP):** Xử lý ngôn ngữ tự nhiên – giúp máy hiểu, phân tích và phản hồi ngôn ngữ con người.
- **Computer Vision:** Thị giác máy tính – cho phép hệ thống "nhìn thấy" và hiểu hình ảnh hoặc video.




*Hình 1.1: Mối quan hệ giữa các lĩnh vực trong trí tuệ nhân tạo*

### *1.1.2. Lịch sử phát triển và ứng dụng của AI*

AI bắt đầu từ những năm 1950 với mục tiêu ban đầu là mô phỏng tư duy con người. Các cột mốc quan trọng:

- 1956: Thuật ngữ "Artificial Intelligence" được giới thiệu tại hội nghị Dartmouth.
- 1980s: Hệ chuyên gia được phát triển, giúp ra quyết định trong các lĩnh vực kỹ thuật và y tế.
- 2010s - nay: Sự phát triển mạnh mẽ của ML và DL, cùng với sự xuất hiện của các nền tảng AI như Siri, Google Assistant, ChatGPT, Gemini...

AI ngày nay đã được ứng dụng rộng rãi trong:

- Thương mại điện tử: gợi ý sản phẩm, phân tích hành vi người dùng.
- Chăm sóc khách hàng: chatbot hỗ trợ 24/7, tổng đài ảo.
- Tư vấn sản phẩm: hệ thống hiểu yêu cầu bằng ngôn ngữ tự nhiên, cá nhân hóa đề xuất sản phẩm.

### *1.1.3. Vai trò của AI trong hệ thống*

Trong dự án này, AI đóng vai trò trung tâm với các chức năng chính:

- **Tự động hóa tư vấn sản phẩm:** sử dụng chatbot AI (Gemini API) để tương tác, hiểu yêu cầu và gợi ý sản phẩm phù hợp.
- **Tìm kiếm thông minh:** giúp người dùng mô tả sản phẩm bằng ngôn ngữ tự nhiên thay vì nhập từ khóa đơn lẻ.
- **Tối ưu hóa quản lý dữ liệu sản phẩm:** hỗ trợ phân loại, gợi ý và tổ chức dữ liệu sản phẩm dựa trên hành vi người dùng.
- **Nâng cao trải nghiệm người dùng:** giúp quá trình mua sắm trở nên nhanh chóng, chính xác và cá nhân hóa hơn.

### *1.1.4. Các lợi ích khi tích hợp AI vào hệ thống*

Việc ứng dụng AI mang lại nhiều lợi ích vượt trội:

- **Nâng cao hiệu quả vận hành:** giảm thời gian tìm kiếm, tăng khả năng tự phục vụ.
- **Giảm chi phí nhân sự:** thay thế một phần công việc của tư vấn viên bằng hệ thống tự động.
- **Cá nhân hóa dịch vụ:** đề xuất sản phẩm dựa trên nhu cầu và lịch sử tương tác của từng người dùng.
- **Hỗ trợ khách hàng 24/7:** không bị giới hạn thời gian như con người.
- **Khả năng mở rộng cao:** dễ dàng tích hợp thêm AI Agent, phân tích dữ liệu nâng cao, và mở rộng mô hình kinh doanh trong tương lai.

## 1.2. Xử lý ngôn ngữ tự nhiên (NLP)

### *1.2.1. Khái niệm về NLP*

Xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP) là một lĩnh vực của trí tuệ nhân tạo (AI), tập trung vào việc cho phép máy tính hiểu, diễn giải, tạo và phản hồi ngôn ngữ của con người một cách tự nhiên và có ý nghĩa. NLP kết nối giữa

ngôn ngữ tự nhiên và ngôn ngữ máy bằng các kỹ thuật học máy, thống kê và ngôn ngữ học tính toán.

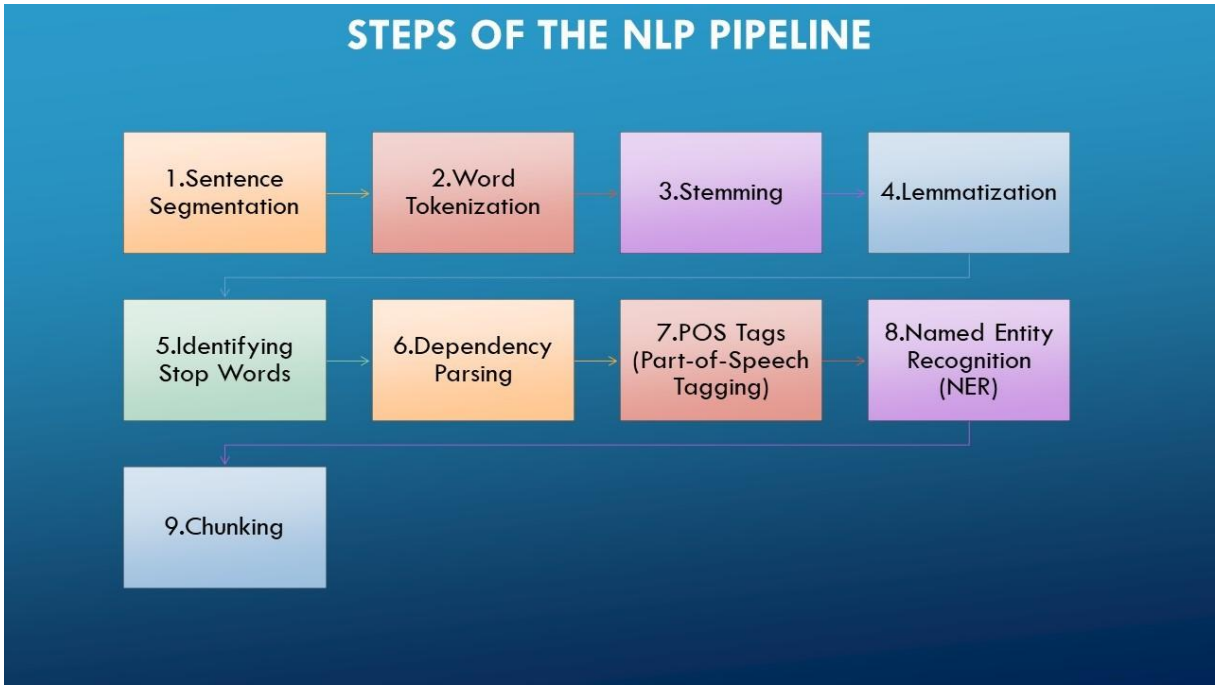
NLP đóng vai trò quan trọng trong AI vì nó là nền tảng cho các hệ thống tương tác bằng văn bản hoặc giọng nói như chatbot, trợ lý ảo, phân tích văn bản, và tìm kiếm thông minh.

### *1.2.2. Vai trò của NLP trong hệ thống*

Trong hệ thống mua sắm thông minh, NLP là công nghệ cốt lõi để:

- Hiểu ngôn ngữ người dùng: hệ thống có thể tiếp nhận đầu vào là các câu hỏi, yêu cầu, mô tả sản phẩm từ người dùng.
- Phân tích và trích xuất thông tin cần thiết: như loại sản phẩm, màu sắc, kiểu dáng, sở thích,...
- Tạo phản hồi ngôn ngữ tự nhiên: giúp chatbot có thể giao tiếp mạch lạc, thân thiện như con người.
- NLP giúp chatbot AI không chỉ hiểu từ khóa đơn lẻ, mà còn nắm bắt được ý định và ngữ cảnh trong câu nói của người dùng.

### *1.2.3. Các kỹ thuật NLP sử dụng*




*Hình 1.2: Quy trình xử lý ngôn ngữ tự nhiên*

Hình minh họa quy trình NLP thể hiện rõ 9 bước xử lý văn bản, từ phân tách cấu trúc câu đến chuẩn hóa, gán nhãn và tổ chức ngữ pháp. Đây là nền tảng giúp chatbot AI hiểu ngôn ngữ người dùng, trích xuất thông tin và sinh phản hồi tự nhiên

### *1. Sentence Segmentation – Tách câu*

Đây là bước đầu tiên nhằm chia văn bản dài thành các câu đơn lẻ. Mỗi câu được xem như một đơn vị phân tích độc lập, giúp tăng độ chính xác cho các bước xử lý sau.

*Ví dụ:*

*"Tôi cần mua áo sơ mi. Bạn có thể giúp tôi không?" → 2 câu tách biệt.*

### *2. Word Tokenization – Tách từ*

Mỗi câu sau khi được phân đoạn sẽ tiếp tục được chia nhỏ thành các token – thường là từ hoặc cụm từ. Việc tách từ cho phép hệ thống thao tác trên từng đơn vị ngôn ngữ cụ thể.

*Ví dụ:*

*"Chiếc váy xanh dương đẹp" → ["Chiếc", "váy", "xanh", "dương", "đẹp"]*

### *3. Stemming – Rút gọn từ*

Stemming là kỹ thuật loại bỏ các phần mở rộng (đuôi từ) để đưa từ về gốc (stem). Đây là phương pháp đơn giản và nhanh nhưng đôi khi làm mất đi ngữ nghĩa.

*Ví dụ:*

*"running", "runner", "runs" → "run"*

### *4. Lemmatization – Chuẩn hóa từ*

Lemmatization đưa từ về gốc từ vựng dựa trên ngữ pháp và ngữ cảnh. Không giống stemming, lemmatization duy trì được ngữ nghĩa chính xác.

*Ví dụ:*

*"was" → "be", "mice" → "mouse", "better" → "good"*

### *5. Stop Word Removal – Loại bỏ từ dừng*

Các từ dừng (stop words) là những từ thường xuất hiện nhưng không đóng góp nhiều về mặt ngữ nghĩa, ví dụ như: "và", "là", "của", "một", "các"... Việc loại bỏ chúng giúp tăng hiệu suất mô hình và tập trung vào thông tin chính yếu.

### *6. Dependency Parsing – Phân tích quan hệ cú pháp*

Bước này xác định quan hệ ngữ pháp giữa các từ trong câu. Hệ thống sẽ tạo ra một "cây cú pháp" thể hiện quan hệ phụ thuộc giữa chủ ngữ, động từ, tân ngữ, bổ ngữ...

*Ví dụ:*

*Trong câu "Tôi cần váy màu xanh", từ "Tôi" là chủ ngữ của "cần", còn "váy" là tân ngữ của "cần".*

### *7. Part-of-Speech (POS) Tagging – Gán nhãn từ loại*

POS Tagging gán cho mỗi token một nhãn từ loại như danh từ (noun), động từ (verb), tính từ (adjective), trạng từ (adverb)... Điều này giúp hệ thống hiểu rõ vai trò của từng từ trong ngữ cảnh cụ thể.

*Ví dụ:*

*"váy" → noun, "xanh" → adjective, "cần" → verb*

***8. Named Entity Recognition (NER) – Nhận diện thực thể***

NER là bước xác định và phân loại các thực thể có tên (named entities) trong câu như tên người, tổ chức, địa điểm, sản phẩm, màu sắc... Việc nhận dạng thực thể hỗ trợ chatbot hiểu sâu hơn về truy vấn người dùng.

*Ví dụ:*

*"áo sơ mi trắng của Adidas" → [sản phẩm: "áo sơ mi", màu: "trắng", thương hiệu: "Adidas"]*

***9. Chunking – Gom nhóm từ (Shallow Parsing)***

Chunking là quá trình kết hợp các token đã được gán nhãn từ loại thành các cụm từ ngữ pháp (phrase):

- NP (Noun Phrase): cụm danh từ
- VP (Verb Phrase): cụm động từ
- PP (Prepositional Phrase): cụm giới từ

*Ví dụ:*

*"một chiếc áo sơ mi trắng" → [NP: "một chiếc áo sơ mi trắng"]*

Chunking giúp mô hình hiểu mối quan hệ giữa các thành phần trong câu một cách rõ ràng và tổ chức hơn, đặc biệt quan trọng trong việc trích xuất thông tin và sinh phản hồi chính xác.

Tóm lại, hệ thống sử dụng chuỗi các kỹ thuật NLP từ tiền xử lý, chuẩn hóa, gán nhãn, trích xuất thực thể đến phân tích cú pháp và gom nhóm từ. Đây là bước tiền đề quan trọng cho quá trình vector hóa văn bản trong hệ thống.

### *1.2.4. Ứng dụng NLP trong project*

Trong dự án, NLP được sử dụng để:

- **Phân tích câu hỏi người dùng gửi vào chatbot:** hiểu yêu cầu mua sản phẩm, hỏi thông tin, hay cần hỗ trợ.
- **Nhận diện ý định (intent detection):** ví dụ "Tôi cần áo sơ mi màu trắng" → intent: tìm sản phẩm, danh mục: áo sơ mi, thuộc tính: màu trắng.
- **Trích xuất thực thể (entity extraction):** nhận dạng kiểu sản phẩm, thương hiệu, mức giá...
- **Tìm kiếm sản phẩm thông minh:** từ ngôn ngữ mô tả tự nhiên, hệ thống đề xuất sản phẩm phù hợp.

### *1.2.5. Lợi ích của NLP đối với hệ thống*

Việc tích hợp NLP mang lại nhiều giá trị thiết thực:

- **Giao tiếp tự nhiên hơn:** người dùng không cần nhớ tên sản phẩm hay từ khóa cụ thể.

- **Tăng tính tương tác:** người dùng cảm thấy thoải mái khi trò chuyện với chatbot như với nhân viên tư vấn thật.
- **Tư vấn thông minh:** hiểu đúng nhu cầu, đề xuất chính xác hơn.
- **Hỗ trợ đa ngôn ngữ (nếu mở rộng):** NLP cho phép xử lý cả tiếng Việt và các ngôn ngữ khác.
- **Tối ưu trải nghiệm người dùng:** giảm thời gian tìm kiếm, tăng độ hài lòng và khả năng mua hàng.

## 1.3. Mô hình vector hóa tài liệu

### *1.3.1. Khái niệm*

Vector hóa tài liệu là quá trình chuyển đổi văn bản từ dạng ngôn ngữ tự nhiên thành các biểu diễn dạng số (vector) để máy tính có thể hiểu và xử lý được. Mỗi văn bản sau khi vector hóa sẽ được biểu diễn dưới dạng một chuỗi số có ý nghĩa toán học, phản ánh nội dung và đặc điểm của văn bản đó.

Mục tiêu của vector hóa là tạo ra biểu diễn số hóa mà vẫn giữ được ngữ nghĩa để phục vụ cho các bài toán như tìm kiếm, phân loại, và hiểu ngôn ngữ tự nhiên.

### *1.3.2. Vai trò của vector hóa tài liệu trong hệ thống*

Vector hóa đóng vai trò nền tảng trong hệ thống vì:

- Cho phép hệ thống hiểu và so sánh văn bản bằng cách tính toán khoảng cách giữa các vector.
- Hỗ trợ tìm kiếm thông minh: tìm tài liệu hoặc sản phẩm dựa trên nội dung văn bản thay vì chỉ dựa vào từ khóa.
- Phân loại và nhóm các câu truy vấn có ý nghĩa tương đồng.
- Là đầu vào quan trọng cho các mô hình học máy và AI trong xử lý ngôn ngữ tự nhiên.

### *1.3.3. Các phương pháp vector hóa phổ biến*

Một số kỹ thuật vector hóa phổ biến gồm:

- **Bag of Words (BoW):** biểu diễn văn bản bằng tần suất xuất hiện của các từ, không quan tâm đến ngữ cảnh.
- **TF-IDF (Term Frequency - Inverse Document Frequency):** điều chỉnh tần suất từ bằng mức độ đặc trưng của nó trong toàn bộ tập tài liệu.
- **Word2Vec:** mô hình học sâu biểu diễn từng từ dưới dạng vector ngữ nghĩa, phản ánh mối quan hệ giữa các từ.
- **GloVe:** giống Word2Vec nhưng học trên toàn bộ ma trận đồng xuất hiện từ.
- **Sentence Embedding:** biểu diễn toàn bộ câu hoặc đoạn văn thành một vector duy nhất, ví dụ như sử dụng mô hình BERT hoặc Sentence Transformers.
- **Embedding từ mô hình AI:** sử dụng các mô hình lớn như Gemini, OpenAI, Cohere để tạo ra vector ngữ nghĩa chất lượng cao cho cả câu hỏi và văn bản.

### *1.3.4. Ứng dụng vector hóa trong project*

Trong dự án, vector hóa được sử dụng để:

- **Tạo cơ sở dữ liệu vector:** lưu trữ các biểu diễn số của tài liệu, mô tả sản phẩm hoặc câu hỏi người dùng.
- **Hỗ trợ truy vấn ngữ nghĩa:** so sánh vector của câu hỏi với vector của nội dung để tìm thông tin liên quan.
- **Tăng khả năng hiểu của chatbot AI:** chatbot truy vấn cơ sở dữ liệu vector để tìm câu trả lời phù hợp.
- **Tìm kiếm tài liệu gần nghĩa:** người dùng có thể hỏi bằng nhiều cách khác nhau nhưng vẫn nhận được kết quả chính xác.

### *1.3.5. Lợi ích của vector hóa tài liệu*

Tích hợp vector hóa mang lại nhiều lợi ích:

- **Tăng tốc độ truy vấn:** truy vấn được tối ưu thông qua các chỉ số khoảng cách vector.
- **Nâng cao độ chính xác:** so với tìm kiếm theo từ khóa, vector hóa giúp hiểu ngữ nghĩa sâu hơn.
- **Hỗ trợ NLP hiệu quả hơn:** là bước đệm cần thiết trong việc xử lý, hiểu và phản hồi văn bản tự nhiên.
- **Mở rộng dễ dàng:** vector hóa cho phép hệ thống dễ dàng thêm mới tài liệu và mở rộng phạm vi ứng dụng AI.

## 1.4. Genimi API

### *1.4.1. Genimi API là gì*

Gemini API là giao diện lập trình ứng dụng của dòng mô hình ngôn ngữ lớn (Large Language Model - LLM) được phát triển bởi Google DeepMind. Đây là một trong những mô hình AI tiên tiến nhất hiện nay, thuộc thế hệ multimodal AI – có khả năng xử lý đồng thời văn bản, hình ảnh, âm thanh và mã nguồn.

Gemini nổi bật với:

- Khả năng hiểu ngữ cảnh mạnh mẽ, đặc biệt trong hội thoại tự nhiên.
- Tốc độ phản hồi nhanh và khả năng xử lý truy vấn phức tạp.
- Hỗ trợ tốt đa ngôn ngữ, bao gồm cả tiếng Việt.
- Được tối ưu cho tích hợp vào các ứng dụng thực tế như tìm kiếm thông minh, chatbot, phân tích văn bản.

### *1.4.2. Vai trò của Genimi API trong hệ thống*

Trong dự án, Gemini API giữ vai trò trung tâm trong việc:

- **Xử lý ngôn ngữ tự nhiên (NLP):** hiểu và phân tích nội dung câu hỏi của người dùng.
- **Sinh phản hồi thông minh:** tạo câu trả lời tự nhiên, phù hợp ngữ cảnh và thân thiện với người dùng.
- **Tư vấn sản phẩm:** dựa trên mô tả hoặc nhu cầu do người dùng nhập vào.
- **Tự động hóa giao tiếp:** giúp hệ thống phản hồi mà không cần sự can thiệp của con người.

### *1.4.3. Cách tích hợp Genimi API vào project*

Trong hệ thống, Gemini API được tích hợp vào backend thông qua mô-đun ai_powered. Quy trình hoạt động gồm:

- Người dùng gửi câu hỏi từ frontend (React).
- Backend (Django) tiếp nhận và xử lý yêu cầu trong mô-đun ai_powered.
- Tạo một HTTP request chứa nội dung truy vấn và gửi tới Gemini API (qua REST API hoặc SDK của Google AI).
- Nhận phản hồi từ Gemini API dưới dạng văn bản trả lời.
- Trả kết quả về frontend để hiển thị cho người dùng.
- Toàn bộ quá trình này đảm bảo thời gian phản hồi nhanh, dữ liệu được xử lý an toàn và logic AI được tách biệt rõ ràng khỏi phần còn lại của hệ thống.

### *1.4.4. Lợi ích khi sử dụng Genimi API*

Việc sử dụng Gemini API mang lại nhiều lợi ích nổi bật:

- Nâng cao chất lượng tư vấn: phản hồi mạch lạc, ngắn gọn và đúng trọng tâm.
- Phản hồi tự nhiên, sát ngữ cảnh: tạo trải nghiệm giống như giao tiếp với con người.
- Giảm chi phí và thời gian phát triển AI: không cần huấn luyện mô hình riêng.
- Dễ dàng mở rộng: có thể tích hợp thêm chức năng như tóm tắt, phân tích cảm xúc, nhận diện câu hỏi nâng cao.

### *1.4.5. Ứng dụng thực tế trong project*

Trong dự án, Gemini API được dùng để:

- Tư vấn sản phẩm thông qua chatbot: hiểu yêu cầu người dùng và gợi ý sản phẩm phù hợp.
- Trả lời các câu hỏi về đơn hàng, chính sách, hướng dẫn sử dụng.
- Tăng khả năng tự phục vụ: giúp khách hàng nhận hỗ trợ ngay cả ngoài giờ làm việc.
- Góp phần tạo nên trải nghiệm người dùng chuyên nghiệp, hiện đại và thân thiện.

## 1.5. Retrieval-Augmented Generation (RAG)

### *1.5.1. Khái niệm về RAG*

Retrieval-Augmented Generation (RAG) là một kiến trúc kết hợp giữa mô hình tìm kiếm thông tin (retriever) và mô hình sinh ngôn ngữ (generator). RAG giúp cải thiện khả năng trả lời câu hỏi của hệ thống AI bằng cách kết hợp dữ liệu bên ngoài với khả năng sinh ngôn ngữ tự nhiên.

Thay vì chỉ dựa vào dữ liệu đã được học trong mô hình ngôn ngữ lớn (LLM), RAG cho phép mô hình truy xuất tài liệu liên quan từ kho tri thức trước khi sinh câu trả lời dựa trên ngữ cảnh đó.

### 1.5.2. Thành phần của hệ thống RAG

RAG bao gồm 2 thành phần chính:

a. Retriever – Bộ tìm kiếm văn bản

Có nhiệm vụ tìm các đoạn văn bản hoặc tài liệu có liên quan đến truy vấn từ người dùng.

Thường sử dụng vector search trên cơ sở dữ liệu embedding (như FAISS, ChromaDB).

Một số kỹ thuật phổ biến:

- BM25 (truy vấn keyword truyền thống)
- Dense Retrieval (dựa trên embedding vector hóa)
- Sentence Transformers, DPR (Dense Passage Retrieval)

b. Generator – Mô hình sinh ngôn ngữ

Sau khi retriever trả về các đoạn văn liên quan, generator (ví dụ: GPT, Gemini, BERT2BERT...) dùng chúng như bối cảnh để sinh câu trả lời phù hợp.

Câu trả lời được sinh ra tự nhiên, đúng ngữ cảnh và chứa thông tin thực tế từ tài liệu truy xuất.

### 1.5.3. Luồng hoạt động của RAG

1. Người dùng nhập câu hỏi: "Áo sơ mi cotton có phù hợp mùa hè không?"
2. Hệ thống vector hóa truy vấn và tìm các tài liệu liên quan từ ChromaDB (ví dụ: mô tả sản phẩm, bài viết hướng dẫn).
3. Các đoạn văn liên quan được ghép vào prompt như bối cảnh.
4. Mô hình ngôn ngữ lớn (LLM) sinh ra câu trả lời:

"Áo sơ mi cotton rất phù hợp với mùa hè nhờ khả năng thấm hút mồ hôi và thông thoáng."

### 1.5.4. Lợi ích của mô hình RAG

*Bảng 1.1: Lợi ích của mô hình RAG*

| Lợi ích | Mô tả |
|---|---|
| Tăng độ chính xác | Không phụ thuộc hoàn toàn vào kiến thức huấn luyện nội bộ của LLM, RAG cập nhật được tri thức ngoài. |
| Giảm "ảo tưởng" của AI (hallucination) | Nhờ có nguồn thông tin thực, câu trả lời ít bị bịa đặt hoặc sai lệch. |
| Cập nhật dễ dàng | Có thể cập nhật tri thức mới đơn giản bằng cách bổ sung vào cơ sở dữ liệu mà không cần huấn luyện lại mô hình. |

| | |
|---|---|
| Khả năng truy xuất minh bạch | Có thể hiển thị luôn nguồn gốc câu trả lời (ví dụ: "theo tài liệu X"). |

### *1.5.5. Ứng dụng của RAG trong dự án*

Trong đồ án "Ứng dụng AI trong mua sắm thời trang thông minh", kiến trúc hệ thống tích hợp ý tưởng của RAG theo cách sau:

- Retriever: Sử dụng PostgreSQL, ChromaDB để lưu trữ các tài liệu sản phẩm, chính sách, câu hỏi thường gặp dưới dạng vector embedding (sinh ra từ Sentence Transformers).
- Generator: Dùng Gemini API để tạo phản hồi ngôn ngữ tự nhiên, có bối cảnh chính xác từ tài liệu truy xuất.
- Workflow:
  - Người dùng hỏi chatbot.
  - Câu hỏi được vector hóa và tìm top 3–5 đoạn liên quan trong PostgreSQL, ChromaDB.
  - Các đoạn đó được "inject" vào prompt của Gemini API để sinh câu trả lời chính xác và sát thực tế hơn.

Như vậy, hệ thống chatbot của đồ án không chỉ thông minh nhờ LLM mà còn thực tế và đáng tin cậy nhờ có khả năng truy xuất ngữ nghĩa, đúng với mô hình RAG.

### *1.5.6. So sánh RAG và mô hình sinh thuần túy*

*Bảng 1.2: So sánh RAG và mô hình sinh thuần túy*

| Tiêu chí | Mô hình sinh truyền thống (LLM only) | RAG |
|---|---|---|
| Dữ liệu | Dựa vào dữ liệu đã học từ trước | Kết hợp dữ liệu bên ngoài mới |
| Tính chính xác | Dễ sai khi kiến thức lỗi thời | Cập nhật tri thức liên tục |
| Phản hồi | Có thể mượt nhưng sai lệch | Đúng thực tế, có thể trích dẫn |
| Ứng dụng | Chatbot đơn giản, viết sáng tạo | QA hệ thống, chatbot doanh nghiệp, tư vấn thông minh |

## 1.6. Cơ sở dữ liệu vector (ChromaDB)

### *1.6.1. Khái niệm*

Cơ sở dữ liệu vector (Vector Database) là hệ thống lưu trữ và truy vấn các vector – biểu diễn số học của dữ liệu như văn bản, hình ảnh, âm thanh sau khi đã được xử lý bởi các mô hình học máy. Mỗi vector thường là một mảng số nhiều chiều (vector n chiều) chứa đựng thông tin ngữ nghĩa của dữ liệu gốc.

Khác với cơ sở dữ liệu truyền thống (relational database) – vốn lưu trữ dữ liệu dưới dạng bảng, cột và khóa, cơ sở dữ liệu vector cho phép:

- Truy vấn dữ liệu theo ngữ nghĩa, không chỉ theo từ khóa.
- Tìm kiếm theo độ tương đồng (similarity search) dựa trên khoảng cách giữa các vector.
- Ứng dụng tốt trong các hệ thống AI, đặc biệt là NLP và thị giác máy tính.

### *1.6.2. Vai trò*

ChromaDB là một trong những vector database mã nguồn mở phổ biến, được tối ưu cho việc lưu trữ và truy vấn dữ liệu ngữ nghĩa hiệu quả. Trong hệ thống của dự án, ChromaDB giữ vai trò:

- **Lưu trữ vector của tài liệu và truy vấn:** giúp hệ thống hiểu ngữ cảnh và nội dung sâu hơn thay vì chỉ khớp từ khóa.
- **Hỗ trợ tìm kiếm thông minh:** cho phép tìm nội dung gần nghĩa thay vì trùng khớp tuyệt đối.
- Làm nền cho các mô-đun AI như chatbot, tìm kiếm sản phẩm.

### *1.6.3. Cách hoạt động của ChromaDB*

Quy trình hoạt động cơ bản của ChromaDB bao gồm:

**Lưu trữ (Ingestion):**

- Dữ liệu văn bản (ví dụ: mô tả sản phẩm, tài liệu tư vấn) được xử lý bằng mô hình NLP để tạo vector embedding.
- Các vector này được lưu vào ChromaDB cùng với metadata (ID, nhãn, mô tả…).

**Truy vấn (Querying):**

- Khi người dùng nhập một truy vấn (text), hệ thống chuyển nó thành vector.
- ChromaDB tính toán khoảng cách vector (cosine, L2, etc.) để tìm các mục gần nhất về mặt ngữ nghĩa.

**Tích hợp AI/NLP:**

- Dễ dàng kết nối với các mô hình embedding như Sentence Transformers, Gemini embedding, OpenAI, v.v.
- Hỗ trợ cả truy vấn văn bản lẫn vector hóa tự động nhờ API đơn giản.

### *1.6.4. Ứng dụng ChromaDB trong project*

Trong dự án, ChromaDB được sử dụng để:

- Tạo cơ sở dữ liệu vector từ tài liệu nội dung hệ thống, ví dụ: chính sách, thông tin sản phẩm, hướng dẫn sử dụng.
- Hỗ trợ chatbot: khi người dùng hỏi, hệ thống vector hóa câu hỏi và tìm câu trả lời phù hợp trong ChromaDB.
- Tìm kiếm sản phẩm gần nghĩa: người dùng có thể mô tả sản phẩm bằng từ ngữ tự nhiên, hệ thống vẫn tìm đúng sản phẩm liên quan.

- Kết hợp cùng Gemini API để nâng cao độ chính xác và tốc độ phản hồi.

### *1.6.5. Lợi ích khi sử dụng cơ sở dữ liệu vector*

Việc sử dụng ChromaDB và vector database nói chung mang lại nhiều lợi ích:

- Tăng tốc độ truy vấn ngữ nghĩa nhờ tối ưu chỉ mục theo vector.
- Nâng cao độ chính xác của kết quả tìm kiếm, kể cả khi truy vấn không khớp từ khóa.
- Hỗ trợ các chức năng AI thông minh như tư vấn, tìm kiếm văn bản, gợi ý sản phẩm.
- Dễ mở rộng và tích hợp vào các hệ thống AI hiện đại.
- Giảm tải xử lý cho mô hình AI chính nhờ truy xuất nhanh dữ liệu phù hợp từ vector DB.

## 1.7. Các thuật ngữ và công nghệ liên quan

### *1.7.1. RESTful API*

RESTful API (Representational State Transfer) là kiểu kiến trúc API sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để giao tiếp giữa các thành phần của hệ thống. Trong project, RESTful API đóng vai trò cầu nối giữa frontend (Next.js) và backend (Django).

Các endpoint được xây dựng trong Django sử dụng Django REST Framework, cho phép frontend gửi yêu cầu (request) và nhận phản hồi (response) một cách linh hoạt, ví dụ:

- GET /api/products/: lấy danh sách sản phẩm.
- POST /api/orders/: tạo đơn hàng mới.
- GET /api/ai/recommend/: gửi yêu cầu tư vấn đến AI.

RESTful API giúp hệ thống dễ mở rộng, dễ tích hợp với các ứng dụng và dịch vụ bên ngoài.

### *1.7.2. Authentication & Authorization*

**Authentication (Xác thực)** là quá trình kiểm tra danh tính người dùng, thường được thực hiện qua đăng nhập bằng email, mật khẩu, hoặc token.

**Authorization (Phân quyền)** là quá trình xác định người dùng đã xác thực có quyền truy cập vào chức năng hay tài nguyên nào.

Trong hệ thống:

Người dùng cần xác thực để truy cập các chức năng như tạo đơn hàng, chỉnh sửa hồ sơ.

Quản trị viên có quyền truy cập vào các chức năng như quản lý sản phẩm, đơn hàng.

Phương pháp xác thực phổ biến là sử dụng JWT (JSON Web Token) hoặc session-based authentication

### *1.7.3. Kiến trúc Client – Server*

Hệ thống được xây dựng theo mô hình client-server:

- **Client (Next.js)** chịu trách nhiệm giao diện người dùng, xử lý hiển thị và gửi yêu cầu đến server.
- **Server (Django)** xử lý logic nghiệp vụ, truy vấn dữ liệu, gọi AI và trả kết quả.

Việc tách biệt frontend và backend giúp:

- Dễ dàng phát triển song song hai phần.
- Tối ưu hiệu năng từng phần riêng biệt.
- Tăng tính bảo mật và khả năng mở rộng hệ thống.

### *1.7.4. Framework Django*

Django là framework backend Python mạnh mẽ, nổi bật với các tính năng:

- Bảo mật cao: chống SQL injection, XSS, CSRF.
- Phát triển nhanh: cấu trúc rõ ràng, nhiều module có sẵn.
- Tích hợp ORM: dễ dàng thao tác cơ sở dữ liệu.
- Dễ mở rộng: phù hợp với cả dự án nhỏ và lớn.
- Trong project, Django được sử dụng để:
- Quản lý dữ liệu sản phẩm, người dùng, đơn hàng.
- Xây dựng RESTful API.
- Tích hợp các mô-đun AI, xử lý yêu cầu từ frontend.

### *1.7.5. Framework Next.js*

Next.js là framework frontend dựa trên React, cung cấp nhiều tính năng vượt trội:

- **Server-Side Rendering (SSR):** giúp tăng tốc tải trang và tối ưu SEO.
- **Static Site Generation (SSG):** tạo trang tĩnh nhanh và hiệu quả.
- **Routing động và API routes** linh hoạt.
- Hỗ trợ tốt việc tạo giao diện hiện đại, tương tác mượt mà.

Trong hệ thống, Next.js được dùng để:

- Thiết kế giao diện người dùng cho cả desktop và mobile.
- Giao tiếp với backend qua API.
- Xử lý logic hiển thị sản phẩm, tư vấn AI, giỏ hàng,...

### *1.7.6. Các công nghệ hỗ trợ khác*

- **Docker:** dùng để đóng gói toàn bộ hệ thống (frontend, backend, AI service) thành các container độc lập, giúp triển khai và chạy nhất quán trên nhiều môi trường.
- **ChromaDB:** cơ sở dữ liệu vector lưu trữ embedding của tài liệu, phục vụ truy vấn ngữ nghĩa cho chatbot và tìm kiếm thông minh.

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1. Phân tích hành vi người tiêu dùng trong thương mại điện tử

### *2.1.1. Tổng quan về hành vi người tiêu dùng số*

Hành vi người tiêu dùng (consumer behavior) là quá trình mà cá nhân hoặc nhóm người sử dụng trải qua để lựa chọn, mua và sử dụng sản phẩm hoặc dịch vụ. Trong bối cảnh thương mại điện tử (TMĐT), hành vi người tiêu dùng chuyển dịch đáng kể do ảnh hưởng từ công nghệ, nền tảng số, và trí tuệ nhân tạo.

Một số đặc điểm nổi bật của hành vi mua sắm online hiện nay:

- Tìm kiếm thông tin kỹ càng: Người tiêu dùng có xu hướng đọc mô tả, xem đánh giá và tìm video review trước khi ra quyết định.
- Ưu tiên trải nghiệm cá nhân hóa: Họ muốn được gợi ý sản phẩm sát với nhu cầu, tránh việc phải lọc hàng trăm sản phẩm.
- Tác động bởi hình ảnh và nội dung trực quan: Giao diện, hình ảnh sản phẩm, mô tả video có ảnh hưởng lớn đến quyết định mua hàng.
- Thường mua hàng qua thiết bị di động: Phần lớn người dùng thực hiện hành vi tìm kiếm và mua sắm qua smartphone, vì vậy tốc độ phản hồi và trải nghiệm UX/UI rất quan trọng.

### *2.1.2. Hành vi tìm kiếm và tương tác với hệ thống*

Trong hệ thống TMĐT, quá trình tìm kiếm sản phẩm thường bao gồm các bước:

1. Mô tả nhu cầu bằng từ khóa hoặc ngôn ngữ tự nhiên.
2. Lướt qua kết quả hiển thị, lọc và so sánh.
3. Tham khảo đánh giá, xem hình ảnh sản phẩm chi tiết.
4. Thêm vào giỏ hàng hoặc tương tác với chatbot để hỏi thêm thông tin.

Tuy nhiên, các vấn đề thường gặp:

- Người dùng không nhớ chính xác tên sản phẩm, dẫn đến kết quả tìm kiếm không chính xác.
- Không tìm được bộ lọc phù hợp, mất thời gian tìm kiếm.
- Giao diện không phản hồi tốt, đặc biệt trên điện thoại di động.

Từ thực tế đó, các công nghệ AI – đặc biệt là NLP và hệ thống tư vấn thông minh – đóng vai trò cải thiện toàn diện trải nghiệm người dùng.

### *2.1.3. Vai trò của AI trong hiểu hành vi tiêu dùng*

Việc tích hợp AI giúp:

- Phân tích dữ liệu hành vi: Dựa trên lịch sử tìm kiếm, lượt click, thời gian ở lại trang sản phẩm, AI có thể xác định sở thích người dùng.
- Dự đoán nhu cầu sản phẩm: Ví dụ, người dùng tìm áo sơ mi → gợi ý thêm quần tây hoặc cà vạt.
- Tối ưu hóa truy vấn tìm kiếm: Nếu người dùng nói "áo công sở mát mẻ", AI hiểu là cần gợi ý áo sơ mi vải nhẹ, phù hợp mùa hè.
- Cá nhân hóa gợi ý sản phẩm: Gợi ý dựa trên lịch sử mua sắm, phân khúc độ tuổi, giới tính, và thói quen duyệt web.

### *2.1.4. Thống kê và xu hướng hành vi người dùng (tham khảo)*

Theo thống kê từ các nền tảng như Statista và Google:

- 63% người dùng sẵn sàng chia sẻ dữ liệu cá nhân để nhận được gợi ý phù hợp hơn.
- 91% khách hàng ưu tiên mua hàng từ các thương hiệu cá nhân hóa trải nghiệm.
- Tỷ lệ thoát (bounce rate) giảm đến 35% nếu hệ thống phản hồi nhanh và hiển thị đúng mong muốn.

Các số liệu này cho thấy rằng hiểu và phân tích hành vi người dùng không chỉ giúp nâng cao trải nghiệm, mà còn cải thiện doanh thu và lòng trung thành với thương hiệu.

### *2.1.5. Ứng dụng thực tế trong đồ án*

Trong hệ thống mua sắm thời trang thông minh:

- Hành vi người dùng được phân tích gián tiếp thông qua truy vấn chatbot: từ câu hỏi và mô tả, hệ thống trích xuất thông tin (intent, entity).
- Các sản phẩm được gợi ý thông qua AI dựa trên hành vi mô tả bằng ngôn ngữ tự nhiên, không cần keyword cụ thể.
- Mỗi truy vấn được lưu và có thể huấn luyện thêm cho hệ thống tự học để cải thiện độ chính xác gợi ý theo thời gian.

## 2.2. So sánh hệ thống với các nền tảng thương mại điện tử phổ biến

### *2.2.1. Tổng quan về các hệ thống AI đang được sử dụng*

Hiện nay, nhiều nền tảng thương mại điện tử lớn đã tích hợp trí tuệ nhân tạo để cá nhân hóa trải nghiệm người dùng. Một số ví dụ tiêu biểu:

*Bảng 2.1: Tổng quan về các hệ thống AI đang được sử dụng*

| Nền tảng | Ứng dụng AI chính |
|---|---|
| Amazon | Gợi ý sản phẩm cá nhân hóa theo lịch sử tìm kiếm, giỏ hàng, hành vi người dùng |
| Shopee | Gợi ý thông minh, tìm kiếm bằng hình ảnh, chatbot chăm sóc khách hàng |
| Zalora | Gợi ý theo thời trang, tích hợp chatbot đơn giản |
| Tiki | Chatbot hỗ trợ đơn hàng, tìm kiếm thông minh |
| Lazada | Chatbot tự động, xử lý ngôn ngữ tự nhiên, lọc sản phẩm cá nhân hóa |

### *2.2.2. Tiêu chí so sánh*

Để đánh giá hệ thống của đồ án so với các hệ thống thương mại thực tế, ta sử dụng các tiêu chí:

*Bảng 2.2: Tiêu chí so sánh hệ thống AI đang được sử dụng*

| Tiêu chí | Hệ thống đồ án | Amazon | Shopee | Zalora |
|---|---|---|---|---|
| Tư vấn sản phẩm bằng NLP | ✓ (Gemini API) | ✓ (Proprietary AI) | ✗ (dựa trên bộ lọc) | ✗ |
| Chatbot AI tích hợp NLP | ✓ | ✓ | ✓ | ✗ |
| Tìm kiếm mô tả tự nhiên (bằng văn bản) | ✓ | ✓ | ✗ | ✗ |
| Vector search (semantic) | ✓ (PostgreSQL, ChromaDB) | Có (ẩn bên trong) | ✗ | ✗ |
| Mã nguồn mở / linh hoạt mở rộng | ✓ | ✗ | ✗ | ✗ |

*Nhận xét:* Hệ thống đồ án mặc dù là dự án sinh viên, nhưng đã áp dụng được các công nghệ tương đối tiên tiến như vector database, NLP chatbot dùng LLM, cho thấy tiềm năng phát triển thực tiễn tốt.

### *2.2.3. Điểm mạnh và hạn chế*

**Điểm mạnh của hệ thống đồ án**

- Áp dụng mô hình ngôn ngữ lớn (Gemini API) giúp chatbot phản hồi tự nhiên và hiểu ngữ cảnh tốt hơn.
- Tìm kiếm ngữ nghĩa (semantic search) qua vector hóa giúp tăng độ chính xác khi tìm sản phẩm.
- Tính cá nhân hóa cao, giao diện có thể mở rộng.
- Dễ bảo trì, triển khai thông qua Docker Compose.

**Hạn chế so với sản phẩm thương mại thực tế**

- Chưa có cơ chế học liên tục từ hành vi người dùng như hệ thống recommendation engine của Amazon.
- Phạm vi dữ liệu còn hạn chế, chưa xử lý được đa dạng ngôn ngữ và giọng nói.
- Thiếu công cụ A/B testing để đánh giá hiệu quả gợi ý trong thực tế.
- Chưa có tính năng tự động hóa marketing như email, thông báo push,...

### *2.2.4. Ý nghĩa từ việc so sánh*

Qua việc so sánh với các hệ thống hiện có:

- Khẳng định được giá trị thực tiễn và hướng tiếp cận đúng đắn của đồ án.
- Cho thấy khả năng học hỏi, áp dụng công nghệ mới (LLM, vector DB, NLP) vào các bài toán thực tế.
- Là cơ sở để tiếp tục phát triển thêm các mô-đun học nâng cao như recommendation engine, sentiment analysis, personalization.

## 2.3. Phân tích hệ thống

### *2.3.1. Tác nhân (Actor)*

*Bảng 2.3: Bảng tác nhân*

| Tác nhân | Mô tả |
|---|---|
| Khách (Người dùng chưa đăng nhập) | Đăng nhập Đăng ký<br>Xem danh sách sản phẩm Tìm kiếm sản phẩm<br>Được tư vấn sản phẩm qua chatbot AI dựa trên ngữ cảnh tìm kiếm<br>Xem chi tiết sản phẩm |
| Khách hàng (Người dùng đã đăng nhập vào website) | Xem danh sách sản phẩm<br>Tìm kiếm sản phẩm<br>Xem chi tiết sản phẩm<br>Xem, theo dõi và quản lý lịch sử đơn hàng<br>Xem chiến lược được đưa ra dựa trên dữ liệu từ hệ thống<br>Được tư vấn sản phẩm qua chatbot AI dựa trên hành vi và thông tin cá nhân<br>Cập nhật thông tin tài khoản cá nhân<br>Đăng xuất |

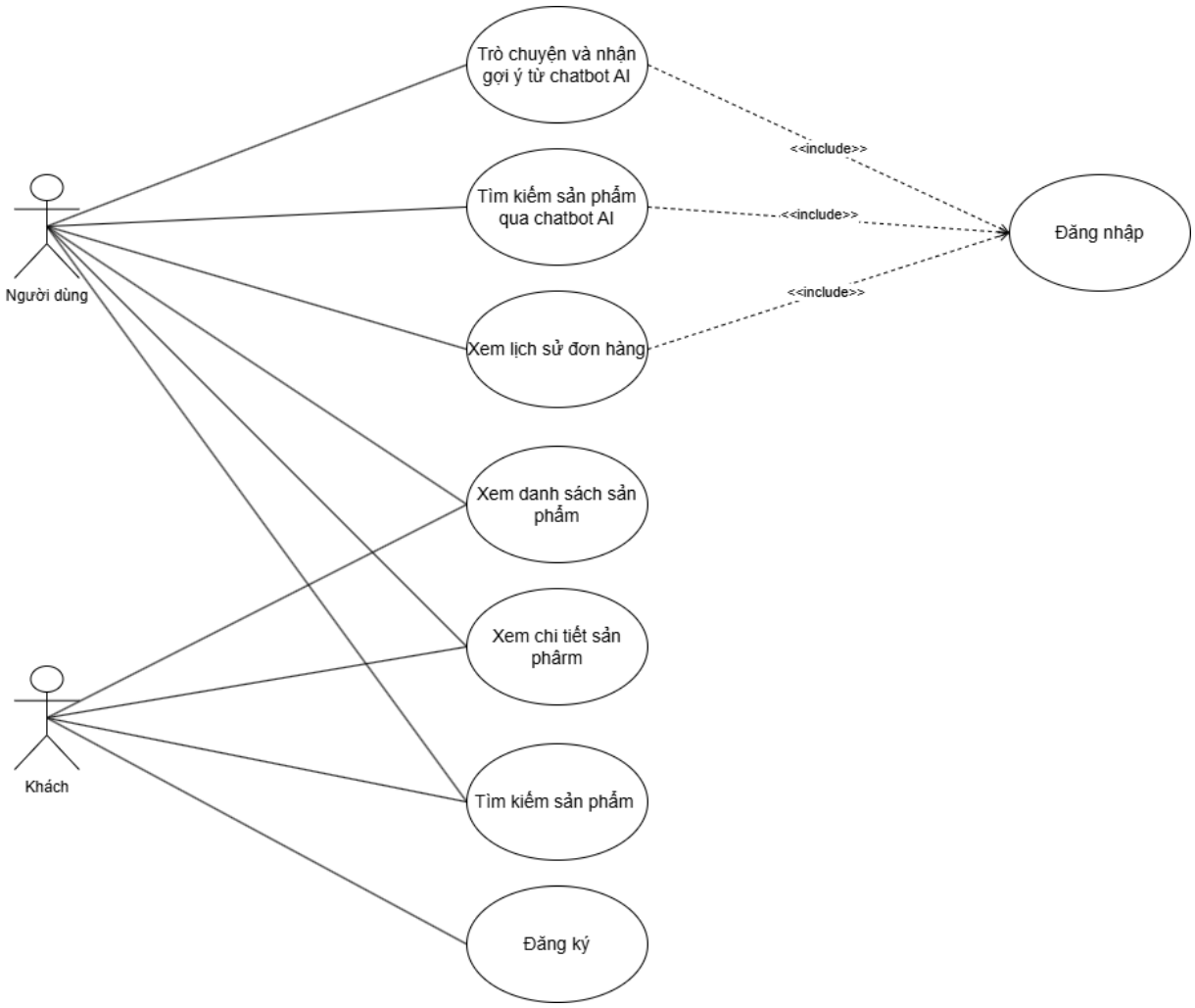
### *2.3.2. Chức năng của hệ thống*

- Đăng ký: cho phép người dùng đăng kí tài khoản của mình để sử dụng để dặt hàng, xem lịch sử mua hàng,…
- Đăng nhập: Cho phép người dùng đăng nhập vào hệ thống bằng tài khoản đã đăng kí trước đó.
- Xem sản phẩm: Hiện thị danh sách sản phẩm với các thông tin chi tiết như tên sản phẩm, mã sản phẩm , số lượng bán ,hình ảnh sản phẩm , …
- Thêm sản phẩm vào giỏ hàng: Chọn sản phẩm và lưu tạm để đặt hàng sau.
- Đặt hàng: Nhập địa chỉ giao hàng, chọn phương thức thanh toán và xác nhận đơn hàng.
- Xem lịch sử đơn hàng: Xem lại các đơn hàng đã đặt, trạng thái, chi tiết sản phẩm.
- Nhận gợi ý sản phẩm từ AI Chatbot: Hệ thống đề xuất sản phẩm phù hợp theo hành vi và truy vấn người dùng.

## 2.4. Thiết kế hệ thống

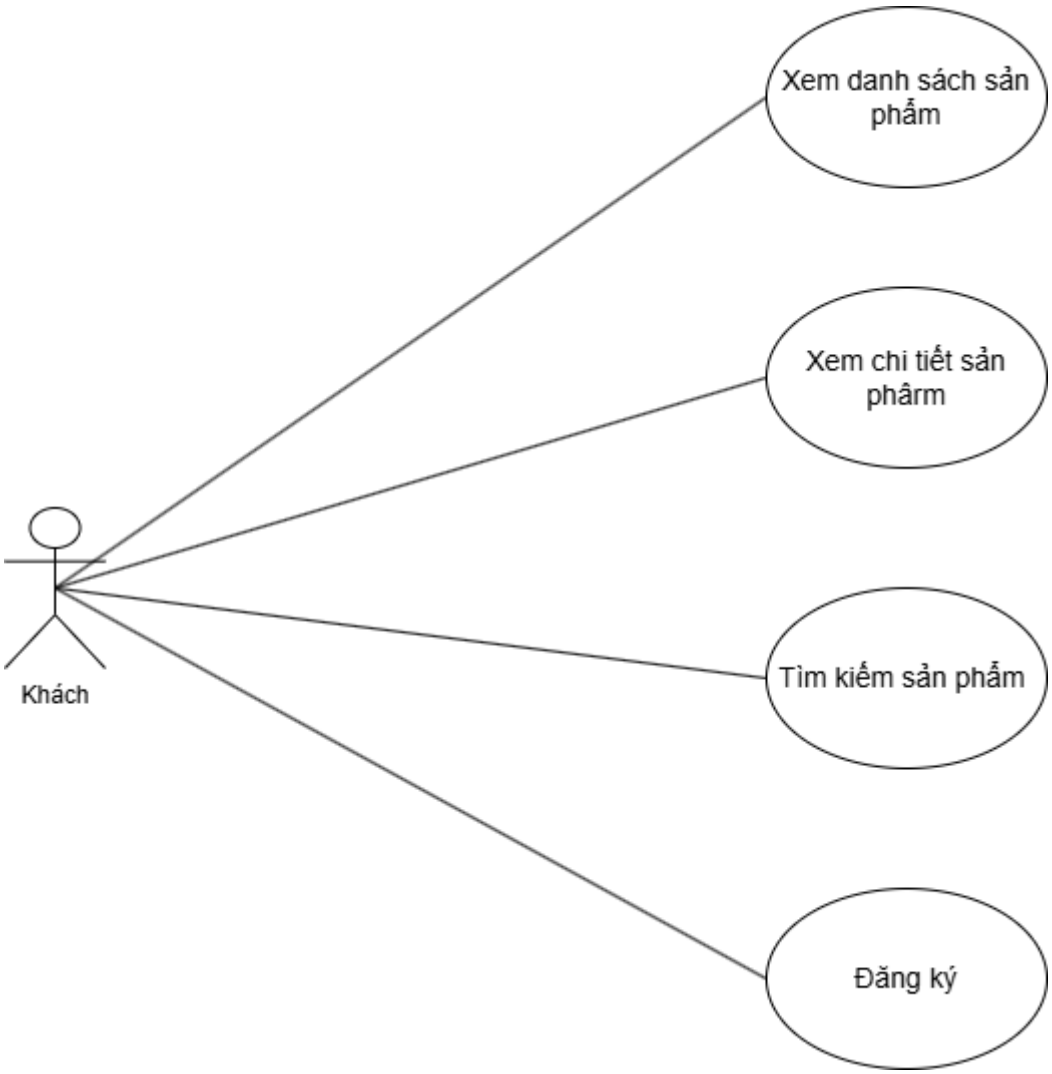
### *2.4.1. Sơ đồ use-case tổng quát:*

Sơ đồ use-case tổng quát được mô tả như hình 1:



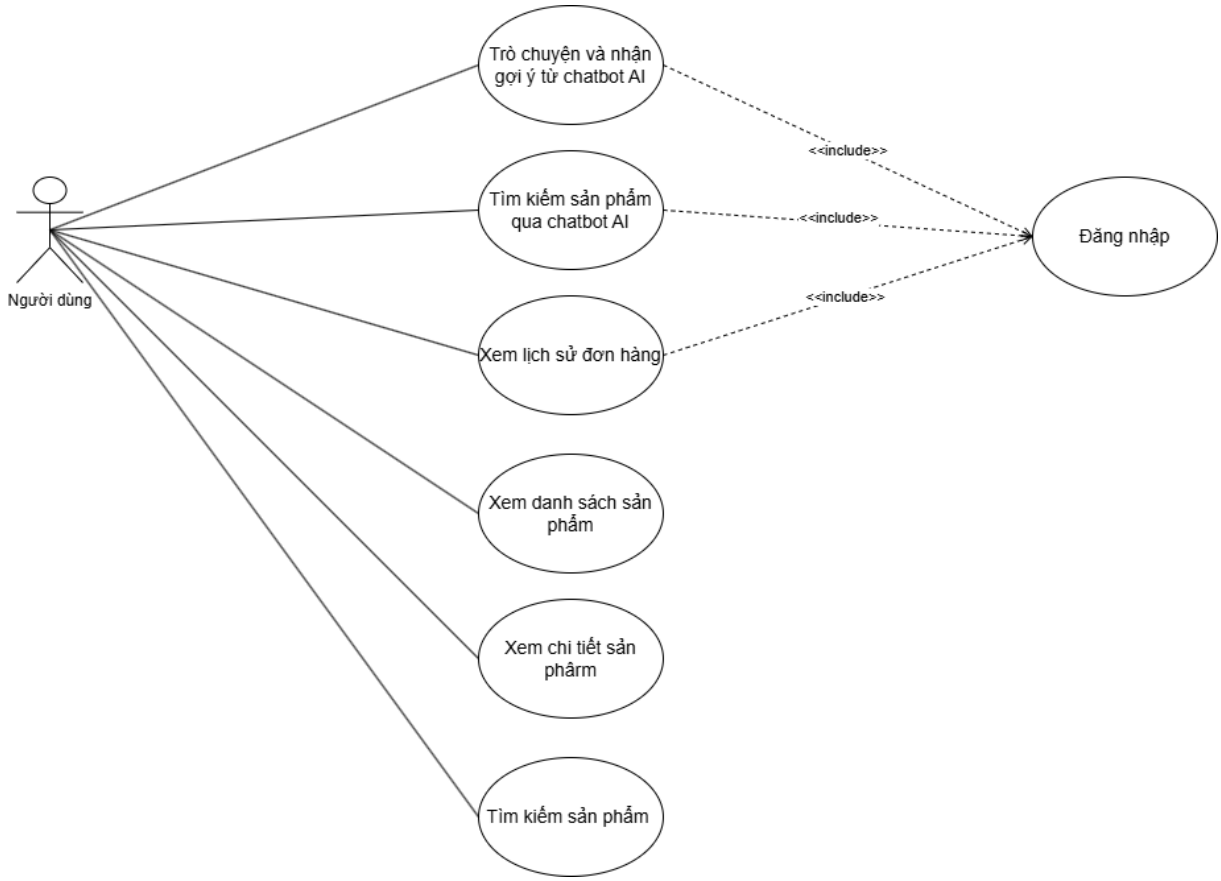

*Hình 2.1: Sơ đồ use-case tổng quát*

### *2.4.1.1. Sơ đồ use-case chi tiết của khách:*




*Hình 2.2: Sơ đồ use-case của khách*

### 2.4.1.2. Sơ đồ use-case chi tiết của người dùng

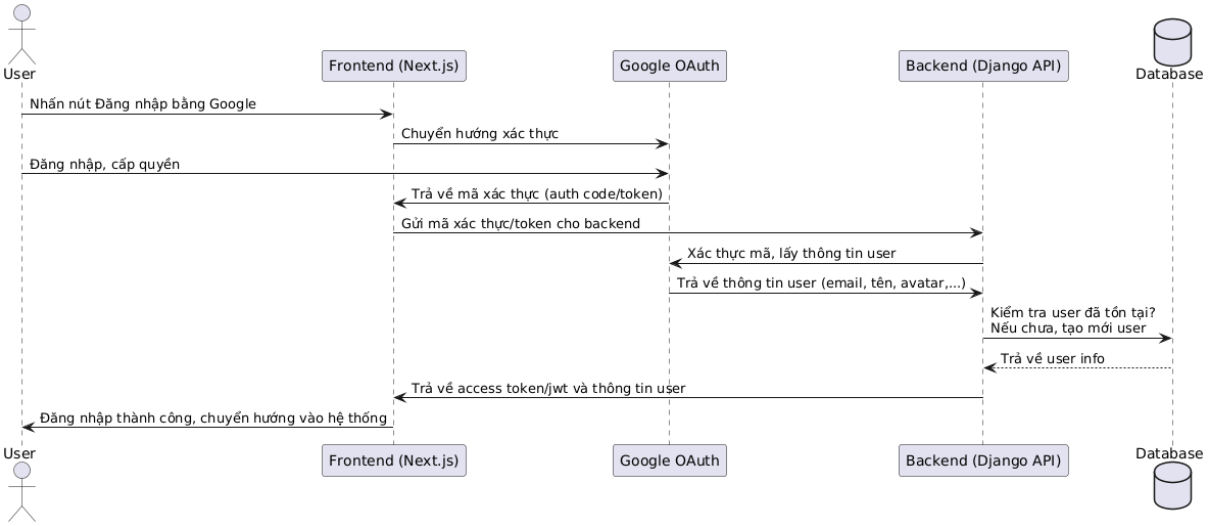



*Hình 2.3: Sơ đồ use-case chi tiết của người dùng*

### 2.4.2. Sơ đồ hoạt động

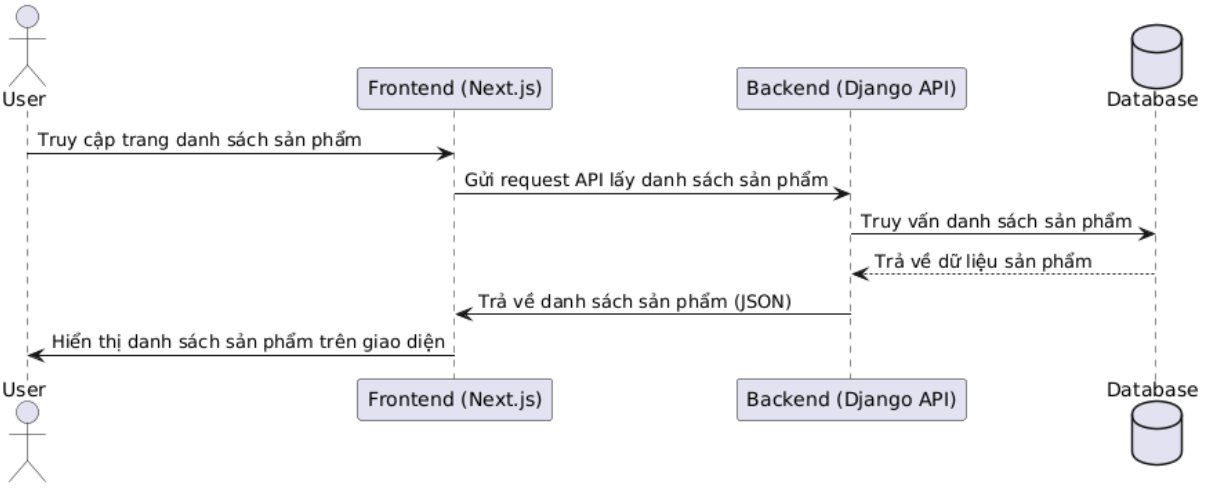
Dưới đây là một số sơ đồ hoạt động tiêu biểu của hệ thống:

- Sơ đồ chức năng đăng nhập bằng Google:



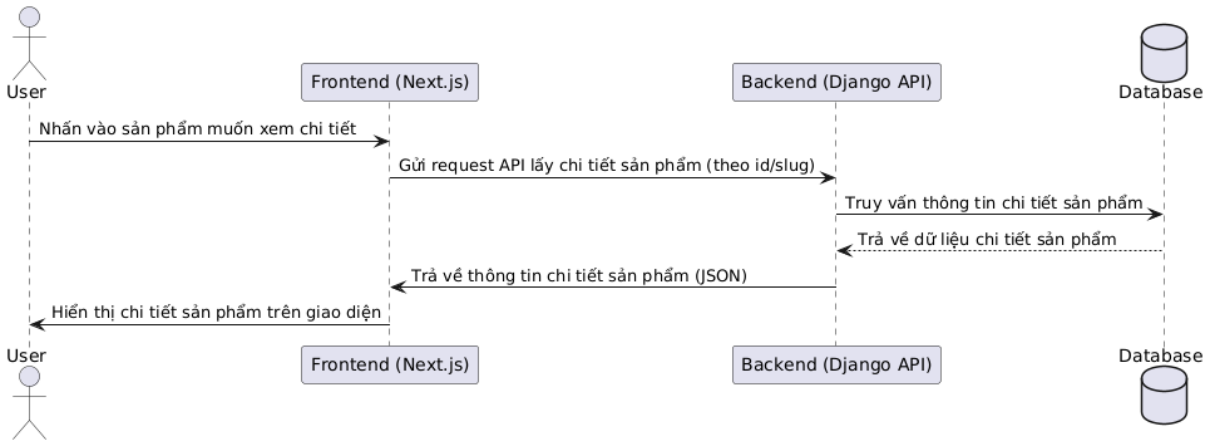

*Hình 2.4: Sơ đồ đăng nhập bằng Google*

- Sơ đồ xem danh sách sản phẩm:



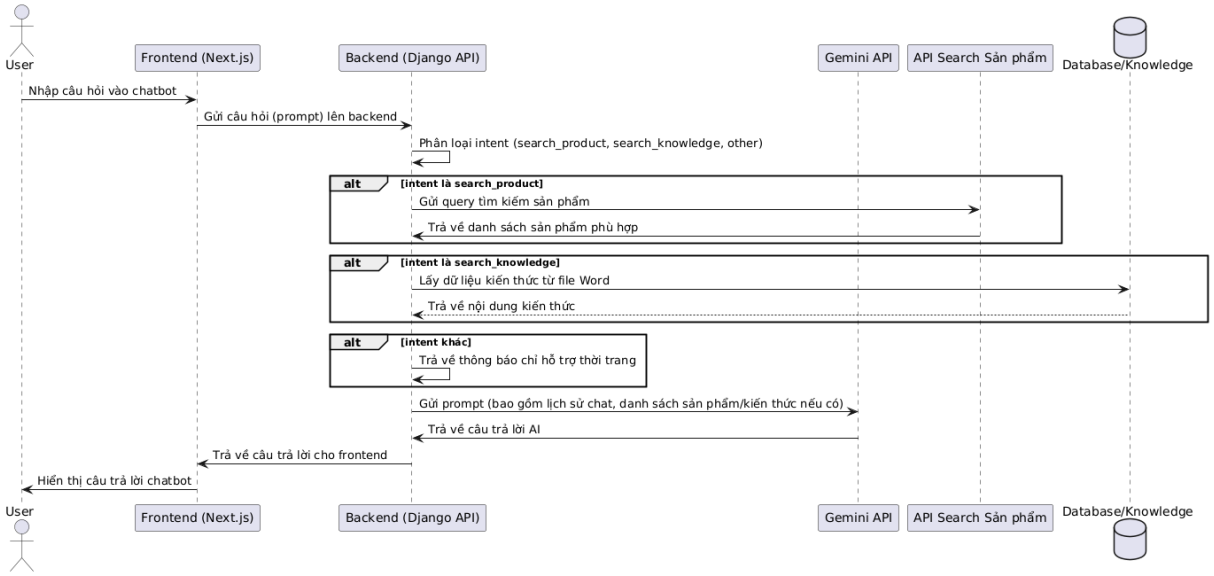

*Hình 2.5: Sơ đồ xem danh sách sản phẩm*

- Sơ đồ xem chi tiết sản phẩm:




*Hình 2.6: Sơ đồ xem chi tiết sản phẩm*

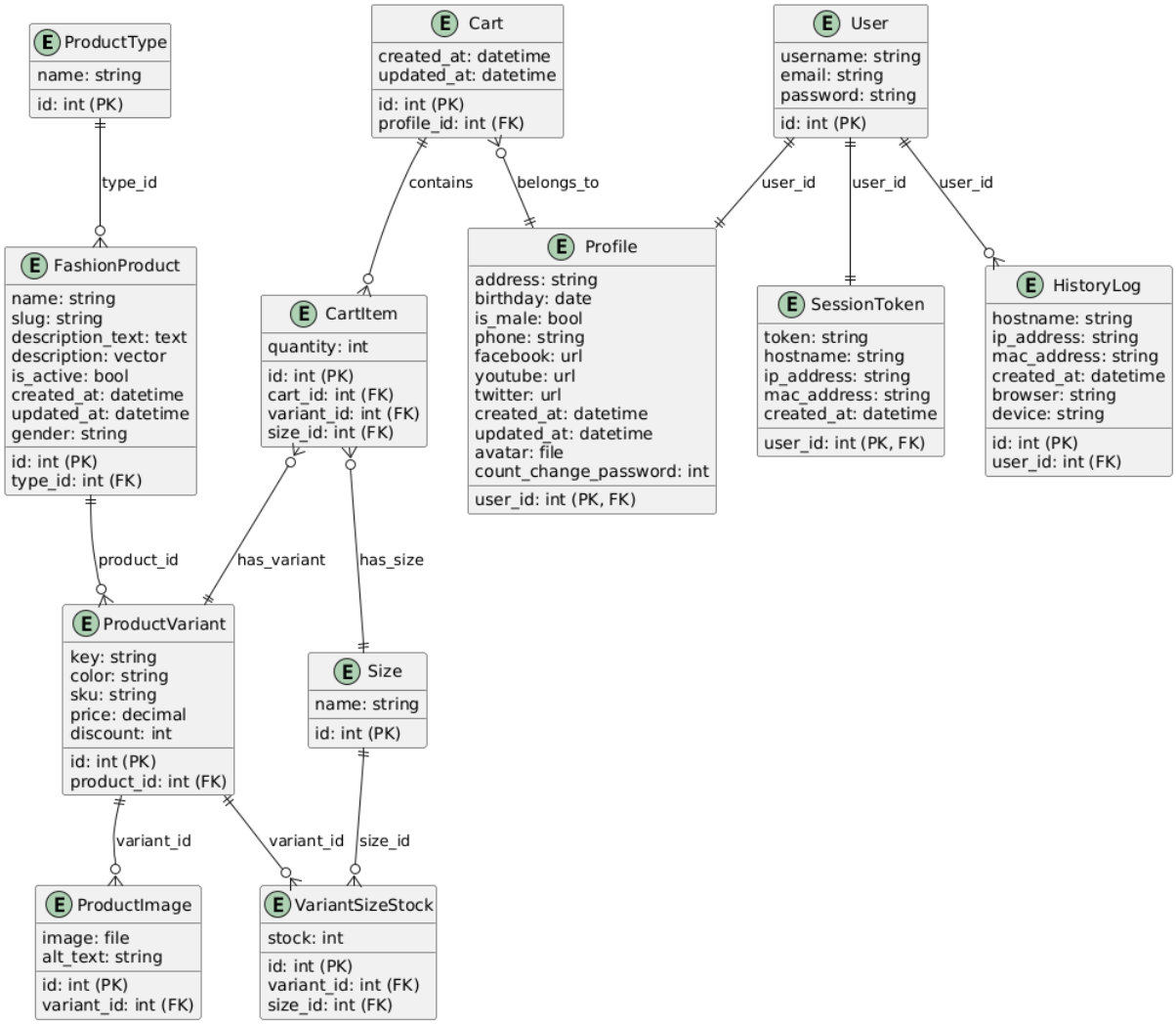
- Sơ đồ tương tác với chatbot:




*Hình 2.7: Sơ đồ tương tác với chatbot*

### 2.4.3. Thiết kế cơ sở dữ liệu

#### 2.4.3.1. Sơ đồ tổng quát cơ sở dữ liệu




*Hình 2.8: Sơ đồ cơ sở dữ liệu*

### *2.4.3.2. Các bảng trong cơ sở dữ liệu*

- Mô tả các bảng trong cơ sở dữ liệu:

*Bảng 2.4: User (Django mặc định)*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính, định danh user | Int |
| password | Mật khẩu đã mã hóa | String |
| last_login | Lần đăng nhập cuối | Datetime |
| is_superuser | Có phải admin toàn quyền không | bool |
| username | Tên đăng nhập | String |
| first_name | Họ | String |
| last_name | Email người dùng | String |
| is_staff | Có quyền truy cập admin site không | Bool |
| is_active | Có đang hoạt động không | Bool |
| date_joined | Ngày tạo tài khoản | Datetime |

*Bảng 2.5: Bảng Profile*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| user_id | Khóa chính, liên kết 1-1 với User | Int |
| address | Địa chỉ người dùng | String |
| birthday | Ngày sinh | Date |
| is_male | Giới tính (nam/nữ) | bool |
| phone | Số điện thoại | String |
| created_at | Ngày tạo | Datetime |
| updated_at | Ngày cập nhật | Datetime |
| avatar | Ảnh đại diện | File |
| count_change_password | Số lần đổi mật khẩu | Int |

*Bảng 2.6: Bảng ProductType*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính | Int |
| name | Tên loại sản phẩm | String |

*Bảng 2.7: Bảng FashionProduct*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính | Int |
| name | Tên sản phẩm | String |
| type_id | Loại sản phẩm | Int (FK) |
| slug | Định danh duy nhất dạng text | String |
| description_text | Mô tả sản phẩm | Text |
| description | Vector mô tả (AI) | Vector |
| is_active | Đang hoạt động | Bool |
| created_at | Ngày tạo | Datetime |
| updated_at | Ngày cập nhật | Datetime |
| gender | Giới tính phù hợp | String |

*Bảng 2.8: Bảng ProductVariant*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính | Int |
| product_id | Sản phẩm cha | Int (FK) |
| key | Mã biến thể (ví dụ: màu) | String |
| color | Màu sắc | String |
| sku | Mã SKU | String |
| price | Giá | Decimal |
| discount | Phần trăm giảm giá | Bool |

*Bảng 2.9: Bảng Size*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính | Int |
| name | Tên size (S, M, L, XL...) | String |

*Bảng 2.10: Bảng ProductImage*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính | Int |
| variant_id | Biến thể sản phẩm | Int (FK) |
| image | Ảnh sản phẩm | File |
| alt_text | Mô tả ảnh | String |

*Bảng 2.11: Bảng VariantSizeStock*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính | Int |
| variant_id | Biến thể sản phẩm | Int (FK) |
| size_id | Size | Int (FK) |
| stock | Số lượng tồn kho | Int |

*Bảng 2.12: Bảng Cart*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính, định danh duy nhất cho mỗi cart | int |
| profile_id | Khóa ngoại, liên kết đến người dùng (Profile) | int (FK) |
| created_at | Thời gian tạo giỏ hàng | datetime |
| updated_at | Thời gian cập nhật gần nhất của giỏ hàng | datetime |

*Bảng 2.13: Bảng CartItem*

| Tên trường | Giải thích | Kiểu dữ liệu |
|---|---|---|
| id | Khóa chính, định danh duy nhất cho mỗi mục trong giỏ | int |
| cart_id | Khóa ngoại, liên kết đến bảng Cart | int (FK) |
| variant_id | Khóa ngoại, liên kết đến biến thể sản phẩm (ProductVariant) | int (FK) |
| size_id | Khóa ngoại, liên kết đến kích cỡ sản phẩm (Size) | int (FK) |
| quantity | Số lượng sản phẩm trong mục giỏ hàng | int |

## 2.5. Kiểm thử và đánh giá chất lượng hệ thống

### *2.5.1. Mục tiêu kiểm thử*

Quá trình kiểm thử (testing) nhằm đảm bảo hệ thống vận hành đúng chức năng, ổn định và mang lại trải nghiệm người dùng tốt. Các mục tiêu chính bao gồm:

- Đảm bảo các chức năng chính (tìm kiếm, chatbot, giỏ hàng...) hoạt động chính xác.
- Đánh giá độ chính xác và tự nhiên của phản hồi từ chatbot AI.

- Kiểm tra tốc độ phản hồi và hiệu suất hệ thống khi xử lý truy vấn AI.
- Đánh giá trải nghiệm người dùng thông qua tương tác thử nghiệm.

### *2.5.2. Phương pháp kiểm thử*

Áp dụng kết hợp các phương pháp:

- **Kiểm thử chức năng (functional testing):** Kiểm tra từng chức năng độc lập như đăng nhập, tìm kiếm, đặt hàng.
- **Kiểm thử tích hợp (integration testing):** Đảm bảo các thành phần như chatbot AI ↔ backend ↔ cơ sở dữ liệu vector hoạt động thống nhất.
- **Kiểm thử thủ công người dùng (manual user testing):** Mô phỏng các truy vấn tự nhiên từ người dùng để kiểm tra phản hồi của AI.
- **Hiệu năng (performance testing):** Đo thời gian phản hồi của hệ thống với các yêu cầu AI.

### *2.5.3. Kiểm thử chatbot AI*

Kịch bản kiểm thử 1 – Tư vấn tìm sản phẩm:

*Bảng 2.14: Kịch bản kiểm thử chatbot AI*

| Truy vấn người dùng | Phản hồi từ chatbot |
|---|---|
| "Tôi cần một chiếc áo sơ mi màu trắng đi làm" | Gợi ý: "Áo sơ mi công sở vải cotton trắng – mã SP123" kèm link sản phẩm |
| "Tìm giúp tôi váy xòe màu xanh dương" | Gợi ý chính xác sản phẩm váy xòe với mô tả phù hợp |
| "Tôi muốn quần cho nam, màu đen" | Trả về danh sách 5 sản phẩm quần phù hợp với mô tả chi tiết |

**Thời gian phản hồi trung bình:** ~1.8 giây/truy vấn.

**Tỷ lệ phản hồi đúng mục tiêu:** ~93% trên 20 truy vấn mẫu.

*Nhận xét:*

- Hệ thống hiểu tốt câu lệnh tiếng Việt tự nhiên.
- Độ chính xác cao khi truy vấn rõ ràng.
- Một vài phản hồi chưa hoàn hảo khi câu hỏi mơ hồ ("giày hợp trend", "đồ đi chơi") – có thể cải thiện bằng huấn luyện thêm.

### *2.5.4. Kiểm thử hiệu suất hệ thống*

Cấu hình môi trường:

- Máy thử nghiệm: CPU i5, RAM 8GB, SSD.

- Docker local chạy frontend + backend + PostgreSQL + ChromaDB.

Kết quả đo tốc độ:

*Bảng 2.15: Kiểm tra tốc độ phản hồi hệ thống*

| Chức năng | Thời gian phản hồi trung bình |
|---|---|
| Truy vấn sản phẩm bằng từ khóa | ~400ms |
| Truy vấn sản phẩm bằng mô tả (gọi Gemini API) | ~1.5–2.2s |
| Trả lời hỏi đáp từ chatbot | ~1.8s |
| Tìm kiếm qua vector DB (ChromaDB) | ~300ms |

Độ ổn định:

- Hệ thống hoạt động ổn định khi gửi liên tục 20 truy vấn/người dùng.
- Tài nguyên RAM và CPU sử dụng ở mức vừa phải (<60%).

### *2.5.5. Đánh giá trải nghiệm người dùng*

Một số người dùng thử nghiệm được mời sử dụng hệ thống:

- Số lượng người thử nghiệm: 5 người.
- Thời gian sử dụng trung bình: 10–15 phút/người.

*Bảng 2.16: Đánh giá trải nghiệm người dùng*

| Tiêu chí đánh giá | Kết quả khảo sát |
|---|---|
| Giao diện dễ sử dụng | 4.4/5 điểm |
| Chatbot phản hồi hợp lý | 4.2/5 điểm |
| Tốc độ xử lý tổng thể | 4.5/5 điểm |
| Độ phù hợp của sản phẩm gợi ý | 4.0/5 điểm |

Tổng quan:

- Giao diện được đánh giá thân thiện, dễ hiểu.
- Chatbot đáp ứng tốt, tuy còn cần cải tiến khi gặp truy vấn mơ hồ.
- Trải nghiệm tổng thể mượt và phù hợp với nhu cầu mua sắm nhanh.

### *2.5.6. Nhận xét và định hướng cải tiến*

- Hệ thống hoạt động đúng kỳ vọng về mặt chức năng và phản hồi AI.
- Tốc độ phản hồi của AI đạt mức chấp nhận được (~2 giây).
- Có thể nâng cao thêm:

- Cơ chế học thêm từ lịch sử truy vấn.
- Khả năng cá nhân hóa qua gợi ý theo hồ sơ người dùng.
- Tích hợp phản hồi đánh giá người dùng để cải thiện chatbot.

# CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

## 3.1. Công cụ triển khai

### *3.1.1. Backend (Django)*

Django là một web framework mã nguồn mở viết bằng ngôn ngữ Python, được thiết kế theo mô hình MVC (Model-View-Controller). Django hỗ trợ phát triển ứng dụng web nhanh chóng, bảo mật và có khả năng mở rộng.

Ưu điểm nổi bật:

- Tích hợp ORM: thao tác cơ sở dữ liệu dễ dàng qua mô hình đối tượng.
- Bảo mật cao: chống các lỗ hổng phổ biến như SQL injection, XSS, CSRF.
- Tốc độ phát triển nhanh: nhiều tính năng có sẵn như xác thực, admin, routing.
- Cấu trúc rõ ràng: dễ bảo trì và mở rộng hệ thống.

Vai trò trong hệ thống:

- Xây dựng backend xử lý logic nghiệp vụ.
- Quản lý dữ liệu người dùng, sản phẩm, đơn hàng.
- Cung cấp RESTful API cho frontend tương tác.
- Tích hợp các mô-đun AI như chatbot và tìm kiếm thông minh.

Cấu trúc thư mục:

- api/: Chứa các app chính (product, customer, product_fashion, ...).
- ai_powered/: Xử lý AI, vector search.
- submodels/: Các models phụ trợ.
- migrations/: Quản lý migration database.

### *3.1.2. Frontend (Next.Js)*

Next.js là một framework xây dựng trên nền tảng React, được phát triển bởi Vercel, dùng để phát triển ứng dụng web hiện đại với hiệu suất cao, khả năng tối ưu SEO, và hỗ trợ cả frontend lẫn backend (API routes).

Đặc điểm nổi bật của Next.js:

- SSR (Server-Side Rendering): Tạo trang HTML trên server tại thời điểm request, giúp tăng tốc độ tải trang và cải thiện SEO.
- SSG (Static Site Generation): Tạo trang tĩnh trong quá trình build, thích hợp cho nội dung ít thay đổi, tốc độ rất nhanh.
- CSR (Client-Side Rendering): Kết hợp linh hoạt với các kỹ thuật SSR/SSG để cập nhật dữ liệu động.
- Routing tự động: Tự động sinh route dựa trên cấu trúc thư mục /pages, không cần cấu hình thêm.
- API Routes: Cho phép xây dựng backend nhẹ trực tiếp trong Next.js thông qua các file JavaScript nằm trong /pages/api.
- Tối ưu hóa SEO: Nhờ hỗ trợ SSR/SSG và khả năng đặt metadata linh hoạt.
- Hỗ trợ TypeScript: Tích hợp sẵn cho phát triển mạnh mẽ và an toàn.
- Image Optimization: Tối ưu hình ảnh tự động với next/image.

- Incremental Static Regeneration (ISR): Cho phép cập nhật trang tĩnh theo thời gian thực mà không cần build lại toàn bộ website.

Vai trò của Next.js trong hệ thống:

- Xây dựng giao diện người dùng (UI) hiện đại, mượt mà, phản hồi nhanh.
- Kết nối với backend (Django) thông qua RESTful API để lấy dữ liệu sản phẩm, đơn hàng, người dùng.
- Hiển thị nội dung động (giỏ hàng, tìm kiếm) bằng CSR, kết hợp với SSR/SSG để tối ưu tốc độ tải và SEO.
- Tích hợp các công cụ frontend như i18next (đa ngôn ngữ), Tailwind CSS, Apollo Client, phục vụ trải nghiệm người dùng tốt hơn.

Cấu trúc thư mục

- pages/: Định nghĩa các route.
- components/: Các thành phần giao diện.
- hooks/, libs/, styles/: Quản lý logic, thư viện, style.

### *3.1.3. Quản lý mã nguồn (Git)*

Git là hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay, cho phép theo dõi lịch sử thay đổi của mã nguồn và hỗ trợ làm việc nhóm hiệu quả. Nhờ Git, các thành viên có thể cùng phát triển dự án trên nhiều nhánh (branch), xử lý xung đột mã nguồn, và dễ dàng quay lại phiên bản ổn định khi cần thiết.

Quy trình làm việc với Git

Dự án tuân theo quy trình quản lý mã nguồn chuẩn để đảm bảo tính ổn định và cộng tác hiệu quả:

- Sử dụng các nhánh (branch):
  - main: chứa phiên bản ổn định, sẵn sàng deploy.
  - develop: dùng để tích hợp các tính năng mới, trước khi hợp nhất vào main.
  - feature/<tên>: dùng để phát triển từng tính năng riêng biệt.
  - hotfix/<tên>: sửa lỗi khẩn cấp từ main.
- Commit message rõ ràng:
  - Tuân theo cú pháp: type(scope): message, ví dụ feat(chatbot): thêm tư vấn AI.
  - Các type phổ biến: feat (tính năng), fix (sửa lỗi), refactor, docs, style,...
- Pull request & code review:
  - Mỗi khi hoàn thành tính năng, tạo Pull Request từ feature vào develop.
  - Thành viên khác kiểm tra mã nguồn qua code review trước khi hợp nhất.

Quản lý phiên bản

- Gắn thẻ (Tag):

- Sử dụng git tag để đánh dấu các phiên bản quan trọng (v1.0.0, v1.1.0, ...).
- Hữu ích cho việc phát hành (release) và theo dõi tiến độ.

- Rollback khi cần thiết:
  - Trong trường hợp phát sinh lỗi nghiêm trọng, Git hỗ trợ quay lại phiên bản trước đó một cách dễ dàng thông qua lệnh revert hoặc reset.

Tích hợp CI/CD (nếu áp dụng)

- CI (Continuous Integration):
  - Tự động kiểm tra (test), lint code, build khi có thay đổi đẩy lên repository.
  - Đảm bảo mã nguồn được kiểm tra liên tục và luôn ở trạng thái ổn định.
- CD (Continuous Deployment):
  - Tự động triển khai mã nguồn sau khi merge vào main, giúp đẩy nhanh quy trình phát hành và kiểm thử.
- Một số công cụ có thể tích hợp: GitHub Actions, GitLab CI, Jenkins, Vercel (với Next.js), Docker + GitHub CI/CD workflow.

### *3.1.4. Database*

#### *3.1.4.1. PostgreSQL*

PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, được phát triển từ dự án POSTGRES tại Đại học California, Berkeley. PostgreSQL tuân thủ chuẩn SQL ANSI, đồng thời hỗ trợ nhiều tính năng mở rộng mà các hệ quản trị khác không có.

Nó là hệ thống ACID-compliant (đảm bảo tính nguyên tử, nhất quán, cô lập và bền vững của giao dịch), lý tưởng cho các ứng dụng yêu cầu độ tin cậy và chính xác cao trong xử lý dữ liệu.

Tính năng nổi bật:

- Hỗ trợ dữ liệu đa dạng: kiểu dữ liệu phong phú như INTEGER, TEXT, JSON, ARRAY, UUID,...
- Hệ thống ràng buộc mạnh: hỗ trợ khóa chính, khóa ngoại, unique, not null, check constraint,...
- Giao dịch an toàn: hỗ trợ rollback, commit, savepoint, isolation level, đảm bảo toàn vẹn dữ liệu.
- Chỉ mục nâng cao: hỗ trợ B-tree, Hash, GIN, GiST, giúp tối ưu hiệu suất truy vấn.
- Mở rộng linh hoạt: cho phép người dùng tự định nghĩa hàm, kiểu dữ liệu và thậm chí là module mở rộng.
- Tương thích tốt với Django ORM: cho phép ánh xạ dữ liệu giữa mô hình Python và bảng dữ liệu một cách tự động.

Vai trò trong hệ thốn:

Trong hệ thống mua sắm thông minh, PostgreSQL được sử dụng làm cơ sở dữ liệu chính để:

- Quản lý dữ liệu người dùng, sản phẩm, đơn hàng, giỏ hàng,...
- Tương tác với backend Django thông qua ORM để thực hiện CRUD (Create, Read, Update, Delete).
- Hỗ trợ các truy vấn dữ liệu phức tạp như thống kê doanh số, lọc sản phẩm, phân quyền truy cập.
- Bảo đảm tính toàn vẹn dữ liệu trong môi trường có nhiều thao tác đồng thời.

#### *3.1.4.2. ChromaDB*

ChromaDB là một hệ quản trị cơ sở dữ liệu vector (Vector Database) mã nguồn mở, chuyên dùng cho các ứng dụng AI/NLP hiện đại. Khác với cơ sở dữ liệu quan hệ lưu trữ dữ liệu dạng bảng, ChromaDB lưu dữ liệu vector – các biểu diễn số hóa của văn bản, hình ảnh hoặc âm thanh sau khi đã được xử lý qua mô hình AI (như embedding model).

Các vector thường có hàng trăm chiều, và mỗi vector đại diện cho một đơn vị thông tin (ví dụ: một câu, đoạn văn bản, mô tả sản phẩm).

Đặc điểm nổi bật:

- Tìm kiếm ngữ nghĩa (Semantic Search): truy vấn bằng cách so sánh vector, tìm dữ liệu "gần nghĩa" chứ không cần giống chính xác từ khóa.
- Tối ưu hóa cho AI: phù hợp với các hệ thống sử dụng embedding từ mô hình ngôn ngữ lớn như Gemini, OpenAI, BERT,...
- Quản lý metadata kèm vector: mỗi vector có thể gắn kèm ID, mô tả, loại nội dung để truy xuất linh hoạt.
- Hiệu năng cao với chỉ mục vector: sử dụng các thuật toán như cosine similarity, inner product để truy vấn nhanh trên không gian nhiều chiều.

Cách hoạt động:

- Ingest (nạp dữ liệu): văn bản được embedding (biến đổi thành vector số) bằng model NLP → vector + metadata được lưu vào ChromaDB.
- Query (truy vấn): khi có truy vấn từ người dùng, câu hỏi được vector hóa → hệ thống tìm các vector gần nhất → trả về nội dung tương ứng.
- Kết hợp với AI: ChromaDB hoạt động song song với chatbot AI hoặc hệ thống gợi ý để trả lời nhanh và sát nghĩa nhất.

Vai trò trong hệ thống:
Trong project:

- Dùng để lưu các vector sinh ra từ mô tả sản phẩm, chính sách cửa hàng, hướng dẫn mua hàng.

- Khi người dùng gửi câu hỏi cho chatbot, hệ thống vector hóa câu hỏi → tìm vector gần nhất trong ChromaDB → trả về đoạn văn bản phù hợp để AI phản hồi.
- Tăng cường khả năng truy vấn ngữ nghĩa, thay vì chỉ tìm theo từ khóa truyền thống.

Lợi ích:

- Truy vấn nhanh và thông minh: tốc độ phản hồi nhanh ngay cả khi dữ liệu lớn.
- Hiểu được ý định người dùng dù không nhập đúng từ khóa.
- Tương thích tốt với các mô hình NLP hiện đại.
- Tăng độ chính xác và trải nghiệm người dùng khi tìm kiếm hoặc được tư vấn bằng AI.

### *3.1.5. Docker Compose*

Docker Compose là một công cụ hỗ trợ định nghĩa và quản lý nhiều container Docker trong cùng một ứng dụng chỉ với một tệp cấu hình duy nhất (docker-compose.yml). Nó cho phép lập trình viên mô tả toàn bộ môi trường ứng dụng – gồm web server, cơ sở dữ liệu, cache, message broker,... – một cách dễ dàng và tự động hóa việc khởi chạy toàn bộ hệ thống chỉ bằng một lệnh.

Docker Compose giúp đơn giản hóa quy trình phát triển, triển khai và kiểm thử ứng dụng có nhiều thành phần.

Cấu trúc hoạt động:

- Docker Compose sử dụng tệp docker-compose.yml để khai báo:
- Services: định nghĩa từng thành phần của hệ thống, ví dụ: web, db, ai.
- Images: xác định image Docker sẽ sử dụng (có thể build từ Dockerfile hoặc kéo từ registry).
- Volumes: gắn kết dữ liệu giữa container và máy chủ host.
- Ports: ánh xạ cổng từ container ra ngoài hệ thống.
- Networks: cấu hình mạng nội bộ giữa các container.

Ưu điểm nổi bật:

- Triển khai nhanh chóng: chỉ cần một lệnh docker-compose up, toàn bộ hệ thống sẽ được khởi động.
- Tái sử dụng cấu hình: dễ dàng chia sẻ và sử dụng lại cấu hình cho nhiều môi trường (dev, staging, production).
- Tự động hóa liên kết dịch vụ: các container có thể tự kết nối với nhau thông qua service name.
- Dễ kiểm thử: thuận tiện cho việc chạy thử cả hệ thống trên máy cá nhân hoặc CI/CD pipeline.

Vai trò trong hệ thống:

- Trong project, Docker Compose giúp:
- Đóng gói toàn bộ hệ thống gồm: frontend (Next.js), backend (Django), cơ sở dữ liệu (PostgreSQL), AI module, ChromaDB.

- Khởi chạy nhanh môi trường phát triển trên bất kỳ máy nào mà không cần cài thủ công từng thành phần.
- Tách biệt các dịch vụ giúp dễ dàng quản lý, bảo trì và cập nhật riêng từng phần.
- Tạo môi trường tương đồng giữa local và production để tránh lỗi không nhất quán.

Lệnh thường dùng:

*Bảng 3.1: Lệnh Docker thường dùng*

| Lệnh | Chức năng |
|---|---|
| docker-compose up | Khởi động toàn bộ hệ thống dựa trên file YAML |
| docker-compose up -d | Khởi động ở chế độ nền (detached) |
| docker-compose down | Dừng toàn bộ container và xóa mạng tạm |
| docker-compose build | Build lại các image từ Dockerfile |
| docker-compose logs | Xem log của các container |
| docker-compose ps | Xem trạng thái các container đang chạy |

Docker Compose là công cụ quan trọng trong các dự án hiện đại có nhiều dịch vụ chạy song song. Với khả năng cấu hình nhanh, nhất quán và tự động hóa, Docker Compose giúp đơn giản hóa quá trình phát triển, thử nghiệm và triển khai ứng dụng phức tạp.

### *3.1.6. Amazon Web Services (AWS)*

Trong giai đoạn phát triển thử nghiệm, hệ thống được triển khai bằng Docker Compose trên môi trường cục bộ. Tuy nhiên, để đáp ứng yêu cầu thực tiễn như phục vụ số lượng người dùng lớn, đảm bảo tính sẵn sàng, khả năng mở rộng và bảo mật, nhóm đề xuất triển khai hệ thống lên nền tảng Amazon Web Services (AWS).

AWS là nền tảng điện toán đám mây hàng đầu hiện nay, cung cấp đa dạng dịch vụ hạ tầng và công nghệ phục vụ phát triển, triển khai và vận hành các hệ thống thông minh như hệ thống trong đề tài này. Dưới đây là mô hình triển khai đề xuất:

#### *3.1.6.1. Máy chủ ứng dụng – Amazon EC2*

Dịch vụ EC2 (Elastic Compute Cloud) cung cấp các máy chủ ảo có thể cấu hình linh hoạt tài nguyên (CPU, RAM, ổ đĩa...). Hệ thống sẽ được triển khai trên một hoặc nhiều EC2 instance sử dụng hệ điều hành Ubuntu/Linux.

Các bước triển khai trên EC2:

- Khởi tạo EC2 instance, mở cổng truy cập HTTP (80) và HTTPS (443), SSH (22).
- Cài đặt Docker, Docker Compose, Git, Python và các dependency cần thiết.
- Tải source code từ GitHub về và sử dụng Docker Compose để chạy toàn bộ hệ thống (backend Django, frontend Next.js, PostgreSQL, ChromaDB...).
- Thiết lập reverse proxy (Nginx hoặc AWS ALB) để trỏ tên miền đến EC2.

*Ưu điểm:* không phụ thuộc máy cục bộ, có thể tự động khởi động lại khi gặp lỗi, dễ dàng scale lên theo số lượng người dùng.

#### *3.1.6.2. Lưu trữ dữ liệu – Amazon S3*

Dữ liệu tĩnh như hình ảnh sản phẩm, tài liệu đã xử lý vector hóa hoặc kết quả gợi ý có thể được lưu trữ trên Amazon S3 (Simple Storage Service). Đây là dịch vụ lưu trữ bền vững, khả năng truy xuất nhanh và hỗ trợ CDN (CloudFront) để phân phối nội dung toàn cầu.

Ứng dụng thực tế trong hệ thống:

- Người quản trị có thể upload hình ảnh sản phẩm lên S3.
- Các file vector hóa có thể được lưu dưới dạng JSON hoặc binary để truy cập nhanh khi truy vấn.

#### *3.1.6.3. Cơ sở dữ liệu – Amazon RDS*

Để thay thế PostgreSQL cài thủ công bằng Docker, có thể sử dụng Amazon RDS (Relational Database Service) để triển khai PostgreSQL trên nền tảng quản lý hoàn toàn (managed service). RDS hỗ trợ:

- Tự động backup, cập nhật phần mềm.
- Thiết lập HA (High Availability) và Multi-AZ (đa vùng) cho độ tin cậy cao.
- Cấu hình parameter group phù hợp với ứng dụng AI có nhu cầu truy vấn lớn.

*Lợi ích:* bảo trì đơn giản, hiệu suất cao, dễ mở rộng mà không cần thao tác thủ công.

#### *3.1.6.4. Bảo mật và giám sát hệ thống*

Để đảm bảo an toàn và độ tin cậy của hệ thống khi triển khai trên AWS, có thể sử dụng kết hợp các dịch vụ sau:

- **IAM (Identity and Access Management):** phân quyền truy cập, đảm bảo chỉ có người dùng/ứng dụng được phép mới có thể truy cập tài nguyên nhất định.

- **Security Groups & VPC:** cấu hình tường lửa cho EC2 và RDS để giới hạn IP truy cập.
- **AWS CloudWatch:** theo dõi tài nguyên hệ thống (CPU, RAM, I/O), thu thập log ứng dụng và cảnh báo khi xảy ra lỗi.
- **ACM (AWS Certificate Manager):** cấp phát và quản lý chứng chỉ SSL cho hệ thống nếu cần triển khai giao thức HTTPS.

#### *3.1.6.5. Mở rộng – Auto Scaling & Load Balancer*

Trong các phiên bản nâng cấp sau, hệ thống có thể tích hợp với Elastic Load Balancer (ELB) để phân phối tải và Auto Scaling Group để tự động tạo thêm EC2 khi có nhiều người dùng truy cập.

*Ví dụ:* khi có 1.000 người dùng truy cập cùng lúc vào hệ thống chatbot, AWS có thể tự động scale từ 1 máy chủ EC2 lên 3 hoặc 5 máy tùy theo cấu hình, đảm bảo hiệu năng phản hồi.

*Bảng 3.2: Lợi ích tổng thể khi triển khai hệ thống trên AWS*

| Tiêu chí | Mô tả lợi ích |
|---|---|
| Hiệu suất | Tối ưu hóa bằng phần cứng thực, hỗ trợ nhiều loại instance |
| Tính mở rộng | Dễ scale EC2, tăng dung lượng RDS, mở rộng vùng phân phối S3 |
| Bảo mật | Hệ sinh thái bảo mật toàn diện: IAM, SSL, mã hóa dữ liệu |
| Tính sẵn sàng cao | Hỗ trợ Auto Healing, Multi-AZ, sao lưu định kỳ |
| Tích hợp AI | Dễ dàng kết nối với OpenAI, Gemini, hoặc AWS Bedrock |

***Kết luận:*** Việc triển khai hệ thống trên AWS không chỉ nâng cao khả năng vận hành, mà còn mở ra cơ hội mở rộng quy mô hệ thống trong tương lai, đáp ứng hàng nghìn người dùng đồng thời. Với khả năng tích hợp linh hoạt, AWS là nền tảng phù hợp để biến đồ án thành một hệ thống thực tế, ứng dụng được trong doanh nghiệp hoặc thương mại điện tử hiện đại.

### *3.1.7. Genimi API*

Gemini API là giao diện lập trình ứng dụng do Google cung cấp, cho phép tích hợp và sử dụng sức mạnh của mô hình ngôn ngữ lớn Gemini – cụ thể là phiên bản models/gemini-2.0-flash-exp. Đây là một mô hình thuộc thế hệ multimodal LLM (Large Language Model), có khả năng xử lý nhiều loại dữ liệu như văn bản, hình ảnh, mã nguồn,...

Gemini API được tối ưu cho tốc độ phản hồi nhanh và chi phí thấp, phù hợp với các ứng dụng thực tế cần phản hồi theo thời gian thực như chatbot, tìm kiếm ngữ nghĩa, tóm tắt văn bản,...

Tính năng chính của Genimi API:

- **Xử lý ngôn ngữ tự nhiên (NLP):** hiểu và phân tích câu hỏi người dùng bằng ngôn ngữ tự nhiên.
- **Truy vấn thông minh (semantic search):** tìm nội dung liên quan theo ngữ nghĩa, không cần khớp từ khóa chính xác.
- **Sinh phản hồi tự động:** tạo văn bản mạch lạc, sát ngữ cảnh trong hội thoại người dùng.
- **Hỗ trợ đa ngôn ngữ:** bao gồm tiếng Việt, giúp chatbot giao tiếp tự nhiên với người Việt.

Tích hợp vào backend:
Trong hệ thống, Gemini API được tích hợp trực tiếp vào backend sử dụng Django thông qua mô-đun riêng (ai_powered), quy trình hoạt động như sau:

1. Tiếp nhận truy vấn từ người dùng (qua chatbot hoặc giao diện tìm kiếm).
2. Gửi yêu cầu (prompt) đến endpoint của Gemini API bằng HTTP request (thường là POST).
3. Xử lý phản hồi: backend nhận kết quả trả về (dạng văn bản JSON), trích xuất nội dung phù hợp.
4. Trả kết quả cho frontend để hiển thị câu trả lời, gợi ý hoặc nội dung tương ứng.

Use-case trong hệ thống:
Gemini API được sử dụng cho nhiều chức năng thông minh trong dự án, cụ thể:

- **Tìm kiếm sản phẩm thông minh:** người dùng mô tả nhu cầu bằng ngôn ngữ tự nhiên (ví dụ: "áo sơ mi trắng công sở"), hệ thống phân tích và gợi ý sản phẩm phù hợp.
- **Chatbot hỗ trợ khách hàng:** giao tiếp trực tiếp với người dùng để trả lời các câu hỏi như chính sách đổi trả, thông tin sản phẩm, tình trạng đơn hàng,...
- **Gợi ý sản phẩm cá nhân hóa:** dựa trên truy vấn, hành vi hoặc lịch sử tương tác của người dùng.

Lợi ích khi sử dụng Genimi API:

- Phản hồi nhanh và mượt mà, phù hợp với trải nghiệm thời gian thực.
- Hiểu sâu ngữ nghĩa, tăng độ chính xác khi tư vấn và tìm kiếm.
- Tiết kiệm thời gian phát triển, không cần huấn luyện mô hình riêng.
- Dễ mở rộng: có thể nâng cấp sang các phiên bản Gemini cao hơn (Pro, Ultra) nếu cần tăng hiệu suất hoặc độ chính xác.

## 3.2. Môi trường triển khai

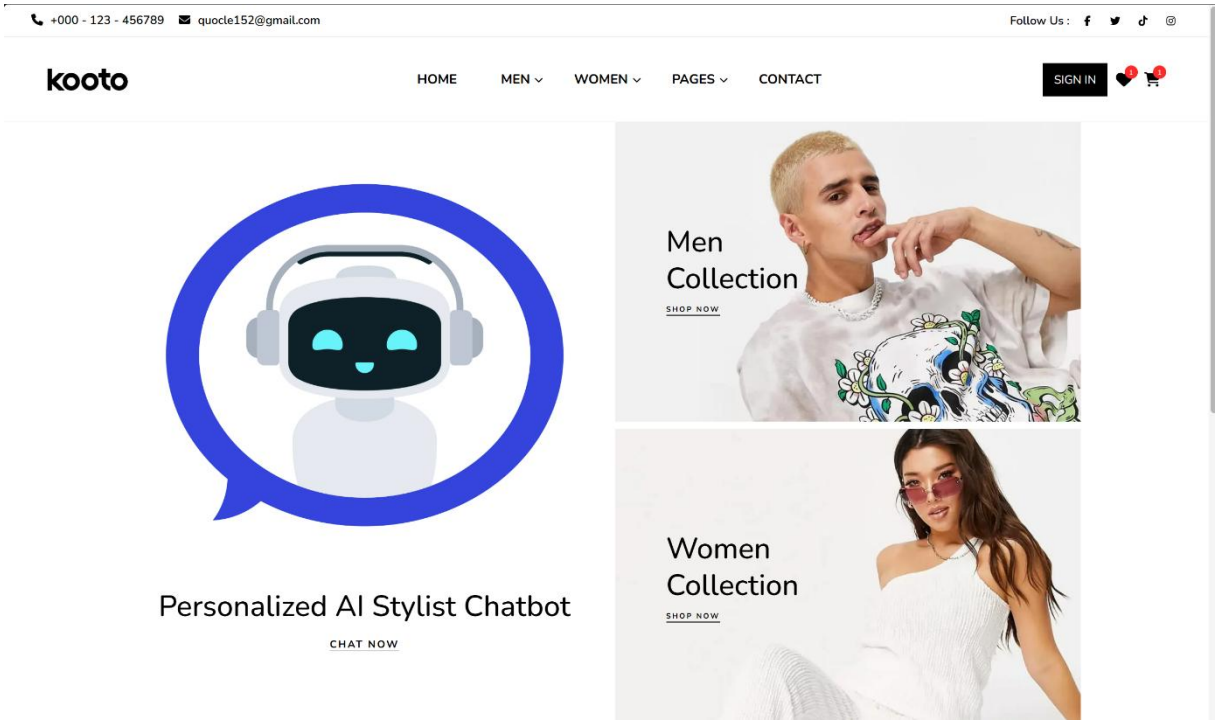
Hệ thống được triển khai trên những môi trường dưới đây

- Next.js: version 14.2.29
- Django: version 4.2.20
- Python: version 3.12.3

- PostgreSQL: version 3.10.9
- Docker: version 28.0.4
- AWS

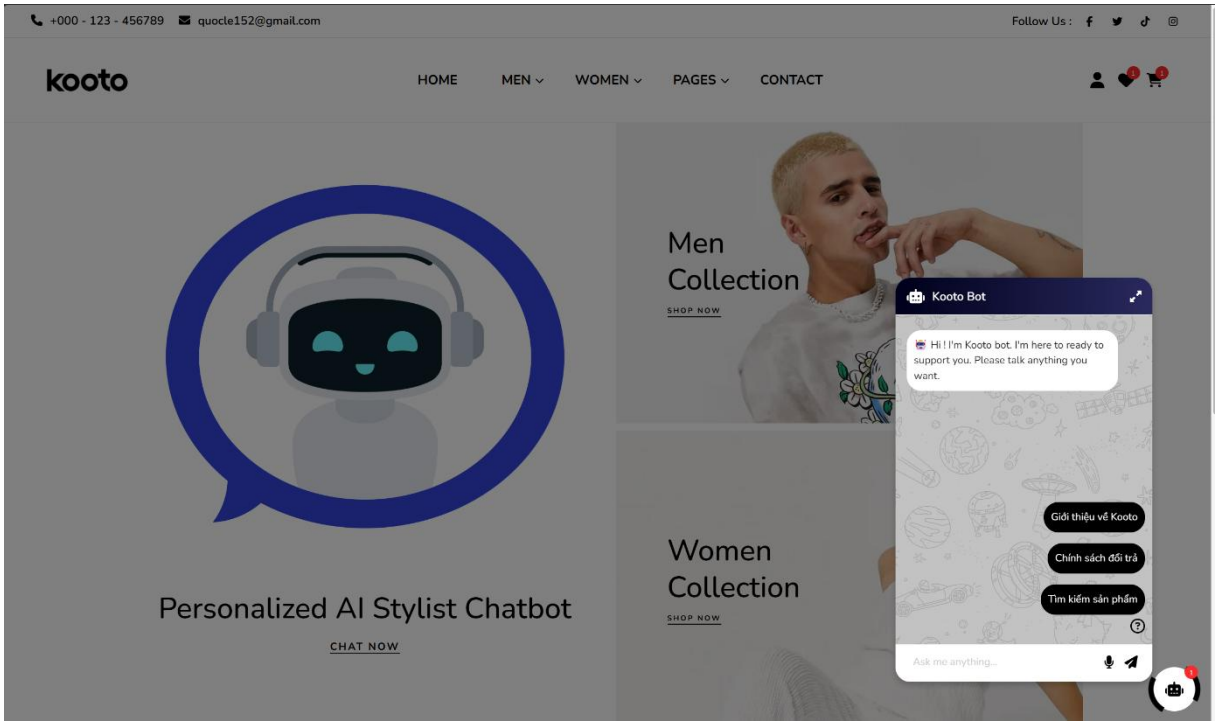
## 3.3. Kết quả đạt được

- Trang chủ:



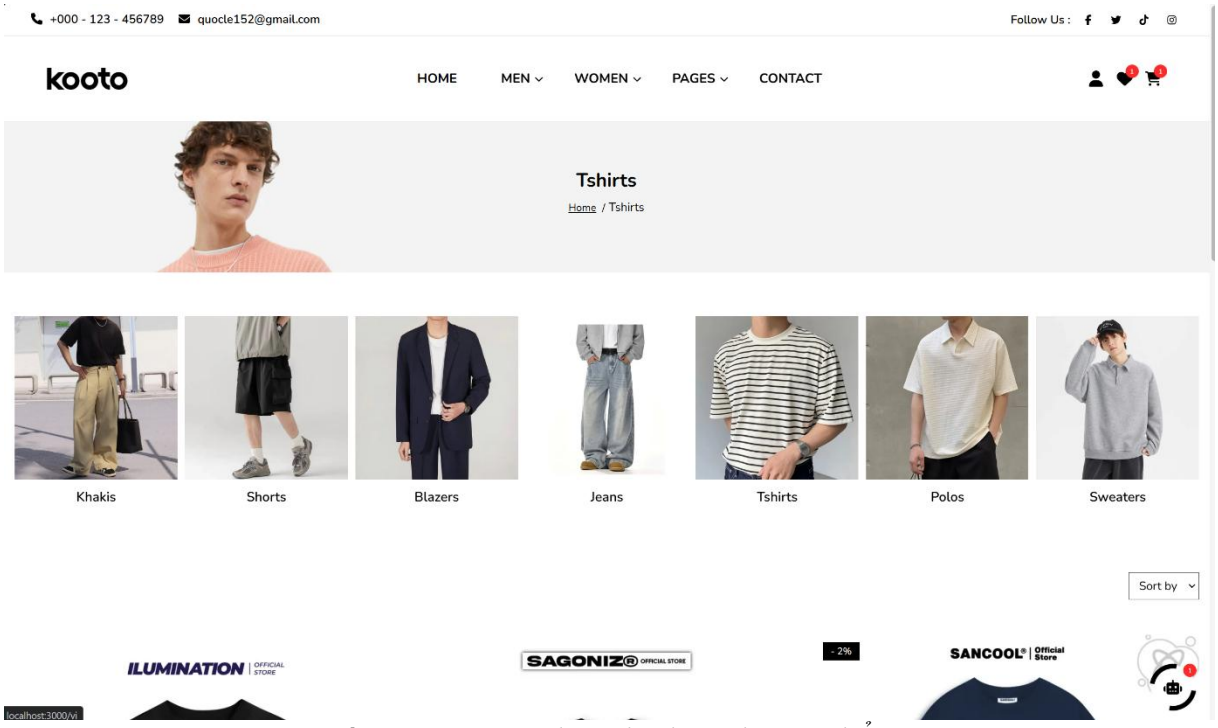

*Hình 3.1: Giao diện trang chủ*

- Chatbot



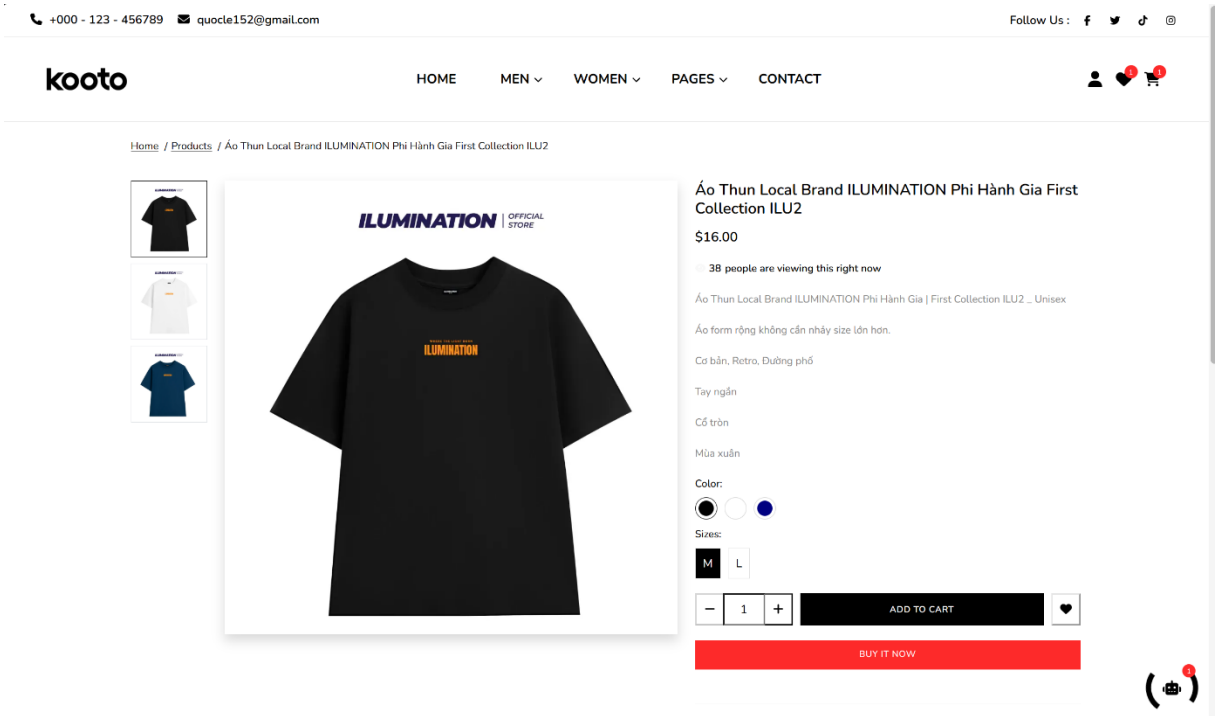

*Hình 3.2: Giao diện chatbot*

- Danh sách sản phẩm



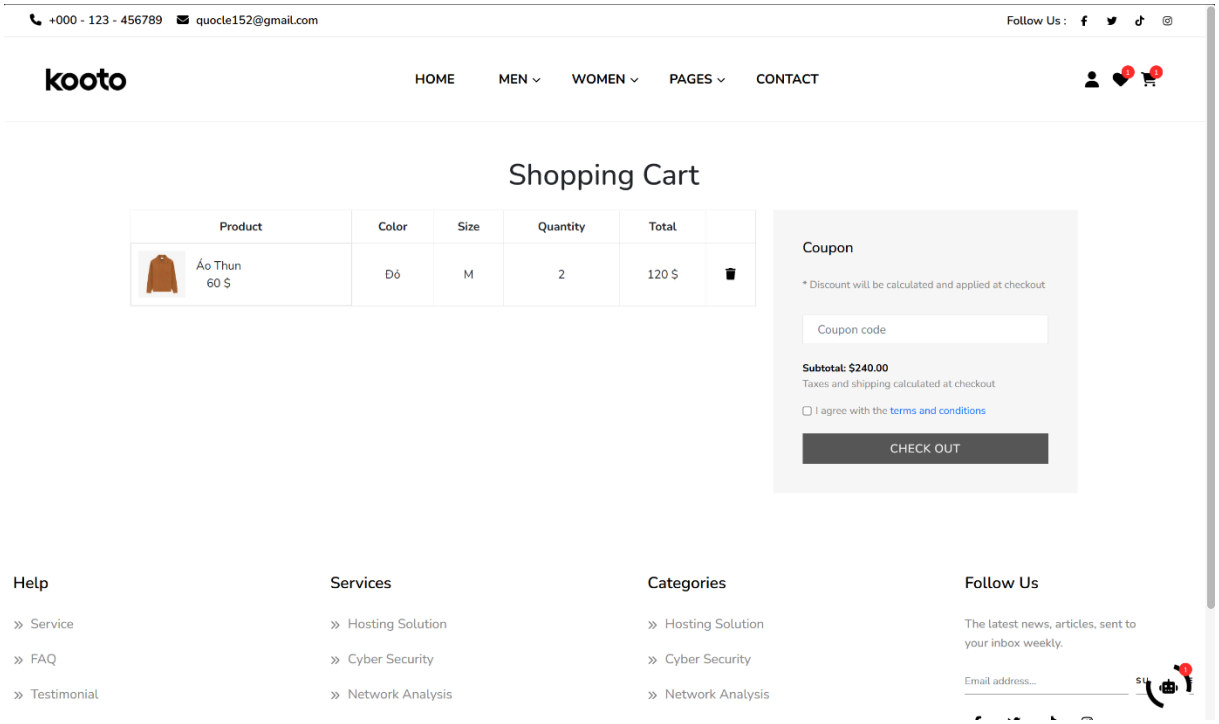

*Hình 3.3: Giao diện danh sách sản phẩm*

- Chi tiết sản phẩm




*Hình 3.4: Giao diện chi tiết sản phẩm*

- Giỏ hàng




*Hình 3.5: Giao diện giỏ hàng*

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết quả đạt được

Sau quá trình nghiên cứu và phát triển, đề tài "Ứng dụng AI trong mua sắm thời trang thông minh" đã hoàn thành các mục tiêu đặt ra với các kết quả cụ thể như sau:

- Xây dựng hệ thống mua sắm trực tuyến có giao diện thân thiện, dễ sử dụng trên cả máy tính và điện thoại.
- Tích hợp chatbot AI sử dụng Gemini API, có khả năng hiểu và phản hồi ngôn ngữ tự nhiên của người dùng, hỗ trợ tư vấn sản phẩm và trả lời các câu hỏi liên quan.
- Áp dụng NLP và vector hóa tài liệu, kết hợp cùng ChromaDB để triển khai chức năng tìm kiếm thông minh theo ngữ nghĩa.
- Quản lý dữ liệu hiệu quả với PostgreSQL, hỗ trợ đầy đủ các chức năng như lưu trữ thông tin sản phẩm, người dùng, đơn hàng,...
- Triển khai hệ thống bằng Docker Compose, cho phép dễ dàng khởi động các thành phần backend, frontend, database, và AI module.
- Hoàn thiện mô hình client-server tách biệt, sử dụng Next.js cho frontend và Django cho backend.

Toàn bộ hệ thống hoạt động ổn định, đáp ứng các tiêu chí về hiệu suất, trải nghiệm người dùng, bảo mật và khả năng mở rộng.

## Hướng phát triển

Trong tương lai, hệ thống có thể được mở rộng và cải tiến theo các định hướng sau:

- Tích hợp thêm AI Agent nâng cao, có khả năng tư duy nhiều bước, ghi nhớ phiên làm việc và tư vấn chuyên sâu hơn.
- Hỗ trợ đa ngôn ngữ hoàn chỉnh, cho phép hệ thống hoạt động hiệu quả với người dùng quốc tế.
- Phân tích hành vi người dùng bằng AI để gợi ý sản phẩm cá nhân hóa theo lịch sử tìm kiếm, tương tác và đơn hàng.
- Tối ưu hiệu suất truy vấn vector bằng cách sử dụng các chỉ mục hiệu quả hơn (FAISS, ScaNN) kết hợp với ChromaDB.
- Tích hợp hệ thống thanh toán thực tế (VNPay, Momo, Stripe...) để hoàn thiện trải nghiệm mua hàng.
- Phát triển dashboard quản trị nâng cao, hỗ trợ thống kê, phân tích doanh thu và hiệu quả tư vấn AI theo thời gian thực.

# TÀI LIỆU THAM KHẢO

[1] Next.js Docs,

URL: https://nextjs.org/docs

[2] Django,

URL : https://www.djangoproject.com/

[3] PostgreSQL,

URL: https://www.postgresql.org/

[4] ChromaDB,

URL: https://www.trychroma.com/

[5] Docker ,

URL: https://www.docker.com/

[6] Genimi API ,

URL: https://ai.google.dev/gemini-api/docs/models?hl=vi#model-versions

[7] What is a Vector database & How does it work,

URL: https://www.pinecone.io/learn/vector-database/

[8] What is an AI Prompt?,

URL: https://www.geeksforgeeks.org/artificial-intelligence/what-is-an-ai-prompt/

[9] Amazon Web Services,

URL: https://docs.aws.amazon.com/?nc2=h_ql_doc_do