**Đề tài 1.** Hỏi đáp tư vấn sản phẩm cho các trang thương mại điện tử

Yêu cầu:

1. Chọn một hoặc một vài loại mặt hàng (điện thoại, máy tính, máy tính bảng, quạt, điều hòa, máy hút ẩm, máy rửa bát,…)

2. Thu thập dữ liệu ở nhiều website khác nhau

a. Dữ liệu có cấu trúc

b. Review, đánh giá của khách hàng

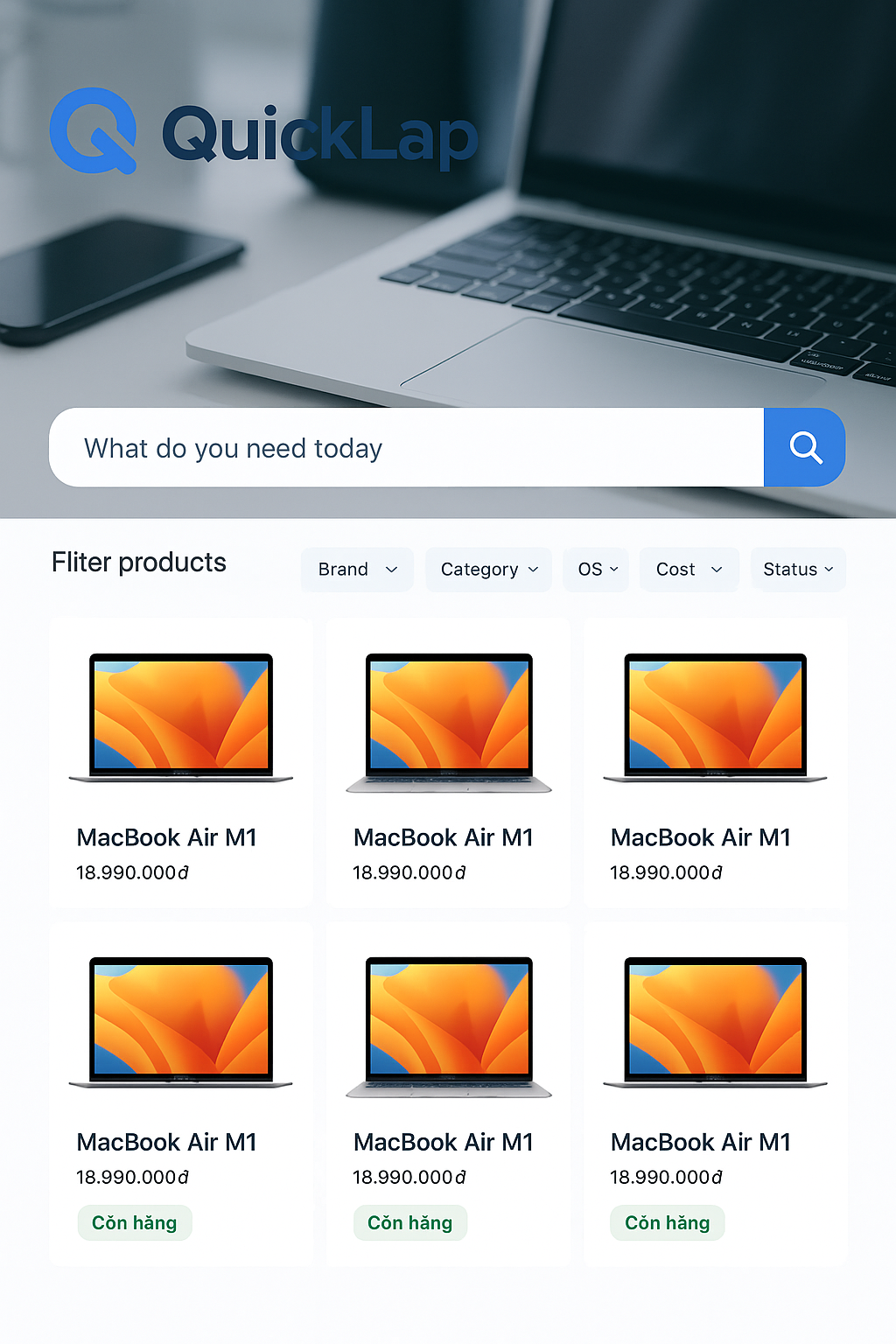
c. Dữ liệu văn bản mô tả sản phẩm

Sử dụng nhiều giải pháp khác nhau để hỗ trợ người dùng tìm kiếm được sản phẩm mong muốn trong thời gian ngắn nhất: VD: (1): LLM với mô hình RAG (Retrieval-Augmented Generation) – cần thử nghiệm với nhiều mô hình LLM khác nhau, (2) Tìm kiếm xử lý thông tin đơn thuần, (3)…

*LM với RAG: Tích hợp mô hình ngôn ngữ lớn (LLM) để trả lời câu hỏi người dùng, sử dụng RAG để tìm kiếm và cung cấp thông tin chính xác từ dữ liệu đã thu thập.*

**Đề tài: QuickLap - Tư vấn Laptop dựa trên mô tả người dùng.**

**Giao diện + Tổng quát**



Dự kiến: 300 máy

What do you need today: Người dùng nhập một đoạn mô tả nhu cầu, hệ thống đọc và lọc sản phẩm phù hợp

Lọc sản phẩm theo tiêu chí:

+ Brand: Dell, MacBook, Lenovo,…

|  |  |
| --- | --- |
| + Category: | GamingLaptop: Laptop chơi game (có GPU mạnh, tần số làm tươi cao).  Ultrabook: Laptop mỏng nhẹ (trọng lượng thấp, thời lượng pin dài).  WorkstationLaptop: Laptop cho công việc chuyên sâu (CPU mạnh, RAM lớn). |

+ OS: Window, macOS, Linux

|  |  |
| --- | --- |
| + Cost: (tăng dần hoặc giảm dần) | 5 – 10tr  10 – 15tr  15 – 20tr  25 – 35tr  35 – 50tr  > 50tr |

+ Status: Mới/Cũ

**Công cụ và thư viện**

Java: Ngôn ngữ lập trình chính.

Jsoup: Thu thập dữ liệu web.

MySQL/MongoDB: Lưu trữ dữ liệu.

Apache Lucene/Elasticsearch: Xây dựng chỉ mục tìm kiếm.

Stanford NLP: Phân tích cảm xúc đánh giá.

Hugging Face API: Tích hợp LLM.

JavaFX/Swing: Xây dựng giao diện người dùng.

**Các lớp**

**public class Laptop**

**- private String id** # Mã sản phẩm TIKI\_ASUS\_123456

**- private String name** # Tên sản phẩm

**- private String brand** # Thương hiệu (Dell, HP, Lenovo,...)

**- private String category**

**- private String OS**

**- private int price**

**- private String description** # Mô tả sản phẩm

**- private List<Review> reviews** # Danh sách đánh giá

**- private String cpu**

**- private String gpu**

**- private int screenRate** # Tần số quét màn hình

**- private int ram**

**- private int storage** # Bộ nhớ

**- private String diskType** # SSD, HDD, …

**- private String resolution** # Độ phân giải màn hình (HD, FHD,…)

**- private float screenSize**  # Kích thước màn hình

**- private float weight** # Cân nặng

**- private float batteryLife** # Thời lượng pin

**- private String ports** # cổng kết nối, ví dụ: USB, HDMI

# Constructor

# Getter + Setter

**+** public void addReview(Review review) {}

# Phương thức tính điểm đánh giá trung bình

# Loại bỏ đánh giá không hợp lệ (score <= 0).

# Đảm bảo không chia cho 0.

# Tính trung bình có trọng số dựa trên nguồn (ví dụ: Tiki có trọng số cao hơn).

# Chỉ tính trong 6 tháng gần đây

+ public double getAverageRating() {}

# Phương thức lấy thông số kỹ thuật

public Map<String, String> getSpecifications() {}

# Phương thức tạo bản tóm tắt đánh giá của một laptop, bao gồm: điểm trung bình tổng quát, một vài bình luận mẫu (tối đa 5, từ Review.getComment() )

# Chức năng

# Hỗ trợ người dùng: Cung cấp tóm tắt ngắn gọn, dễ đọc trong giao diện JavaFX.

# Hỗ trợ RAG: Tạo ngữ cảnh phong phú (điểm số + bình luận) để AI trả lời câu hỏi như "Laptop ASUS có nhận xét tích cực không?".

# Tiết kiệm thời gian: Tránh phân tích toàn bộ reviews mỗi lần cần thông tin.

public String getReviewSummary()

# Phương thức toString để hiển thị thông tin cơ bản

public String toString()

**public class Review**

- private String userId

- private double rating

- private String comment

- private LocalDate date

+ public Review(String userId, double rating, String comment, LocalDate date)

+ getSentiment() (**Stanford NLP**)

+ public boolean isRecent()

+ public String toString

**public class DataCollector**

+ public List<Laptop> collectStructuredData(String url)

+ public List<Review> collectReviews(String url)

**public abstract class SearchStrategy**

# protected List<Laptop> laptops

+ public SearchStrategy(List<Laptop> laptops)

+ public abstract List<Laptop> search(String query);

# protected List<Laptop> filterByMaxPrice(List<Laptop> results, double maxPrice)

**public class KeywordSearchStrategy extends SearchStrategy**

+ public KeywordSearchStrategy(List<Laptop> laptops)

@Override

+ public List<Laptop> search(String query)

public class FilterSearchStrategy extends SearchStrategy

- private Map<String, String> filters;

+ public FilterSearchStrategy(List<Laptop> laptops, Map<String, String> filters)

@Override

+ public List<Laptop> search(String query)

**public class RAGIntegration**

Tích hợp với các mô hình LLM (ví dụ: thông qua API của Hugging Face, OpenAI, hoặc Grok)

Phương thức: retrieveAndGenerate() (truy xuất dữ liệu liên quan và tạo câu trả lời)

Triển khai RAG

1. Truy xuất thông tin

Sử dụng **Apache Lucene** hoặc **Elasticsearch** để xây dựng chỉ mục tìm kiếm cho dữ liệu sản phẩm, đánh giá, và mô tả.

Khi người dùng đặt câu hỏi (ví dụ: "Điện thoại nào dưới 10 triệu có camera tốt?"), truy xuất các sản phẩm liên quan dựa trên từ khóa và ngữ cảnh.

2. Tích hợp LLM

Sử dụng API của các mô hình LLM như **Grok (xAI)**, **BERT**, hoặc **LLaMA** (thông qua Hugging Face).

Gửi dữ liệu truy xuất được (context) cùng với câu hỏi của người dùng tới LLM để tạo câu trả lời tự nhiên.

3. Thử nghiệm nhiều mô hình

**Grok**: Phù hợp với câu hỏi yêu cầu phân tích sâu, tích hợp tốt với hệ sinh thái xAI.

**BERT**: Tốt cho phân tích ngữ nghĩa và tìm kiếm ngữ cảnh.

**LLaMA**: Hiệu quả với các tác vụ tạo văn bản dài.

- private final String llmApiUrl = "https://api-inference.huggingface.co/models/bert-base-uncased";

- private final String apiKey = "your\_huggingface\_api\_key";

+ public String retrieveAndGenerate(String query, List<String> context)

+ private static class RequestPayload

**UserInterface**

- private SearchEngine searchEngine;

+ public void start(Stage primaryStage) {

- private List<Product> loadProducts() {

- public static void main(String[] args) {