**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**ĐỒ ÁN TỐT NGHIỆP**

# Xây dựng hệ thống tìm kiếm, khai thác thông tin từ dữ liệu Facebook

**Hồ Mạnh Thắng**

[thang.hm183627@sis.hust.edu.vn](mailto:thang.hm183627@sis.hust.edu.vn)

**Ngành Công nghệ thông tin và truyền thông Chuyên ngành Khoa học máy tính**

**Giảng viên hướng dẫn:** TS. Ban Hà Bằng

Chữ kí GVHD

**Khoa:** Khoa học máy tính

**Trường:** Công nghệ thông tin và Truyền thông

**HÀ NỘI, 01/2023**

# LỜI CẢM ƠN

Lời cảm ơn của sinh viên (SV) tới người yêu, gia đình, bạn bè, thầy cô, và chính bản thân mình vì đã chăm chỉ và quyết tâm thực hiện ĐATN để đạt kết quả tốt nhất, nên viết phần cảm ơn ngắn gọn, tránh dùng các từ sáo rỗng, giới hạn trong khoảng 100-150 từ.

# TÓM TẮT NỘI DUNG ĐỒ ÁN

Đồ án tốt nghiệp của em mang tên "Xây dựng hệ thống tìm kiếm, khai thác thông tin từ dữ liệu Facebook". Trong thời dữ liệu hiện nay, dữ liệu về người dùng đặc biệt là từ các nền tảng mạng xã hội đang được coi là mỏ vàng mới của thế giới, và Facebook là mỏ vàng lớn nhất trong số đó. Từ dữ liệu của một số lượng rất lớn các thực thể tương tác trên Facebook (thông qua GraphAPI) người ta có thể trích xuất ra các thông tin ẩn hữu ích về đa lĩnh vực, đa chủ đề, sử dụng chúng để đưa ra các kết quả tìm kiếm như với Google, các dự đoán xu hướng tương lai và nhiều ứng dụng khác. Đơn giản như khi ta muốn tìm kiếm thông tin về một người trên Facebook hay muốn xem các chủ đề đang là xu hướng trong lĩnh vực chứng khoán. Tuy nhiên các thông tin từ kết quả trả về đều bị giới hạn phạm vi trong những đối tượng ta tương tác, giới hạn về số lượng hoặc kết quả không sát với truy vấn mong muốn, chúng xuất phát từ việc sử dụng, tương tác với dữ liệu từ mạng xã hội này những năm gần đây đang gặp cản trở lớn từ việc Facebook dừng việc cung cấp quyền truy cập tùy ý vào GraphAPI-bộ khung của các dòng dữ liệu và các tương tác của nó, thêm nữa là chính sách giới hạn khả năng tìm kiếm thông tin vì lí do bảo vệ thông tin người dùng, hạn chế về các tính năng insight data trên nền tảng Facebook hiện tại. Các nhà phát triển, lập trình viên làm việc với dữ liệu realtime từ Facebook cũng đang có những cách tiếp cận đa dạng để giải quyết vấn đề này, tiêu biểu như 1 số ít công ty lớn hợp tác trực tiếp với Meta (công ty mẹ của Facebook ) để có khả năng truy cập trực tiếp vào GraphAPI đầy đủ chức năng, hoặc phổ biến hơn là cách gián tiếp-sử dụng các API tự tạo ra để request đến máy chủ Facebook nhằm tìm kiếm và lấy thông tin trả về bằng các phương pháp crawl dữ liệu, phương pháp này yêu cầu phải đăng ký trở thành nhà phát triển thử nghiệm của Meta. Các hướng đi như trên đảm bảo tính chính thống trong việc sử dụng dữ liệu, tính chuẩn xác của dữ liệu. Tuy nhiên chúng có những khuyết điểm khiến chúng không trở nên tối ưu cho đa số các nhà phát triển như: Chi phí cao (để có thể sử dụng đầy đủ GraphAPI của Meta), số lượng và số loại dữ liệu được sử dụng bị giới hạn, việc đăng ký không dễ dàng, thông tin không đa dạng và khó tùy biến các cài đặt theo mục đích của người sử dụng (bị giới hạn theo tài khoản đang sử dụng). Xuất phát từ vấn đề và thách thức như vậy, em đã chọn hướng tiếp cận khác, về khái quát như sau: Thu thập dữ liệu từ các nguồn có sẵn trên Facebook (fanpage, group,…) sau đó qua các bước trích xuất dữ liệu - lưu trữ - xử lí – làm sạch – làm giàu – lưu trữ vào hệ CSDL NoSQL – trả về dữ liệu qua API. Phương pháp của em có thể tùy ý lấy được dữ liệu mà không bị các ràng buộc giới hạn như các phương pháp trước, lượng dữ liệu có thể thu thập tùy ý dựa vào độ lớn và khả năng của server chạy crawl bots, dữ liệu có thể được truy xuất nhanh chóng và sử dụng lại vì được lưu trong các kho dữ liệu đa dạng từ SQL đến NoSQL, ngoài ra giá trị của dữ liệu cũng được tăng đáng kể vì đã đi qua các bước làm giàu dữ liệu sử dụng các thuật toán học máy, các ứng dụng của NLP. Việc tìm kiếm và dự đoán thông tin từ dữ liệu được tổ chức theo chức năng riêng biệt, tùy biến không giới hạn theo cấu hình của người dùng, và chi phí vận hành theo thực tế đã và đang triển khai là hợp lý với cả quy mô cá nhân sử dụng. Với các khả năng trên phương pháp và phát triển này có thể khắc phục các nhược điểm cố hữu của những phương pháp trước, ngoài ra đây có thể mở ra hướng nghiên cứu mới về dữ liệu và việc sử dụng dữ liệu từ các mạng xã hội, vốn là địa hạt có tiềm năng rất lớn mà các doanh nghiệp và tổ chức còn chưa để ý và khai thác có hiệu quả. Kết quả thu được từ phát triển có thể đóng góp một kênh tham khảo không chỉ về lĩnh vực tìm kiếm thông tin mà còn có thể dùng trong công tác điều tra, dân số, dự đoán, phát hiện các thành phần xã hội. Nếu như trong tương lai phát triển này có thể mở rộng từ nền tảng có sẵn hiện tại sẽ đem lại lợi ích trước hết cho doanh nghiệp và người sử dụng, xa hơn là thúc đẩy tốc độ phát triển và ứng dụng dữ liệu lớn.

Sinh viên viết tóm tắt ĐATN của mình trong mục này, với 200 đến 350 từ. Theo trình tự, các nội dung tóm tắt cần có: (i) Giới thiệu vấn đề (tại sao có vấn đề đó, hiện tại được giải quyết chưa, có những hướng tiếp cận nào, các hướng này giải quyết như thế nào, hạn chế là gì), (ii) Hướng tiếp cận sinh viên lựa chọn là gì, vì sao chọn hướng đó, (iii) Tổng quan giải pháp của sinh viên theo hướng tiếp cận đã chọn, và (iv) Đóng góp chính của ĐATN là gì, kết quả đạt được sau cùng là gì. Sinh viên cần viết thành đoạn văn, không được viết ý hoặc gạch đầu dòng.

# ABSTRACT

Mục này khuyến khích sinh viên viết lại mục “Tóm tắt” đồ án tốt nghiệp ở trang trước bằng tiếng Anh. Phần này phải có đầy đủ các nội dung như trong phần tóm tắt bằng tiếng Việt. Sinh viên không nhất thiết phải trình bày mục này.

Nhưng nếu lựa chọn trình bày, sinh viên cần đảm bảo câu từ và ngữ pháp chuẩn tắc, nếu không sẽ có tác dụng ngược, gây phản cảm.

## MỤC LỤC

**CHƯƠNG 1.** [**GIỚI THIỆU ĐỀ TÀI**](#_heading=h.30j0zll) **1**

* 1. [Đặt vấn đề](#_heading=h.3znysh7) 1
  2. [Mục tiêu và phạm vi đề tài](#_heading=h.tyjcwt) 2
  3. [Định hướng giải pháp](#_heading=h.1t3h5sf) 2
  4. [Bố cục đồ án](#_heading=h.2s8eyo1) 2

**CHƯƠNG 2.** [**KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU**](#_heading=h.3rdcrjn) **4**

* 1. [Khảo sát hiện trạng](#_heading=h.lnxbz9) 4
  2. [Tổng quan chức năng](#_heading=h.1ksv4uv) 4
     1. [Biểu đồ use case tổng quát](#_heading=h.2jxsxqh) 4
     2. [Biểu đồ use case phân rã XYZ](#_heading=h.3j2qqm3) 4
     3. [Quy trình nghiệp vụ](#_heading=h.4i7ojhp) 4
  3. [Đặc tả chức năng](#_heading=h.1ci93xb) 5
     1. [Đặc tả use case A](#_heading=h.2bn6wsx) 5
     2. [Đặc tả use case B](#_heading=h.3as4poj) 5
  4. [Yêu cầu phi chức năng](#_heading=h.49x2ik5) 5

**CHƯƠNG 3.** [**CÔNG NGHỆ SỬ DỤNG**](#_heading=h.147n2zr) **6**

**CHƯƠNG 4.** [**THỰC NGHIỆM VÀ ĐÁNG GIÁ**](#_heading=h.23ckvvd) **7**

* 1. [Thiết kế kiến trúc](#_heading=h.1hmsyys) 7
     1. [Lựa chọn kiến trúc phần mềm](#_heading=h.1hmsyys) 7
     2. [Thiết kế tổng quan.](#_heading=h.2grqrue) 7
     3. [Thiết kế chi tiết gói](#_heading=h.1v1yuxt) 7
  2. [Thiết kế chi tiết](#_heading=h.3tbugp1) 8
     1. [Thiết kế giao diện](#_heading=h.3tbugp1) 8
     2. [Thiết kế lớp](#_heading=h.nmf14n) 8
     3. [Thiết kế cơ sở dữ liệu](#_heading=h.1mrcu09) 9
  3. [Xây dựng ứng dụng](#_heading=h.2lwamvv) 9
     1. [Thư viện và công cụ sử dụng](#_heading=h.3l18frh) 9
     2. [Kết quả đạt được](#_heading=h.2zbgiuw) 9
     3. [Minh họa các chức năng chính](#_heading=h.3ygebqi) 9
  4. [Kiểm thử](#_heading=h.sqyw64) 9
  5. [Triển khai](#_heading=h.1rvwp1q) 10

**CHƯƠNG 5.** [**CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT**](#_heading=h.2r0uhxc) **11**

**CHƯƠNG 6.** [**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**](#_heading=h.25b2l0r) **12**

* 1. [Kết luận](#_heading=h.34g0dwd) 12
  2. [Hướng phát triển](#_heading=h.43ky6rz) 12

[**TÀI LIỆU THAM KHẢO**](#_heading=h.1baon6m) **15**

[**PHỤ LỤC**](#_heading=h.pkwqa1) **17**

1. [**HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP**](#_heading=h.pkwqa1) **17**
   1. [Ngành học](#_heading=h.1opuj5n) 18
   2. [Đánh dấu (bullet) và đánh số (numering)](#_heading=h.2nusc19) 18
   3. [Cách thêm bảng](#_heading=h.haapch) 19
   4. [Chèn hình ảnh](#_heading=h.1gf8i83) 19
   5. [Tài liệu tham khảo](#_heading=h.2fk6b3p) 19
   6. [Cách viết phương trình và công thức toán học](#_heading=h.3ep43zb) 20
   7. [Qui cách đóng quyển](#_heading=h.2szc72q) 20
2. [**ĐẶC TẢ USE CASE**](#_heading=h.meukdy) **22**
   1. [Đặc tả use case “Thống kê tình hình mượn sách”](#_heading=h.1ljsd9k) 22
   2. [Đặc tả use case “Đăng ký làm thẻ mượn”](#_heading=h.2koq656) 22

**DANH MỤC HÌNH VẼ**

Hình 3.1.a: Giao diện web của FastAPI Docs………………………………..

Hình 3.1.b: Thuật toán LambdaMART………………………………………

Hình 3.1.c: Công thức của DCG……………………………………………..

Hình 3.1.d: Giao diện Redis Queue Dashboard……………………………...

Hình 4.2.1.a: Giao diện minh họa Postman và input, output chuẩn cho các API…………………………………………………………………………

Hình 4.2.1.b: Minh họa giao diện cửa sổ thực thi của các micro service….

Hình 4.2.1.c: Giao diện console phía người dùng khi gửi request…………..

Hình 4.2.1.d: Giao diện console phía người dùng khi nhận phản hồi………

Hình 4.2.3.1: Mô hình thực thể liên kết của các đối tượng trong hệ thống thực tế……………………………………………………………………….

Hình 4.3.3.a: Giao diện dashboard Ngóng tin MXH Facecbook…………..

Hình 4.3.3.b: Giao diện truy vấn theo từ khóa qua API sử dụng Postman…

Hình 4.3.3.c: Giao diện hiển thị kết quả tìm kiếm theo từ khóa “Hà Nội” và các từ khóa thịnh hành ……………………………………………………

Hình 4.3.3.d: Giao diện kết quả tìm kiếm theo bộ lọc thời gian………….

## DANH MỤC BẢNG BIỂU

[Bảng 4.1 Danh sách thư viện và công cụ sử dụng](#_heading=h.206ipza) 9

[Bảng A.1 Table to test captions and labels.](#_heading=h.1302m92) 19

Bảng 2.3.a: Đặc tả chức năng các use case quan trọng

Bảng 4.3.1.a: Danh sách các công cụ sử dụng

Bảng 4.3.1.2: Danh sách ngôn ngữ lập trình, các thư viện và API sử dụng

Bảng 4.3.2.a: Các service đã đóng gói và triển khai thực tế

Bảng 4.3.2.b: Các thông số về mã nguồn và service

Bảng 4.4.1: Kết quả tổng kết kiểm thử

Bảng 4.5.1: Thông tin thống kê các kết quả thông số trong thực tế triển khai hệ thống

Biểu đồ 2.2.1: Biểu đồ Use Case tổng quát

Biểu đồ 2.2.2.1: Biểu đồ phân rã use case gửi truy vấn dữ liệu đến hệ thống

Biểu đồ 2.2.2.2: Biểu đồ phân rã cho use case thu thập dữ liệu

Biểu đồ 2.2.2.3: Biểu đồ phân rã use case quản trị viên quản lý cấu hình hệ thống

Biểu đồ 2.2.2.4: Biểu đồ use case phân rã cho use case gửi yêu cầu lập lịch

Biểu đồ 4.1.2 : Thiết kế gói tổng quan toàn hệ thống

Biểu đồ 4.1.3.a: Thiết kế package ManageDatabase

Biểu đồ 4.1.3.b: Thiết kế package AutoScheduleCrawler

Biểu đồ 4.1.3.c: Thiết kế package FacebookCrawler

Biểu đồ 4.1.3.d: Thiết kế package PrivateFB

## DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

| API | Giao diện lập trình ứng dụng |
| --- | --- |
| Service | Dịch vụ |
| Cloud server | Hệ thống máy chủ đám mây |
| Beta | Giai đoạn thử nghiệm sản phẩm |
| Live | Giai đoạn triển khai thực tế |

## 

## CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Lưu ý: **Trước khi viết ĐATN, sinh viên cần đọc kỹ hướng dẫn và quy định chi tiết** về cách viết ĐATN trong Phụ lục A. Sinh viên tuân theo mẫu tài liệu này để viết báo cáo đồ án tốt nghiệp, vì tài liệu này đã được căn chỉnh, chỉnh sửa theo đúng chuẩn báo cáo kỹ thuật đồ án tốt nghiệp (ISO 7144:1986). Sinh viên viết trực tiếp vào file này, chỉ chỉnh sửa nội dung, và không viết trên file mới.

**Khi đóng quyển ĐATN**, sinh viên cần lưu ý tuân thủ hướng dẫn ở phụ lục A.9

**SV cần đặc biệt lưu ý cách hành văn**. Mỗi đoạn văn không được quá dài và cần có ý tứ rõ ràng, bao gồm duy nhất một ý chính và các ý phân tích bổ trợ để làm rõ hơn ý chính. Các câu văn trong đoạn phải đầy đủ chủ ngữ vị ngữ, cùng hướng đến chủ đề chung. Câu sau phải liên kết với câu trước, đoạn sau liên kết với đoạn trước. Trong văn phong khoa học, sinh viên không được dùng từ trong văn nói, không dùng các từ phóng đại, thái quá, các từ thiếu khách quan, thiên về cảm xúc, về quan điểm cá nhân như “tuyệt vời”, “cực hay”, “cực kỳ hữu ích”, v.v. Các câu văn cần được tối ưu hóa, đảm bảo rất khó để thể thêm hoặc bớt đi được dù chỉ một từ. Cách diễn đạt cần ngắn gọn, súc tích, không dài dòng.

Mẫu ĐATN này được thiết kế phù hợp nhất với đa số các đề tài xây dựng phần mềm ứng dụng. Với các dạng đề tài khác (giải pháp, nghiên cứu, phần mềm đặc thù, v.v.), sinh viên dựa trên cấu trúc và hướng dẫn của báo cáo này để đề xuất và trao đổi với giáo viên hướng dẫn để thiết kế khung báo cáo đồ án cho phù hợp. Sinh viên lưu ý **trong mọi trường hợp, SV luôn phải sử dụng định dạng báo cáo này, và phải đọc kỹ toàn bộ các hướng dẫn từ đầu tới cuối.** Các hướng dẫn không chỉ áp dụng riêng cho đề tài ứng dụng, mà còn phù hợp với các dạng đề tài khác. Ngoài ra, trong mẫu ĐATN này đã được tích hợp một số hướng dẫn dành riêng cho đề tài nghiên cứu.

Chương 1 có độ dài từ 3 đến 6 trang với các nội dung sau đây

## Đặt vấn đề

Trong thời đại dữ liệu hiện nay, dữ liệu về người dùng đặc biệt là từ các nền tảng mạng xã hội đang được coi là mỏ vàng mới của thế giới, và Facebook là mỏ vàng lớn nhất trong số đó. Với khoảng 2,96 tỷ người dùng hoạt động hàng tháng tính đến quý 3 năm 2022 (1), Facebook là mạng xã hội trực tuyến được sử dụng nhiều nhất trên toàn thế giới. Từ dữ liệu của một số lượng rất lớn các thực thể tương tác trên Facebook (thông qua GraphAPI) người ta có thể trích xuất ra các thông tin hữu ích về đa lĩnh vực, đa chủ đề, sử dụng chúng để đưa ra các kết quả tìm kiếm như với Google, các dự đoán xu hướng tương lai, lập các quan sát thống kê về các mô hình nhóm, cộng đồng trong xã hội và nhiều ứng dụng khác. Đơn cử như trường hợp người dùng muốn tìm kiếm thông tin về một sự kiện thời sự đương thời qua các bài đăng trên Facebook hay muốn xem các chủ đề đang là xu hướng trong lĩnh vực chứng khoán. Các kết quả từ việc nghiên cứu và khai thác dữ liệu Facebook hoàn toàn có thể ứng dụng vào nhiều ngành khoa học và đời sống xã hội như phục vụ công tác lập thống kê điều tra dân số, phân cụm các thành phần văn hóa xã hội, nghiên cứu tâm lý học và tương tác hành vi, nghiên cứu thị trường và marketing, báo chí tin tức hoặc sử dụng như một nguồn số liệu tham khảo cho việc giảng dạy và phân tích. Lượng dữ liệu hàng ngày từ Facebook đáp ứng đủ các tiêu chí 5V (velocity, volume, value, variety, veracity) của dữ liệu lớn, vậy nên tiềm năng về lợi ích khi tận dụng được nguồn dữ liệu này cần được quan tâm xứng đáng. Thách thức về việc khai thác tiềm năng này đặt ra bài toán về việc xây dựng một hệ thống có khả năng trích xuất, biến đổi và chắt lọc dữ liệu thô từ nền tảng Facebook từ đó tạo ra những thông tin có giá trị để trả về cho người dùng.

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## Mục tiêu và phạm vi đề tài

Hiện nay các doanh nghiệp, nhà phát triển, lập trình viên làm việc với dữ liệu từ Facebook đang có những cách tiếp cận đa dạng để tận dụng tối đa tiềm năng từ dữ liệu Facebook, nhưng chung quy lại mục tiêu đầu tiên cần đạt được là lấy được nguồn dữ liệu hoặc nguồn cấp dữ liệu ổn định. Tiêu biểu như 1 số ít công ty lớn hợp tác trực tiếp với Meta (công ty mẹ của Facebook ) để có khả năng truy cập trực tiếp vào Data Source của cá nhân/tổ chức trên Facebook với nhiều chức năng quản lý. Có thể kể đến NVIDIA có quyền truy cập vào kho dữ liệu của Meta để hợp tác xây dựng hệ sinh thái Metaverse; Roblox (1 hãng game lớn) nắm giữ đầy đủ dữ liệu về người dùng của họ thông qua liên kết với tài khoản Facebook. Tùy vào chức năng và mục đích mà mỗi công ty này có những đặc quyền sử dụng dữ liệu công khai/riêng tư khác nhau, chi phí cho những đặc quyền này cũng không hề nhỏ với đa số cá nhân và tổ chức. Cách tiếp cận phổ biến hơn (chủ yếu dùng để tự nghiên cứu và sử dụng cho quảng cáo-advertising) với các nhà phát triển, người kinh doanh là gián tiếp tiếp cận dữ liệu qua các API tạo ra theo hướng dẫn sẵn có của Meta để request đến máy chủ Facebook nhằm tìm kiếm và lấy thông tin trả về theo tham số đầu vào, phương pháp này yêu cầu phải đăng ký trở thành nhà phát triển thử nghiệm của Meta với các lựa chọn gói hợp tác từ miễn phí đến trả phí, phạm vi dữ liệu được truy cập bị hạn chế hơn nhiều nếu so với các công ty hợp tác dữ liệu theo hướng tiếp cận đầu tiên. Trong trường hợp không muốn thông qua đăng ký với các API giới hạn của Meta thì một số nhà phát triển chọn giải pháp tự tạo ra các API hoặc giao thức riêng liên tục truy vấn trực tiếp đến bộ tìm kiếm của Facebook rồi lấy về dữ liệu cần thiết như các công cụ OSINT (viết tắt của Open-source intelligence: Trí thông minh nguồn mở) đang phổ biến trên mạng. Đây thực chất chỉ là việc thay thế truy vấn thông tin Facebook bằng các cú pháp hỗ trợ và giao diện. Các hướng đi như trên đảm bảo tính chính thống trong việc sử dụng dữ liệu, tính chuẩn xác của dữ liệu. Tuy nhiên vẫn có những khuyết điểm khiến chúng không trở nên tối ưu để áp dụng theo phương hướng service product ở nước ta hiện nay như: Chi phí cao (để có thể sử dụng đầy đủ GraphAPI của Meta), số lượng và số loại dữ liệu được sử dụng bị giới hạn (ảnh hưởng từ thời điểm Facebook ngừng cung cấp đầy đủ tính năng của GraphAPI công khai), việc đăng ký không dễ dàng, thông tin không đa dạng và khó tùy biến các cài đặt theo mục đích của người sử dụng (bị giới hạn theo tài khoản đang sử dụng). Có thể thấy các giải pháp trên đều chỉ giải quyết một phương diện của bài toán lớn, chủ yếu chỉ tập trung xử lí vấn đề đặc thù của doanh nghiệp/người kinh doanh/nhà nghiên cứu mà chưa tận dụng hết tiềm năng sức mạnh của toàn bộ kho dữ liệu khổng lồ đang cập nhật hàng ngày để đem đến khả năng ứng dụng rộng khắp cho nhiều lĩnh vực khác nhau như đã nêu. Xuất phát từ các vấn đề còn tồn tại này, em đã lên ý tưởng nghiên cứu và xây dựng một hệ thống trên nguyên lý, cách vận hành khác biệt có khả năng thu thập nguồn dữ liệu không bị giới hạn phạm vi, có tính ổn định, sâu hơn là tích hợp các tính năng chọn lọc, xử lí, làm giàu dữ liệu có sẵn để chắt lọc những thông tin có giá trị cho các yêu cầu khác nhau, cấu hình có tính tùy biến và linh hoạt cao kèm tiêu chí quan trọng là hiệu quả sử dụng đạt tiêu chuẩn của một service product với giá thành vận hành hợp lí nhằm phục vụ tối đa nhu cầu của người dùng và xã hội.

Sinh viên trước tiên cần trình bày tổng quan các kết quả của các nghiên cứu hiện nay cho bài toán giới thiệu ở phần [1.1](#_heading=h.3znysh7) (đối với đề tài nghiên cứu), hoặc về các sản phẩm hiện tại/về nhu cầu của người dùng (đối với đề tài ứng dụng). Tiếp đến, sinh viên tiến hành so sánh và đánh giá tổng quan các sản phẩm/nghiên cứu này.

Dựa trên các phân tích và đánh giá ở trên, sinh viên khái quát lại các hạn chế hiện tại đang gặp phải. Trên cơ sở đó, sinh viên sẽ hướng tới giải quyết vấn đề cụ thể gì, khắc phục hạn chế gì, phát triển phần mềm **có các chức năng chính gì**, tạo nên đột phá gì, v.v.

Trong phần này, sinh viên lưu ý chỉ trình bày tổng quan, không đi vào chi tiết của vấn đề hoặc giải pháp. Nội dung chi tiết sẽ được trình bày trong các chương tiếp theo, đặc biệt là trong Chương 5.

## Định hướng giải pháp

Định hướng chủ đạo trong phát triển này là các phương pháp thu thập dữ liệu được triển khai bằng các công cụ/nền tảng lưu trữ xử lí dữ liệu như Redis/Elastic Search kết hợp với các thuật toán, thành tựu trong lĩnh vực Data Science/NLP/Machine Learning. Quy trình làm việc với dữ liệu về khái quát sẽ trải qua các bước: cấu hình - thu thập dữ liệu từ Facebook - xử lí - làm giàu - lưu trữ - trích xuất và trả về kết quả. Lưu ý các bước trên là các bước chính và toàn bộ phát triển này bao gồm nhiều quy trình theo các trình tự, số lượng tác vụ khác nhau tùy theo mỗi nhiệm vụ. Hệ thống này là ví dụ tham khảo nghiên cứu và cơ sở để phát triển các hệ thống hoàn thiện hơn trong tương lai nhằm giải quyết bài toán đặt ra, và hiện tại vẫn đang được thử nghiệm và sử dụng thực tế trong môi trường product.

Đồ án này khái quát cơ chế và đi sâu giải thích các khía cạnh về nguyên nhân và mục tiêu phát triển, cách vận hành, cơ sở lý thuyết, thực tế áp dụng và triển khai của hệ thống này. Có thể xem đồ án này như tài liệu trình bày cơ sở lý thuyết và hướng dẫn nghiệp vụ cho hệ thống được nói đến. Bước khảo sát và phân tích yêu cầu hệ thống sẽ được trình bày ở chương tiếp theo.

## Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau. Chương 2 trình bày về các bước trong quy trình khảo sát và phân tích yêu cầu hệ thống thông qua các biểu đồ use case, quy trình nghiệp vụ, đặc tả chức năng, phân tích yêu cầu phi chức năng.

Trong Chương 3, em giới thiệu về các công nghệ chính sử dụng để phát triển hệ thống trong đồ án này.

Chương 4 trình bày về quá trình thực nghiệm và đánh giá hệ thống khi triển khai thực tế qua quy trình thiết kế kiến trúc, giao diện, chi tiết gói và lớp, cơ sở dữ liệu, các thư viện công cụ sử dụng, minh họa chức năng chính, kết quả kiểm thử và triển khai.

Chương 5 trình bày các giải pháp và đóng góp nổi bật theo đánh giá chủ quan và khách quan từ kết quả triển khai bên cạnh những khó khăn thách thức bản thân gặp phải khi thực hiện dự án.

Chương 6 đưa ra kết luận về kết quả đã đạt được và hướng phát triển tiếp theo cho sản phẩm trong đồ án này.

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

Chương 2 trình bày về thực trạng về nhu cầu sử dụng trong môi trường doanh nghiệp và phân tích các yêu cầu cần thiết của hệ thống này, đó là một hệ thống cho phép thu thập, khai phá, làm giàu dữ liệu từ Facebook để từ đó sử dụng kết hợp với các hệ thống, hệ sinh thái thông tin khác nhằm phục vụ cho các ngành dịch vụ sản xuất liên quan đến dữ liệu mạng xã hội.

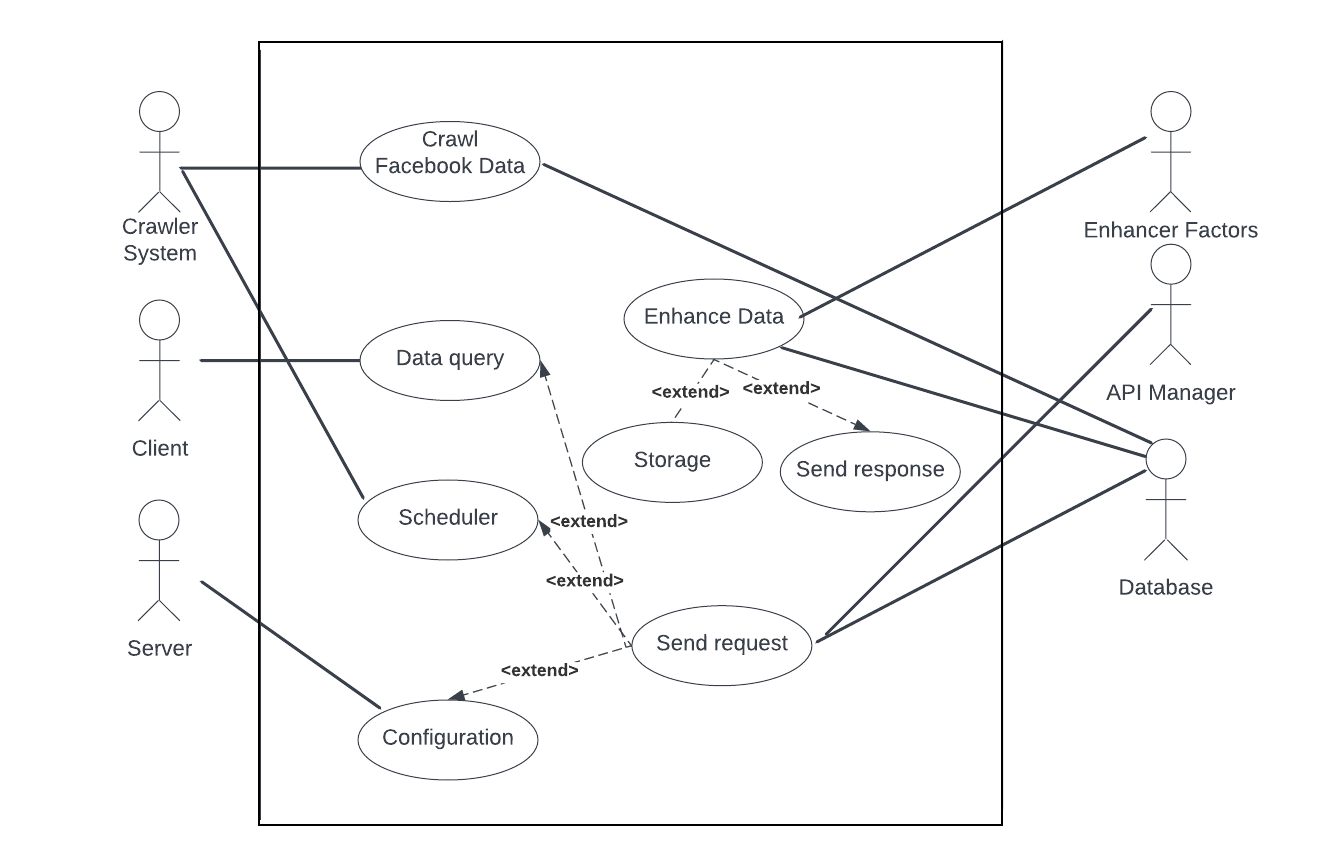
## Khảo sát hiện trạng

Trong môi trường phát triển thực tế của hệ thống, nhu cầu thiết yếu cần đáp ứng ưu tiên nhất đến từ việc cung cấp nguồn tin đa dạng và có tính cập nhật cao từ một số fanpage hoặc nhóm Facebook cho người sáng tạo nội dung, biên tập viên các đầu báo online như CafeF, ThanhNien, VnExpress,...So với các nguồn tin truyền thống phải tự thu thập thì Facebook có lượng tin bài lớn, dễ thống kê tương tác cũng như có sẵn các liên kết ngoài, link đa phương tiện gắn kèm. Sau đây là một số thống kê về nhu cầu và thực tế sử dụng nguồn dữ liệu gốc từ Facebook và các ứng dụng con của nó tại một số đầu báo lớn. Ngoài ra luôn có nhu cầu ứng dụng dữ liệu từ Facebook từ lớp người dùng cá nhân và doanh nghiệp khác qua các khảo sát và báo cáo từ thực tế phân tích thị trường.

Thị trường dữ liệu ở nước ta hiện nay đang chỉ cung cấp một số ít giải pháp tiềm năng cho khách hàng tùy thuộc yêu cầu sử dụng. Một sản phẩm/hệ thống đáng chú ý đang được triển khai cho môi trường product có thể kể đến “Hệ thống theo dõi phân tích tin tức mạng xã hội SMCC” của Công ty Cổ phần công nghệ chọn lọc InfoRe Technology. SMCC là một PaaS (Platform as a service) có thể sử dụng qua các bản dùng thử của người dùng ưu tiên hoặc bản đầy đủ cần trả phí duy trì. Theo giới thiệu trên trang chủ của SMCC (2): “SMCC giúp bạn thu thập, chọn lọc, phân tích thông tin từ các kênh social media công khai, đặc biệt là Page/Group/Profile Facebook và các báo điện tử. Dữ liệu công khai trên internet là thượng vàng hạ cám, để tìm ra nhiều thông tin quý cần phải chọn lọc kĩ càng. SMCC cung cấp cho người sử dụng cách chọn lọc thông tin bằng các mô hình dữ liệu được phân tích tự động bởi AI, chúng tôi gọi là các Dataverse.” So với các hệ thống trong cùng phân khúc đây là hệ thống có tính ứng dụng thực tiễn và quy mô lớn, khả năng mở rộng (scability) cao do hoạt động qua mạng lưới Data Miner rộng khắp phụ thuộc vào quy mô người dùng, nhờ đó tốc độ cập nhật dữ liệu khá nhanh (tuy chưa đạt mức độ near-realtime) và độ phủ của dữ liệu là rộng khắp đa số chủ đề, lĩnh vực. Dashboard của SMCC cung cấp các thống kê phân tích về số lượng, tần suất, các sắc thái NLP của các đối tượng dữ liệu để hiển thị theo dõi trending theo thời gian , độ liên quan giữa các chủ đề, tính tích cực tiêu cực của chúng. Em cũng đã trải nghiệm thử hệ thống này qua vai trò người dùng ưu tiên để rút ra những ưu điểm vừa rồi của SMCC. Bên cạnh đó theo đánh giá chủ quan của em và khảo sát qua những người sử dụng khác kết hợp môi trường thực tế doanh nghiệp có thể thấy những điểm trừ tồn đọng của SMCC như: Giá thành vận hành không nhỏ nhằm duy trì lượng lớn Data Miner phân tán (một phần vì nhu cầu thu thập dữ liệu từ nhiều nguồn khác không chỉ Facebook), tốc độ cập nhật dữ liệu chưa đủ nhanh để bắt kịp các sự kiện hàng giờ, hàng phút, tính năng truy vấn còn hạn chế trong phạm vi những từ khóa được phân tích, không có khả năng cấu hình kết quả trả về theo yêu cầu đơn cử như sắp xếp các kết quả theo bộ lọc về số lượng reaction, thiếu tính năng phân tích xếp hạng các kết quả. Những điểm trừ này khiến tính ứng dụng thực tế bị giảm sút do chưa đáp ứng yêu cầu E2ES (End to end solution) của một sản phẩm công nghệ. Để giải quyết triệt để các vấn đề trên cần một hệ thống được xây dựng với mục đích gần tương tự SMCC với các yêu cầu về tính năng sau: (i) Khả năng nâng cao tối đa tính tùy biến của người dùng trong việc cấu hình và đưa ra yêu cầu (yêu cầu việc thu thập các thông tin ở mức sâu hơn để thu được các đối tượng chi tiết như số tương tác, xây dựng công cụ phân tích trả về kết quả linh hoạt theo truy vấn), (ii) sử dụng mô hình thu thập dữ liệu mới thu hẹp về quy mô phân tán (vì lí do không cần thiết thu thập các nguồn dữ liệu khác), (iii) tạo tính năng tự động cấu hình lịch trình thu thập dữ liệu để tối ưu tốc độ cập nhật, (iv) kết hợp các công cụ phân tích dữ liệu dựa vào học máy, NLP để làm giàu giá trị dữ liệu, nâng cao tính chính xác của kết quả trả về.

## Tổng quan chức năng

## Biểu đồ use case tổng quát



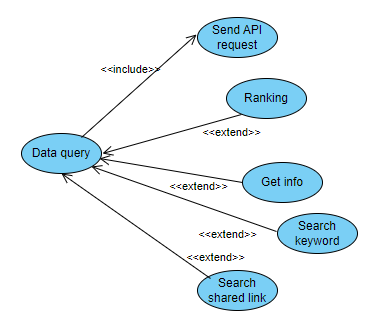
**Biểu đồ 2.2.1:** Biểu đồ Use Case tổng quát

Các tác nhân tham gia gồm: Người dùng (Người sử dụng kết quả của hệ thống, có vai trò đưa ra truy vấn cho hệ thống), Server (Người quản lý vận hành hệ thống, có vai trò cấu hình hệ thống để yêu cầu từ Client cũng như khả năng của hệ thống, ra lệnh trực tiếp cho hệ thống), Crawler System (Hệ thống thu thập và tiền xử lý dữ liệu với trung tâm là Crawler Bot, vai trò thu thập-xử lý dữ liệu từ nguồn cho trước theo cấu hình của Server rồi lưu trữ vào các DataLake trong Database), API Manager (Module quản lý các API cho việc giao tiếp giữa Client/Server với các thành phần của hệ thống, bao gồm các request truy vấn dữ liệu của Client, request cấu hình hệ thống của Server), Enhance Factors ( Các nhân tố tăng cường dữ liệu, có vai trò làm giàu giá trị dữ liệu thô qua các xử lý, phân tích, biến đổi dữ liệu sẵn có thành các dữ liệu quy chuẩn có thể áp dụng trực tiếp thông qua các công cụ của Học máy và NLP), Databases ( Các cơ sở dữ liệu, dùng cho việc lưu trữ và trích xuất các phiên bản/cấu trúc dữ liệu khác nhau, sử dụng gồm cả 2 loại CSDL là SQL và NoSQL nhằm tận dụng tối đa điểm mạnh của chúng. Cả dữ liệu được thu thập và dữ liệu về cấu hình được lưu giữ ở trên các CSDL này và có thể tương tác bởi Client/Server với phân quyền khác nhau qua API Manager).

Các Use-case chính gồm (i) Client-Data query-API Manager và Databases: Quá trình Client gửi request truy vấn dữ liệu qua API đến cho API Manager và Databases, từ đây API Manager sẽ kết hợp làm việc với Databases, (ii) Databases và Enhance Factors-Enhance Data: Dữ liệu từ Databases sẽ được các nhân tố ML, NLP phân tích và tăng cường rồi tiếp tục lưu trữ hoặc trả về kết quả cho Client.

## Các biểu đồ use case phân rã

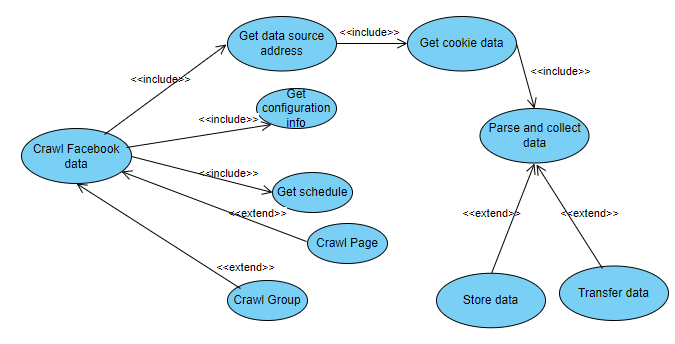
## 2.2.2.1 Biểu đồ use case phân rã cho use case gửi truy vấn dữ liệu đến hệ thống (Data query)



**Biểu đồ 2.2.2.1:** Biểu đồ phân rã use case gửi truy vấn dữ liệu đến hệ thống

Use case Data query có hoạt động bắt buộc là gửi API request đến hệ thống và có các use case quan hệ mở rộng khác là các tùy chọn của yêu cầu truy vấn gồm yêu cầu xếp hạng kết quả (Ranking); yêu cầu trích xuất thông tin dữ liệu tổng quát (Get info); tìm kiếm theo từ khóa và liên kết. Tại một thời điểm người dùng chỉ sử dụng một tùy chọn yêu cầu.

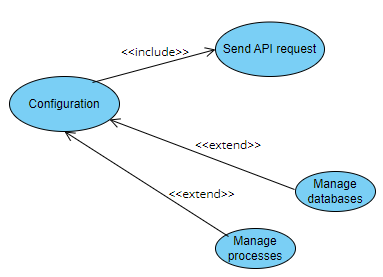
## 2.2.2.2 Biểu đồ use case phân rã cho use case thu thập dữ liệu (Crawl Facebook data)



**Biểu đồ 2.2.2.2:** Biểu đồ phân rã cho use case thu thập dữ liệu

Use case Crawl Facebook data có các hoạt động bắt buộc là lấy thông tin nguồn dữ liệu từ đó lấy dữ liệu cookie-phân tích và thu thập dữ liệu tiêu chuẩn-lưu trữ hoặc chuyển tiếp dữ liệu sang các kho dữ liệu khác, lấy thông tin cấu hình và lịch thu thập; các use case mở rộng gồm tùy chọn thu thập dữ liệu trang hoặc nhóm. Hệ thống có thể hoạt động song song các tùy chọn này.

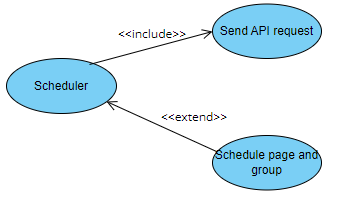
## 2.2.2.3 Biểu đồ use case phân rã cho use case quản trị viên quản lý cấu hình hệ thống (Configuration)



**Biểu đồ 2.2.2.3:** Biểu đồ phân rã use case quản trị viên quản lý cấu hình hệ thống

Use case Configuration có hoạt động bắt buộc là gửi yêu cầu quản lý qua API; các use case mở rộng gồm các tùy chọn quản lý cấu hình chung hệ thống và quản lý việc vận hành các tiến trình (Manage process), ví dụ như chạy hoặc tạm dừng tiến trình, tùy chọn quản lý các cơ sở dữ liệu với các tương tác thay đổi, cập nhật, thêm mới, xóa, truy xuất dữ liệu. Một quản trị viên tại một thời điểm chỉ thao tác với một tùy chọn use case quản lý.

## 2.2.2.4 Biểu đồ use case phân rã cho use case gửi yêu cầu lập lịch (Scheduler)



**Biểu đồ 2.2.2.4:** Biểu đồ use case phân rã cho use case gửi yêu cầu lập lịch

Use case Scheduler có hoạt động bắt buộc là gửi yêu cầu lập lịch qua API; use case mở rộng là lập lịch thu thập dữ liệu cho các nguồn cấp (trang và nhóm). Tại một thời điểm hệ thống chỉ nhận một yêu cầu lập lịch.

## 2.2.3 Quy trình nghiệp vụ

Quy trình nghiệp vụ của hệ thống bao gồm nhiều quy trình nghiệp vụ thành phần cấu thành, một số quy trình chính gồm có (i) Quy trình nghiệp vụ gửi yêu cầu truy vấn đầu vào từ khách hàng: Khách hàng đăng ký mã định danh ID qua đội ngũ xử lý và chờ xác nhận quyền sử dụng dịch vụ, sau khi đã có quyền truy cập sẽ chọn đăng ký các nguồn dữ liệu cần sử dụng. Khách hàng được xác nhận đầy đủ quyền và thông tin sẽ thông qua giao thức kết nối được cung cấp (VD: tương tác qua giao diện phần mềm) gửi các tham số chứa thông tin truy vấn bằng API tiêu chuẩn đến hệ thống và chờ kết quả trả về qua giao diện sử dụng. (ii) Quy trình nghiệp vụ quản lý cấu hình hệ thống: Được thực hiện bởi quản trị viên và người vận hành hệ thống. Chủ thể thực hiện tiến hành kiểm tra thông tin và quyền truy cập hệ thống, tạo yêu cầu quản lý bằng API theo nghiệp vụ, gửi yêu cầu đến hệ thống, kiểm tra và xác nhận kết quả thực hiện yêu cầu từ hệ thống. (iii) Quy trình đăng ký các nguồn dữ liệu để thu thập: người dùng lập danh sách các nguồn dữ liệu cần sử dụng rồi gửi về đội ngũ quản lý, đội ngũ quản lý kiểm tra các quyền của người dùng và tính hợp pháp của nguồn dữ liệu rồi đưa ra các quyết định chấp nhận hoặc từ chối mỗi đăng ký. Nếu đăng ký thành công, địa chỉ nguồn dữ liệu sẽ được lưu trữ liên kết đến ID khách hàng trong cơ sở dữ liệu và được kích hoạt trạng thái hoạt động.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

## Đặc tả chức năng

Mục này trình bày đặc tả chức năng chi tiết của các use case quan trọng nhất trong hệ thống. Đặc tả được thể hiện ở bảng 2.3.a dưới đây.

| Tên use case và các tác nhân tham gia | Luồng sự kiện | Tiền điều kiện | Hậu điều kiện |
| --- | --- | --- | --- |
| Người dùng sử dụng dịch vụ hệ thống (người dùng, đội ngũ quản lý, hệ thống) | -Người dùng tiến hành đăng ký sử dụng dịch vụ theo quy trình có sẵn  -Đội ngủ quản lý tiếp nhận đăng ký và cấp quyền sử dụng, ID và dữ liệu khách hàng được tạo mới trong cơ sở dữ liệu  -Tài khoản người dùng được phân quyền  -Người dùng sử dụng tài khoản được cấp quyền để sử dụng dịch vụ tùy chọn  -Hệ thống tiếp nhận, thực hiện và phản hồi yêu cầu từ người dùng | -Người dùng đã đăng ký sử dụng hệ thống và được đội ngũ quản lý xác nhận  -ID và dữ liệu liên quan của người dùng đã được tạo trong cơ sở dữ liệu  -Tài khoản người dùng đã được phân quyền | -Yêu cầu của người dùng được gửi thành công  -Hệ thống ghi nhận yêu cầu và thực hiện yêu cầu |
| Người dùng hủy đăng ký sử dụng dịch vụ (người dùng, đội ngũ quản lý, hệ thống) | -Người dùng gửi yêu cầu hủy đăng ký sử dụng dịch vụ cho đội ngũ quản trị  -Đội ngũ quản trị xác nhận yêu cầu và cập nhật lại trạng thái, dữ liệu người dùng trong cơ sở dữ liệu  -Hệ thống cập nhật lại cấu hình theo dữ liệu mới, hủy quyền sử dụng dịch vụ của người dùng và gửi thông báo xác nhận cho đội ngũ quản trị  -Đội ngũ quản trị xác nhận đã hủy đăng ký sử dụng dịch vụ thành công cho khách hàng | -Người dùng đã có ID và dữ liệu được lưu trữ trong hệ thống, có tài khoản được phân quyền từ trước  -Người dùng đã tiến hành gửi yêu cầu hủy đăng ký sử dụng dịch vụ theo đúng quy trình | -Đội ngũ quản trị nhận được thành công các yêu cầu xác nhận từ người dùng và hệ thống  -Hệ thống ghi nhận và thực hiện yêu cầu hủy đăng ký thành công |
| Người dùng tích hợp thêm dịch vụ mới vào hệ thống | -Người dùng gửi yêu cầu và thông tin dịch vụ tích hợp cho đội ngũ quản trị  -Đội ngũ quản trị đánh giá quyền hạn của người dùng và tính hợp lệ, quyền hạn của dịch vụ rồi gửi yêu cầu cho hệ thống  -Hệ thống kiểm tra khả năng tích hợp từ thông tin của đội ngũ quản trị  -Hệ thống tích hợp dịch vụ mới và thông báo xác nhận tích hợp thành công | -Người dùng đã có ID và dữ liệu được lưu trữ trong hệ thống, có tài khoản được phân quyền từ trước  -Người dùng đã tiến hành gửi yêu cầu tích hợp và thông tin dịch vụ theo đúng quy trình  -Hệ thống hiện tại cho phép tích hợp thêm dịch vụ mới | -Đội ngũ quản trị nhận được thành công các yêu cầu xác nhận từ người dùng và hệ thống  -Dịch vụ mới được đăng ký tích hợp thành công vào hệ thống  -Hệ thống xác nhận tích hợp thành công |
| Quản trị viên gửi yêu cầu quản lý hệ thống | -Quản trị viên gửi yêu cầu quản lý qua API với tài khoản đã được phân quyền  -Hệ thống tiếp nhận và xác thực yêu cầu  -Hệ thống thực hiện yêu cầu và thông báo xác nhận đã thực hiện cho quản trị viên | -Tài khoản quản trị viên đã được phân quyền và có quyền quản lý tính năng yêu cầu  -Quản trị viên đã gửi yêu cầu quản lý theo chuẩn API được cung cấp | -Hệ thống xác thực thành công quyền và yêu cầu API của quản trị viên  -Hệ thống thực thi thành công yêu cầu và có phản hồi |

**Bảng 2.3.a:** Đặc tả chức năng các use case quan trọng

## Yêu cầu phi chức năng

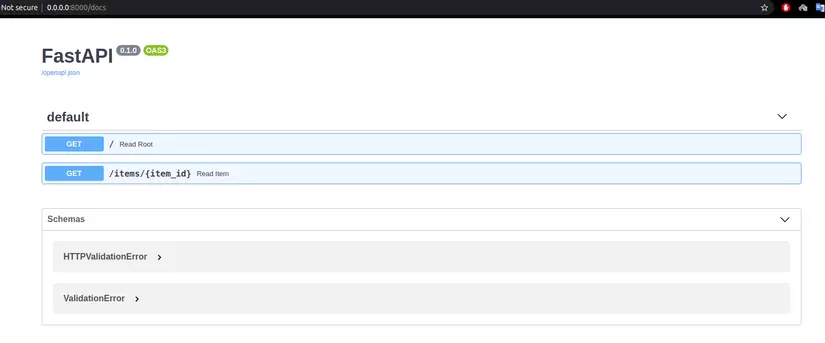
Các yêu cầu phi chức năng cần đáp ứng khi triển khai thực tế gồm có: (i) Về hiệu suất hệ thống: Thời gian phản hồi yêu cầu từ phía người dùng không quá 2 giây; Có khả năng chịu tải với lưu lượng yêu cầu hàng trăm request mỗi giây; Có khả năng thu thập và lưu trữ dữ liệu từ 10 đến 50 trang/nhóm Facebook ổn định mỗi ngày với mỗi máy con (worker) và khoảng 1000 trang/nhóm với một cụm máy con/toàn hệ thống để dễ dàng duy trì mở rộng, (ii) Về khả năng mở rộng: Yêu cầu hệ thống có khả năng tích hợp thêm các máy con dễ dàng hoặc giảm bớt khi cần thiết mà không ảnh hưởng các luồng nhiệm vụ khác, (iii) Về cơ sở dữ liệu: Yêu cầu phải có các cơ sở dữ liệu riêng biệt cho các mục đích lưu trữ dữ liệu thô, dữ liệu backup-cache, dữ liệu làm giàu; cơ sở dữ liệu phải có kích thước đủ lớn để hệ thống hoạt động lâu dài và có khả năng mở rộng, chuyển đổi sang các địa chỉ/kho dữ liệu mới, (iv) Về bảo mật và an toàn: Toàn bộ hệ thống bao gồm mã nguồn, quy trình nghiệp vụ xử lý, cơ sở dữ liệu, các API và khóa mã hóa, thông tin khách hàng nếu có cần được bảo mật nhất có thể, đảm bảo tính toàn vẹn dữ liệu và ngăn ngừa mọi truy cập trái phép vào hệ thống, (v) Về khả năng bảo trì: Mã nguồn và quy trình xử lý của hệ thống cần được trình bày theo quy chuẩn nghiệp vụ, tuân theo các mẫu thiết kế đã thống nhất và đi kèm với tài liệu hướng dẫn đễ dễ dàng duy trì, nâng cấp sau này, (vi) Khả năng sử dụng: Hệ thống khi triển khai đến khách hàng cần có giao diện/cách thức tương tác dễ hiểu, dễ sử dụng, ít gây nhầm lẫn cho người dùng, (vii) Khả năng tương thích: Hệ thống khi được tích hợp với các hệ thống biên cần đạt sự tương thích hiệu quả để vận hành, có các tùy chọn và hướng dẫn kết nối tới các hệ thống khác cần tích hợp, (viii) Tính hợp pháp: Mọi quy trình nghiệp vụ và mã nguồn hệ thống cần đảm bảo việc không vi phạm bất cứ luật hoặc quy định hiện hành nào trong lĩnh vực công nghệ thông tin, bản quyền trí tuệ, nội dung số và các lĩnh vực khác từ trong và ngoài nước. Các công nghệ sử dụng đều phải là các công nghệ đã được công khai mã nguồn chính thức hoặc có sự cho phép về mặt pháp lý của người sở hữu.

Chương này đã trình bày các kết quả khảo sát, tìm hiểu nghiệp vụ về chức năng và yêu cầu cần có của hệ thống. Chương tiếp theo sẽ trình bày về các công nghệ sử dụng trong quá trình phát triển hệ thống qua cơ sở lý thuyết và áp dụng thực tiễn.

## CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

Chương này trình bày về các công nghệ, nền tảng sử dụng để giải quyết các vấn đề và yêu cầu đã nêu ở trong chương 2 . Ngoài ra em cũng liệt kê qua một số công nghệ tương tự trong mục đích ứng dụng và giải thích nguyên nhân lựa chọn sử dụng những công nghệ nào cho đồ án này.

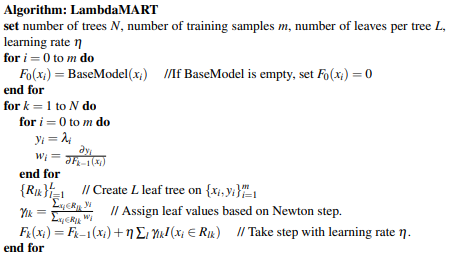
* FastAPI: Theo định nghĩa trên trang chủ Amazon Web Service (AWS), API là “cụm từ viết tắt của Giao diện chương trình ứng dụng. Trong ngữ cảnh API, từ “Ứng dụng” đề cập đến mọi phần mềm có chức năng riêng biệt. Giao diện có thể được xem là một hợp đồng dịch vụ giữa 2 ứng dụng. Hợp đồng này xác định cách thức hai ứng dụng giao tiếp với nhau thông qua các yêu cầu và phản hồi. Tài liệu API của các ứng dụng này chứa thông tin về cách nhà phát triển xây dựng cấu trúc cho những yêu cầu và phản hồi đó.” Để cung cấp các kênh giao tiếp giữa các tác nhân trong hệ thống em đã sử dụng framework FastAPI (3), một “web framework hiện đại, nhanh (hiệu suất cao) để xây dựng API với Python 3.7+ dựa trên các kiểu giá trị của Python tiêu chuẩn” (trích từ trang chủ của nhà phát triển FastAPI). Với các ưu điểm vượt trội về tốc độ, hiệu suất (ngang với các framework phát triển web API nổi tiếng như NodeJS, Go theo đánh giá từ người dùng), tính đơn giản, sử dụng Python làm ngôn ngữ chính vốn rất phổ biến với các nhà phát triển, hỗ trợ cho việc debug, được xây dựng dựa trên các tiêu chuẩn API có sẵn từ OpenAPI và Json Schema, hơn hết là khả năng tạo và triển khai, kiểm thử API nhanh gọn; trực quan hóa dễ dàng bằng FastAPI Docs (document có thể tương tác trực tiếp)



**Hình 3.1.a:** Giao diện web của FastAPI Docs

FastAPI tận dụng sức mạnh từ những framework lớn khác như Pydantic: Tăng tốc độ parsing (phân tích cú pháp) dữ liệu JSON và kiểm tra dữ liệu có dạng email; Starlette: tạo request, config và hỗ trợ GraphQL. Ngoài ra để triển khai sản phẩm trong thực tế, FastAPI cung cấp dịch vụ ASGI server như Uvicorn ( ASGI kế thừa từ WSGI-một chuẩn giao tiếp giữa web server và Python application server cho quá trình triển khai ứng dụng, ASGI hỗ trợ cả cơ chế đồng bộ và bất đồng bộ cho các nhiệm vụ cần xử lý nhiều request cùng lúc). Các công nghệ, framework có thể sử dụng cho cùng mục đích tạo và quản lý các API giao tiếp như FastAPI có thể kể đến như Django với Django REST framework, Flask. Chúng đều sử dụng ngôn ngữ Python và có tuổi đời lâu hơn so với FastAPI cũng như cộng đồng sử dụng lớn hơn. Tuy nhiên lí do để em lựa chọn FastAPI cho đồ án này bởi tính đơn giản dễ sử dụng nhưng vẫn đáp ứng đủ các yêu cầu cần thiết về triển khai và quản lý các API cho các thành phần hệ thống; hỗ trợ xử lý bất đồng bộ giúp tăng khả năng và tốc độ xử lý request nhờ cơ chế ASGI trong khi Django và Flask chỉ hỗ trợ cơ chế đồng bộ; FastAPI là framework có tốc độ nhanh hơn nhiều so với 2 framework trên, phù hợp cho nhiệm vụ yêu cầu đáp ứng về tốc độ phản hồi như đồ án này; FastAPI có sẵn module giúp kiểm tra tính đúng đắn của dữ liệu đầu vào, giảm thiểu công việc kiểm tra cho người quản lý API.

* LambdaMART Ranking: Thuật toán Learning to Rank-Học để xếp hạng LambdaMART. Learning to Rank là một ứng dụng của máy học, điển hình là học có giám sát, bán giám sát hoặc tăng cường, trong việc xây dựng các mô hình xếp hạng kết quả cho hệ thống truy xuất thông tin. “ LambdaMART là phiên bản cây tăng cường của LambdaRank, dựa trên RankNet. RankNet, LambdaRank và LambdaMART đã được chứng minh là những thuật toán rất thành công để giải quyết các vấn đề xếp hạng trong thế giới thực: ví dụ kết quả xếp hạng từ LambdaMART đã giành được vị trí thứ nhất trong cuộc thi Thử thách Learning to Rank của Yahoo năm 2010” (4). Bản chất của LambdaMART là một kỹ thuật trong đó việc xếp hạng các ứng viên được chuyển đổi thành vấn đề hồi quy hoặc phân loại theo cặp ứng viên. Các thuật toán xem xét mỗi cặp mục cùng một lúc, đưa ra thứ tự khả thi của các mục đó trước khi đánh giá thứ tự của toàn bộ danh sách. Em sử dụng LambdaMART trong đồ án này để phục vụ cho nhiệm vụ xếp hạng các kết quả trả về cho truy từ người dùng, đưa ra các thông tin phù hợp cho tiêu chí của người dùng bao gồm mức độ tương tác, thời gian, độ phù hợp về từ khóa. Các thuật toán tiền thân của LambdaMART như LambdaRank hay MART cũng thường được dùng để giải quyết bài toán trên. Trong khi MART sử dụng cây quyết định được tăng cường độ dốc (gradient boosted) cho các nhiệm vụ dự đoán thì LambdaMART cải thiện điều này bằng cách sử dụng cây quyết định tăng cường độ dốc với hàm chi phí bắt nguồn từ LambdaRank để sắp xếp mọi tình huống xếp hạng. Qua đánh giá LambdaMART đã cho thấy kết quả tốt hơn LambdaRank và RankNet ban đầu trên bộ dữ liệu Facebook được thu thập bởi chính hệ thống của em, so về cả tốc độ hoàn thành nhiệm vụ và độ tin cậy của xếp hạng. (6-cần biểu đồ so sánh)



**Hình 3.1.b:** Thuật toán LambdaMART

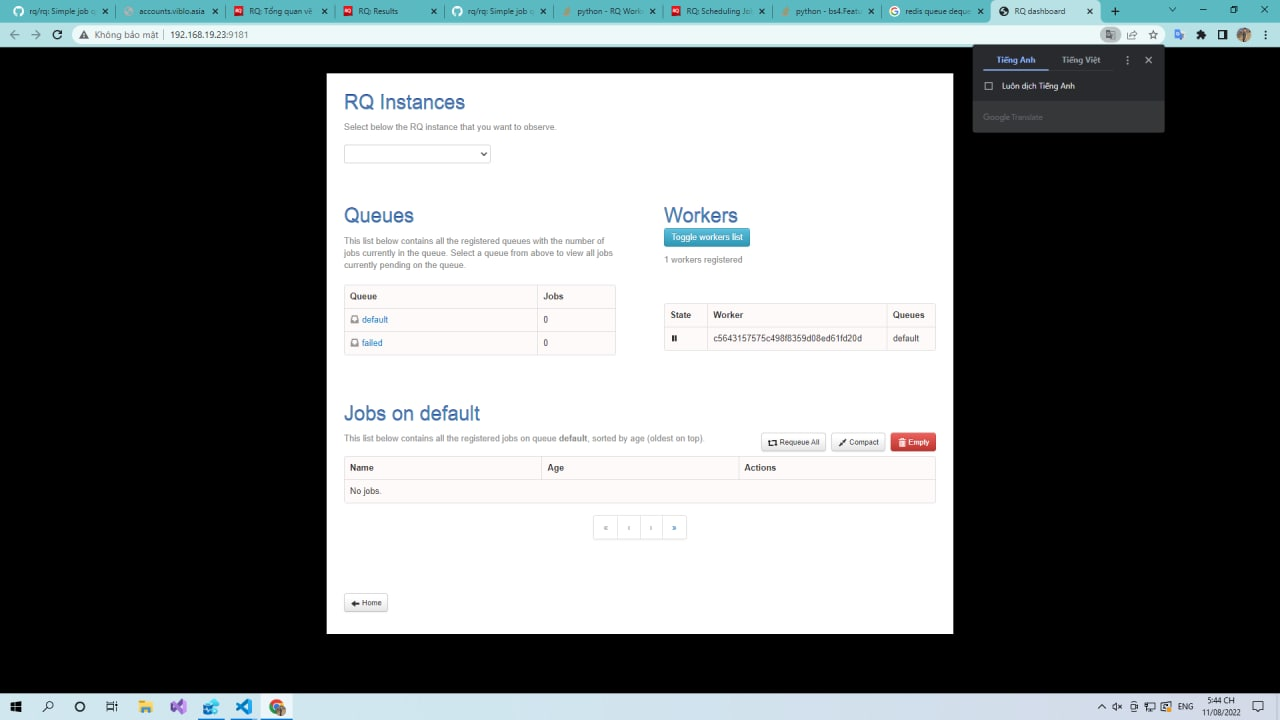
Dữ liệu đầu vào cho mô hình xếp hạng sẽ là các kết quả từ truy vấn từ điển của người dùng-là các bài viết với thông tin số tương tác, thời gian, nội dung. Bước biến đổi sẽ đưa các thông tin này thành dữ liệu đầu vào (các tài liệu) cho việc xếp hạng. Trước tiên LambdaMART sẽ có 2 giả định để nó coi như các mục tiêu mà việc xếp hạng cần đạt được, đó là: (i) Các tài liệu có liên quan cao sẽ hữu ích hơn khi xuất hiện sớm hơn trong danh sách kết quả của công cụ tìm kiếm (có thứ hạng cao hơn); (ii) Các tài liệu có liên quan cao sẽ hữu ích hơn các tài liệu có liên quan ít, và ngược lại, các tài liệu này lại hữu ích hơn các tài liệu không liên quan (8). Với LambdaMART, vấn đề học cách xếp hạng tương tự như một vấn đề phân loại nhị phân h(xu, xv) giữa hai tài liệu trong bộ dữ liệu, cho biết tài liệu nào xếp hạng cao hơn trong một cặp tài liệu nhất định. Bộ phân loại có mục tiêu là giảm thiểu hàm mất mát L(h,xu, xv,yuv). Hàm L dựa trên điểm đánh giá DCG (Mức tăng tích lũy được chiết khấu) là một chỉ số dùng làm thước đo chất lượng xếp hạng; nó tính toán mức độ “phù hợp” của một tài liệu dựa trên vị trí của nó trong bảng xếp hạng rồi tích lũy điểm số này từ đầu đến cuối danh sách theo công thức chiết khấu sau:



**Hình 3.1.c:** Công thức của DCG, với p là số tài liệu trong xếp hạng

Ở đây reli là mức độ phù hợp được phân loại của kết quả tại vị trí i theo mục tiêu của 2 giả định từ đầu của LambdaMART (nếu một tài liệu có thứ hạng cao và nó nằm ở các vị trí đầu tiên đồng nghĩa với điểm relcủa nó sẽ cao hơn nếu nó nằm ở các vị trí có thứ tự sau). Thông qua quá trình xếp hạng theo cặp và hoán đổi vị trí các cặp liên tục nhằm đánh giá hàm mất mát tổng quát, LambdaMART sẽ đưa ra kết quả xếp hạng phù hợp nhất theo tiêu chí đánh giá đầu vào.

* Redis/Redis Queue: Nhiệm vụ thu thập lưu trữ dữ liệu trực tiếp từ Facebook tuy không quá phức tạp về mặt thuật toán nhưng lại tiêu tốn phần lớn tài nguyên và thời gian trong cả hệ thống. Các tác vụ I/O được thực hiện liên tục (có thể trùng thời điểm thực hiện) để đọc, ghi, xuất, chuyển đổi dữ liệu theo các định dạng dễ gây chồng chéo, tắc nghẽn dẫn đến giảm hiệu năng chung toàn hệ thống, thậm chí làm treo hệ thống. Việc quản lý tốt quá trình có tính thứ tự này sẽ đảm bảo tiêu chí tốc độ cập nhật dữ liệu cũng như dễ dàng debug, truy vết các luồng con khi gặp lỗi hay cần sửa đổi hệ thống. Để giải quyết yêu cầu này em đã sử dụng module hàng đợi Redis-Queue chạy tích hợp trên cơ sở backend của Redis: Kho lưu trữ cấu trúc dữ liệu trong bộ nhớ, cũng được dùng như một cơ sở dữ liệu khóa-giá trị phân tán trong bộ nhớ, bộ đệm của nhiều hệ thống với khả năng cấu hình đa dạng, hỗ trợ nhiều loại cấu trúc dữ liệu, tốc độ truy cập nhanh chóng (thường được dùng cho các ứng dụng message broker yêu cầu chuyển tiếp thông tin nhanh chóng), khả năng mở rộng tới hàng triệu node cho mỗi cụm, khả năng chịu lỗi với các bản sao dự phòng (9). Việc sử dụng cả Redis và Redis Queue không chỉ giải quyết vấn đề nêu trên mà còn cung cấp khả năng chịu lỗi cho dữ liệu đi vào hệ thống với khả năng backup (dữ liệu vẫn tồn tại trong Redis trong thời gian người quản lý cấu hình). Trên nền tảng của Redis, Redis Queue được em sử dụng để tạo một queue với các nhiệm vụ đi vào queue là các lệnh thu thập và lưu trữ dữ liệu cho mỗi fanpage, group Facebook. Sở dĩ các task-nhiệm vụ này có tính thứ tự bởi chúng có thời gian bắt đầu khác nhau, tùy thuộc vào bộ lên lịch cho các tác vụ mà em đã trình bày ở chương 2; Redis Queue giúp đảm bảo tính tuần tự và liên tục của các nhiệm vụ này; ngoài ra còn hỗ trợ các thao tác sắp xếp, chèn, xóa queue bằng tham số, tính năng này được em tận dụng để quản lý tiến trình này. Khi yêu cầu về số nhiệm vụ (tỉ lệ với lượng dữ liệu theo thời gian hay số lượng cá thể Facebook cần thu thập dữ liệu) tăng lên, quản trị viên hệ thống có thể tăng số hàng đợi chạy song song để đáp ứng; mỗi hàng đợi sẽ chạy trên một worker và các worker đều phục vụ chung một cơ sở dữ liệu Redis duy nhất. Với yêu cầu về kết nối hệ thống local với server để tập trung dữ liệu, Redis cung cấp chức năng kết nối qua IP-Port ngoài tùy chọn localhost. Redis Queue cũng có sẵn dashboard trực quan các nhiệm vụ trong hàng đợi và cả các nhiệm vụ failed vẫn còn được lưu lại trong Redis.



**Hình 3.1.d:** Giao diện Redis Queue Dashboard

Bên cạnh Redis/Redis Queue thì RabbitMQ cũng là một giải pháp hàng đợi mã nguồn mở phổ biến có tính ổn định, chịu lỗi, khả năng mở rộng và đáng tin cậy đã được kiểm chứng. RabbitMQ được viết bằng Erlang OTP, cơ chế message broker của nó hỗ trợ “giao thức xếp hàng tin nhắn nâng cao” (10) giúp xử lý nhiều tác vụ đồng thời . Không chỉ có những ưu điểm chính của Redis, RabbitMQ còn có hệ thống plugin đa dạng cho người dùng lựa chọn, dễ dàng tích hợp vào các ứng dụng khác. Cả hai giải pháp này đều có ưu nhược điểm không quá khác biệt nhau, và lựa chọn sử dụng Redis hoàn toàn bởi sự phù hợp hơn của nó xét trong quy mô và cách thức vận hành của hệ thống này. Một số đặc điểm đó có thể kể đến như sau: (i) RabbitMQ yêu cầu sử dụng Erlang ở mức độ nào đó, trong khi Redis dễ dàng triển khai bằng Python (ngôn ngữ chính để xây dựng hệ thống); (ii) RabbitMQ không ổn định trong môi trường có lượng truy cập I/O cao, vấn đề này Redis có thể giải quyết; (iii) Tốc độ xử lý của Redis nhanh hơn so với RabbitMQ do Redis là một in-memory database; (iv) Khả năng cập nhật dữ liệu: Vì hệ thống có thể có nhiều worker hoạt động cùng lúc, dữ liệu cũng cần được cập nhật chính xác và kịp thời, và đây là điểm mạnh của Redis.

* Elasticsearch: là một công cụ tìm kiếm dựa trên nền tảng Apache Lucene. Nó cung cấp một bộ máy tìm kiếm dạng phân tán, có đầy đủ công cụ với một giao diện web HTTP có hỗ trợ dữ liệu JSON. Phát triển bằng Java và hoạt động như một web server và tương tác qua giao thức RESTful khiến nó dễ dàng tích hợp vào hệ thống bất kể ngôn ngữ gì. Nhờ ưu điểm về tốc độ truy vấn và khả năng mở rộng hướng phân tán, Elasticsearch (viết tắt ES) đang được các hệ thống lớn sử dụng cho chức năng tìm kiếm như Wikimedia, Quora, Github, Facebook, Netflix. Thông thường ES sẽ được dùng song song với một database chính lưu trữ dữ liệu như SQL, MongoDB. Trong phạm vi đồ án này em sử dụng Elasticsearch như một công cụ tìm kiếm dữ liệu theo truy vấn đầu vào của người dùng sau khi đã được xử lý về dạng truy vấn của Elasticsearch. Elasticsearch còn đóng vai trò như một DataWarehouse lưu trữ các dữ liệu đã được phân tích và làm giàu từ DataLake nhằm tối ưu khả năng tìm kiếm, đáp ứng tiêu chí end-to-end solution cho dữ liệu. Ngoài ES, MongoDB hay SQL cũng là các lựa chọn cho việc lưu trữ và tìm kiếm dữ liệu, nhưng điểm mạnh của các database này nằm ở việc lưu trữ chứ không phải tìm kiếm như ES. Dữ liệu thực tế của hệ thống này là các đơn vị văn bản dưới dạng bài viết; trong khi bộ truy vấn của SQL tỏ ra không phù hợp trong việc tìm kiếm các từ khóa (đặc biệt là tiếng Việt) do không phân biệt các dấu câu và đơn vị từ ngữ thì ES có cách làm việc với từ khóa rất khác: đánh index cho từng từ khóa trong văn bản theo cấu trúc inverted index để xác định mỗi từ khóa trong database thuộc văn bản nào. Nhờ đó ES có thể tìm kiếm bất cứ từ khóa nào bất kể các định dạng theo dấu câu hay viết hoa viết thường , đây cũng là lí do ES có hỗ trợ chế độ tìm kiếm mờ (fuzzy search) thường được dùng đến trong các tìm kiếm không chắc chắn của người dùng (chiếm đa số lượng tìm kiếm trên các nền tảng). So với MongoDB thì ES có nhiều điểm tương đồng hơn khi so sánh với SQL; tuy vậy ES vẫn có ưu thế hơn về khả năng xử lý các yêu cầu bằng các API Restful; tự động đánh index khi có dữ liệu mới. Tất cả các đặc điểm trên khiến Elasticsearch là lựa chọn không thể thay thế trong đồ án này.

Chương này đã trình bày chi tiết về các công nghệ quan trọng được áp dụng vào hệ thống của đồ án này gồm Elasticsearch, Redis, FastAPI, LambdaMART ranking. Chương tiếp theo sẽ trình bày các kết quả từ quá trình thực nghiệm và đánh giá kết quả triển khai của hệ thống.

## CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ

Chương này sẽ trình bày các phương diện thiết kế từ tổng quan đến chi tiết của hệ thống, từ đó đưa ra các kết quả kiểm thử đánh giá, minh họa tính năng khi triển khai trong thực tế.

## Thiết kế kiến trúc

## Lựa chọn kiến trúc phần mềm

Hệ thống trong đồ án này sử dụng kiến trúc Microservices. Mẫu kiến trúc này là cách thức phát triển phần mềm như một bộ các service có thể triển khai và chạy độc lập được. Chúng tách biệt về mặt mã nguồn, về hoạt động và dữ liệu. Do các microservice là độc lập, chúng không giao tiếp trực tiếp với nhau mà qua một thành phần trung gian được gọi là API gateway. Mô hình này có các đặc trưng về tính độc lập, tính chuyên biệt và khả năng phòng chống lỗi, khả năng mở rộng. Trong đồ án này với một hệ thống được triển khai theo hướng backend nhằm trở thành công cụ tích hợp vào các hệ thống tương tác trực tiếp với người dùng (ví dụ thực tế là web service Ngóng tin Facebook) yêu cầu phải có nhiều service khác nhau có thể dùng chung hoặc riêng cơ sở dữ liệu nhưng phải tách biệt về mặt chức năng nghiệp vụ, có thể mở rộng bằng cách thêm vào các service mới hoặc loại bỏ một service sẵn có mà không làm ảnh hưởng các luồng công việc của hệ thống chung. Đây là ưu điểm lớn nhất của Microservices nếu so với mô hình kiến trúc cổ điển khá phổ biến là thiết kế nguyên khối vốn không phù hợp với hệ thống mang tính dịch vụ và cần tính linh hoạt như hệ thống trong đồ án này. Cụ thể các micro service trong hệ thống này có thể kể đến: (i) Service quản lý cấu hình hệ thống qua API: Được triển khai bằng các module sql\_interactive, es\_interactive, dữ liệu sau khi phân tích yêu cầu API sẽ được các module quản lý dữ liệu trên xử lý. Do thông tin cấu hình hệ thống lưu ở vùng dữ liệu riêng nên service này chủ yếu tương tác với cơ sở dữ liệu này. Service có thể nâng cấp hoặc loại bỏ độc lập. Các nhiệm vụ chính mà service này đảm nhiệm là quản lý thời gian chạy CrawlerBot, quản lý trạng thái các fanpage trong database (ii) Service thu thập phân tích dữ liệu nhóm kín Facebook: Triển khai bằng package Private\_FB với cơ chế thu thập xử lý dữ liệu qua CrawlerBot chỉ dành riêng cho nhóm kín hoàn toàn khác với cơ chế cho fanpage. Luồng dữ liệu từ service này sẽ đi vào và lưu trữ tại các cơ sở dữ liệu tương tự những service thu thập phân tích dữ liệu từ Facebook khác. Service này chỉ được tích hợp vào hệ thống đích cho số ít các yêu cầu đặc biệt do đặc thù về tính riêng tư và tài nguyên sử dụng. (iii) Service thu thập phân tích dữ liệu fanpage Facebook: Triển khai bằng package FacebookCrawler, luồng dữ liệu có đích đến như service nhóm kín Facebook, kết nối với cơ sở dữ liệu nhằm quản lý hàng đợi các nhiệm vụ crawler cũng như cấu hình các fanpage, user. (iv) Service truy vấn Elasticsearch: có chức năng tạo kênh giao tiếp giữa hệ thống với Elasticsearch để nhận , trả dữ liệu cho các truy vấn từ API. Đây là dịch vụ giúp chuyển đổi yêu cầu từ web service đang tích hợp hệ thống, biến hệ thống thành phần backend xử lý và quản lý lưu trữ dữ liệu gián tiếp. Module phục vụ service này là ES\_utils\_query, thay thế cho việc tương tác trực tiếp với các kho dữ liệu của hệ thống, đảm bảo tính toàn vẹn và che dấu thông tin hệ thống. (v) Các Service xử lý phân tích dữ liệu: Gồm các module chuyên biệt xử lý dữ liệu văn bản, chuỗi ký tự được gắn vào những giai đoạn xử lý dữ liệu cho việc lưu trữ hay kết xuất như module xếp hạng kết quả ranking\_response, module phân tích thực thể ner\_entity\_extracting, module xử lý văn bản lưu trữ định dạng JSON json\_interaction. Các module này có thể dễ dàng thêm vào trong luồng xử lý chính, có vị trí và tần suất xuất hiện khác nhau bởi yêu cầu khác nhau về dữ liệu trong từng giai đoạn của hệ thống. Các dữ liệu được thu thập từ những nguồn khác nếu được yêu cầu xử lý phân tích cũng có thể sử dụng service này như một tiện ích phụ trợ thông qua API. Cấu hình và cơ chế xử lý của các tác vụ trong service này cũng có thể thay đổi theo yêu cầu riêng như ví dụ trong thực tế triển khai hệ thống: Thay vì sử dụng module phân tích thực thể NER, để so sánh và đánh giá các phương án xử lý dữ liệu em còn dùng thêm module phân tích từ khóa chính keyphrase\_extracting ở phần backend cho API xử lý.

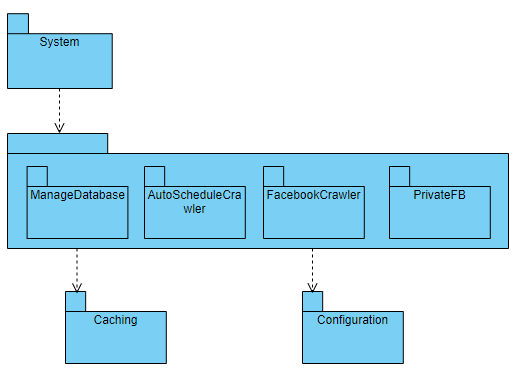
Đối với phần các API gateway kết nối các service, em sử dụng framework FastAPI đã nêu ở chương 3, với tiêu chuẩn của RESTful API để tạo các kênh giao tiếp với các tính năng xác thực bảo mật OAuth2 bằng khóa bí mật, hỗ trợ truyền tham số rỗng, tùy chọn địa chỉ IP và Port. Có một số cấu hình API sẽ được tạo mặc định bởi quản trị viên và những cấu hình khác có thể thay đổi trong file cấu hình bởi người dùng service.

Kiến trúc Microservice em sử dụng cho hệ thống này có một vài điểm thay đổi với mô hình Microservice nguyên bản có thể trình bày qua bảng sau:

| Đặc điểm | Kiến trúc hệ thống hiện tại | Kiến trúc Microservice hiện tại |
| --- | --- | --- |
| Tính độc lập giữa các service | Một số micro-service như service xử lý dữ liệu và service thu thập dữ liệu tuy tách biệt hoàn toàn về luồng vận hành nhưng bắt buộc phải kết hợp với nhau để thu được kết quả cuối cùng theo yêu cầu của hệ thông; ngoài ra chúng vẫn có thể sử dụng để chạy độc lập theo yêu cầu; một số khác sử dụng cùng một nguồn dữ liệu khi cần | Các micro-service hoàn toàn tách biệt nhau và không cần (hoặc không có khả năng) kết hợp chung, chúng chỉ có cùng kết nối chung với phần trung tâm xử lý (server) |
| Khả năng mở rộng | Các service mới có thể tích hợp trực tiếp vào các service có sẵn qua cổng kết nối API luôn mở | Service tích hợp cần đi qua phần xử lý, quản lý của server trung tâm |

## Thiết kế gói tổng quan

Mục này trình bày biểu đồ thiết kế gói tổng quan bao trùm toàn bộ hệ thống, gồm 4 package con phụ thuộc có từng chức năng riêng sẽ nêu ở mục 4.1.3 bên dưới và các gói cơ bản phụ trợ chức năng caching, cấu hình.

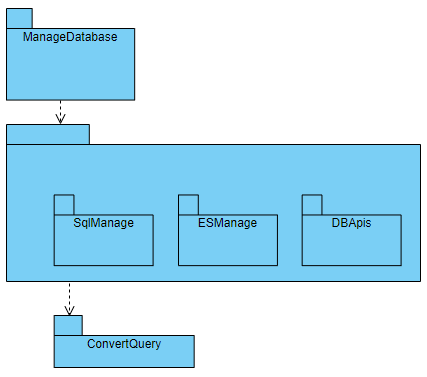


**Biểu đồ 4.1.2:** Thiết kế gói tổng quan toàn hệ thống

## Thiết kế chi tiết gói

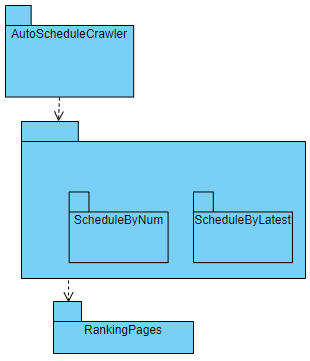
Mục này trình bày biểu đồ thiết kế cho các nhóm package giải quyết các nhiệm vụ cơ bản trong hệ thống cùng với giải thích về các thiết kế.

Package ManageDatabase: Bao gồm ba package con phụ thuộc gồm SqlManage, ESManage, DBApis với các module có nhiệm vụ tạo khung quản lý các cơ sở dữ liệu thuộc SQL và Elasticsearch mà hệ thống sử dụng thông qua các API chứa tham số, logging các sự kiện chính xảy ra trong cơ sở dữ liệu như thêm/xóa/sửa. Package con cơ bản thấp nhất là ConvertQuery có nhiệm vụ chuyển đổi các yêu cầu API thành các truy vấn database và ngược lại.

****

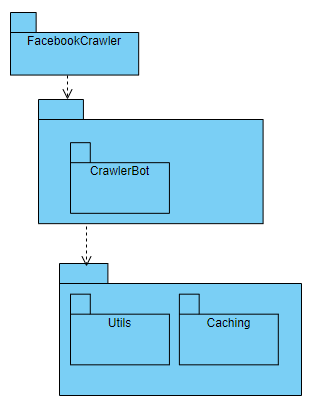
**Biểu đồ 4.1.3.a:** Thiết kế package ManageDatabase

Package AutoScheduleCrawler: Bao gồm các module có nhiệm vụ tự động tạo lịch quét và thu thập xử lý dữ liệu các fanpage, group có trong danh sách cấu hình cho trước. Các package này đều sử dụng kết quả từ tác vụ của gói cơ bản RankingPages để sắp xếp thứ hạng các nguồn dữ liệu theo bài viết.



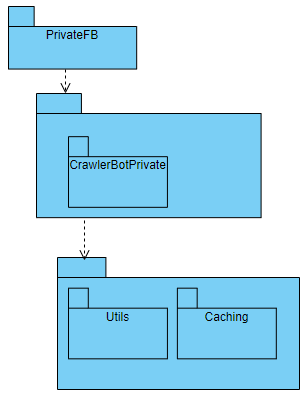
**Biểu đồ 4.1.3.b:** Thiết kế package AutoScheduleCrawler

Package FacebookCrawler: Bao gồm các package con với các chức năng logging, thực thi tác vụ, tiện ích (utils), kiểm thử, cấu hình. Nhiệm vụ chính là quản lý và triển khai Crawler Bot để thu thập xử lý dữ liệu thô từ các fanpage Facebook rồi lưu trữ vào database thông qua package cùng tên, sử dụng các tiện ích và thao tác caching từ các package con.



**Biểu đồ 4.1.3.c:** Thiết kế package FacebookCrawler

Package PrivateFB: Với nhiệm vụ và các package con có chức năng tương tự như FacebookCrawler nhưng nhiệm vụ vủa Crawler Bot trong package này hướng đến dữ liệu từ các group (nhóm) trên Facebook. Cấu trúc thiết kế tương tự như package FacebookCrawler



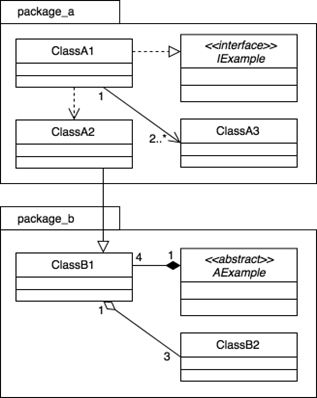
**Biểu đồ 4.1.3.d:** Thiết kế package PrivateFB

Sinh viên thiết kế và lần lượt vẽ biểu đồ thiết kế cho từng package, hoặc một nhóm các package liên quan để giải quyết một vấn đề gì đó. Khi vẽ thiết kế gói, sinh viên chỉ cần đưa tên lớp, không cần chỉ ra các thành viên phương thức và thuộc tính. SV tham khảo ví dụ minh họa trong Hình [4.2.](#_heading=h.4f1mdlm)

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ

Sinh viên cần vẽ rõ ràng quan hệ giữa các lớp trong biểu đồ. Các quan hệ bao gồm: phụ thuộc (dependency), kết hợp (association), kết tập (aggregation), hợp thành (composition), kế thừa (inheritance), và thực thi (implementation). Các quan hệ này đều đã được minh họa trong [4.2.](#_heading=h.4f1mdlm)

Sau khi vẽ hình minh họa, sinh viên cần giải thích ngắn gọn về thiết kế của mình.



**Hình 4.2:** Ví dụ thiết kế gói

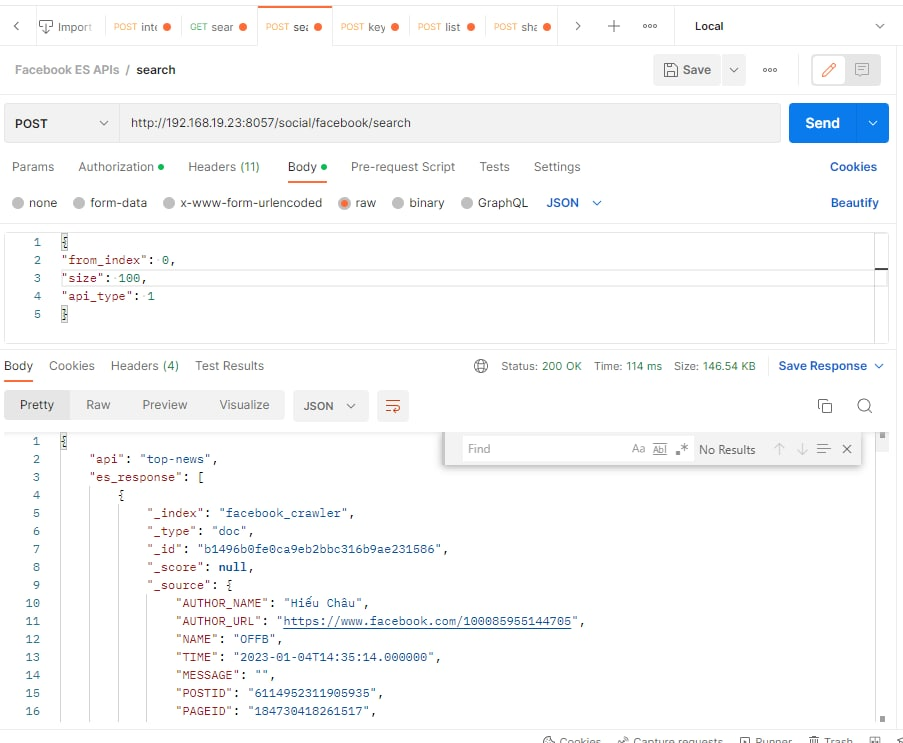
## Thiết kế chi tiết

## Thiết kế giao diện

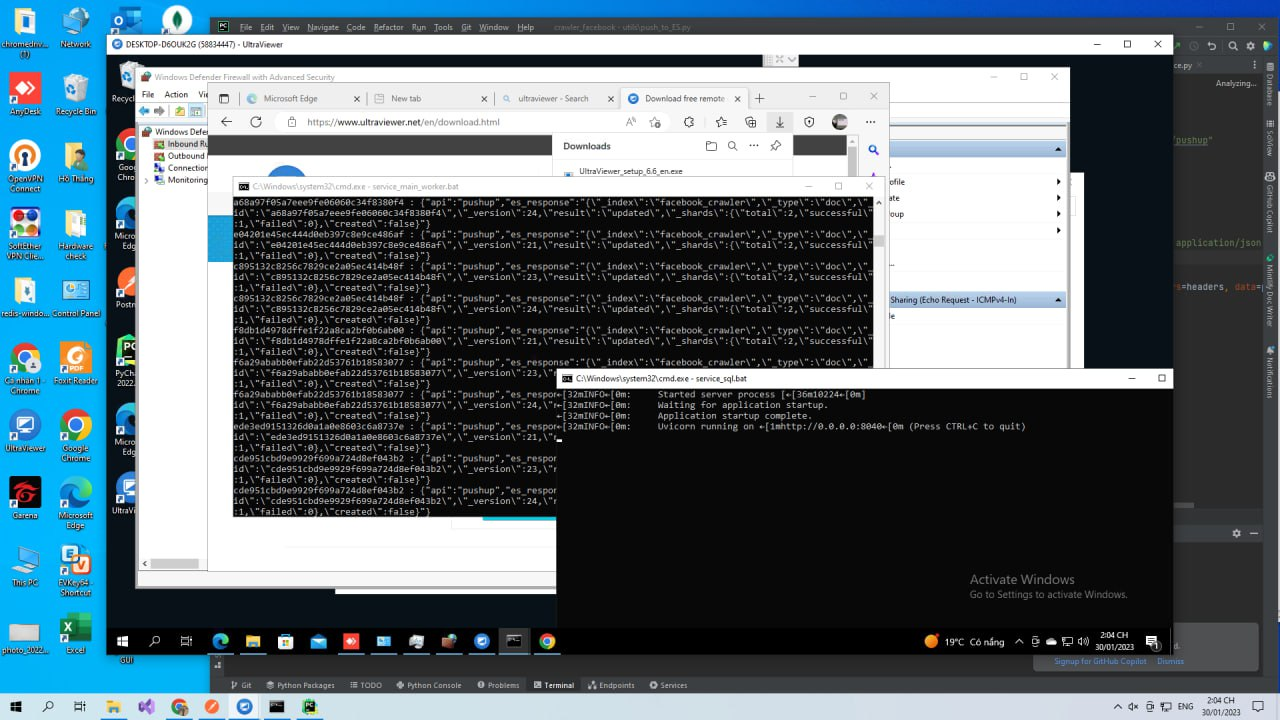
Vì hệ thống trong đồ án này có kiến trúc là một hệ thống gồm các micro-services hoạt động độc lập hoặc được tích hợp với nhau nên giao diện thiết kế sử dụng sẽ chia ra hai thiết kế dành cho quản trị viên/người bảo trì hệ thống và khách hàng là các đối tác có hệ thống tích hợp với hệ thống này.

Với giao diện thiết kế cho quản trị viên/người bảo trì hệ thống sẽ sử dụng giao diện tương tự giao diện khi xây dựng hệ thống, gồm màn hình độ phân giải 1920 x 1080 theo chuẩn màu của màn hình máy tính thông dụng hiện nay. Các công cụ sử dụng để giao tiếp và tương tác hệ thống gồm (i) Giao diện công cụ Postman: Dùng để quản lý và tương tác các API điều khiển và gửi yêu cầu hệ thống. Màn hình Postman có giao diện tương tác UI trực quan, tương tác bằng các thao tác nhấn thả, gõ ký tự trên các thành phần chính như Settings (cài đặt), Collections (lưu trữ thông tin các API hoặc folder), bảng API content (nội dung của API kèm các tham số, thành phần hỗ trợ thao tác API). Chuẩn thống nhất chung về giao diện làm việc với API là thao tác thông qua dạng JSON request nhập vào từ bàn phím theo định dạng raw, thông điệp phản hồi bao gồm tên API và nội dung thông điệp dưới dạng JSON (dictionary theo Python), các API tương tác cùng một đối tượng cần nằm cùng trong một Collection riêng. Minh họa về giao diện tương tác, input và output ở hình 4.2.1.a. (ii) Giao diện tương tác dưới dạng các service định dạng Windows batch file (.bat). Các micro-service được tổ chức và quản lý ở server có dạng Windows batch file để gọi và thực thi theo yêu cầu người quản trị. Các service này được gọi trực tiếp qua thao tác khởi động máy hoặc gọi và tắt qua tương tác trực tiếp, với phần tên của service hiển thị trên cùng và nội dung nhiệm vụ thực thi được logging và in ra màn hình cửa sổ dòng lệnh như hình minh họa 4.2.1.b.

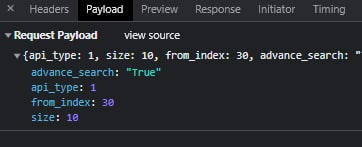
Với giao diện thiết kế cho người dùng sẽ là giao diện gián tiếp tùy vào cách thức tích hợp các service của hệ thống. Chuẩn chung đang sử dụng trong triển khai thực tế là dưới dạng các API Restful truyền vào từ cookie của người dùng, các API này tuân theo định dạng JSON-raw như các API đã nêu ở phần thiết kế của Postman phía trên. Kết quả trả về tương tự với thông điệp trả về ở Postman, có thể tùy chỉnh thông điệp cần lấy phù hợp bởi người dùng. Minh họa console cookie phía người dùng đang tương tác với hệ thống ở hình 4.2.1.c và 4.2.1.d.



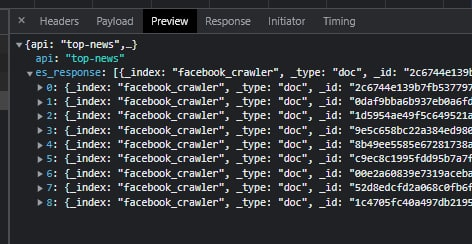
**Hình 4.2.1.a:** Giao diện minh họa Postman và input, output chuẩn cho các API



**Hình 4.2.1.b:** Minh họa giao diện cửa sổ thực thi của các micro service



**Hình 4.2.1.c:** Giao diện console phía người dùng khi gửi request



**Hình 4.2.1.d:** Giao diện console phía người dùng khi nhận phản hồi

## Thiết kế lớp

Mục này trình bày thiết kế chi tiết các thuộc tính (gồm các thuộc tính lớp) và phương thức (không kể đến phương thức khởi tạo) thuộc một số lớp chủ đạo tham gia vào quá trình chính của hệ thống. Các lớp này thuộc các package lớn đã nêu ở mục 4.1.3, gồm các lớp sau: (i) Lớp CrawlerBot: Thuộc package FacebookCrawler, nằm trong module *crawler\_bot.py* là lớp đại diện cho đối tượng CrawlerBot có nhiệm vụ tương tác trực tiếp với cookie dữ liệu Facebook để thu thập các thông tin liên quan của một bài viết từ một nguồn cấp dữ liệu công khai (fanpage, nhóm công khai). Các thuộc tính gồm có *pageurl* là địa chỉ đầy đủ của nguồn cấp có kiểu string; *checking\_req* là thuộc tính kiểm tra điều kiện đầu vào của nguồn cấo có kiểu bool. Các phương thức gồm các instance method như *get\_cookieid*, *get\_pageid* có nhiệm vụ lấy các dữ liệu liên quan đến ID cookie, headers và ID đối tượng; *parsing\_edge*, *parsing\_ProfileComet*, *parsing\_CometModern* có nhiệm vụ truy xuất thông tin như nội dung, các liên kết từ dữ liệu phản hồi thu được nhờ các request được gửi để thu thập cookie; *extract\_reaction* trích xuất thông tin về số lượng các tương tác cảm xúc, *Crawl\_PagePosts* và *Crawl\_GroupPosts* có nhiệm vụ kết hợp các phương thức chuyên biệt kể trên vào quá trình thu thập dữ liệu của trang và nhóm, nhóm phương thức tĩnh gồm *get\_pageid\_to\_check* và *caching* có nhiệm vụ kiểm tra tính sẵn có pageurl của đối tượng trong bộ nhớ cache và hỗ trợ caching dữ liệu để tái sử dụng. Minh họa về luồng truyền thông điệp giữa các đối tượng trong hoạt động của lớp CrawlerBot thể hiện ở biểu đồ 4.2.2.a. (ii) Lớp SQL\_accounts: Thuộc package ManageDatabase, nằm trong module *manage\_sql\_database.py*, có nhiệm vụ cung cấp các tương tác gián tiếp để truy cập vào các bảng dữ liệu trong SQL Servers như DataLake, dữ liệu về các fanpage, dữ liệu quản trị viên, dữ liệu người dùng phục vụ cho mục đích quản lý hệ thống hoặc lưu trữ, truy xuất, chỉnh sửa dữ liệu thô. Các thuộc tính gồm có *server, host, database, user, password* lưu giữ thông tin truy cập vào cơ sở dữ liệu; *columns* đại diện cho danh sách các cột mà dữ liệu thể hiện khi đưa vào SQL; *filepath* là địa chỉ file lưu dữ liệu tạm thời trong bộ nhớ hệ thống; *pag\_table, acc\_table* là tên định dang của các bảng dữ liệu riêng. Các phương thức instance method gồm *interact\_datalake, interact\_pages, interact\_accounts* có nhiệm vụ gần tương tự là tương tác với các bảng dữ liệu được thiết kế riêng với các tính năng thao tác với dữ liệu như cập nhật, chèn, xóa; *read\_data\_from\_table* đọc dữ liệu từ bảng được chọn, các phương thức tĩnh gồm *check\_active* kiểm tra trạng thái của đối tượng trong một bảng cho trước; một số phương thức tĩnh khác có vai trò phụ trợ nhằm thao tác với dữ liệu JSON và nhập xuất dữ liệu từ SQL. Các phương thức instance method sử dụng một vài phương thức khác trong cùng lớp để tái sử dụng chức năng như *interact\_pages* sử dụng *check\_active* để lấy thông tin từ những trang nhóm có trạng thái active. Minh họa về luồng truyền thông điệp giữa các đối tượng trong hoạt động của lớp SQL\_accounts thể hiện ở biểu đồ 4.2.2.b. (iii) Lớp ESAPIManager: Thuộc package lớn ManageDatabase, package cha trực tiếp là DBApis, nằm trong module integrated\_es\_apis: Nhiệm vụ triển khai và quản lý các API tương tác với dữ liệu trong Elasticsearch, phục vụ trực tiếp cho quá trình trả kết quả đầu ra của hệ thống. Các thuộc tính gồm có *port, ip* đại diện cho địa chỉ kết nối đến Elasticsearch, *config* lưu giữ thông tin cấu hình API. Các phương thức lần lượt là các hàm chức năng API như *search\_es, top\_news\_es, top\_keywords\_es, shared\_posts\_es, info\_es* có nhiệm vụ tạo lập cơ chế hoạt động của API theo tên gọi và một số phương thức phụ trợ tương tác với Elasticsearch như *convert\_request* để chuyển đổi yêu cầu API thành truy vấn Elasticsearch. Các phương thức triển khai API dưới dạng các hàm xử lý bất đồng bộ theo định dạng request function của FastAPI để phục vụ quá trình nhận gửi request và được triển khai cùng một địa chỉ ip và port. Mỗi phương thức có một chuẩn tham số đầu vào riêng, hỗ trợ truyền vào các tham số rỗng và null cũng như có sẵn các đầu ra mặc định trả về cho các trường hợp lỗi đầu vào. Minh họa về luồng truyền thông điệp giữa các đối tượng trong hoạt động của lớp ESAPIManager thể hiện ở biểu đồ 4.2.2.c.

Phần này có độ dài từ ba đến bốn trang. Sinh viên trình bày thiết kế chi tiết các thuộc tính và phương thức cho một số lớp chủ đạo/quan trọng nhất của ứng dụng

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ

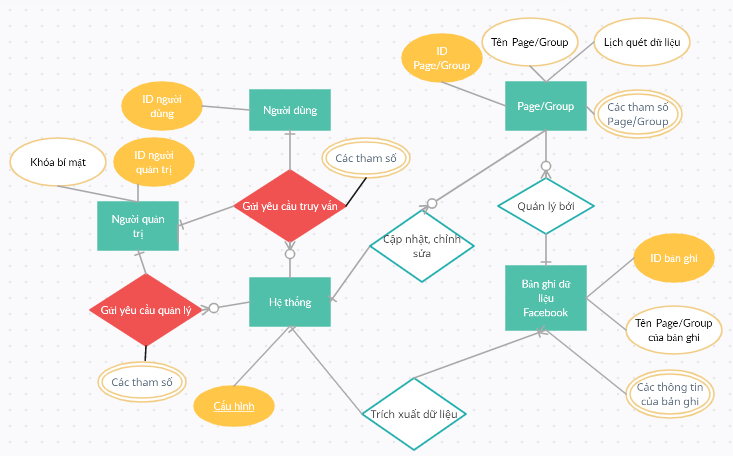
(từ 2-4 lớp). Thiết kế chi tiết cho các lớp khác, nếu muốn trình bày, sinh viên đưa vào phần phụ lục.

Để minh họa thiết kế lớp, sinh viên thiết kế luồng truyền thông điệp giữa các đối tượng tham gia cho 2 đến 3 use case quan trọng nào đó bằng biểu đồ trình tự (hoặc biểu đồ giao tiếp).

## Thiết kế cơ sở dữ liệu

Mục này trình bày mô hình thực thể liên kết thể hiện các đối tượng thành phần của hệ thống trong thực tế sử dụng và mối quan hệ giữa chúng, từ đó đưa ra bản thiết kế các cơ sở dữ liệu.

Mô hình thực thể liên kết của các đối tượng trong hệ thống thực tế:



**Hình 4.2.3.1:** Mô hình thực thể liên kết của các đối tượng trong hệ thống thực tế

Giải thích biểu đồ thực thể liên kết: Gồm các thực thể “Người quản trị”, “Người dùng”, “Hệ thống”, “Bản ghi dữ liệu Facebook”, “Page/Group”. (i) Thực thể “Người quản trị” có các thuộc tính ID, khóa bí mật và có quan hệ quản lý cấu hình hệ thống thông qua yêu cầu bằng API. Các thuộc tính được lưu trữ trong table “Admins” của CSDL SQL Server. (ii) Thực thể “Người dùng” có thuộc tính ID người dùng, có quan hệ gửi yêu cầu truy vấn với “Hệ thống” qua API. (iii) Thực thể “Hệ thống” có thuộc tính cấu hình hệ thống là thực thể trung tâm xử lý các quan hệ với “Người quản trị”, “Người dùng” thông qua thực hiện các yêu cầu bằng quan hệ trích xuất dữ liệu từ các “Bản ghi dữ liệu Facebook” và cập nhật, chỉnh sửa thông tin cấu hình của “Page,Group”. (iv) Thực thể “Bản ghi dữ liệu Facebook” là các đơn vị dữ liệu cơ bản của hệ thống dưới dạng các bài viết, thuộc và được quản lý bởi thực thể “Page/Group”, gồm các thuộc tính ID, các thông tin dữ liệu bản ghi, Page/Group quản lý. Các đối tượng thuộc thực thể này có nhiệm vụ trả dữ liệu theo yêu cầu từ “Hệ thống”. (v) Thực thể “Page/Group” là các fanpage, group Facebook có thông tin dữ liệu trong hệ thống. Gồm các thuộc tính ID, tên, lịch quét, các tham số khác như trạng thái active/inactive. Các đối tượng của nó quản lý các bản ghi Facebook phụ thuộc theo ID và tên Page/Group và được quản lý bởi “Hệ thống”.

Từ mô hình biểu đồ liên kết thực thể đã xây dựng, em đã thiết kế hệ cơ sở dữ liệu cho hệ thống bao gồm hai loại cơ sở dữ liệu có quan hệ (SQL Server) và phi quan hệ (Elasticsearch). CSDL SQL Server đảm nhiệm vai trò lưu trữ các table chứa thông tin về các thực thể “Page/Group”, “Người quản trị”, “Người dùng”, “Hệ thống” và dữ liệu thô của thực thể “Bản ghi dữ liệu Facebook” dưới dạng một DataLake. Với các yêu cầu cần ưu tiên cho việc lưu trữ số lượng lớn và có các quan hệ SQL qua lại giữa các table, có các trường dữ liệu dùng chung, yêu cầu quản lý theo khóa ngoài như ID page/group, yêu cầu về tính cập nhật dữ liệu đồng thời giữa các thực thể đã dẫn đến lựa chọn sử dụng CSDL SQL. Ngoài ra dữ liệu từ DataLake của SQL Server có thể dễ dàng trích xuất theo các trường dữ liệu phục vụ cho quá trình biến đổi làm giàu dữ liệu. Elasticsearch được sử dụng để lưu trữ và tìm kiếm dữ liệu đã được biến đổi từ DataLake. Elasticsearch có thế mạnh về khả năng tìm kiếm nhanh chóng và dễ dàng truy vấn các dữ liệu văn bản để sử dụng như một công cụ backend cho việc tìm kiếm thông tin được truy vấn; khả năng mở rộng và cập nhật đồng thời các document (đơn vị lưu trữ cơ bản của Elasticsearch) dễ dàng giúp người thiết kế và nâng cấp hệ thống có nhiều lựa chọn trong việc mở rộng các trường dữ liệu theo dạng key-value, tăng tính tương thích của dữ liệu theo thời gian. Dưới đây là bảng thiết kế và quan hệ giữa các CSDL trong hệ thống.

Phần này có độ dài từ hai đến bốn trang. Sinh viên thiết kế, vẽ và giải thích biểu đồ thực thể liên kết (E-R diagram). Từ đó, sinh viên thiết kế cơ sở dữ liệu tùy theo hệ quản trị cơ sở dữ liệu mà mình sử dụng (SQL, NoSQL, Firebase, v.v.)

## Xây dựng ứng dụng

## Thư viện và công cụ sử dụng

Các công cụ sử dụng:

| **Mục đích** | **Công cụ** | **Địa chỉ URL** |
| --- | --- | --- |
| IDE lập trình | PyCharm 2022.3.1 | https://www.jetbrains.com/pycharm/ |
| Thiết kế, triển khai, kiểm thử các yêu cầu API | Postman Windows 64 bit | https://www.postman.com/ |
| Lưu trữ, quản lý, chia sẻ mã nguồn | Git | https://git-scm.com/ |
| Quản lý, thao tác với cơ sở dữ liệu SQL | Azure data studio | https://azure.microsoft.com/en-us/products/data-studio/ |
| Tạo kết nối VPN | OpenVPN | https://openvpn.net/ |

**Bảng 4.3.1.a:** Danh sách các công cụ sử dụng

Ngôn ngữ lập trình, các thư viện chính và API sử dụng:

| **Mục đích** | **Ngôn ngữ/thư viện/API** | **Địa chỉ URL** |
| --- | --- | --- |
| Ngôn ngữ lập trình chính | Python | https://www.jetbrains.com/pycharm/ |
| Framework phát triển các API RESTful | FastAPI | https://fastapi.tiangolo.com/ |
| Triển khai web server cho các giao tiếp API | Uvicorn | https://www.uvicorn.org/ |
| Gửi các yêu cầu HTTP trong quá trình crawl dữ liệu cũng như giao tiếp API | Requests | https://pypi.org/project/requests/ |
| Kết nối và tương tác với cơ sở dữ liệu SQL Server | pymssql | https://pypi.org/project/pymssql/ |
| Tạo và quản lý hàng đợi, kết nối tới Redis | rq | https://github.com/rq/rq |
| Phân tích cú pháp HTML và XML của trang web | beautifulsoup4 | https://pypi.org/project/beautifulsoup4/ |
| Triển khai các thuật toán học máy cho nhiệm vụ xếp hạng | xgboost | https://xgboost.readthedocs.io/en/stable/ |

**Bảng 4.3.1.2:** Danh sách ngôn ngữ lập trình, các thư viện và API sử dụng

## Kết quả đạt được

Kết quả thực tế đã đạt được là hệ thống hoàn chỉnh gồm có đủ các chức năng, đáp ứng các yêu cầu cốt lõi đã được vạch ra ở chương 1 và chương 2. Hệ thống đã được ứng dụng thực tế cho dịch vụ web Ngóng tin Facebook nằm trong hệ sinh thái web service về tin tức và mạng xã hội ở công ty VCCorp. Hệ thống cung cấp nguồn dữ liệu có giá trị với tính ổn định cho việc lưu trữ vào cơ sở dữ liệu; tạo phương thức tìm kiếm và đánh giá các thông tin theo bộ lọc được yêu cầu và trả thông tin qua các API tích hợp. Hệ thống hiện tại bao gồm các service đã đóng gói và triển khai thực tế sau (Lưu ý các service triển khai thực tế có thể là các micro-service thiết kế cơ bản đã nêu ở mục 4.1.1 hoặc là được tích hợp từ một số micro-service này):

| **Service** | **Chức năng** | **Các thành phần** |
| --- | --- | --- |
| Facebook page crawler | Thu thập, xử lý, lưu trữ dữ liệu từ các fanpage Facebook theo cấu hình | Crawler Bot, công cụ lập lịch thu thập, cơ sở dữ liệu, cache, API quản lý |
| Facebook group crawler | Thu thập, xử lý, lưu trữ dữ liệu từ các group Facebook theo cấu hình | Crawler Bot, công cụ lập lịch thu thập, cơ sở dữ liệu, cache, API quản lý |
| Enhance data factors | Phân tích, làm giàu dữ liệu thô; trích xuất các thông tin quan trọng; xếp hạng kết quả trả về | Công cụ phân tích thực thể NER, công cụ xếp hạng bởi LambdaMART, API tương tác, cơ sở dữ liệu lưu trữ kết quả |
| Scheduler | Tự động tạo lịch quét dữ liệu cho Crawler Bot theo các chế độ cài đặt | Module theo dõi trạng thái dữ liệu các đối tượng trong cơ sở dữ liệu, bộ lập lịch, API quản lý lịch, cơ sở dữ liệu lưu trữ lịch quét |
| System response APIs | Các API có nhiệm vụ thực hiện yêu cầu cấu hình từ phía người dùng hoặc trả về kết quả cho các yêu cầu truy vấn | Các API trả dữ liệu, API quản lý chung hệ thống, bộ chuyển đổi yêu cầu API, công cụ tìm kiếm Elasticsearch, module quản lý hệ thống |
|  |  |  |

**Bảng 4.3.2.a:** Các service đã đóng gói và triển khai thực tế

Thống kê các thông số về mã nguồn toàn bộ hệ thống và các service chính:

| **Tên thông số** | **Số lượng** | **Đơn vị** |
| --- | --- | --- |
| Số dòng code (chỉ tính trong các file định dạng ngôn ngữ lập trình như .py) | 15336 | Dòng |
| Số lớp (class) | 59 | Lớp |
| Số gói | 34 | Gói |
| Dung lượng tổng | 323902 | kB |
| Dung lượng service Facebook page crawler | 124858 | kB |
| Dung lượng service Facebook group crawler | 147100 | kB |
| Dung lượng dữ liệu hiện có trong cơ sở dữ liệu SQL | 4275 | MB |
| Dung lượng dữ liệu hiện có trong Elasticsearch | 920 | MB |
| Tổng số fanpage và group Facebook hệ thống đang chạy quét | 193 | Trang/Nhóm |

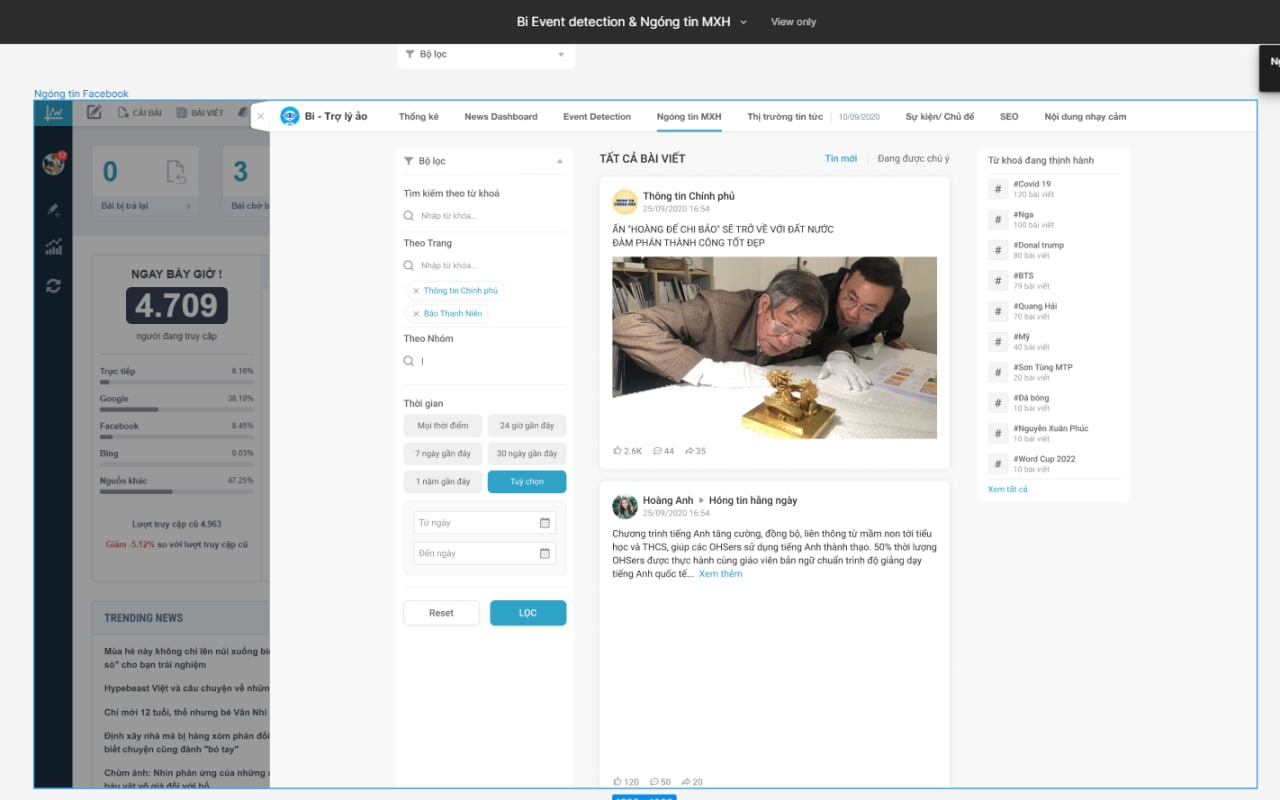
**Bảng 4.3.2.b:** Các thông số về mã nguồn và service

Sinh viên trước tiên mô tả kết quả đạt được của mình là gì, ví dụ như các sản phẩm được đóng gói là gì, bao gồm những thành phần nào, ý nghĩa, vai trò?

Sinh viên cần thống kê các thông tin về ứng dụng của mình như: số dòng code, số lớp, số gói, dung lượng toàn bộ mã nguồn, dung lượng của từng sản phẩm đóng gói, v.v. Tương tự như phần liệt kê về công cụ sử dụng, sinh viên cũng nên dùng bảng để mô tả phần thông tin thống kê này.

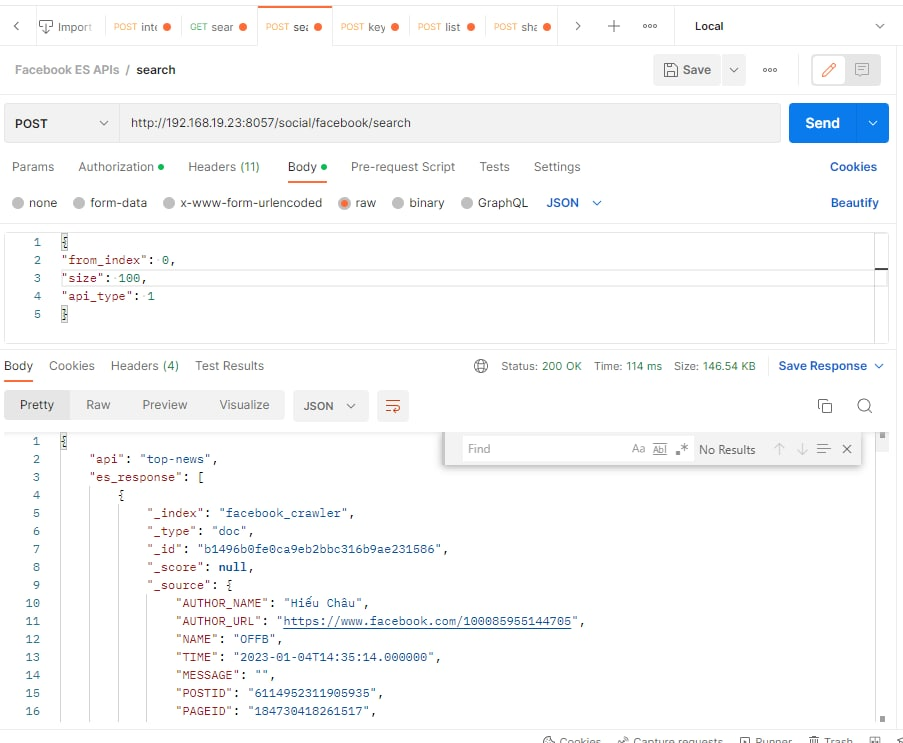
## Minh họa các chức năng chính

* Giao diện dashboard Ngóng tin Facebook với phần dữ liệu và API backend tích hợp từ hệ thống (các chức năng chính tìm kiếm theo truy vấn và xếp hạng kết quả trả về, hiển thị top từ khóa trong 24h, newfeed gồm các bài viết đã được xếp hạng theo thời gian)



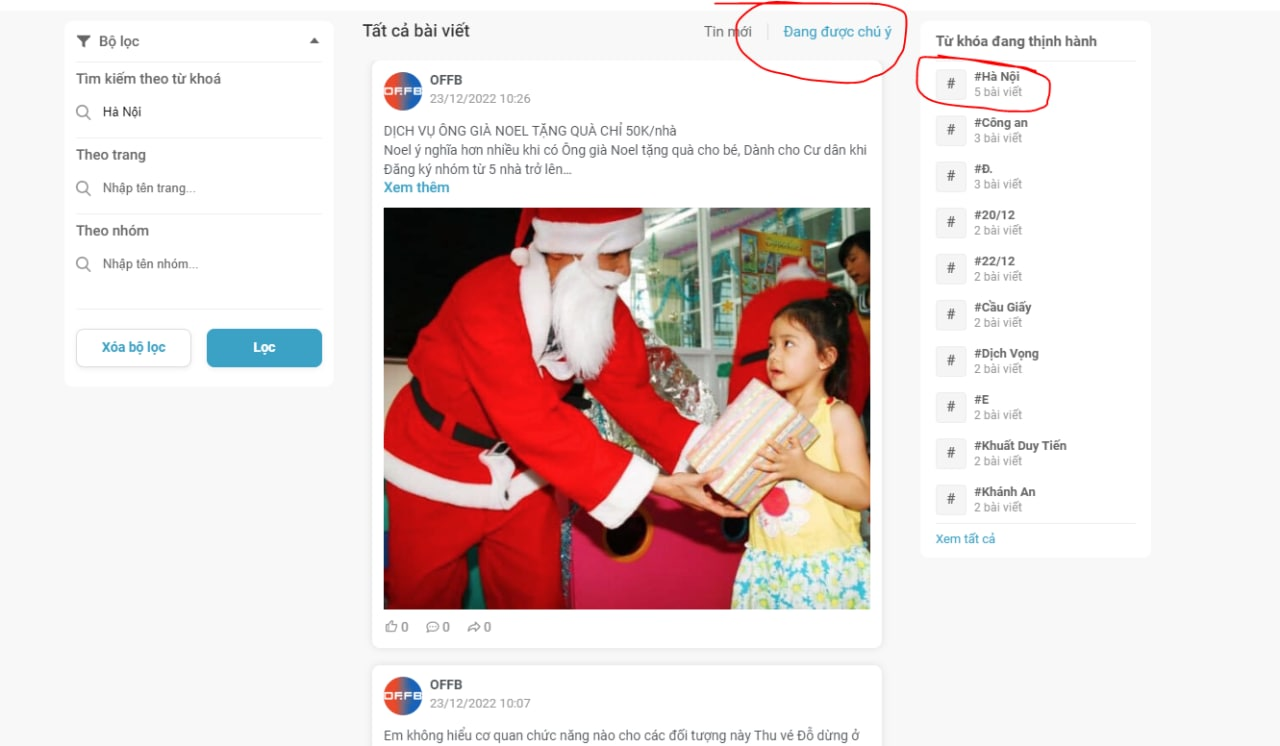
**Hình 4.3.3.a:** Giao diện dashboard Ngóng tin MXH Facecbook

* Truy vấn theo từ khóa qua API bằng công cụ Postman (dành cho quản trị viên và tester):



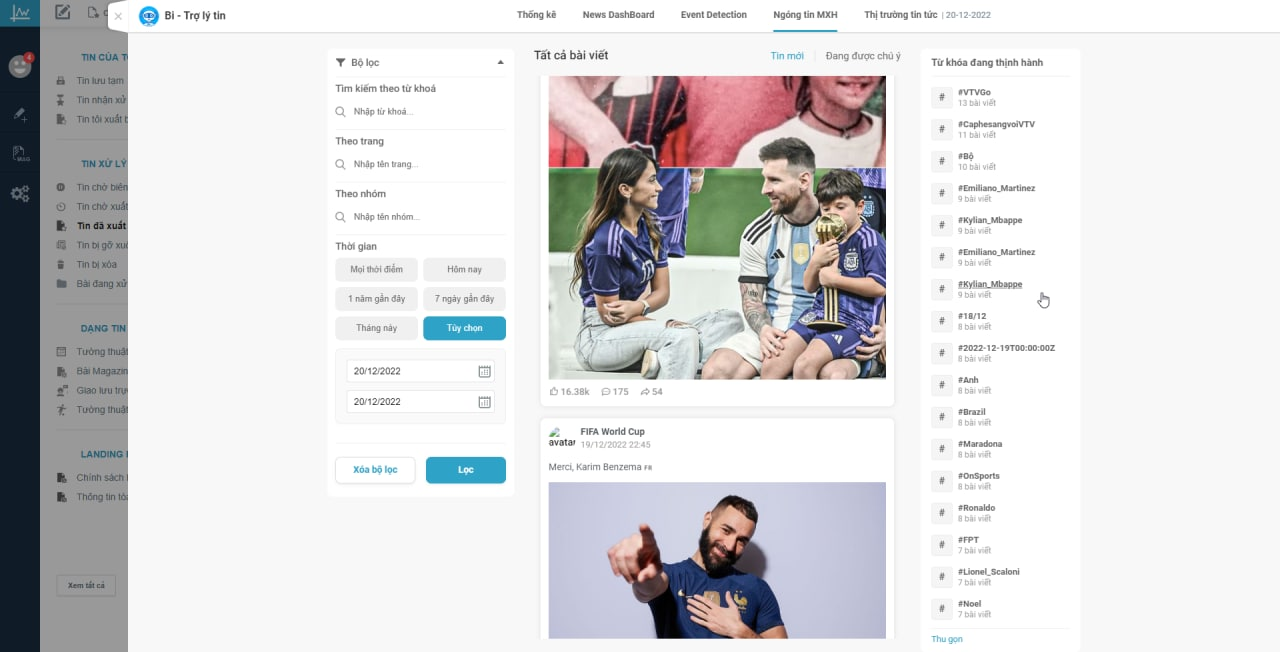
**Hình 4.3.3.b:** Giao diện truy vấn theo từ khóa qua API sử dụng Postman

* Kết quả tìm kiếm cho từ khóa Hà Nội (có hai chế độ tìm kiếm theo từ khóa đã phân tích và tìm kiếm đầy đủ văn bản) hiển thị ở khung giữa; Top các từ khóa đang thịnh hành gần đây ở khung bên phải:



**Hình 4.3.3.c:** Giao diện hiển thị kết quả tìm kiếm theo từ khóa “Hà Nội” và các từ khóa thịnh hành

* Kết quả tìm kiếm theo bộ lọc tìm kiếm theo thời gian:



**Hình 4.3.3.d:** Giao diện kết quả tìm kiếm theo bộ lọc thời gian

Sinh viên lựa chọn và đưa ra màn hình cho các chức năng chính, quan trọng, và thú vị nhất. Mỗi giao diện cần phải có lời giải thích ngắn gọn. Khi giải thích, sinh viên có thể kết hợp với các chú thích ở trong hình ảnh giao diện.

## Kiểm thử

Mục này trình bày những trường hợp kiểm thử từ các kịch bản kiểm thử đã trải qua thực tế đánh giá từ đội ngũ tester và người dùng thử nghiệm. (i) Kiểm thử về khả năng thực hiện truy vấn tại giao diện sử dụng thực tế: Sử dụng phương pháp kiểm thử hộp đen mà cụ thể là phương pháp phân tích giá trị biên kết hợp phân tích phân vùng tương đương. Thao tác kiểm thử kết hợp kiểm thử khả năng hoạt động của các API với khả năng tương tác truy vấn dữ liệu qua API và giao diện, thực hiện ngay trên giao diện công cụ Ngóng tin Facebook, đi từ việc nhập các giá trị truy vấn (dưới dạng các kí tự, từ khóa) vào thanh tìm kiếm, có thể kết hợp sử dụng bộ lọc thông tin rồi rút ra đánh giá từ các kết quả trả về. Với kỹ thuật phân tích giá trị biên, kiểm thử viên (tester) tiến hành chọn ra các giá trị biên từ các thuộc tính có tính indexing (sắp xếp theo thứ tự) đặc biệt như size (số lượng kết quả trả về) lớn nhất, size nhỏ nhất, size âm, size rất lớn; paging (phân trang kết quả theo chỉ số trong danh sách kết quả sẵn có của kết quả đầu tiên trả về. VD: paging=5 sẽ chọn kết quả từ vị trí thứ 5 trong danh sách xếp hạng các kết quả sẵn có) với các giá trị biên tương tự như size; time\_create (thời gian tạo bài viết). Với kỹ thuật phân vùng tương đương, tester chọn ra các giá trị đại diện các vùng giá trị khác nhau bao quát các trường hợp có thể của tham số nhập vào máy tìm kiếm với hai phân loại là các lớp giá trị hợp lệ và không hợp lệ. Trình tự kiểm thử của phương án này như sau: Liên tục gán các giá trị cho các lớp tương đương hợp lệ cho đến khi tất cả các lớp tương đương hợp lệ được bao phủ bởi các ca kiểm thử, viết 1 ca kiểm thử mới bao phủ càng nhiều các lớp tương đương đó mà chưa được bao phủ càng tốt. Tương tự như vậy cho đến khi các ca kiểm thử đã bao phủ tất cả các lớp tương đương không hợp lệ, viết 1 ca kiểm thử mà bao phủ một và chỉ một trong các lớp tương đương không hợp lệ chưa được bao phủ. Lý do mà mỗi ca kiểm thử riêng bao phủ các trường hợp không hợp lệ là vì các kiểm tra đầu vào không đúng nào đó che giấu hoặc thay thế các kiểm tra đầu vào không đúng khác. Ví dụ minh họa kiểm thử này là chọn ra các lớp tương đương cho giá trị của size với số lượng kết quả có sẵn giả thuyết đã được xác định và giữ nguyên, gồm: Các giá trị hợp lệ, các giá trị không hợp lệ âm (size yêu cầu giá trị dương), các giá trị hợp lệ lớn hơn size thực tế, các giá trị không có dạng số (như null, rỗng, ký tự).Áp dụng các bước đã nêu trên để tiến hành kiểm thử. (ii) Kiểm thử về chức năng xếp hạng kết quả: Thực hiện nhằm đánh giá các nhiệm vụ xếp hạng như trả về kết quả truy vấn theo các mức độ liên quan phù hợp, thời gian, số lượt tương tác; đưa ra các bảng xếp hạng từ khóa thịnh hành, bài viết nổi bật. Vì yêu cầu kiểm thử này đi sâu hơn vào dữ liệu giao tiếp giữa phần front-end và back-end nên cần sử dụng các phương pháp kiểm thử hộp xám như kiểm thử hồi quy nhằm đánh giá lại chức năng sẵn có khi chức năng mới được thêm vào; kiểm thử mẫu để xác định các lỗi còn tồn tại. Quy trình kiểm thử này cần phối hợp giữa quản trị viên hệ thống và tester để truy cập một phần vào mã nguồn và cơ sở dữ liệu. Trình tự khái quát sẽ có các bước: Tester dựa trên đặc tả chức năng và cơ chế tính năng xếp hạng tạo các input từ phía front-end qua các API giao tiếp được cung cấp sẵn bởi service API (bảng 4.3.2.a) gửi đến hệ thống rồi nhận dữ liệu trả về. Đánh giá sẽ dựa trên độ chính xác trong thứ tự các kết quả, khả năng xếp hạng khi có thay đổi về dữ liệu tại cơ sở dữ liệu cũng như cơ chế giao tiếp, sự thay đổi kết quả xếp hạng theo thay đổi của yêu cầu đầu vào, các “điểm mù” tồn tại trong xếp hạng như trường dữ liệu cần để xếp hạng bị lỗi, giá trị null. (iii) Kiểm thử phân hệ cấu hình và quản lý hệ thống có xác thực của quản trị viên: Thực hiện nhằm đánh giá khả năng quản lý, cập nhật thay đổi cho cấu hình chung của hệ thống cũng như dữ liệu người dùng được thực hiện bởi quản trị viên; đánh giá tính phân quyền trong quản lý, độ sâu của phân quyền được áp dụng; khả năng hoạt động của module quản lý khi được thực hiện bởi nhiều tác nhân quản trị viên. Tương tự như trường hợp kiểm thử thứ nhất về khả năng thực hiện yêu cầu truy vấn (vì quy trình nghiệp vụ đều có hoạt động bắt buộc là gửi yêu cầu đến hệ thống qua API), kiểm thử thực hiện bởi phương pháp kiểm thử hộp đen; ngoài ra để so sánh với các thay đổi khi quản lý bằng tương tác trực tiếp như truy cập trực tiếp vào file cấu hình, phương pháp kiểm thử hộp xám cũng được áp dụng để kết hợp các ca kiểm thử bề mặt và bản chất hệ thống quản lý. Hoạt động kiểm thử cốt lõi là sử dụng tài khoản quản trị viên với các quyền quản lý khác nhau để thay đổi các thông tin cấu hình và so sánh phản hồi, khả năng thực thi của hệ thống với khi sử dụng tài khoản có quyền khác hoặc sử dụng thao tác quản lý trực tiếp đã nêu ở trên.

Sau đây là bảng đánh giá tổng kết kiểm thử:

| **Chức năng được kiểm thử** | **Số test case** | **Kết quả** |
| --- | --- | --- |
| Tìm kiếm truy vấn thông tin tại giao diện sản phẩm thực tế | 40 | Hệ thống hoàn thành tất cả test case trong giới hạn thời gian sử dụng thực tế, không có lỗi và gián đoạn dịch vụ nào được ghi nhận |
| Xếp hạng thông tin kết quả | 30 | Hệ thống hoàn thành tất cả test case trong giới hạn thời gian sử dụng thực tế, không có lỗi và gián đoạn dịch vụ nào được ghi nhận |
| Kiểm thử phân hệ cấu hình và quản lý hệ thống có xác thực của quản trị viên | 30 | Hệ thống đáp ứng tất cả yêu cầu quản lý theo quyền hạn của mỗi quản trị viên, yêu cầu được thực thi và phản hồi trong thời gian cho phép, không có ngoại lệ xảy ra |

**Bảng 4.4.1**: Kết quả tổng kết kiểm thử

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ

## Triển khai

Mục này đề cập đến mô hình và cách thức triển khai thực tế của hệ thống .

Tại môi trường thực tế ở công ty VCCorp, hệ thống đang được thử nghiệm và ứng dụng như một cloud service phục vụ cho việc vận hành hệ thống Ngóng tin Mạng xã hội nhằm cung cấp thông tin từ nền tảng Facebook làm dữ liệu tham khảo cho đối tác là các kênh báo chí có quan hệ khách hàng. Hệ thống bao gồm các micro service đã trình bày trong mục 4.3.2 được kết nối thông qua các API gate có tên miền “<https://apigate.cnnd.vn>”, được tích hợp theo tùy chọn để tham gia vào quá trình thu thập xử lý, biến đổi và cung cấp dữ liệu đầu ra. Chi tiết hạ tầng triển khai hệ thống thực tế gồm: (i) Server: Máy chủ điều khiển toàn bộ quá trình vận hành, lưu trữ dữ liệu và các API gate. Sử dụng cloud server lưu trữ dữ liệu và service của VCCloud, cấu hình máy ảo 24GB RAM, 4 nhân CPU Intel Xeon Silver 4214r, hệ điều hành Windows Server. (ii) Worker: Worker là các máy con hoạt động bên dưới máy chủ điều khiển của server, độc lập với nhau và cùng liên kết tới máy chủ nhằm nhận và cung cấp dữ liệu. Mục đích của các worker chủ yếu là thu thập/cào dữ liệu từ trang nguồn (Facebook) rồi đẩy vào DataLake ở server. Vì công việc này không yêu cầu cao về khả năng xử lý và lưu trữ nên các máy con đa dạng về cấu hình cũng như khả năng mở rộng cao (thêm các máy con mới hoặc các máy ảo thay thế máy vật lý). Thực tế đang sử dụng các máy con đồng bộ RAM 16GB, 2 nhân CPU, hệ điều hành Windows 10 cập nhật theo phiên bản mới nhất. Sau đây là bảng thông tin về kết quả triển khai thử nghiệm mô hình thực tế hệ thống.

| **Thông số** | **Số lượng** | **Đơn vị** |
| --- | --- | --- |
| Số lượng đầu báo sử dụng hệ thống như một dịch vụ | 12 | Đầu báo |
| Lưu lượng truy cập tổng vào các micro service từ client | 60 | POST request/phút |
| Lưu lượng dữ liệu đổ về DataWarehouse (gồm dữ liệu mới và dữ liệu cập nhật) | 105 | MB/ngày |
| Thời gian phản hồi trung bình của API gates | 0.3 | giây/request |
| Khả năng chịu tải của API gates | 200 | Request API cùng lúc |

**Bảng 4.5.1**: Thông tin thống kê các kết quả thông số trong thực tế triển khai hệ thống

Phản hồi của người dùng phần lớn từ các biên tập viên của các đầu báo mạng hợp tác với công ty cho thấy người dùng hài lòng với kết quả sử dụng ban đầu của hệ thống thông qua dịch vụ Ngóng tin MXH. Đa số các trường hợp sử dụng không gặp trở ngại về tắc nghẽn yêu cầu hay nhận dữ liệu; dữ liệu cung cấp đúng theo yêu cầu và format theo thiết kế ban đầu; tổng quan hệ thống hoạt động ổn định và luôn đảm bảo hoàn thành các yêu cầu nhận được. Tuy nhiên bên cạnh đó vẫn có một vài góp ý cần cải thiện thêm khả năng chịu tải cho hệ thống cho những đợt cao điểm về nhu cầu sử dụng thông tin như các mùa thể thao (gần nhất là World Cup), cần nâng cao khả năng chịu lỗi và backup dữ liệu cho các tình huống dữ liệu bị thiếu/sai/mất mát hoặc những thời điểm nâng cấp hệ thống cần có cơ chế đảm bảo toàn vẹn khả năng hoạt động, có các thông báo và logging đầy đủ để người dùng cũng như quản trị viên, người duy trì hệ thống hiểu rõ tình trạng hoạt động nhằm có những giải pháp kịp thời.

Chương này đã trình bày các thiết kế chi tiết từ giao diện bên ngoài đến kiến trúc, cấu trúc các thành phần bên trong hệ thống và đưa ra các kết quả đánh giá sử dụng thực tế. Chương tiếp theo sẽ nói về các giải pháp đóng góp nổi bật cũng như thách thức khi phát triển đồ án này từ bản thân em.

Sinh viên trình bày mô hình và/hoặc cách thức triển khai thử nghiệm/thực tế. Ứng dụng của sinh viên được triển khai trên server/thiết bị gì, cấu hình như thế nào. Kết quả triển khai thử nghiệm nếu có (số lượng người dùng, số lượng truy cập, thời gian phản hồi, phản hồi người dùng, khả năng chịu tải, các thống kê, v.v.)

## CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

Chương này có độ dài tối thiểu 5 trang, tối đa không giới hạn.[1](#_heading=h.1664s55)Sinh viên cần trình bày tất cả những nội dung đóng góp mà mình thấy tâm đắc nhất trong suốt quá trình làm ĐATN. Đó có thể là một loạt các vấn đề khó khăn mà sinh viên đã từng bước giải quyết được, là giải thuật cho một bài toán cụ thể, là giải pháp tổng quát cho một lớp bài toán, hoặc là mô hình/kiến trúc hữu hiệu nào đó được sinh viên thiết kế.

Chương này là tổng kết tất cả những nội dung đóng góp mà bản thân em tâm đắc nhất trong suốt quá trình làm đồ án bao gồm những giải pháp đặc thù giải quyết các vấn đề liên quan và những khó khăn gặp phải khi nghiên cứu và phát triển dự án này.

Kiến trúc thiết kế tổng quát của hệ thống này (đã trình bày ở mục 4.1) có thể coi là nhân tố quan trọng ảnh hưởng đến không chỉ kết quả triển khai thực tế của dự án mà còn là phương diện tạo nên tính độc đáo, khác biệt cho hệ thống khi so sánh với các hướng tiếp cận hiện có. (i) Đặt vấn đề: Khi bắt tay thực hiện dự án này, bước đầu tiên để tạo bản vẽ sơ bộ cho hệ thống là lựa chọn kiến trúc thiết kế. Kiến trúc phù hợp sẽ tối ưu khả năng hệ thống hiện tại và trong tương lai, vì vậy khi lựa chọn em cần chú ý không chỉ các yêu cầu chức năng, phi chức năng được giao mà còn phải tính toán khía cạnh mở rộng, phát triển về quy mô và chiều sâu sau này. Các sản phẩm hiện có trên thị trường cũng như các giải pháp về lý thuyết, tài liệu trong ngành hẹp được em tham khảo với tinh thần cầu thị bên cạnh các hoạt động thử nghiệm đánh giá để thu được kết quả khách quan. Những khảo sát thực tế này đã đem đến đánh giá tổng quan chung về các cách thức tiếp cận hiện nay cũng như ưu nhược điểm của chúng (đã trình bày ở mục 2.1), từ đó vấn đề đặt ra là cần lựa chọn một hướng kiến trúc mới có tính linh hoạt cao hơn, dễ dàng thiết kế độc lập theo chức năng cũng như quản lý, bảo trì nhưng vẫn đảm bảo khả năng giao tiếp giữa các thành phần để phục vụ cho các nhiệm vụ chung yêu cầu tính kết hợp. (ii) Quá trình đóng góp giải pháp: Quá trình tìm ra phương án tối ưu cho kiến trúc hệ thống dựa trên những lần lựa chọn, áp dụng, đánh giá các kiến trúc khác nhau. Ban đầu hệ thống đi theo hướng mô hình MVC truyền thống, với thiết kế cơ bản gồm các thành phần Model, View, Controller triển khai bằng các package chức năng tương ứng, giao tiếp thông qua các lệnh gọi đối tượng và không có giao thức kết nối bên ngoài. Tuy nhiên mô hình MVC có mục tiêu hướng đến thiết kế giao diện sử sụng cho phần mềm đóng gói hoàn chỉnh, khó để tích hợp thêm các tính năng và giao tiếo ngoại vi, trong khi bản thân hệ thống hướng đến nhiệm vụ trở thành công cụ tích hợp như một backend xử lý cho các hệ thống biên sử dụng và không đặt nặng yêu cầu về View (giao diện). Khi chuyển đổi qua các mô hình cải tiến như MVP thì kết quả thực nghiệm cũng không đáp ứng những yêu cầu đã đề ra. Dựa trên đánh giá sơ bộ và yêu cầu đầu vào, em đã lựa chọn sử dụng kiến trúc hệ thống theo hướng micro services cải tiến. Với kiến trúc micro services nguyên gốc, các service (dịch vụ) tách biệt hoàn toàn và chỉ có quan hệ phụ thuộc cùng nằm trong một hệ thống quản lý lớn; kiến trúc sử dụng trong hệ thống của em cũng triển khai các tính năng dưới dạng các service riêng nhưng khả năng quản lý, tính liên kết sẽ được đảm bảo chặt chẽ hơn thông qua hệ cơ sở các API quản lý và giao tiếp, khả năng mở rộng cũng được tối ưu theo cùng cách thức để dễ dàng tích hợp tính năng, nâng cao tính tùy biến theo yêu cầu người dùng. (iii) Kết quả: Trong triển khai thực tế hệ thống sử dụng kết hợp các tổ hợp dịch vụ con để giải quyết từng nhiệm vụ, việc có nhiều tổ hợp hướng tiếp cận cũng hỗ trợ cho việc kiểm thử đánh giá hiệu năng hệ thống và tối đa hóa khả năng xử lý các yêu cầu phát sinh như thu thập dữ liệu nhóm kín, phân quyền hóa các nguồn dữ liệu, xử lý xếp hạng thông tin-đây đều là các tính năng chuyên biệt hệ thống có được nhờ kiến trúc thiết kế này. Thị trường sản phẩm ứng dụng hiện nay theo khảo sát từ dự án vẫn còn thiếu những sản phẩm tương tự có khả năng phát triển các tính năng mở rộng như đã nêu theo thời gian, một phần nằm ở kiến trúc sử dụng còn chưa đủ linh hoạt và còn theo hướng đóng gói cứng toàn bộ sản phẩm. Với kết quả nghiệm thu từ thực tế triển khai, bản thân hệ thống cũng như khung thiết kế của nó có thể coi là thành tựu có giá trị mà bản thân em đã nghiên cứu và xây dựng để áp dụng vào sản phẩm chung của đội ngũ và công ty nói chung.

Khó khăn lớn nhất về nghiệp vụ và quy trình trong quá trình phát triển hệ thống này có thể kể đến việc thiết kế các giao thức giao tiếp chung giữa các thành phần riêng biệt về mặt vật lý cũng như chức năng. (i) Vấn đề: Về mặt vật lý, cấu trúc hệ thống bắt buộc bao gồm các máy con phân tán để thực hiện nhiệm vụ thu thập dữ liệu từ nguồn bắt đầu (Facebook) và máy chủ server để quản lý hệ thống, tương tự với các cơ sở dữ liệu có địa chỉ khác nhau được sử dụng như SQL Server và Elasticsearch. Yêu cầu tiên quyết là cần phải tạo được các đường kết nối thông tin giao tiếp cho việc gửi nhận thông tin quản lý và trao đổi dữ liệu. Mỗi đối tượng tham gia các quan hệ này có một cấu hình cổng giao tiếp khác nhau bên cạnh sự khác biệt về cấu trúc dữ liệu, cơ chế xử lý phản hồi. Thách thức tiếp theo là đảm bảo yêu cầu về tính đồng bộ dữ liệu giữa các đối tượng sao cho các bước trong quá trình đi và đến, gửi, nhận, xử lý, phản hồi, lưu trữ đều phải được xác nhận đầy đủ, đảm bảo điều kiện cần và đủ cho các bước xử lý Input-Output nhằm tránh mất trật tự luồng xử lý thông tin cũng như thực hiện nhiệm vụ chung. Ví dụ như dữ liệu đi từ SQL Server phải được biến đổi đúng theo cấu trúc chuẩn và được xác nhận tính đúng đắn trước khi đi đến Elasticsearch, kèm theo đó khi đã có xác nhận về dữ liệu đi vào Elasticsearch thì máy chủ mới gửi các API quản lý thông báo hoàn thành một quá trình đẩy dữ liệu tới các file log trạng thái để các máy con có đủ điều kiện thực thi nhiệm vụ mới, nếu có lỗi hoặc ngoại lệ xảy ra sẽ phải thực hiện lại nhiệm vụ và log ra file thông báo cho quản trị viên theo dõi. Về mặt chức năng nói đến việc kết nối các service thành phần của hệ thống. Mỗi service có yêu cầu dữ liệu sử dụng riêng và giao thức kết nối có thể khác nhau. Vì thực tế triển khai cần tích hợp các service theo tổ hợp để xử lý nhiệm vụ nên yêu cầu về mặt đồng bộ thông tin cũng cần quan tâm như với các thành phần vật lý/bán vật lý. Xử lý giao tiếp giữa các service có phần không phức tạp như các thành phần trước nhưng có điểm khác về yêu cầu thống nhất chuẩn giao thức kết nối. Khi một service bất kỳ được ngắt, thay đổi hoặc thêm mới vào hệ thống thì quy trình kết nối, cấu trúc xử lý cần phải tương đương, không phân biệt chức năng và phải đảm bảo không làm ảnh hưởng đến các thành phần đang hoạt động. Không chỉ phải xử lý kết nối giữa các thành phần có sẵn của hệ thống mà hệ thống cần có khả năng cập nhật, ghi nhận liên tục các thay đổi đến từ yêu cầu truyền vào từ bên ngoài (người dùng, quản trị viên) một cách đồng bộ, đảm bảo công việc hiện tại luôn được thực hiện trơn tru bất kể ngoại cảnh. (ii) Xử lý: Để giải quyết sự khác biệt cố hữu giữa các thành phần vật lý (hoặc bán vật lý), trước tiên em tìm hiểu bản chất của các thành phần này từ cách thức hoạt động và cơ chế giao tiếp. Với các kiến thức lý thuyết, nghiệp vụ tự tìm hiểu và được hỗ trợ về kỹ thuật bởi leader và các đồng nghiệp, em dần nắm rõ đặc điểm của các thành phần và tìm ra những cách tiếp cận xử lý theo hướng tịnh tiến, chia để trị. Từ nhiệm vụ lớn cần giải quyết ban đầu, em chia thành các nhiệm vụ con có khả năng kết nối với nhau để xử lý dần rồi tổng kết lại thành pipeline xử lý chung. Mỗi nhiệm vụ là một yêu cầu kết nối hai thành phần riêng biệt, chỉ cần triển khai được kênh giao tiếp hoàn chỉnh giữa chúng mà không cần quan tâm tới hai nút đầu cuối bởi dữ liệu ở hai đầu đều có cấu trúc tuân theo thành phần cục bộ đã biết. Khi đã hoàn thiện việc liên kết các thành phần vật lý/bán vật lý thì yêu cầu kế đến cần xử lý là tối ưu hiệu năng và tốc độ xử lý thông tin luồng dữ liệu. Các phiên bản thử nghiệm ban đầu của hệ thống tuy đảm bảo yêu cầu về việc hoàn thành chức năng nhiệm vụ ổn định nhưng phương diện về cấu hình trong thời gian chạy vẫn còn phải thực hiện thủ công ở một số bước bởi người duy trì hệ thống điển hình như bước đẩy dữ liệu làm giàu lên kho dữ liệu tại cloud server phải kết nối qua OpenVPN có mã OTP thay đổi theo ngày nhằm đảm bảo tính bảo mật, mã này chỉ được cấp bởi quản trị viên khiến quá trình làm việc chung của hệ thống không liền mạch; một số thời diểm luồng dữ liệu bị tắc nghẽn cục bộ do mất kết nối VPN vô hình chung làm giảm sút đáng kể tính cập nhật dữ liệu như tiêu chí thứ ba ở mục 2.1. Sau quá trình thống nhất và xác nhận các tiêu chuẩn về bảo mật trong các kết nối của hệ thống, em đã chuyển đổi và triển khai thành công các thành phần cấu hình và quản lý này ngay tại phía máy chủ để giảm thiểu các thao tác kết nối, xác thực từ xa; mã nguồn nghiệp vụ của các thành phần liên quan cũng được cập nhật lại trong suốt giai đoạn thử nghiệm để dần đạt được hiệu năng theo tiêu chuẩn sản phẩm yêu cầu (thông tin ở bảng 4.5.1). Về nhiệm vụ tối ưu thiết kế cho thành phần chức năng mà cụ thể là các API giao tiếp trong và ngoài hệ thống, em đã thử nghiệm và sử dụng framework FastAPI (đã nêu ở chương ba) để tận dụng ưu thế về tốc độ, dễ sử dụng và sự đa dạng trong cách thiết kế. Khi so sánh đánh giá các phương án để lựa chọn framework phù hợp, bản thân em đã được tiếp cận và làm việc trực tiếp với các công nghệ API mới cũng như các giao thức giao tiếp mạng hiện nay. Quy trình triển khai dịch vụ API trong hệ thống này trải qua các bước thử nghiệm tại máy local, thử nghiệm tại server, đẩy lên bản beta và cuối cùng là bản live chính thức. Suốt quá trình này các Restful API tạo bởi FastAPI đã thể hiện tính ổn định tại mọi thời điểm, khả năng chịu tải yêu cầu đạt tiêu chí đề ra với các thông số cũng đã nêu trong bảng 4.5.1. (iii) Kết quả: Sự hoàn thiện kết nối và thống nhất chuẩn giao thức giữa các thành phần tham gia hệ thống này có thể được coi như chất keo kết dính các service chức năng độc lập để tích hợp sử dụng khi cần thiết, đảm bảo chất lượng yêu cầu chung khi triển khai hệ thống. Từ việc đảm bảo mối liên kết trong và ngoài hệ thống, các tính năng và phát triển trong tương lai có thể được tính hợp và sử dụng dễ dàng theo tinh thần ban đầu của hệ thống.

Làm việc với hệ thống đa thành phần yêu cầu khả năng làm việc độc lập cũng như phối hợp tốt với các nhóm và cá nhân tham gia khác để đi đến sự thống nhất chung trong quy tắc phối hợp xây dựng và sử dụng hệ thống. (i) Vấn đề: Như đã đề cập ở các phần trước, hệ thống này có nhiều use case sử dụng và bản thân cũng bao gồm các thành phần chức năng khác nhau thuộc sử quản lý vận hành bởi nhiều đội nhóm trong phòng phát triển. Bản thân em chịu trách nhiệm nghiên cứu và xây dựng thiết kế hệ thống theo yêu cầu, viết mã nguồn và tạo lập các thành phần cục bộ cần thiết cho phần lõi nghiệp vụ của hệ thống. Khi đến các giai đoạn thử nghiệm triển khai có tính mở rộng quy mô thì hệ thống cần chuyển đổi các dữ liệu và cấu hình sẵn có về các thành phần bên ngoài (server, dịch vụ bên ngoài, hệ thống tích hợp) có chức năng chuyên biệt được dùng cho bước triển khai thực tế thì yêu cầu về phối hợp làm việc lại càng cấp thiết hơn. Bất kỳ thiếu sót hay nhầm lẫn trong các nhiệm vụ tương tác đồng thời bởi các bên có thể dẫn đến xung đột toàn cục vô hình chung làm chậm tiến độ của dự án. Đây cũng là dự án quy mô đầu tiên em được tham gia vào một đội ngũ làm việc chung ở vai trò nghiên cứu phát triển chính, vậy nên các khó khăn trong quá trình phối hợp làm việc là không ít. (ii) Xử lý: Từ khi bắt đầu công việc ở môi trường hiện tại, em không ngừng học hỏi và quan sát quy trình làm việc phát triển sản phẩm từ những người đi trước để nắm bắt được vai trò cũng như trách nhiệm, nhiệm vụ cần thực hiện khi phát triển dự án này. Đặc điểm điển hình của môi trường phát triển này là yêu cầu sự phối hợp nhịp nhàng giữa các đội nhóm bởi mỗi tác nhân tham gia sẽ đảm nhiệm xử lý một thành phần hoặc phương diện của sản phẩm đầy đủ như phần front-end, backend, dữ liệu, server, thiết kế giao diện. Không ít thời điểm khi thử nghiệm các tính năng dự án này giữa em và các cá nhân tham gia có những bất đồng ý kiến khi tìm cách thống nhất một phương án chung xuất phát từ khác biệt về cách tiếp cận vào bản chất hệ thống hoặc cách sử dụng một tính năng nào đó. Có thể kể đến khi bản thiết kế giao diện web dạng Figma của hệ thống Ngóng tin Facebook (mục 4.3.3) được đưa ra, nhóm thiết kế đã thiết kế bảng thông tin bộ lọc dữ liệu đầu vào với vector liên kết đầy đủ cho mọi trường dữ liệu kết nối tới tính năng xếp hạng bài viết trong khi tính năng này theo thiết kế nghiệp vụ cũng như cơ chế xử lý không hỗ trợ lọc thông tin theo tất cả trường dữ liệu hiện có bởi đây là tính năng sử dụng thuật toán xếp hạng chỉ áp dụng cho các dữ liệu đã biết (đã được truy vấn liệt kê từ trước) chứ không phải truy vấn các thông tin chưa xác định tại thời điểm gửi yêu cầu. Mâu thuẫn này xuất phát từ việc thiếu thông tin nghiệp vụ và tài liệu hướng dẫn từ cả hai phía. Để giải quyết vấn đề các bên tham gia và em đã thống nhất tạo và sử dụng tài liệu giao tiếp chia sẻ chung bên cạnh việc thảo luận lại các phương án xử lý để tái triển khai tính năng còn gặp khúc mắc. Những bất đồng trong quá trình phối hợp sau này vẫn xuất hiện nhưng đã có thể xử lý nhanh chóng và chủ yếu xuất phát từ các lỗi phát sinh khách quan thay vì chủ quan. Việc phối hợp nhóm không chỉ đem đến cơ hội khám phá học hỏi thêm các kiến thức nghiệp vụ, kỹ năng giao tiếp cho bản thân mà còn giúp phát hiện ra những thiếu sót trong phát triển của em. Khi có một lỗi xuất hiện trong luồng chức năng, bên sử dụng tính năng sẽ thông báo chi tiết đến người thiết kế hệ thống là em để khắc phục. Những lần khắc phục lỗi này đã chỉ ra không ít lỗi, ngoại lệ còn tồn tại trong mã nguồn mà khi thiết kế chính em và những người lập trình nói chung ít để ý đến bởi thiếu sót trong việc lường trước các tình huống use case có thể xảy ra hoặc một vài chức năng chưa được tối ưu đủ tốt yêu cầu em phải xem lại thiết kế hiện tại và tìm hướng xử lý mới đảm bảo yêu cầu từ các bên. (iii) Kết quả: Quá trình làm việc chung để phát triển dự án đã giúp nâng cao chất lượng thực tế của hệ thống và đối với bản thân một lập trình viên như em là những kiến thức, kỹ năng, ý thức về trách nhiệm khi làm công việc trong lĩnh vực công nghệ thông tin.

Chương này đã trình bày chi tiết về các quy trình nghiệp vụ được thực hiện để giải quyết các khó khăn nội tại của bài toán xây dựng hệ thống, quá trình làm việc để tìm ra các đóng góp quan trọng góp phần hoàn thiện hệ thống này từ bản thân người thực hiện. Chương tiếp theo sẽ đưa ra các kết luận sau cùng cũng như hướng phát triển tiếp theo của hệ thống trong đồ án.

Chương này **là cơ sở quan trọng** để các thầy cô đánh giá sinh viên. Vì vậy, sinh viên cần phát huy tính sáng tạo, khả năng phân tích, phản biện, lập luận, tổng quát hóa vấn đề và tập trung viết cho thật tốt. Mỗi giải pháp hoặc đóng góp của sinh viên cần được trình bày trong một mục độc lập bao gồm ba mục con: (i) dẫn dắt/giới thiệu về bài toán/vấn đề, (ii) giải pháp, và (iii) kết quả đạt được (nếu có).

Sinh viên lưu ý **không trình bày lặp lại nội dung**. Những nội dung đã trình bày chi tiết trong các chương trước không được trình bày lại trong chương này. Vì vậy, với nội dung hay, mang tính đóng góp/giải pháp, sinh viên chỉ nên tóm lược/mô tả sơ bộ trong các chương trước, đồng thời tạo tham chiếu chéo tới đề mục tương ứng trong Chương 5 này. Chi tiết thông tin về đóng góp/giải pháp được trình bày trong mục đó.

Ví dụ, trong Chương 4, sinh viên có thiết kế được kiến trúc đáng lưu ý gì đó, là sự kết hợp của các kiến trúc MVC, MVP, SOA, v.v. Khi đó, sinh viên sẽ chỉ mô tả ngắn gọn kiến trúc đó ở Chương 4, rồi thêm các câu có dạng: “Chi tiết về kiến trúc này sẽ được trình bày trong phần 5.1".

## CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chương này trình bày các kết luận được đúc rút từ quá trình thực hiện đồ án và các phương hướng tiếp theo nhằm duy trì và phát triển cho hệ thống này.

## Kết luận

Từ kết quả thử nghiệm triển khai thực tế đã trình bày ở mục 4.5 cũng như các phân tích tổng quát về các sản phẩm/ hệ thống tương tự trên thị trường hiện nay ở mục 2.1 em có một vài so sánh với hệ thống của mình như sau: (i) Về chi phí vận hành và giá thành sử dụng dịch vụ: Chi phí vận hành càng cao sẽ kéo theo giá thành sử dụng của khách hành tăng theo. Với việc chỉ sử dụng một số lượng nhỏ các máy con để thu thập dữ liệu và dịch vụ cloud server để lưu trữ trích xuất, hệ thống của em có quy mô nhỏ gọn khi so với các hệ thống làm việc với dữ liệu theo hướng phân tán (như ví dụ về SMCC), chi phí chủ yếu nằm ở tài nguyên sử dụng cho các request và tính toán phân tích dữ liệu và do đó tài nguyên, chi phí vận hành thực tế đã được đánh giá là phù hợp, có tính ứng dụng và cạnh tranh. (ii) Về tốc độ cập nhật, cung cấp dữ liệu: Nhờ vào công cụ lập lịch và sử dụng các hàng đợi như đã nêu ở chương 3, luồng dữ liệu feed từ các nguồn trên Facebook được thu thập lưu trữ vào kho dữ liệu tối ưu về tính cập nhật đến từng giờ, từng ngày, khắc phục tình trạng chậm/thiếu dữ liệu do việc thu thập không được cấu hình dựa vào lượng thông tin hàng ngày. (iii) Về quy mô hệ thống: Vì hệ thống tập trung làm việc với dữ liệu Facebook nên quy mô khá nhỏ so với các hệ thống lớn tương tự khác, do đó khả năng lưu trữ và xử lý dữ liệu lớn tương đương có thể không mạnh như các hệ thống khác. Tuy nhiên theo nhu cầu sử dụng thực tế và mục đích thiết kế, hệ thống đã đáp ứng đủ và phù hợp với yêu cầu đề ra; hơn nữa khả năng mở rộng hệ thống theo mục 4.5 là dễ dàng và hoàn toàn có thể dùng để mở rộng quy mô, khả năng xử lý cho các nhiệm vụ lớn hơn. (iv) Về tính đảm bảo giải pháp end-to-end solution cho người dùng: Như đã nêu ở mục 2.1, các hệ thống tương tự hiện nay hầu hết chỉ cung cấp dữ liệu cho người dùng theo format thô dưới dạng danh sách các kết quả (bài viết, kết quả thống kê tương tác) , chưa được tái phân tích để đáp ứng nhu cầu tìm hiểu insight dữ liệu của khách hàng. Với các micro-service phân tích làm giàu dữ liệu thô tích hợp bên trong, hệ thống có khả năng đáp ứng yêu cầu đầu-cuối từ client qua các kết quả như xếp hạng bài viết theo các mức độ, hiển thị các từ khóa trending, khả năng tăng cường thu thập dữ liệu bằng quan sát từ nhu cầu sử dụng dữ liệu từ một nguồn, ngoài ra các tính năng về dự đoán xu hướng dữ liệu vẫn đang được xây dựng và thử nghiệm để áp dụng trong tương lai. (v) Về khả năng ứng dụng thực tế: Vì hệ thống vẫn trong giai đoạn liên tục nâng cấp phát triển, mới chỉ áp dụng cho một số lượng nhỏ các đối tác liên quan nên thực tế triển khai có quy mô khá nhỏ, chưa có độ phổ biến rộng rãi như các hệ thống lớn hiện nay trên thị trường. Tuy vậy tiềm năng của hệ thống là có, và thực tế đã chứng minh độ ổn định đáng tin cập trong môi trường doanh nghiệp, vậy nên nếu có thêm thời gian và đánh giá thì hệ thống hoàn toàn có thể ứng dụng rộng rãi trong các mặt đời sống, kinh tế, xã hội có nhu cầu sử dụng.

Với đề tài xây dựng hệ thống này, bản thân em đã trải qua các bước chính trên con đường sản xuất tạo ra một sản phẩm công nghệ thông tin ứng dụng thực tế. Em đã tham gia các bước từ tìm hiểu nhu cầu, khảo sát hiện trạng, khảo sát khách hàng, nghiên cứu các sản phẩm tương tự hiện có, tìm tòi học hỏi các công nghệ cần thiết để làm tiền đề xây những mảnh ghép đầu tiên cho hệ thống. Nhờ có sự phối hợp làm việc của các đồng nghiệp và sự hướng dẫn, đánh giá trong từng giai đoạn phát triển của leader và khách hàng, em đã hoàn thiện cơ bản mã nguồn cũng như các API, cấu hình các cơ sở dữ liệu, thiết lập mô hình ứng dụng phù hợp với tài nguyên và yêu cầu. Em vẫn đang tiếp tục mở rộng và phát triển phần lõi hệ thống để tích hợp các tính năng mới, cập nhật kịp thời các thay đổi từ dữ liệu Facebook. Trong cả quá trình đó em vẫn còn thiếu sót ở một số bước phát triển như còn vội vàng trong việc triển khai áp dụng mà sơ sài trong bước kiểm thử, còn phụ thuộc một phần vào sự hỗ trợ và cung cấp kiến thức công nghệ từ leader, mới chỉ chú trọng vào xây dựng mã nguồn theo yêu cầu mà chưa quan tâm về khả năng bảo trì, nâng cấp mã nguồn đó để tái sử dụng, còn thiếu sót trong hiểu biết về quy trình nghiệp vụ xây dựng một hệ thống hoàn chỉnh. Từ những đóng góp và thiếu sót thông qua dự án này bản thân em đã được trau dồi rất nhiều không chỉ về kiến thức chuyên ngành và quy trình nghiệp vụ mà cả về kỹ năng phối hợp đội nhóm, kỹ năng tìm kiếm và áp dụng thông tin, khả năng chịu áp lực và hiểu sâu hơn về trách nhiệm và tinh thần mà một người lập trình viên/nhà phát triển cần phải có trong môt trường hiện nay.

Sinh viên so sánh kết quả nghiên cứu hoặc sản phẩm của mình với các nghiên cứu hoặc sản phẩm tương tự.

Sinh viên phân tích trong suốt quá trình thực hiện ĐATN, mình đã làm được gì, chưa làm được gì, các đóng góp nổi bật là gì, và tổng hợp những bài học kinh nghiệm rút ra nếu có.

## Hướng phát triển

Ở mục này em sẽ trình bày các công việc cần thiết để hoàn thiện các chức năng hiện tại của hệ thống và phân tích các hướng phát triển mới trong tương lai.

Về các nhiệm vụ chức năng chính hiện tại cần hoàn thiện bao gồm: (i) Chức năng xác định các từ khóa keyword cho mỗi tài liệu: Đây là chức năng với nhiệm vụ như tên gọi nhằm xác định các đối tượng, chủ thể chính mà một bản ghi/bài viết nhắc đến, phục vụ cho mục đích phân tích và SEO (Search engine optimization). Hiện tại em vẫn đang kết hợp và thử nghiệm các phương pháp học máy, học sâu khác nhau có thể kể đến phương pháp phân tích thực thể NER, công cụ phân tích từ khóa chính nhằm tìm ra phương án phân tích tối ưu nhất khi so sánh với kết quả tự dán nhãn từ khóa của khách hàng (các biên tập viên). Các yếu tố cần được đánh giá để xác định độ chính xác của chức năng bao gồm khả năng tìm kiếm các từ khóa chính, tỉ lệ từ khóa chính trên tổng số từ khóa có mặt, độ phủ từ khóa chính khi so với các phương án khác. (ii) Khả năng tự cập nhật module thu thập dữ liệu theo sự thay đổi của dữ liệu Facebook: Chức năng thu thập dữ liệu của hệ thống này dựa vào cookie tiêu chuẩn của Facebook, do đó mọi thay đổi trong định dạng cookie có thể dẫn đến lỗi hệ thống khi cơ chế thu thập được cấu hình mặc định không theo kịp sự thay đổi. Để tăng khả năng phòng chống lỗi cho hệ thống nói chung, em đang nghiên cứu thử nghiệm các phương pháp thông minh hơn trong việc tìm kiếm các thành phần dữ liệu cần thiết trong cookie Facebook, có thể tự động xác định các thông tin bao hàm trong khung cookie lớn (dưới dạng JSON) và thay đổi cơ chế nhằm lấy được các thông tin đó. Tuy nhiên hướng phát triển này vẫn cần nhiều thời gian và thử nghiệm do khó khăn về việc xác định các thành phần động trong dữ liệu Facebook và độ lớn của dữ liệu cũng ảnh hưởng hiệu suất của phương án này. Cách tiếp cận hiện tại của em dựa vào tương tác với cookie dưới dạng JSON và string, áo dụng regex, sử dụng cơ chế truy xuất-tìm kiếm theo tên riêng mặc định của mỗi thành phần thông tin-cập nhật lại cơ chế theo vị trí thông tin tìm được. (iii) Như đã nêu qua ở mục 6.1, chức năng trả kết quả và thực hiện yêu cầu của hệ thống sẽ được nâng cấp trong tương lai, cụ thể hiện tại hệ thống chỉ có khả năng phân tích để đưa ra các kết quả trong quá khứ và hiện tại. Để tăng tính ứng dụng thực tiễn cho các nhu cầu phân tích dữ đoán hiện nay, em đang nghiên cứu và đưa vào thử nghiệm một số tính năng dự đoán xu hướng dữ liệu theo chủ đề và từ khóa. Sử dụng các phương pháp xử lý dữ liệu theo dạng time-series và ứng dụng các kỹ thuật học sâu trong dự đoán hồi quy như mạng LSTM, tính năng này có thể cung cấp thông tin về những chủ đề có khả năng trở thành hot trend trong tương lai hoặc đưa ra thống kê đánh giá về lượng dữ liệu/bài viết theo thời gian. Các công cụ NLP phân tích chủ đề cũng được em nghiên cứu và áp dụng thêm nhằm hoàn thiện tính năng này.

Trong phần này, sinh viên trình bày định hướng công việc trong tương lai để hoàn thiện sản phẩm hoặc nghiên cứu của mình.

Trước tiên, sinh viên trình bày các công việc cần thiết để hoàn thiện các chức năng/nhiệm vụ đã làm. Sau đó sinh viên phân tích các hướng đi mới cho phép cải thiện và nâng cấp các chức năng/nhiệm vụ đã làm.

## MỘT SỐ LƯU Ý VỀ TÀI LIỆU THAM KHẢO

Lưu ý: Sinh viên không được đưa bài giảng/slide, các trang Wikipedia, hoặc các trang web thông thường làm tài liệu tham khảo.

Một trang web được phép dùng làm tài liệu tham khảo **chỉ khi** nó là công bố chính thống của cá nhân hoặc tổ chức nào đó. Ví dụ, trang web đặc tả ngôn ngữ XML của tổ chức W3C [https://www.w3.org/TR/2008/REC-xml-200](https://www.w3.org/TR/2008/REC-xml-20081126/) [81126/](https://www.w3.org/TR/2008/REC-xml-20081126/) là TLTK hợp lệ.

Có năm loại tài liệu tham khảo mà sinh viên phải tuân thủ đúng quy định về cách thức liệt kê thông tin như sau. Lưu ý: các phần văn bản trong cặp dấu < > dưới đây chỉ là hướng dẫn khai báo cho từng loại tài liệu tham khảo; sinh viên cần xóa các phần văn bản này trong ĐATN của mình.

<**Bài báo đăng trên tạp chí khoa học**: Tên tác giả, tên bài báo, tên tạp chí, volume, từ trang đến trang (nếu có), nhà xuất bản, năm xuất bản >

* + 1. E. H. Hovy, “Automated discourse generation using discourse structure rela- tions," *Artificial intelligence*, vol. 63, no. 1-2, pp. 341–385, 1993

<**Sách**: Tên tác giả, tên sách, volume (nếu có), lần tái bản (nếu có), nhà xuất bản, năm xuất bản>

* + 1. L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.
    2. N. T. Hải, *Mạng máy tính và các hệ thống mở*. Nhà xuất bản giáo dục, 1999.

<**Tập san Báo cáo Hội nghị Khoa học**: Tên tác giả, tên báo cáo, tên hội nghị, ngày (nếu có), địa điểm hội nghị, năm xuất bản>

* + 1. M. Poesio and B. Di Eugenio, “Discourse structure and anaphoric accessibil- ity," in *ESSLLI workshop on information structure, discourse structure and dis- course semantics*, Copenhagen, Denmark, 2001, pp. 129–143.

<**Đồ án tốt nghiệp, Luận văn Thạc sĩ, Tiến sĩ**: Tên tác giả, tên đồ án/luận văn, loại đồ án/luận văn, tên trường, địa điểm, năm xuất bản>

* + 1. A. Knott, “A data-driven methodology for motivating a set of coherence relations," Ph.D. dissertation, The University of Edinburgh, UK, 1996.

<**Tài liệu tham khảo từ Internet**: Tên tác giả (nếu có), tựa đề, cơ quan (nếu có), địa chỉ trang web, thời gian lần cuối truy cập trang web>

* + 1. T. Berners-Lee, *Hypertext transfer protocol (HTTP)*. [Online]. Available:

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

ftp:/info.cern.ch/pub/www/doc/http-spec.txt.Z (visited on 09/30/2010).

* + 1. Princeton University, *Wordnet*. [Online]. Available: [http://www.cogs ci.princeton.edu/~wn/index.shtml](http://www.cogsci.princeton.edu/~wn/index.shtml) (visited on 09/30/2010).

## TÀI LIỆU THAM KHẢO

* + - 1. E. H. Hovy, “Automated discourse generation using discourse structure rela- tions,” *Artificial intelligence*, vol. 63, no. 1-2, pp. 341–385, 1993.
      2. L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. El- sevier, 2007.
      3. N. T. Hải, *Mạng máy tính và các hệ thống mở*. Nhà xuất bản giáo dục, 1999.
      4. M. Poesio and B. Di Eugenio, “Discourse structure and anaphoric accessibil- ity,” in *ESSLLI workshop on information structure, discourse structure and discourse semantics, Copenhagen, Denmark*, 2001, pp. 129–143.
      5. A. Knott, “A data-driven methodology for motivating a set of coherence rela- tions,” Ph.D. dissertation, The University of Edinburgh, UK, 1996.
      6. T. Berners-Lee, *Hypertext transfer protocol (HTTP)*. [Online]. Available: ftp:

/info. cern. ch/ pub/ www/ doc/ http- spec. txt. Z (visited on 09/30/2010).

* + - 1. Princeton University, *Wordnet*. [Online]. Available: [http://www.cogsci. princeton.edu/~wn/index.shtml](http://www.cogsci.princeton.edu/~wn/index.shtml) (visited on 09/30/2010).

**PHỤ LỤC**

## HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP

**Quy định chung**

Dưới đây là một số quy định và hướng dẫn viết đồ án tốt nghiệp mà bắt buộc sinh viên phải đọc kỹ và tuân thủ nghiêm ngặt.

Sinh viên cần đảm bảo tính thống nhất toàn báo cáo (font chữ, căn dòng hai bên, hình ảnh, bảng, margin trang, đánh số trang, v.v.). Để làm được như vậy, sinh viên chỉ cần sử dụng các định dạng theo đúng template ĐATN này. Khi paste nội dung văn bản từ tài liệu khác của mình, sinh viên cần chọn kiểu Copy là “Text Only” để định dạng văn bản của template không bị phá vỡ/vi phạm.

Tuyệt đối cấm sinh viên đạo văn. Sinh viên cần ghi rõ nguồn cho tất cả những gì không tự mình viết/vẽ lên, bao gồm các câu trích dẫn, các hình ảnh, bảng biểu,

* 1. Khi bị phát hiện, sinh viên sẽ không được phép bảo vệ ĐATN.

Tất cả các hình vẽ, bảng biểu, công thức, và tài liệu tham khảo trong ĐATN nhất thiết phải được SV giải thích và tham chiếu tới ít nhất một lần. Không chấp nhận các trường hợp sinh viên đưa ra hình ảnh, bảng biểu tùy hứng và không có lời mô tả/giải thích nào.

Sinh viên tuyệt đối không trình bày ĐATN theo kiểu viết ý hoặc gạch đầu dòng. ĐATN không phải là một slide thuyết trình; khi người đọc không hiểu sẽ không có ai giải thích hộ. Sinh viên cần viết thành các đoạn văn và phân tích, diễn giải đầy đủ, rõ ràng. Câu văn cần đúng ngữ pháp, đầy đủ chủ ngữ, vị ngữ và các thành phần câu. Khi thực sự cần liệt kê, sinh viên nên liệt kê theo phong cách khoa học với các ký tự La Mã. Ví dụ, nhiều sinh viên luôn cảm thấy hối hận vì (i) chưa cố gắng hết mình, (ii) chưa sắp xếp thời gian học/chơi một cách hợp lý, (iii) chưa tìm được người yêu để chia sẻ quãng đời sinh viên vất vả, và (iv) viết ĐATN một cách cẩu thả.

Trong một số trường hợp nhất thiết phải dùng các bullet để liệt kê, sinh viên cần thống nhất Style cho toàn bộ các bullet các cấp mà mình sử dụng đến trong báo cáo. Nếu dùng bullet cấp 1 là hình tròn đen, toàn bộ báo cáo cần thống nhất cách dùng như vậy; ví dụ như sau:

* + - Đây là mục 1 – Thực sự không còn cách nào khác tôi mới dùng đến việc bullet trong báo cáo.
    - Đây là mục 2 – Nghĩ lại thì tôi có thể không cần dùng bullet cũng được. Nên tôi sẽ xóa bullet và tổ chức lại hai mục này trong báo cáo của mình cho khoa học hơn. Tôi muốn thầy cô và người đọc cảm nhận được tâm huyết của tôi

17

PHỤ LỤC A. HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP

trong từng trang báo cáo ĐATN.

## Ngành học

Sinh viên lưu ý viết đúng ngành/chuyên ngành trên bìa và trên gáy theo đúng quy định của Trường. Ngành học hay chuyên ngành học phụ thuộc vào ngành học mà sinh viên đăng ký. Sinh viên có thể đăng nhập trên trang quản lý học tập của mình để xem lại chính xác ngành học của mình.

Một số ví dụ sinh viên có thể tham khảo dưới đây, trong trường hợp có chuyên ngành thì sinh viên không cần ghi chuyên ngành:

Đối với kỹ sư chính quy: Từ K61 trở về trước: Ngành Kỹ thuật phần mềm Từ K62 trở về sau: Ngành Khoa học máy tính Đối với cử nhân: Ngành Công nghệ thông tin Đối với chương trình EliteTech: Chương trình Việt Nhật/KSTN: Ngành Công nghệ thông tin Chương trình ICT Global: Ngành Information Technology Chương trình DS&AI: Ngành Khoa học dữ liệu

## Đánh dấu (bullet) và đánh số (numering)

Việc sử dụng danh sách trong LaTeX khá đơn giản và không yêu cầu sinh viên phải thêm bất kỳ gói bổ sung nào. LaTeX cung cấp hai môi trường liệt kê đó là:

* + - Đánh dấu (bullet) là kiểu liệt kê không có thứ tự. Để sử dụng kiểu liệt kê đánh dấu, chúng ta khai báo như sau

\begin{itemize}

\item Nội dung thứ nhất được viết ở đây.

\item Nội dung thứ hai được viết ở đây.

\item ...

\end{itemize}

* + - Đánh số (numering) là kiểu liệt kê có thứ tự. Để sử dụng kiểu liệt kê đánh số, chúng ta khai báo như sau

\begin{enumerate}

\item Nội dung thứ nhất được viết ở đây.

\item Nội dung thứ hai được viết ở đây.

\item ...

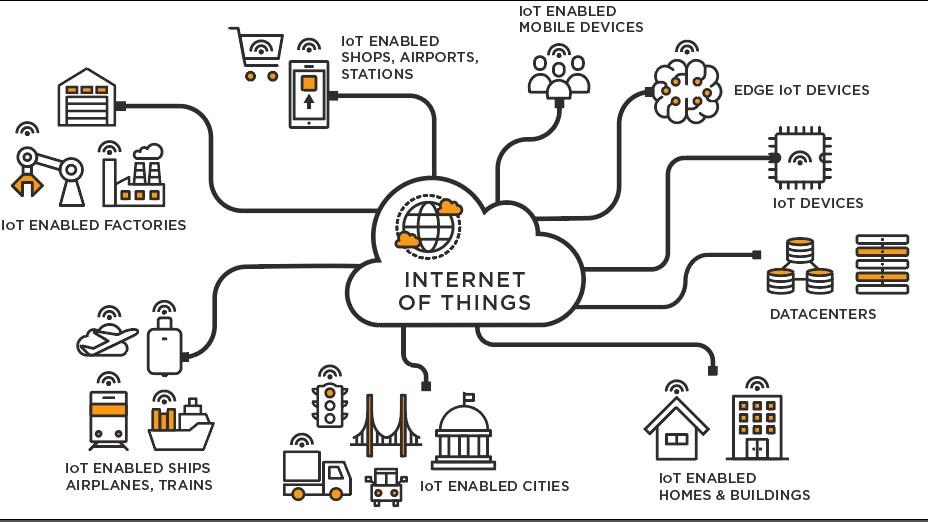
\end{enumerate}

Chú ý các nội dung trình bày trong cả hai môi trường liệt kê theo sau lệnh \item. Ngoài ra LaTeX còn cung cấp một số kiểu liệt kê khác, sinh viên có thể tham khảo tại <https://www.overleaf.com/learn/latex/Lists>

PHỤ LỤC A. HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP

| Col1 | Col2 | Col2 | Col3 |
| --- | --- | --- | --- |
| 1 | 6 | 87837 | 787 |
| 2 | 7 | 78 | 5415 |
| 3 | 545 | 778 | 7507 |
| 4 | 545 | 18744 | 7560 |
| 5 | 88 | 788 | 6344 |

**Bảng A.1:** Table to test captions and labels.



**Hình A.1:** Internet vạn vật

## Cách thêm bảng

Bảng [A.1](#_heading=h.1302m92) là ví dụ về cách tạo bảng. Tất cả các bảng biểu phải được đề cập đến trong phần nội dung và phải được phân tích và bình luận. Chú ý: Tạo bảng trong Latex khá phức tạp và mất thời gian, vì vậy sinh viên có thể sử dụng các công cụ hỗ trợ tạo bảng (Ví dụ: <https://www.tablesgenerator.com/>). Sinh viên có thể tìm hiểu sâu hơn về cách chèn ảnh trong Latex tại link [https:](https://www.overleaf.com/learn/latex/Tables)

[//www.overleaf.com/learn/latex/Tables](https://www.overleaf.com/learn/latex/Tables).

## Chèn hình ảnh

Hình [A.1](#_heading=h.3mzq4wv) là ví dụ về cách chèn ảnh. Lưu ý chú thích của hình vẽ được đặt ngay dưới hình vẽ. Sinh viên có thể tìm hiểu sâu hơn về cách chèn ảnh trong Latex tại <https://www.overleaf.com/learn/latex/Inserting_Images>.

Chú ý, tất cả các hình vẽ phải được đề cập đến trong phần nội dung và phải được phân tích và bình luận.

## Tài liệu tham khảo Cách liệt kê

* 1. (4) https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf

Áp dụng cách liệt kê theo quy định của IEEE. Ví dụ của việc trích dẫn như sau

19

PHỤ LỤC A. HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP

**scott2013sdn**. Cụ thể, sinh viên sử dụng lệnh \cite{} như sau **ashton2009internet**. Chỉ những tài liệu được trích dẫn thì mới xuất hiện trong phần Tài liệu tham khảo.

Tài liệu tham khảo cần có nguồn gốc rõ ràng và phải từ nguồn đáng tin cậy. Hạn chế trích dẫn tài liệu tham khảo từ các website, từ wikipedia.

## Các loại tài liệu tham khảo

Các nguồn tài liệu tham khảo chính là sách, bài báo trong các tạp chí, bài báo trong các hội nghị khoa học và các tài liệu tham khảo khác trên internet.

## Cách viết phương trình và công thức toán học

Các gói amsmath, amssymb, amsfonts hỗ trợ viết phương trình/công thức toán học đã được bổ sung sẵn ở phần đầu của file main.tex. Một ví dụ về tạo phương trình [(A.1)](#_heading=h.1tuee74) như sau

*F* (*x*) =

∫ *a* 1

*x*3 (A.1)

*b* 3

Phương trình [A.1](#_heading=h.1tuee74) là ví dụ về phương trình tích phân. Một phương trình khác không được đánh số thứ tự (gán nhãn)

*N−*1

*x*[*tn*] = *√N*

*k*=0

*X*[*fk*]*e*

*j*2*πnk/N*

Phương trình này thể hiện phép biến đổi Fourier rời rạc ngược (IDFT).

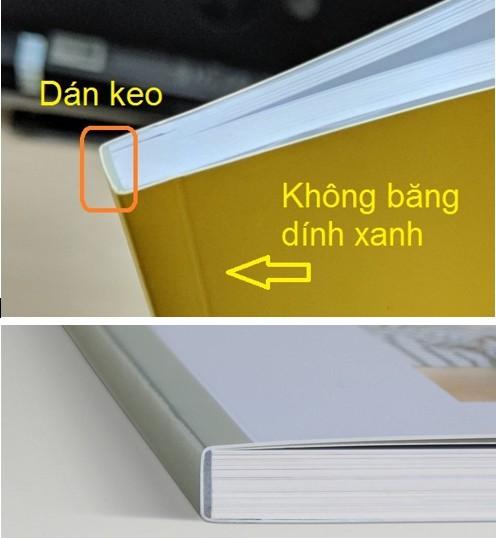
## Qui cách đóng quyển

Phần bìa trước chế bản theo qui định; bìa trước và bìa sau là giấy liền khổ. Sử dụng keo nhiệt để dán gáy khi đóng quyển thay vì sử dụng băng dính và dập ghim như mô tả ở Hình [A.3](#_heading=h.3s49zyc) Phần gáy ĐATN cần ghi các thông tin tóm tắt sau: Kỳ làm ĐATN - Ngành đào tạo - Họ và tên sinh viên - Mã số sinh viên. Ví dụ:

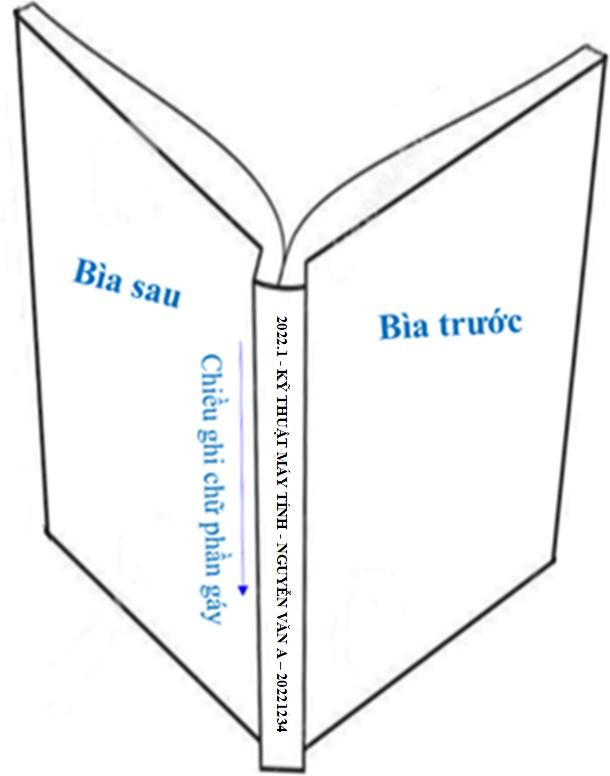
2022.1 - KỸ THUẬT MÁY TÍNH - NGUYỄN VĂN A - 20221234

Qui cách ghi chữ phần gáy như hình dưới đây:

PHỤ LỤC A. HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP



**Hình A.2:** Qui cách đóng quyển đồ án



**Hình A.3:** Qui cách đóng quyển đồ án

21