ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN – TIN HỌC

ĐỒ ÁN MÔN HỌC
KỸ THUẬT XỬ LÝ DỮ LIỆU LỚN

ĐỀ TÀI: ỨNG DỤNG MODEL CONTEXT PROTOCOL TRONG HỆ THỐNG GỢI Ý XU HƯỚNG XÃ HỘI CÁ NHÂN HÓA

Giảng viên: PGS.TS Nguyễn Thanh Bình

Nhóm thực hiện:

Trương Thành Thắng	:	24C01036 (Trưởng nhóm)
Tạ Ngọc Sơn	:	24C01019
Lê Thị Diệu	:	24C01016

Thành phố Hồ Chí Minh, tháng 10 năm 2025

Mục lục

Lời mở đầu

Trong bối cảnh thông tin và xu hướng xã hội thay đổi từng giờ, việc nắm bắt nhanh chóng các chủ đề "nóng" trở thành yếu tố quan trọng cho nghiên cứu, truyền thông và ra quyết định. Tuy nhiên, lượng dữ liệu khổng lồ từ các nền tảng mạng xã hội và website khiến quá trình thu thập, phân tích, và gợi ý xu hướng trở nên phức tạp và khó duy trì ổn định.

Dự án *TrendApp* được thực hiện với mục tiêu ứng dụng thử *Model Context Protocol (MCP)* – một khái niệm mới trong việc chuẩn hóa giao tiếp giữa các công cụ – nhằm tạo nên một hệ thống thu thập và gợi ý xu hướng linh hoạt, độc lập giữa phần Cào dữ liệu (crawler) và Khách (client). Bằng việc kết hợp *LLM (Large Language Model)* để phân tích sở thích người dùng và MCP để quản lý luồng dữ liệu từ nhiều nguồn khác nhau, TrendApp hướng tới việc xây dựng nền tảng gợi ý xu hướng xã hội *cá nhân hóa, mở rộng dễ dàng và tối ưu hiệu suất.*

1. Giới thiệu

1.1 Giới thiệu chung

TrendApp là hệ thống gợi ý xu hướng xã hội cá nhân hóa, được phát triển nhằm giúp người dùng nắm bắt nhanh các chủ đề nổi bật phù hợp với sở thích cá nhân. Hệ thống thu thập dữ liệu từ nhiều nguồn phổ biến như *Reddit, Facebook, YouTube, Finance Yahoo, Google Trending, CFBiz và HuggingFace Paper*, sau đó xử lý và phân tích để nhận diện các xu hướng đang được quan tâm.

Điểm đặc biệt của TrendApp là việc ứng dụng Model Context Protocol (MCP) – một giao thức giúp tách biệt và chuẩn hóa các công cụ crawler. Nhờ đó, mỗi crawler có thể hoạt động độc lập, dễ dàng điều chỉnh hoặc mở rộng mà không cần can thiệp vào phần client. Kiến trúc này giúp tăng tính linh hoạt, giảm rủi ro khi thay đổi hệ thống và hỗ trợ tích hợp thêm nhiều nguồn dữ liệu mới trong tương lai.

Hệ thống đồng thời kết hợp mô hình ngôn ngữ lớn (LLM) để hiểu và phân tích ngôn ngữ tự nhiên từ người dùng, từ đó xác định sở thích và đưa ra gợi ý xu hướng phù hợp nhất. Mô hình triển khai trên nền FastAPI, tận dụng xử lý bất đồng bộ (async) và cơ chế cache để tối ưu hiệu năng khi thu thập và phân tích dữ liệu.

1.2 Mục tiêu dự án

Dự án TrendApp – MCP Crawler Tool hướng đến các mục tiêu chính sau:

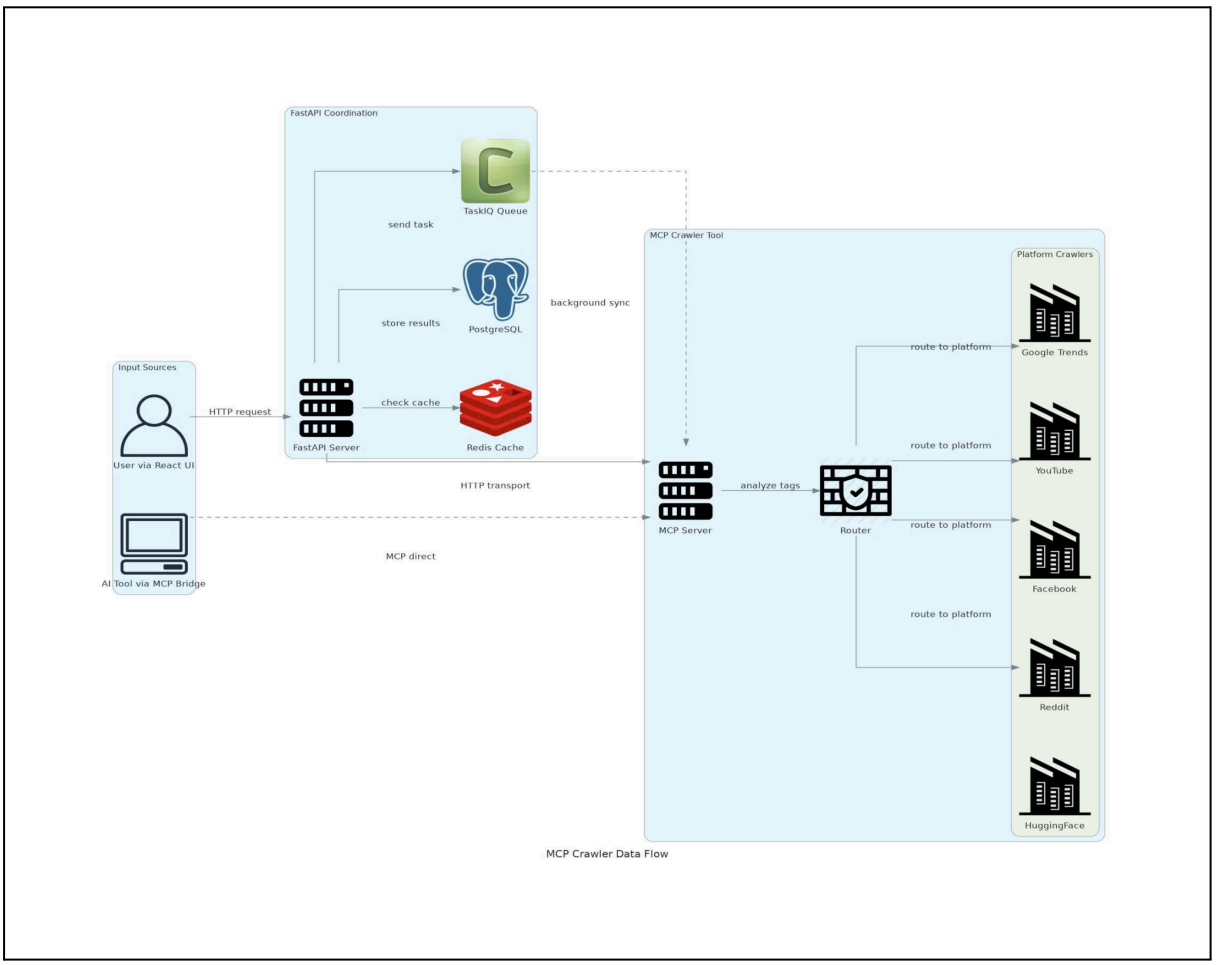
- Xây dựng hệ thống gợi ý xu hướng xã hội cá nhân hóa, dựa trên phân tích sở thích người dùng bằng mô hình LLM.
- Thiết kế và triển khai kiến trúc MCP để chuẩn hóa giao tiếp giữa client và crawler, đảm bảo khả năng mở rộng và bảo trì linh hoạt.

- Phát triển bộ crawler mô-đun hóa, có thể dễ dàng thêm mới hoặc thay thế từng nguồn dữ liệu mà không ảnh hưởng đến toàn hệ thống.
- Đảm bảo hiệu năng và độ tin cậy cao, thông qua cơ chế xử lý bất đồng bộ và caching thông minh.
- Tạo nền tảng mở có thể mở rộng sang các lĩnh vực khác như phân tích tin tức, phát hiện xu hướng đầu tư hoặc theo dõi cảm xúc cộng đồng.

2. Kiến trúc hệ thống

2.1 Tổng quan

Hệ thống TrendApp được thiết kế theo hướng phân lớp và mô-đun hóa, trong đó mỗi thành phần đảm nhận một vai trò riêng biệt nhưng có thể giao tiếp thông qua Model Context Protocol (MCP). Cách tiếp cận này giúp đảm bảo khả năng mở rộng, bảo trì dễ dàng và linh hoạt trong việc tích hợp thêm các nguồn dữ liệu hoặc công cụ xử lý mới.



Hình 1: *Sơ đồ tổng quan mô tả luồng dữ liệu và sự tương tác giữa các thành phần chính*

Hệ thống TrendApp được thiết kế theo một kiến trúc phân tầng rõ ràng và mô-đun, đảm bảo khả năng mở rộng, bảo trì và vận hành hiệu quả. Cấu trúc tổng thể có thể được khái quát với các thành phần chính sau:

- **Giao diện Người dùng (UI):** Đây là điểm tiếp xúc đầu tiên và trực tiếp nhất với người dùng cuối. Nhiệm vụ chính của tầng UI là thu thập các tương tác và hành vi của người dùng (như truy vấn tìm kiếm, thao tác thích/bỏ qua, thiết lập tùy chọn cá nhân hóa) và chuyển chúng thành các yêu cầu (request) đến máy chủ. Đồng thời, UI có trách nhiệm hiển thị kết quả xử lý (response) từ hệ thống—các xu hướng được gợi ý, tin tức, hoặc phân tích—một cách trực quan và dễ hiểu, hoàn tất một vòng lặp tương tác hoàn chỉnh.

- **Client (Lõi Điều khiển Trung tâm):** Đóng vai trò là bộ não của toàn hệ thống, Client là trung tâm điều phối mọi hoạt động. Thành phần này tích hợp và quản lý nhiều module then chốt:

- **API Server (FastAPI):** Xử lý các yêu cầu từ UI, định tuyến và điều phối luồng dữ liệu.

- **Cơ sở dữ liệu:** Sử dụng PostgreSQL làm cơ sở dữ liệu chính để lưu trữ dữ liệu có cấu trúc như thông tin người dùng, lịch sử tương tác, metadata của xu hướng. Redis được tận dụng cho các tác vụ cần hiệu năng cao như cache kết quả truy vấn, lưu trữ phiên làm việc (session) và quản lý hàng đợi tác vụ.

- **LLM Integration:** Module này kết nối với các mô hình ngôn ngữ lớn, chịu trách nhiệm phân tích ngữ nghĩa truy vấn của người dùng, hiểu sở thích ẩn sau hành vi, và tóm tắt hay phân loại nội dung để nâng cao chất lượng gợi ý.

- **MCP Caller:** Là cầu nối giao tiếp giữa Client và các công cụ thu thập dữ liệu bên ngoài. Nó đóng gói các yêu cầu cần lấy dữ liệu và gửi chúng đến đúng MCP Tool thông qua các tags được quy định.

- **Sub-Client - Taskiq (Xử lý Nền):** Để đảm bảo trải nghiệm real-time cho người dùng, các tác vụ tốn nhiều thời gian xử lý hoặc không yêu cầu phản hồi tức thời sẽ được chuyển sang một máy chủ xử lý nền (background task) chạy song song. Sử dụng Taskiq, hệ thống có thể thực hiện các công việc như thu thập dữ liệu định kỳ từ các nguồn, tiền xử lý và làm sạch dữ liệu thô, huấn luyện lại mô hình đề xuất, hoặc gửi thông báo tổng hợp email mà không làm ảnh hưởng đến hiệu năng phản hồi của Client chính.

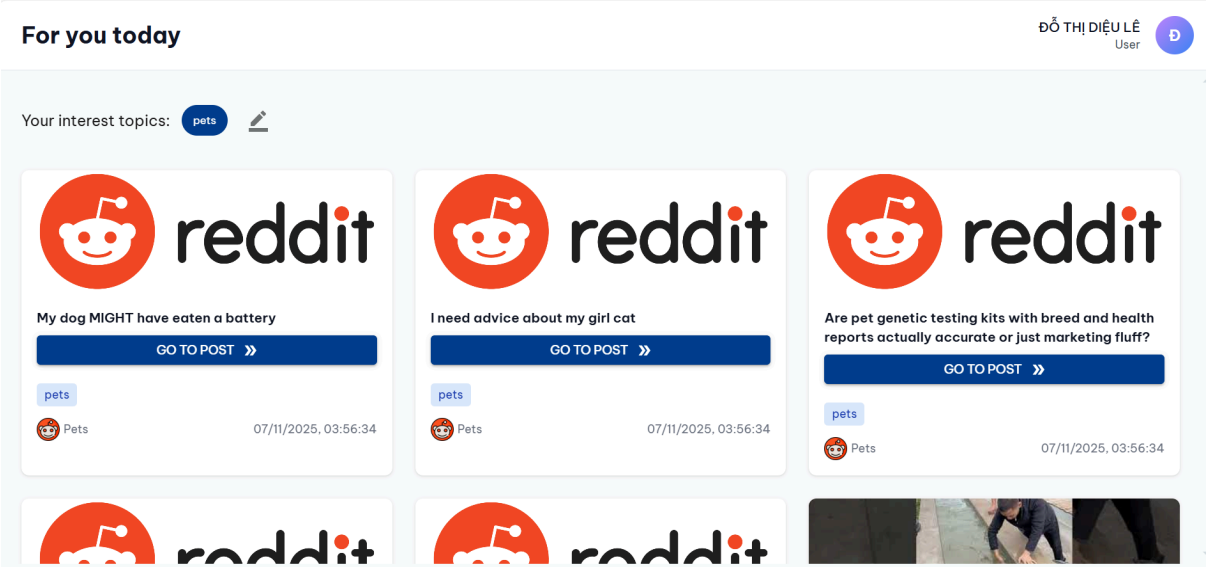
- Công cụ MCP (MCP Tool) (*Introducing the Model Context Protocol*, 2024): Đây là lớp thực thi việc thu thập dữ liệu, được quản lý một cách tập trung. MCP Tool điều khiển một loạt các crawler chuyên biệt cho từng nguồn dữ liệu (Reddit, YouTube, Google Trends...). Mỗi crawler được đăng ký với các tags mô tả đặc tính (ví dụ: news, finance, social-media, real-time). Khi MCP Caller từ Client gửi một yêu cầu kèm theo tags cụ thể (ví dụ: "cần tin tức tài chính và xu hướng mạng xã hội"), MCP Tool sẽ định tuyến và kích hoạt các crawler tương ứng để thu thập dữ liệu, đảm bảo tính mô-đun hóa cao và dễ dàng mở rộng.

2.2 Luồng dữ liệu hệ thống:

Hệ thống TrendApp vận hành thông qua hai luồng dữ liệu chính, được thiết kế để đáp ứng các kịch bản tương tác khác nhau của người dùng.

2.2.1. Luồng lấy bảng tin mặc định (get_user_feed)

Luồng xử lý này được kích hoạt khi người dùng truy cập vào bảng tin chính mà không cung cấp truy vấn tìm kiếm cụ thể. Mục tiêu của luồng này là cung cấp nhanh chóng nội dung được cá nhân hóa dựa trên hồ sơ sở thích hiện có của người dùng.



Hình 2: *Giao diện của bảng tin mặc định*

Quy trình thực hiện được mô tả như sau:

- Bước 1: Xác thực người dùng và thu thập sở thích hiện tại

- Hệ thống tiến hành xác thực danh tính người dùng thông qua cơ chế validate_curr_user()
- Truy vấn cơ sở dữ liệu để lấy danh sách các tags sở thích đã được lưu trữ trước đó của người dùng thông qua phương thức feed_service._get_user_interests()

- Bước 2: Truy xuất và truyền dữ liệu

- Danh sách tags thu được được truyền trực tiếp đến hàm xử lý chính _stream_feed() với tham số save_interests=False

- Bước 3: Xử lý luồng dữ liệu

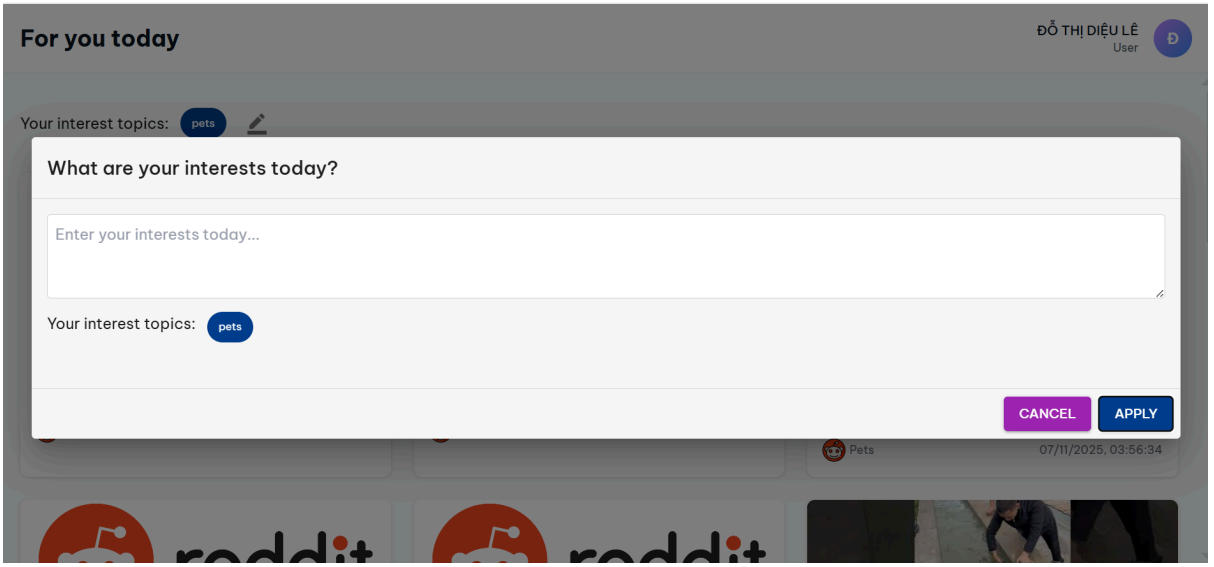
- Hàm _stream_feed() thực hiện truy vấn cơ sở dữ liệu để lấy các bài viết phù hợp với tags
- Các bài viết có sẵn được stream ngay lập tức về client với nhãn source: "cache"
- Đối với các tags chưa được đáp ứng đủ, hệ thống gọi MCP Server để thu thập nội dung mới
- Các bài viết mới được stream về client với nhãn source: "live"

- Bước 4: Đồng bộ hóa nền

- Một tác vụ nền được kích hoạt để đồng bộ các bài viết mới vào cơ sở dữ liệu
- Do tham số save_interests=False, hệ thống không cập nhật lại sở thích người dùng

2.2.2. Luồng lấy bảng tin theo truy vấn (get_user_feed_by_text)

Luồng xử lý này được kích hoạt khi người dùng nhập truy vấn tìm kiếm bằng văn bản. Khác với luồng mặc định, luồng này tập trung vào việc phân tích ý định người dùng và cập nhật hồ sơ sở thích.



Hình 3: *Giao diện cập nhật sở thích bằng Ngôn ngữ tự nhiên*

Quy trình thực hiện được mô tả như sau:

- Bước 1: Phân tích và trích xuất sở thích

- Hệ thống xác thực người dùng và thu thập sở thích hiện tại
- Truy vấn văn bản của người dùng (user_query) cùng với sở thích hiện tại được gửi đến mô hình LLM (Reflector) để phân tích và trích xuất tags mới

- Bước 2: Thông báo trạng thái

- Hệ thống gửi thông báo trạng thái "Analyzing your interests..." kèm theo các tags vừa được trích xuất về client
- Danh sách tags mới được truyền đến hàm _stream_feed() với tham số save_interests=True

- Bước 3: Xử lý luồng dữ liệu

- Hàm _stream_feed() xử lý tương tự như luồng mặc định, ưu tiên dữ liệu cache trước khi bổ sung dữ liệu live từ MCP Server

- Bước 4: Cập nhật hồ sơ người dùng

- Tác vụ nền được kích hoạt với đầy đủ thông tin người dùng (user_info)
- Hệ thống tiến hành lưu trữ các tags mới vào cơ sở dữ liệu, cập nhật chính thức sở thích người dùng cho các tương tác tương lai, cũng như là post mới (nếu có)

2.2.3. So sánh đặc điểm hai luồng dữ liệu:

ĐẶC ĐIỂM	LUỒNG MẶC ĐỊNH	LUỒNG THEO TRUY VẤN
Điều kiện kích hoạt	Truy cập bảng tin chính	Có truy vấn văn bản
Xử lý LLM	Không sử dụng	Có sử dụng để trích xuất tags
Cập nhật sở thích	Không cập nhật	Có cập nhật
Thông báo trạng thái	Không có	Có thông báo phân tích
Hiệu năng	Ưu tiên tốc độ phản hồi	Tập trung vào độ chính xác

Bảng 2.1: *So sánh đặc điểm hai luồng dữ liệu*

Hai luồng dữ liệu này kết hợp với nhau tạo nên một hệ thống linh hoạt, vừa đảm bảo khả năng phản hồi nhanh chóng, vừa cung cấp khả năng cá nhân hóa sâu sắc dựa trên sự thấu hiểu ngữ nghĩa từ truy vấn của người dùng.

3. Triển khai và Công nghệ

3.1. Kiến trúc Triển khai và Hạ tầng Đám mây

Hệ thống TrendApp được triển khai hoàn toàn trên nền tảng đám mây Microsoft Azure, mang lại tính sẵn sàng cao, khả năng mở rộng linh hoạt và bảo mật toàn diện. Kiến trúc triển khai bao gồm:

- **Azure Web App**: Đóng vai trò hosting cho Client chính và API server, cung cấp khả năng auto-scaling tự động dựa trên tải workload, tích hợp sẵn monitoring và logging thông qua Azure Monitor.
- **MCP Cloud Service**: MCP Server được triển khai trên nền tảng MCP Cloud chuyên biệt, được tối ưu hóa cho các dịch vụ Model Context Protocol, đảm bảo hiệu năng và độ ổn định trong việc quản lý và điều phối các crawler.

- **Azure Cache for Redis**: Service managed Redis cung cấp hiệu năng cao với SLA 99.9%, đảm bảo tính sẵn sàng cho cả caching layer và message broker system.

- **Azure Database Services**: Hệ thống database sử dụng Azure Database for PostgreSQL với tính năng read replica để phân tải truy vấn và automatic backups định kỳ.

3.2. Ngôn ngữ Lập trình và Framework Chính

Backend Platform (Python) Toàn bộ phần backend bao gồm Client chính và MCP Server đều được phát triển bằng Python 3.12, tận dụng ưu thế về:

- Hệ sinh thái thư viện phong phú cho xử lý dữ liệu (Pandas, NumPy)
- Hỗ trợ mạnh mẽ cho AI và machine learning (scikit-learn, transformers)
- Async/await syntax cho xử lý bất đồng bộ hiệu quả

FastAPI được lựa chọn làm web framework chính nhờ:

- Hiệu năng vượt trội, tương đương với Node.js và Go
- Hỗ trợ native asynchronous programming
- Tự động generate OpenAPI documentation
- Type hints giúp phát hiện lỗi tại thời điểm development

MCP Framework (FastMCP) MCP Server triển khai trên FastMCP - framework tiên tiến nhất cho MCP server development:

- **HTTP Protocol**: Được sử dụng làm giao thức chính trong production, hỗ trợ RESTful API design, dễ dàng tích hợp với load balancer và API gateway
- **Stdio Protocol**: Cung cấp khả năng tích hợp trực tiếp với các AI client như Claude và ChatGPT thông qua standard input/output

Frontend Platform (TypeScript) Giao diện người dùng phát triển bằng TypeScript 5.0+ với:

- React 18 cho component-based architecture
- Tailwind CSS cho responsive design
- Axios cho API communication
- React Query cho server state management

3.3. Công nghệ Then chốt và Tích hợp

3.3.1. Large Language Model (Google Gemini)

Google Gemini được lựa chọn làm LLM engine chính nhờ những ưu điểm vượt trội:

- **Tốc độ xử lý**: Thời gian phản hồi trung bình dưới 2s cho các task phân tích văn bản
- **Độ chính xác**: Cho kết quả chính xác trong các tác vụ semantic understanding và tag extraction
- **Cost-effective**: Pricing model cạnh tranh so với các LLM khác trên thị trường
- **API Stability**: Độ ổn định cao với rate limit linh hoạt

3.3.2. Bộ nhớ đệm và Trung gian Tác vụ (Redis)

Redis đóng hai vai trò quan trọng trong kiến trúc hệ thống:

- **Lớp Bộ nhớ đệm (Cache Layer):**
  - Lưu trữ tạm thời kết quả truy vấn với thời gian sống (TTL) 30 phút.
  - Lưu trữ phiên đăng nhập (session) cho việc xác thực người dùng.
  - Hỗ trợ giới hạn tốc độ (rate limiting) cho các endpoint API.
- **Trung gian Tác vụ (Task Broker):**
  - Hàng đợi tin nhắn (Message queue) cho các tác vụ xử lý nền.
  - Cơ chế ưu tiên tác vụ (Task prioritization).
  - Hàng đợi tin nhắn lỗi (Dead letter queue) để xử lý các tác vụ thất bại.

3.3.3. Hệ thống Xác thực (Azure Entra ID)

Azure Entra ID cung cấp:

- **Xác thực Tập trung:** Cơ chế đăng nhập một lần (Single sign-on) cho toàn bộ hệ thống.
- **Kiểm soát Truy cập Dựa trên Vai trò (RBAC):** Phân quyền chi tiết cho người dùng và các dịch vụ.
- **Giám sát Bảo mật:** Phát hiện đe dọa thời gian thực và nhật ký kiểm tra (audit logs).
- **Tích hợp:** Khả năng tích hợp liền mạch với các dịch vụ Azure khác.

3.3.4. Xử lý Nền và Quản lý Tác vụ

Hệ thống sử dụng Taskiq kết hợp với Redis làm message broker để xử lý các tác vụ nền:

- **Phân phối Tác vụ:** Tự động phân phối tác vụ cho nhiều worker.
- **Cơ chế Thử lại:** Tự động thử lại các tác vụ thất bại với chiến lược "backoff" theo cấp số nhân.
- **Theo dõi Tiến trình:** Giám sát tiến độ tác vụ thời gian thực.
- **Quản lý Tài nguyên:** Phân bổ tài nguyên thông minh dựa trên mức độ ưu tiên của tác vụ.

Kiến trúc triển khai này không chỉ đảm bảo hệ thống vận hành ổn định với hiệu năng tối ưu, mà còn cung cấp nền tảng vững chắc cho việc mở rộng và phát triển các tính năng trong tương lai.

3.4. Kết quả triển khai và nơi lưu trữ

Sau khi hoàn thiện quá trình phát triển và tích hợp các thành phần chính, hệ thống *TrendApp – MCP Crawler Tool* đã được triển khai thử nghiệm trên môi trường thực tế. Ứng dụng bao gồm hai phần chính: công cụ thu thập và phân tích xu hướng (backend), cùng với giao diện người dùng web (frontend) để hiển thị kết quả gợi ý.

- Kết quả triển khai:

- Hệ thống hoạt động ổn định trong các bài kiểm thử cơ bản, có thể xử lý đồng thời nhiều yêu cầu truy vấn và hiển thị xu hướng tổng hợp từ nhiều nguồn dữ liệu khác nhau. Thời gian phản hồi trung bình đạt khoảng 2 giây cho mỗi truy vấn.

- Kho mã nguồn (GitHub):
  MCP Server: https://github.com/airen95/trendapp_mcp_server
  MCP Backend: https://github.com/airen95/trendapp_mcp_backend
  MCP Frontend: https://github.com/airen95/trendapp_mcp_frontend

- Phiên bản web thử nghiệm:
  Hệ thống webapp (*Introducing the Model Context Protocol*, 2024) được triển khai thử nghiệm tại (TrendApp Demo)

Người dùng có thể truy cập để xem giao diện demo, kết quả truy vấn xu hướng, cũng như cấu trúc mã nguồn chi tiết của dự án.

4. Kết quả và Đánh giá

4.1. Kết quả thực nghiệm

Sau khi hoàn thiện phiên bản đầu tiên của *TrendApp – MCP Crawler Tool*, nhóm tiến hành giai đoạn kiểm thử nhằm đánh giá tính ổn định, tốc độ phản hồi và khả năng mở rộng của hệ thống. Quá trình kiểm thử được thực hiện với các nguồn dữ liệu phổ biến và có độ đa dạng cao, bao gồm *Reddit, YouTube, Google Trending, CFBiz, và HuggingFace Paper.* Các nền tảng này được chọn nhằm phản ánh nhiều loại nội dung khác nhau — từ mạng xã hội, video, tài chính, kinh doanh cho đến học thuật — giúp đánh giá khả năng tổng hợp xu hướng của hệ thống trong nhiều lĩnh vực.

Kết quả thực nghiệm cho thấy:

- **Thời gian phản hồi trung bình** cho mỗi truy vấn gợi ý xu hướng dao động từ **1,8 đến 2,5 giây**, bao gồm cả giai đoạn phân tích sở thích người dùng bằng **mô hình ngôn ngữ lớn (LLM)** và quá trình tổng hợp dữ liệu từ nhiều **crawler** hoạt động song song. Kết quả này được xem là khả quan, đảm bảo trải nghiệm người dùng mượt mà ngay cả khi lượng truy vấn tăng cao.

- **Hiệu suất xử lý tổng thể** được cải thiện đáng kể nhờ áp dụng giao thức **Model Context Protocol (MCP)**. So với mô hình tích hợp truyền thống, MCP giúp **rút ngắn khoảng 35–40% thời gian** khi thêm mới hoặc cập nhật một module crawler, nhờ việc loại bỏ phần lớn khâu kết nối thủ công và giảm chi phí bảo trì giữa các thành phần.

- **Bộ nhớ đệm Redis** phát huy hiệu quả rõ rệt, giúp **tiết kiệm trung bình 60% số lần gọi crawler**, đặc biệt trong các trường hợp nhiều người dùng có mối quan tâm trùng lặp. Điều này không chỉ giúp giảm tải cho hệ thống mà còn cải thiện đáng kể tốc độ phản hồi ở tầng ứng dụng.

- **Khả năng xử lý đồng thời** cũng được xác nhận qua các bài kiểm thử hiệu năng. Hệ thống có thể xử lý nhiều yêu cầu song song mà không làm suy giảm đáng kể tốc độ phản hồi hay tiêu tốn quá mức tài nguyên phần cứng.

Nhìn chung, các kết quả thử nghiệm bước đầu khẳng định tính khả thi của kiến trúc MCP-based TrendApp, thể hiện rõ ưu điểm về hiệu năng, khả năng mở rộng và sự linh hoạt trong quá trình triển khai.

4.2. Đánh giá hiệu quả

Việc ứng dụng *Model Context Protocol (MCP)* trong kiến trúc hệ thống TrendApp đã mang lại nhiều lợi ích thiết thực, cả về mặt kỹ thuật lẫn khả năng phát triển lâu dài:

- **Tính module hóa cao:** Mỗi thành phần trong hệ thống được thiết kế như một module độc lập (crawler, API, phân tích sở thích, tổng hợp kết quả, v.v.), kết nối thông qua giao thức MCP. Cách tiếp cận này giúp việc **bảo trì, cập nhật hoặc mở rộng** trở nên đơn giản, giảm thiểu rủi ro khi triển khai thay đổi.

- **Giảm thiểu phụ thuộc:** MCP đóng vai trò như **lớp trừu tượng trung gian** giữa client và crawler, đảm bảo sự tách biệt rõ ràng giữa các tầng. Nhờ đó, mỗi phần của hệ thống có thể được phát triển, kiểm thử hoặc triển khai độc lập, giúp giảm độ phức tạp và nâng cao khả năng tái sử dụng mã nguồn.

- **Hiệu năng tối ưu:** Cơ chế **xử lý bất đồng bộ (asynchronous)** được kết hợp với **cache đa tầng (Redis + bộ nhớ tạm thời)** giúp hệ thống tận dụng tối đa tài nguyên, giảm độ trễ và hạn chế truy xuất lặp lại vào các nguồn dữ liệu tốn kém thời gian. Điều này đặc biệt hữu ích trong bối cảnh người dùng có hành vi tra cứu lặp lại hoặc xu hướng được quan tâm rộng rãi.

- **Khả năng mở rộng linh hoạt:** Với thiết kế theo hướng **plug-in architecture**, hệ thống có thể tích hợp dễ dàng các nguồn dữ liệu mới, công cụ AI bổ sung hoặc thuật toán phân tích nâng cao. Các điểm mở rộng (extension points) đã được định nghĩa rõ trong kiến trúc MCP, giúp việc thêm module mới chỉ cần cấu hình tối thiểu.

- **Khả năng thích ứng và duy trì ổn định:** Hệ thống chứng minh được khả năng vận hành liên tục với độ ổn định cao. Các bài kiểm thử dài hạn cho thấy hệ thống vẫn duy trì hiệu năng ổn định sau nhiều giờ hoạt động liên tục, không ghi nhận lỗi nghiêm trọng hay tình trạng nghẽn tài nguyên.

Những kết quả trên khẳng định rằng TrendApp không chỉ đạt hiệu quả trong giai đoạn thử nghiệm mà còn có nền tảng kỹ thuật vững chắc để mở rộng thành hệ thống phân tích xu hướng ở quy mô lớn trong tương lai.

4.3. Hạn chế và Hướng phát triển

Bên cạnh những kết quả tích cực, hệ thống vẫn tồn tại một số hạn chế cần khắc phục trong giai đoạn tiếp theo:

- **Phụ thuộc vào chất lượng dữ liệu đầu vào:** Vì hệ thống thu thập dữ liệu từ nhiều nền tảng mạng xã hội, nên **tính chính xác và nhất quán** của dữ liệu có thể bị ảnh hưởng bởi sự thay đổi API, spam hoặc nội dung nhiễu từ phía người dùng.

- **Chi phí xử lý mô hình LLM cao:** Việc phân tích sở thích người dùng và tổng hợp xu hướng dựa trên LLM tiêu tốn tài nguyên đáng kể. Khi mở rộng quy mô người dùng, chi phí tính toán có thể tăng nhanh, ảnh hưởng đến hiệu quả kinh tế.
- **Chưa hỗ trợ thời gian thực (real-time):** Hệ thống hiện mới hoạt động theo cơ chế batch, chưa phản ứng tức thì khi có xu hướng bùng nổ đột ngột trên mạng xã hội. Điều này hạn chế khả năng phát hiện sớm các xu hướng mới nổi.

Để khắc phục các hạn chế trên, nhóm dự án đề xuất một số hướng phát triển trong giai đoạn tới:

1. **Tối ưu pipeline xử lý dữ liệu** theo hướng **streaming** để giảm độ trễ và hỗ trợ cập nhật xu hướng theo thời gian thực.
2. **Bổ sung dashboard trực quan hóa xu hướng**, giúp người dùng quan sát, so sánh và phân tích dữ liệu theo từng chủ đề hoặc nền tảng.
3. **Tích hợp thêm các nguồn dữ liệu mở rộng** như **X (Twitter)**, **Google Trends** và các diễn đàn chuyên ngành để tăng tính toàn diện của hệ thống.
4. **Áp dụng cơ chế caching và sampling thông minh** nhằm cân bằng giữa chi phí vận hành và chất lượng gợi ý.

Những định hướng này không chỉ giúp khắc phục hạn chế hiện tại mà còn mở ra khả năng đưa TrendApp trở thành một nền tảng phân tích xu hướng xã hội có tính thích ứng cao và ứng dụng thực tiễn rộng rãi.

4.4. Bảng phân công công việc

Tuần	Trương Thành Thắng	Tạ Ngọc Sơn	Đỗ Thị Diệu Lê
1	- Thiết kế database - Setup FastAPI - Tạo base models	- Nghiên cứu FastMCP - Setup MCP server - Viết crawler Reddit	- Setup React + TypeScript - Thiết kế UI components - Cài đặt Redis
2	- API user management - Azure Entra ID integration - Post controller	- Crawler YouTube, News - Xử lý data pipeline - Gemini API integration	- Pages: Login, Feed, Profile - Kết nối API - Taskiq background tasks
3	- Feed recommendation API - Interest tracking - Cache integration	- Tag extraction với Gemini - Content analysis - MCP protocol optimization	- Real-time feed display - Search interface - Docker containerization

Tuần	Trương Thành Thắng	Tạ Ngọc Sơn	Đỗ Thị Diệu Lê
4	- Testing & bug fixes - Performance optimization - API documentation	- Testing crawlers - Fine-tuning LLM prompts - Data validation	- Deploy Azure Web App - Monitoring & logging - Final demo preparation

5. Kết luận và Hướng phát triển

5.1. Kết luận

Đồ án đã xây dựng và thử nghiệm thành công phiên bản đầu tiên của *TrendApp – MCP Crawler Tool,* ứng dụng *Model Context Protocol (MCP)* trong việc gợi ý xu hướng xã hội cá nhân hóa. Hệ thống cho phép thu thập, phân tích và tổng hợp dữ liệu từ nhiều nguồn khác nhau, đồng thời duy trì tính linh hoạt trong phát triển nhờ kiến trúc module hóa.

Kết quả thử nghiệm cho thấy việc áp dụng MCP giúp giảm thời gian xử lý, cải thiện hiệu năng và khả năng mở rộng, đáp ứng yêu cầu cơ bản về tốc độ và độ ổn định. Mặc dù còn một số hạn chế về chi phí tính toán và khả năng cập nhật theo thời gian thực, mô hình hiện tại đã chứng minh được tính khả thi và định hướng đúng đắn của đề tài.

5.2. Hướng phát triển

Trong giai đoạn tiếp theo, nhóm dự án dự kiến:

- Nâng cấp pipeline xử lý dữ liệu sang mô hình streaming để hỗ trợ cập nhật xu hướng gần thời gian thực.

- Vector hóa sở thích user để phục vụ cho các tính năng mở rộng như: gợi ý kết bạn, tự động đề xuất feed.

- Bổ sung thêm các nguồn dữ liệu như X (Twitter) và Tiktok để tăng độ phủ và tính đa dạng.

- Tối ưu chi phí và hiệu năng của mô hình LLM bằng cách áp dụng caching thông minh hoặc mô hình nhẹ hơn cho các truy vấn đơn giản.

Những hướng phát triển này nhằm hoàn thiện hệ thống theo hướng thực tiễn hơn, đảm bảo khả năng ứng dụng lâu dài và hiệu quả trong việc phân tích, gợi ý xu hướng xã hội.

References

1. *Introducing the Model Context Protocol*. (2024, November 25). Anthropic. Retrieved November 7, 2025, from https://www.anthropic.com/news/model-context-protocol
2. *Web App Service*. (n.d.). Microsoft Azure. Retrieved November 7, 2025, from https://azure.microsoft.com/en-us/products/app-service/web
3. *Anthropic. (2024, November 25). Introducing the Model Context Protocol (MCP).* Anthropic News. Retrieved November 7, 2025, from https://www.anthropic.com/news/model-context-protocol
4. *Microsoft Azure. (n.d.). Web App Service.* Retrieved November 7, 2025, from https://azure.microsoft.com/en-us/products/app-service/web
5. *FastAPI. (n.d.). FastAPI: Modern, Fast (high-performance) web framework for building APIs with Python 3.7+*. Retrieved November 7, 2025, from https://fastapi.tiangolo.com
6. *Redis. (n.d.). Redis: In-memory data store, used as a database, cache, and message broker*. Retrieved November 7, 2025, from https://redis.io
7. *TaskIQ. (2024). TaskIQ Documentation: Distributed background task processing for Python.* Retrieved November 7, 2025, from https://taskiq-python.github.io
8. *ClickHouse. (n.d.). ClickHouse: Open-source column-oriented database management system.* Retrieved November 7, 2025, from https://clickhouse.com
9. *Apache Airflow. (n.d.). Airflow: Platform to programmatically author, schedule, and monitor workflows. Apache Software Foundation*. Retrieved November 7, 2025, from https://airflow.apache.org
10. *Google. (2024). Gemini: A family of multimodal large language models by Google DeepMind*. Retrieved November 7, 2025, from https://deepmind.google/technologies/gemini
11. R*eact. (n.d.). React: A JavaScript library for building user interfaces. Meta Platforms, Inc.* Retrieved November 7, 2025, from https://react.dev
12. *PostgreSQL. (n.d.). PostgreSQL: The world's most advanced open source relational database*. Retrieved November 7, 2025, from https://www.postgresql.org