Tự động hóa Xử lý Email Hỗ trợ Khách hàng bằng AI

Hoàng Anh Tuấn[1], Nguyễn Hữu Luân[1], Nguyễn Cao Bách[1]

Abstract

Việc xử lý thủ công khối lượng lớn email hỗ trợ khách hàng là một thách thức về hiệu quả và chi phí. Dự án này trình bày một giải pháp *Tích hợp được Tăng cường bởi AI (AI-Enhanced Integration)*, tự động hóa quy trình xử lý email bằng cách đưa Trí tuệ Nhân tạo (AI) vào vai trò trung tâm điều phối và ra quyết định. Hệ thống tích hợp chặt chẽ Gmail API và Google Cloud Pub/Sub để tiếp nhận email theo thời gian thực. AI của Google Gemini (`gemini-2.0-flash`) không chỉ phân loại nội dung email (Spam/Cần xử lý) mà còn đưa ra nội dung chính mà người dùng muốn hỏi về. Điểm cốt lõi là khả năng tạo truy vấn, khi AI có khả năng hiểu cách sử dụng Ebay API và xây dựng tham số truy vấn hợp lý từ thông tin tổng hợp được từ email của khách hàng để tìm kiếm kết quả đúng theo mong muốn. AI tiếp tục đánh giá mức độ phù hợp của thông tin tổng hợp được trả về từ API. Nếu đủ, Gemini tự động tổng hợp và sinh phản hồi phù hợp ngữ cảnh, tích hợp ngược lại vào luồng email qua Gmail API. Nếu không đủ thông tin hoặc yêu cầu phức tạp, AI sẽ kích hoạt quy trình tích hợp với con người, chuyển tiếp thông minh đến nhóm hỗ trợ qua Telegram Bot. Giải pháp AI-Enhanced Integration này hứa hẹn cách mạng hóa quy trình hỗ trợ, giảm thiểu thời gian phản hồi, tăng tính nhất quán và nâng cao trải nghiệm khách hàng.

Keywords: AI-Enhanced Integration, Tự động hóa Email, Hỗ trợ khách hàng, Trí tuệ nhân tạo (AI), Xử lý ngôn ngữ tự nhiên (NLP), Google Gemini, Gmail API, Google Cloud Pub/Sub, Ebay API, Telegram Bot, Phân loại Email, Sinh phản hồi, Tích hợp thông minh, `gemini-2.0-flash`,

1. Giới thiệu

Trong bối cảnh các doanh nghiệp ngày càng phụ thuộc vào tương tác số, việc xử lý hiệu quả các yêu cầu hỗ trợ khách hàng qua email trở nên tối quan trọng. Tuy nhiên, các phương pháp tích hợp và xử lý truyền thống thường gặp giới hạn về tốc độ, khả năng hiểu ngữ cảnh và chi phí vận hành khi đối mặt với khối lượng lớn email. Các quy trình thủ công hoặc dựa trên quy tắc cứng nhắc tỏ ra kém linh hoạt, dẫn đến chậm trễ và thiếu nhất quán trong phản hồi, ảnh hưởng trực tiếp đến sự hài lòng của khách hàng.

Dự án này giới thiệu một cách tiếp cận đột phá: *Tích hợp được Tăng cường bởi AI (AI-Enhanced Integration)* để tự động hóa và thông minh hóa quy trình hỗ trợ khách hàng qua email. Thay vì xem AI chỉ như một công cụ thực thi tác vụ riêng lẻ, giải pháp này đặt AI vào vị trí trung tâm, chủ động điều phối và làm phong phú chính quá trình tích hợp dữ liệu và quy trình làm việc. Mục tiêu không chỉ là tự động hóa, mà còn là tạo ra một hệ thống tích hợp thông minh, có khả năng hiểu ngữ nghĩa, đưa ra quyết định dựa trên ngữ cảnh và tương tác linh hoạt giữa các dịch vụ đám mây, API và con người.

Giải pháp tận dụng hệ sinh thái công nghệ tiên tiến, bao gồm:

- **Google Gemini:** Sử dụng các mô hình ngôn ngữ lớn (LLM - `gemini-2.0-flash`)) để cung cấp năng lực hiểu ngôn ngữ tự nhiên, phân loại, tìm kiếm ngữ nghĩa, đánh giá thông tin và sinh văn bản.
- **Gmail API & Google Cloud Pub/Sub:** Tạo thành xương sống cho việc tiếp nhận email theo thời gian thực (thông qua cơ chế push notification của Pub/Sub) và thực hiện các hành động gửi/nhận/quản lý email một cách tự động.
- **Ebay API:** API của nền tảng thương mại điện tử Ebay, cho phép nhà phát triển tích hợp các chức năng của Ebay như tìm kiếm sản phẩm, mua, bán, quản lý sản phẩm... vào ứng dụng của mình.
- **Telegram Bot API:** Cung cấp kênh tích hợp liền mạch để chuyển tiếp các trường hợp phức tạp hoặc cần sự can thiệp của nhân viên hỗ trợ.

AI đóng vai trò then chốt trong việc định hình luồng tích hợp: từ việc phân tích và định tuyến thông minh email đầu vào, thực hiện tạo truy vấn hợp lý dựa theo tri thức thu được từ email, tổng hợp thông tin để tạo phản hồi động, cho đến việc quyết định thời điểm và cách thức tích hợp với quy trình xử lý thủ công.

Phạm vi của dự án tập trung vào việc xây dựng và tích hợp các thành phần cốt lõi này thành một hệ thống tự động hóa hoàn chỉnh. Dự án không bao gồm việc xây dựng giao diện quản lý người dùng cuối, tích hợp sâu với các hệ

Email addresses: `23025047@vnu.edu.vn` (Hoàng Anh Tuấn), `23025045@vnu.edu.vn` (Nguyễn Hữu Luân), `24025003@vnu.edu.vn` (Nguyễn Cao Bách)

thống CRM bên ngoài, hoặc thực hiện huấn luyện/tinh chỉnh chuyên sâu các mô hình AI (tận dụng các mô hình có sẵn qua API).

Báo cáo này sẽ trình bày chi tiết kiến trúc AI-Enhanced Integration được đề xuất, bao gồm thiết kế các thành phần, luồng xử lý dữ liệu thông minh, hướng dẫn cài đặt và triển khai, cùng với việc đánh giá các khía cạnh quan trọng như hiệu suất, khả năng mở rộng và bảo mật của giải pháp tích hợp được tăng cường bởi AI này.

2. Phân tích và Thiết kế Hệ thống

Hệ thống "Tự động hóa Xử lý Email Hỗ trợ Khách hàng bằng AI" được thiết kế với triết lý cốt lõi là **Tích hợp được Tăng cường bởi AI (AI-Enhanced Integration)**. Kiến trúc này tập trung vào việc sử dụng Trí tuệ Nhân tạo (AI) để làm cho chính quá trình tích hợp dữ liệu và luồng quy trình trở nên thông minh, linh hoạt và đáp ứng nhanh chóng các yêu cầu. Thay vì chỉ tự động hóa các bước rời rạc, AI đóng vai trò trung tâm trong việc phân tích ngữ nghĩa, đưa ra quyết định tích hợp thông minh, và điều phối các tương tác giữa các dịch vụ đám mây và các API bên thứ ba. Việc phân tách rõ ràng trách nhiệm giữa các thành phần module hóa, với AI là nhân tố kết nối và ra quyết định, tạo điều kiện cho việc phát triển, bảo trì và mở rộng hệ thống một cách hiệu quả.

2.1. Yêu cầu Hệ thống (System Requirements)

Phần này mô tả chi tiết các yêu cầu chức năng và phi chức năng mà hệ thống "Tự động hóa Xử lý Email Hỗ trợ Khách hàng bằng AI" cần đáp ứng.

2.1.1. Yêu cầu Chức năng (Functional Requirements)

Các khả năng và hành vi cụ thể mà hệ thống phải thực hiện:

- Hệ thống phải tự động tiếp nhận email mới từ một tài khoản Gmail được chỉ định, gần như thời gian thực thông qua Google Cloud Pub/Sub.
- Hệ thống phải trích xuất được các thông tin cốt lõi từ email nhận được: địa chỉ người gửi, tiêu đề, nội dung chính (ưu tiên plain text), Message ID và Thread ID.
- Hệ thống phải sử dụng Google Gemini để phân loại email thành các loại: SPAM/Không liên quan hoặc Cần xử lý (PROCESS).
- Đối với email SPAM, hệ thống phải tự động đánh dấu là đã đọc và dừng xử lý.
- Đối với email PROCESS, hệ thống phải xây dựng được tham số truy vấn hợp lý dựa theo tài liệu hướng dẫn sử dụng API của Ebay kết hợp với nội dung email từ người dùng (thông qua việc sử dụng Gemini AI).
- Hệ thống phải thực hiện gọi API đến Ebay với các tham số đã được quyết định ở trên, nhận về các kết quả mà API trả về.
- Sau khi thu thập được thông tin, hệ thống phải dùng Google Gemini để đánh giá xem thông tin tìm được có đủ và liên quan để trả lời đầy đủ yêu cầu của khách hàng không.
- Nếu thông tin được đánh giá là đủ liên quan, hệ thống phải dùng Google Gemini (LLM) để tự động tạo nội dung email phản hồi dựa trên ngữ cảnh email gốc và thông tin truy xuất, đảm bảo tính hữu ích, lịch sự, chính xác.
- Hệ thống phải gửi email phản hồi đã tạo đến người gửi ban đầu qua Gmail API, đảm bảo trả lời đúng luồng thư gốc.
- Hệ thống phải tự động chuyển tiếp thông tin email (người gửi, tiêu đề, nội dung tóm tắt, link Gmail) đến nhóm chat Telegram hỗ trợ trong các trường hợp: không thể phân loại email, không tìm thấy thông tin liên quan, thông tin không đủ liên quan, gọi API không thành công hoặc tạo phản hồi tự động thất bại.
- Sau khi phản hồi tự động thành công hoặc chuyển tiếp thành công, hệ thống phải đánh dấu email đó là đã đọc trong Gmail.
- Hệ thống phải ghi log các bước xử lý chính, các quyết định quan trọng, và các lỗi xảy ra.

2.1.2. Yêu cầu Phi chức năng (Non-functional Requirements)

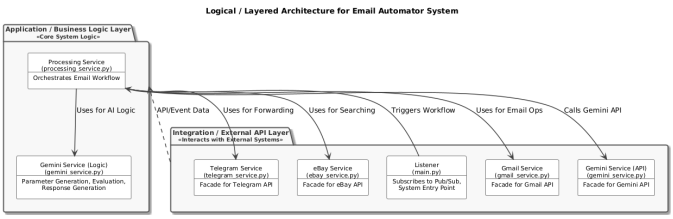
Các tiêu chuẩn về chất lượng, hiệu suất và cách thức hoạt động của hệ thống:

- **Hiệu suất:**
  - Thời gian trung bình từ lúc nhận email mới đến khi gửi phản hồi tự động (cho trường hợp thành công) không nên vượt quá 30-60 giây.
  - Thời gian từ lúc quyết định chuyển tiếp đến khi tin nhắn xuất hiện trên Telegram không nên vượt quá 10-20 giây.
- **Độ Chính xác của AI:**
  - Tỷ lệ phân loại email (SPAM/PROCESS) chính xác cần đạt tối thiểu 90-95%.
  - Phản hồi tự động do AI tạo ra phải hữu ích, lịch sự, phù hợp ngữ cảnh, không sai lệch (đánh giá định tính).
- **Khả năng Mở rộng:** Kiến trúc phải cho phép chạy nhiều instance ứng dụng song song để xử lý lượng email tăng lên.

- **Độ Tin cậy và Tính Sẵn sàng:** Hệ thống cần hoạt động ổn định, có cơ chế retry cho các lỗi API tạm thời, và không làm mất email yêu cầu khi có lỗi nghiêm trọng (Pub/Sub giữ lại message).
- **Bảo mật:**
  - Thông tin nhạy cảm (API keys, tokens) phải được lưu trữ và truy cập an toàn (biến môi trường, file `.env` bảo vệ, hoặc lý tưởng là dịch vụ quản lý bí mật), không hardcode.
  - Cấp quyền tối thiểu cần thiết cho các tài khoản dịch vụ và OAuth client.
  - Ý thức về việc xử lý và lưu trữ (nếu có) thông tin cá nhân nhạy cảm (PII) trong email.
- **Khả năng Bảo trì:** Mã nguồn phải được tổ chức rõ ràng, module hóa, có bình luận. Cấu hình hệ thống quan trọng phải dễ thay đổi mà không cần sửa code.
- **Tính Dễ sử dụng (cho đội ngũ hỗ trợ):** Thông báo Telegram phải rõ ràng, cung cấp đủ thông tin (người gửi, tiêu đề, tóm tắt nội dung, link Gmail) để nhân viên hỗ trợ xử lý nhanh.
- **Giới hạn Tài nguyên:** Hệ thống phải hoạt động trong giới hạn tần suất yêu cầu (rate limits) và quotas của các API sử dụng, có cơ chế xử lý khi vượt giới hạn.

2.2. Sơ đồ Kiến trúc Phân Lớp (Layered Architecture)

Kiến trúc logic của hệ thống được tổ chức thành các lớp chức năng riêng biệt, mỗi lớp đảm nhiệm một tập hợp các trách nhiệm cụ thể. Cách tiếp cận này giúp tăng tính module, dễ bảo trì và phát triển. Luồng tương tác chính giữa các lớp thường theo chiều từ trên xuống: Integration Layer nhận sự kiện và dữ liệu, chuyển cho Application Layer xử lý logic. Application Layer cũng sử dụng lại Integration Layer để gửi phản hồi hoặc thông báo ra bên ngoài.



Hình 1: Biểu đồ Kiến trúc Phân Lớp

- **Lớp Tích hợp / API Bên ngoài (Integration / External API Layer):** Lớp này đóng vai trò là cổng giao tiếp giữa ứng dụng và các hệ thống/dịch vụ bên ngoài (Gmail, Google Cloud Pub/Sub, Google Gemini API, Telegram API). Nó bao gồm các thành phần chịu trách nhiệm lắng nghe sự kiện, nhận và gửi dữ liệu qua các API.
  - *Listener (main.py):* Thành phần đầu vào, chạy liên tục và đăng ký (subscribe) vào Google Cloud Pub/Sub subscription. Khi nhận được thông báo về email mới (do Pub/Sub đẩy đến), nó khởi tạo luồng xử lý chính bằng cách gọi `processing_service.py`. Đồng thời, nó chịu trách nhiệm khởi tạo các client cần thiết cho Gmail và Pub/Sub khi ứng dụng khởi động.
  - *Gmail Service (gmail_ service.py):* Đóng vai trò Facade, trừu tượng hóa việc tương tác phức tạp với Gmail API. Các trách nhiệm chính bao gồm: xác thực người dùng qua OAuth 2.0 (lưu và làm mới token trong `token.json`); lấy chi tiết nội dung email (người gửi, tiêu đề, nội dung plain text, message ID, thread ID); gửi email phản hồi, đảm bảo reply đúng thread của email gốc; và đánh dấu email là đã đọc (xóa nhãn 'UNREAD').
  - *Gemini Service (API Facade - phần trong gemini _ service.py):* Là Facade cho các tương tác API với Google Gemini. Nó đóng gói việc tạo và gửi các yêu cầu đến Gemini API, bao gồm các tác vụ như: gửi yêu cầu phân loại email, yêu cầu tạo tham số truy vấn, yêu cầu đánh giá mức độ liên quan, và yêu cầu sinh nội dung văn bản. Nó cũng xử lý các phản hồi từ Gemini API và logic retry nếu cần.
  - *Ebay Service (API Facade - phần trong ebay _ service.py):* Là Facade cho các tương tác API với Ebay. Nó đóng gói việc tạo và gửi yêu cầu truy vấn về thông tin sản phẩm, nhận dữ liệu trả về và format lại thành định dạng phù hợp hơn để xử lý
  - *Telegram Service (telegram_ service.py):* Facade cho Telegram Bot API. Nó nhận thông tin email cần chuyển tiếp, định dạng thông tin này (người gửi, tiêu đề, trích dẫn nội dung, link Gmail) thành tin nhắn Markdown dễ đọc, và gửi tin nhắn đến chat ID của nhóm hỗ trợ đã được cấu hình (`TELEGRAM_CHAT_ID`) thông qua bot token (`TELEGRAM_BOT_TOKEN`).
- **Lớp Ứng dụng / Logic Nghiệp vụ (Application / Business Logic Layer):** Đây là trái tim của hệ thống, chứa đựng logic nghiệp vụ cốt lõi và điều phối các hoạt động để xử lý email.
  - *Processing Service (processing_ service.py):* Module điều phối trung tâm, thực hiện logic xử lý chính cho mỗi email được kích hoạt bởi Listener. Quy trình của nó bao gồm: nhận thông tin email từ `Gmail Service`; gọi `Gemini Service` để phân loại email (SPAM/PROCESS) và phân loại ý định (FAQ/Product); nếu là PROCESS, tiếp tục gọi `Gemini Service` để trích xuất thuật ngữ sản phẩm (nếu cần) và tạo vector nhúng;

gọi `Database Service` để tìm kiếm thông tin liên quan trong cơ sở tri thức; gọi lại `Gemini Service` để đánh giá độ liên quan của thông tin tìm được; và cuối cùng, dựa trên kết quả đánh giá, quyết định tạo và gửi phản hồi tự động (qua `Gemini Service` và `Gmail Service`) hoặc chuyển tiếp email đến nhóm hỗ trợ (qua `Telegram Service`). Nó cũng đảm bảo email được đánh dấu đã đọc sau khi xử lý thành công.

- *Gemini Service (Logic/Embedding - phần trong gemini_service.py):* Phần này của Gemini Service chứa logic nghiệp vụ liên quan đến AI không chỉ đơn thuần là gọi API. Ví dụ: quyết định nội dung nào (toàn bộ email hay chỉ thuật ngữ sản phẩm) sẽ được dùng để tạo embedding; xử lý và diễn giải kết quả phân loại, kết quả đánh giá độ liên quan từ API để hỗ trợ `Processing Service` đưa ra quyết định tiếp theo trong luồng.

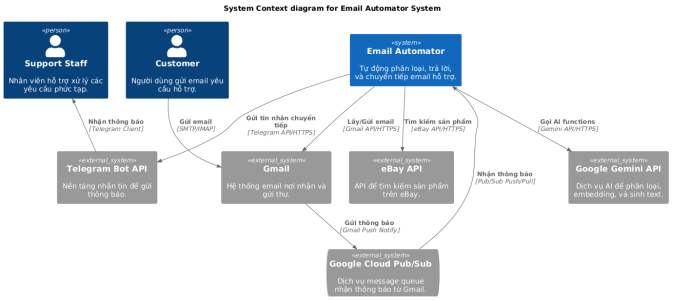
- **Lớp Truy cập Dữ liệu (Data Access / Persistence Layer):** Lớp này trừu tượng hóa việc truy cập và thao tác với nguồn dữ liệu, cụ thể là Knowledge Base.

2.3. Kiến trúc Chi tiết theo Mô hình C4 (C4 Model View)

Để hiểu rõ hơn về cấu trúc và các tương tác ở các mức độ chi tiết khác nhau, chúng ta sử dụng Mô hình C4. Mô hình này cung cấp các sơ đồ có mức độ zoom khác nhau vào kiến trúc phần mềm, giúp làm rõ trách nhiệm và mối quan hệ của từng thành phần.

2.3.1. Sơ đồ Ngữ cảnh Hệ thống (System Context Diagram - C4 Level 1)

Sơ đồ Ngữ cảnh cho thấy cái nhìn tổng quan nhất về hệ thống Email Automator và vị trí của nó trong môi trường hoạt động. Hệ thống được biểu diễn như một hộp đen duy nhất, tương tác với các Người dùng (Actors) và các Hệ thống Phần mềm Bên ngoài (External Software Systems) mà nó phụ thuộc vào.



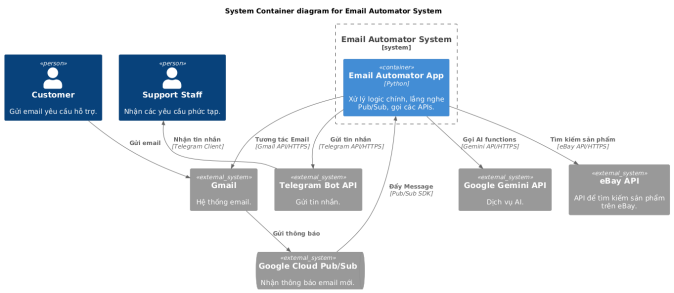
Hình 2: Sơ đồ Ngữ cảnh Hệ thống (C4 Level 1)

- **Người dùng (Actors):**
  - *Customer*: Người dùng cuối, gửi email yêu cầu hỗ trợ đến địa chỉ email được hệ thống giám sát.
  - *Support Staff*: Nhân viên hỗ trợ khách hàng, nhận thông báo về các trường hợp email phức tạp hoặc không thể tự động xử lý được qua kênh Telegram.
- **Hệ thống Bên ngoài (External Systems):**
  - *Gmail*: Hệ thống email chính nơi các yêu cầu hỗ trợ được nhận và các phản hồi được gửi đi. Email Automator sử dụng Gmail API để đọc và gửi thư.
  - *Google Cloud Pub/Sub*: Dịch vụ message queue hoạt động như một bus sự kiện. Gmail gửi thông báo (push notification) đến Pub/Sub khi có email mới, và Email Automator lắng nghe các thông báo này.
  - *Google Gemini API*: Dịch vụ Trí tuệ Nhân tạo cung cấp các mô hình ngôn ngữ lớn (LLM) để phân loại email, hiểu ý định, tạo tham số truy vấn theo ý định, đánh giá độ liên quan và sinh nội dung phản hồi.
  - *Ebay API*: Sàn thương mại điện tử cung cấp thông tin của các sản phẩm, hỗ trợ truy vấn đa dạng
  - *Telegram Bot API*: Nền tảng nhắn tin cho phép Email Automator gửi thông báo và chi tiết email đến một nhóm chat của Support Staff.

Sơ đồ này nhấn mạnh các tương tác tại ranh giới của hệ thống Email Automator, cho thấy luồng dữ liệu vào/ra chính.

2.3.2. Sơ đồ Container (Container Diagram - C4 Level 2)

Sơ đồ Container làm rõ các thành phần chính có thể triển khai (deployable units hoặc các khối logic lớn) bên trong và cách chúng tương tác.



Hình 3: Sơ đồ Container (C4 Level 2)

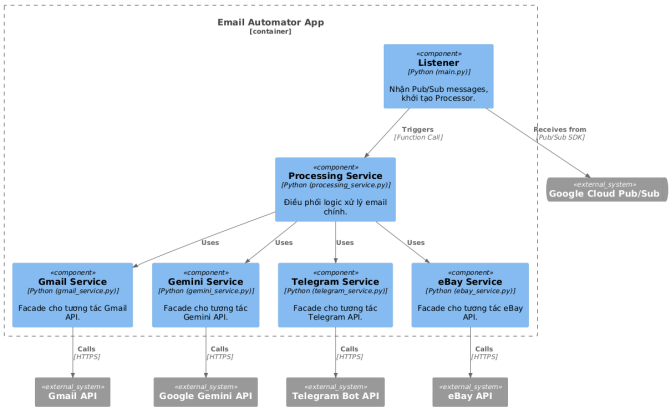
- **Email Automator App (Container - Python Application):** Đây là ứng dụng Python chính, thực thi toàn bộ logic nghiệp vụ của hệ thống.

Nó bao gồm các module như Listener (lắng nghe Pub/Sub), Processing Service (điều phối), và các Service Facade (Gmail, Gemini, Ebay, Telegram). Container này chịu trách nhiệm nhận thông báo, xử lý email, tương tác với AI, gọi API tìm kiếm và gửi phản hồi/thông báo. Công nghệ chính là Python và các thư viện liên quan.

Mối quan hệ chính trong sơ đồ này là 'Email Automator App' duy trì các tương tác với các hệ thống bên ngoài (Gmail, Pub/Sub, Gemini, Telegram) đã được xác định ở Level 1.

2.3.3. Sơ đồ Component (Component Diagram - C4 Level 3)

Sơ đồ Component tập trung vào cấu trúc bên trong container 'Email Automator App', chi tiết hóa các module code chính (component) và cách chúng được tổ chức và tương tác với nhau.



Hình 4: Sơ đồ Component (C4 Level 3)

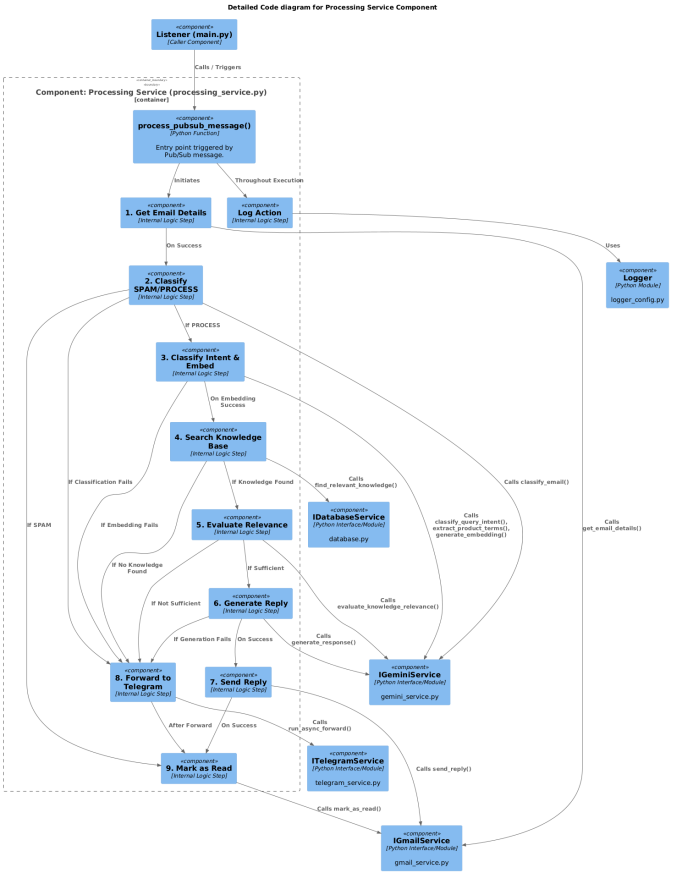
Các component chính trong 'Email Automator App' bao gồm:

- **Listener (main.py):** Component đầu vào của ứng dụng. Chức năng chính là khởi tạo và duy trì kết nối lắng nghe (subscribe) với Google Cloud Pub/Sub subscription. Khi nhận được một message mới từ Pub/Sub (thông báo có email mới), Listener sẽ kích hoạt `Processing Service` để bắt đầu xử lý email đó. Nó cũng chịu trách nhiệm khởi tạo các client dùng chung (ví dụ: Gmail client, Pub/Sub client) khi ứng dụng bắt đầu.
- **Processing Service (processing_service.py):** Là bộ não điều phối chính của ứng dụng. Component này chứa logic nghiệp vụ cốt lõi để xử lý một email từ đầu đến cuối. Nó nhận thông tin email, sau đó tuần tự gọi các service khác: `Gmail Service` để lấy chi tiết email; `Gemini Service` để phân loại, tạo embedding, đánh giá; `Database Service` để tìm kiếm thông tin trong Knowledge Base; và cuối cùng quyết định gửi phản hồi qua `Gmail Service` hoặc chuyển tiếp qua `Telegram Service`.
- **Gmail Service (gmail_service.py):** Đóng vai trò là một Facade, cung cấp một giao diện lập trình ứng dụng (API) đơn giản và tập trung cho tất cả các tương tác cần thiết với Gmail API của Google. Nó ẩn đi sự phức tạp của việc xác thực OAuth 2.0, tạo và gửi các request HTTP, cũng như xử lý các response từ Gmail API cho các tác vụ như đọc nội dung email, gửi email, và sửa đổi nhãn email (ví dụ, đánh dấu đã đọc).
- **Gemini Service (gemini_service.py):** Tương tự Gmail Service, đây là một Facade cho Google Gemini API. Nó đóng gói logic gọi đến các mô hình AI của Gemini cho các nhiệm vụ: phân loại email (SPAM/PROCESS), trích xuất thuật ngữ sản phẩm, tạo ra tham số truy vấn dựa trên nội dung mail, đánh giá mức độ liên quan của thông tin, và sinh ra nội dung email phản hồi một cách tự nhiên.
- **Ebay Service (ebay_service):** Facade cho Ebay API. Nó đóng gói logic gọi truy vấn tìm kiếm sản phẩm đến Ebay, nhận phản hồi và format dữ liệu nhận được theo định dạng
- **Telegram Service (telegram_service.py):** Facade cho việc gửi tin nhắn thông qua Telegram Bot API. Nó nhận dữ liệu email cần chuyển tiếp, định dạng thành một tin nhắn thân thiện với người dùng cho Support Staff, và gửi tin nhắn đó đến kênh Telegram đã định trước.

Trong kiến trúc này, `Processing Service` là component trung tâm, sử dụng các service (Facade) khác để thực hiện các tác vụ chuyên biệt.

2.3.4. Sơ đồ Mã nguồn (Code Diagram - C4 Level 4) cho Processing Service

Sơ đồ này đi sâu vào chi tiết cấu trúc bên trong component `Processing Service`, cụ thể là hàm hoặc lớp chính điều phối logic `process_pubsub_message()`.



Hình 5: Sơ đồ Mã nguồn cho Processing Service (C4 Level 4)

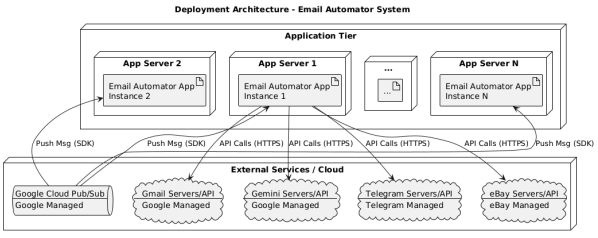
Hàm `process_pubsub_message()` thực hiện một chuỗi các bước logic, mỗi bước có thể gọi đến các phương thức của các interface/module phụ thuộc (được đại diện bởi các Facade service):

1. **Get Email Details:** Phương thức `process_pubsub_message()` trong `processing_service.py` khởi tạo luồng xử lý. Bước này gọi phương thức `get_email_details()` của `Gmail Service` (`gmail_service.py`) để lấy thông tin chi tiết từ email.
2. **Classify SPAM/PROCESS:** Bước này trong `processing_service.py` gọi phương thức `classify_email()` của `Gemini Service` (`gemini_service.py`) để phân loại email là SPAM hoặc PROCESS.
3. **Generate eBay Search Params (cho email PROCESS):** Nếu email được phân loại là PROCESS, luồng xử lý tiếp tục đến bước này trong `processing_service.py`, gọi phương thức `generate_ebay_search_params()` của `eBay Service` (`ebay_service.py`) để tạo các tham số tìm kiếm trên eBay.
4. **Perform eBay Search:** Nếu các tham số tìm kiếm được tạo thành công, bước này trong `processing_service.py` gọi phương thức `search_ebay_items()` của `eBay Service` (`ebay_service.py`) để tìm kiếm các mặt hàng trên eBay.
5. **Evaluate Relevance (using eBay Items):** Nếu tìm thấy các mặt hàng, bước này trong `processing_service.py` gọi phương thức `evaluate_knowledge_relevance()` của `Gemini Service` (`gemini_service.py`) để đánh giá mức độ liên quan của các mặt hàng eBay tìm được.
6. **Generate Reply (using eBay Items):** Nếu mức độ liên quan đủ, bước này trong `processing_service.py` gọi phương thức `generate_response()` của `Gemini Service` (`gemini_service.py`) để tạo nội dung email trả lời dựa trên các mặt hàng eBay.
7. **Send Reply:** Nếu nội dung trả lời được tạo thành công, bước này trong `processing_service.py` gọi phương thức `send_reply()` của `Gmail Service` (`gmail_service.py`) để gửi email phản hồi.
8. **Forward to Telegram:** Nếu phân loại thất bại, tạo tham số thất bại, không tìm thấy mặt hàng, mức độ liên quan không đủ, hoặc tạo trả lời thất bại, luồng xử lý chuyển đến bước này trong `processing_service.py`, gọi phương thức `run_async_forward()` của `Telegram Service` (`telegram_service.py`) để chuyển tiếp thông tin email đến Telegram.
9. **Mark as Read:** Sau khi xử lý thành công (gửi trả lời hoặc chuyển tiếp thành công), bước này trong `processing_service.py` gọi phương thức `mark_as_read()` của `Gmail Service` (`gmail_service.py`) để đánh dấu email là đã đọc.

Xuyên suốt quá trình này, các hành động quan trọng và lỗi phát sinh đều được ghi lại thông qua một module Logger.

2.4. Sơ đồ Kiến trúc Triển khai (Deployment Architecture)

Kiến trúc triển khai mô tả cách các thành phần phần mềm được ánh xạ lên hạ tầng vật lý hoặc các môi trường runtime, cho thấy nơi các phần mềm chạy và cách chúng kết nối.



Hình 6: Biểu đồ Kiến trúc Triển khai

Hệ thống được triển khai qua các tầng vật lý/logic như sau:

– **Tầng Dịch vụ Bên ngoài / Đám mây (External Services / Cloud):** Bao gồm các dịch vụ API

và máy chủ do các nhà cung cấp bên thứ ba quản lý và vận hành (Google, Telegram, Ebay). Hệ thống Email Automator tương tác với các dịch vụ này qua mạng internet, sử dụng thư viện được cung cấp sẵn của Python, wrap trên các API được cung cấp (HTTPS). Các dịch vụ này bao gồm Gmail Servers/API, Google Cloud Pub/Sub, Gemini Servers/API, Ebay Servers/API và Telegram Servers/API.

- **Tầng Ứng dụng (Application Tier):** Đây là nơi ứng dụng `Email Automator App` (mã nguồn Python) được triển khai và thực thi. Tầng này có thể bao gồm một hoặc nhiều **Máy chủ Ứng dụng (App Server 1...N)**. Mỗi máy chủ ứng dụng sẽ chạy một hoặc nhiều instance của ứng dụng Python. Các instance này đều đăng ký và lắng nghe tin nhắn từ Google Cloud Pub/Sub. Để đảm bảo khả năng chịu lỗi và mở rộng, có thể sử dụng các công nghệ container hóa (như Docker) và các hệ thống điều phối container (như Kubernetes), hoặc các dịch vụ PaaS/Serverless của Google Cloud (Cloud Run, App Engine) để triển khai ứng dụng.

Khả năng mở rộng của Tầng Ứng dụng có thể đạt được bằng cách tăng số lượng instance ứng dụng (horizontal scaling), và Google Cloud Pub/Sub sẽ tự động phân phối tải đến các listener đang hoạt động. Tầng Cơ sở dữ liệu có thể được mở rộng theo chiều dọc (vertical scaling - nâng cấp tài nguyên máy chủ) hoặc theo chiều ngang (horizontal scaling - sử dụng các kỹ thuật như replication, sharding, tùy thuộc vào dịch vụ DB được sử dụng).

2.5. Luồng xử lý chính

Luồng xử lý email tự động diễn ra theo các bước sau):

1. **Nhận thông báo:** Email mới đến tài khoản Gmail đích. Gmail gửi thông báo đến Google Cloud Pub/Sub Topic đã cấu hình.
2. **Kích hoạt Listener:** Pub/Sub đẩy thông báo đến Subscription mà `main.py` đang lắng nghe. Callback `process_message_callback` trong `main.py` được thực thi.
3. **Lấy chi tiết Email:** `processing_service` (được gọi bởi callback) yêu cầu `gmail_service` lấy nội dung chi tiết của email dựa trên thông tin từ Pub/Sub (thường là gián tiếp thông qua việc list email unread mới nhất).
4. **Phân loại Email:** `processing_service` gửi tiêu đề và nội dung email đến `gemini_service.classify_email`.
5. **Xử lý SPAM:** Nếu kết quả là SPAM, luồng xử lý chuyển đến bước Mark as Read.
6. **Generate eBay Search Params (Nếu là PROCESS):** Nếu là PROCESS, `processing_service` gửi nội dung email đến `ebay_service.generate_ebay_search_params` để tạo các tham số tìm kiếm trên eBay.
7. **Perform eBay Search:** Nếu các tham số tìm kiếm được tạo thành công, `processing_service` gọi `ebay_service.search_ebay_items` với các tham số đã tạo để tìm kiếm các mặt hàng trên eBay.
8. **Evaluate Relevance (using eBay Items):**
   - Nếu không tìm thấy mặt hàng eBay (`eBay Items` rỗng): Chuyển sang bước Chuyển tiếp Telegram.
   - Nếu tìm thấy mặt hàng: `processing_service` gọi `gemini_service.evaluate_knowledge_relevance` với nội dung email gốc và các mặt hàng eBay tìm được.
9. **Tạo và Gửi Phản hồi:** Nếu kết quả đánh giá là đủ liên quan (`True`):
   - `processing_service` gọi `gemini_service.generate_response` để tạo nội dung email trả lời dựa trên các mặt hàng eBay.
   - Nếu tạo response thành công, `processing_service` gọi `gmail_service.send_reply` để gửi email.
   - Nếu gửi thành công, `processing_service` gọi `gmail_service.mark_as_read`. Kết thúc luồng.
10. **Chuyển tiếp Telegram:** Nếu xảy ra một trong các trường hợp sau:
    - Phân loại email ban đầu thất bại (không phải SPAM/PROCESS).
    - Lỗi xảy ra trong quá trình tạo tham số tìm kiếm eBay.
    - Không tìm thấy mặt hàng eBay.
    - Mặt hàng eBay tìm thấy nhưng không được đánh giá là đủ liên quan.
    - Lỗi xảy ra trong quá trình tạo/gửi phản hồi.

    Thì `processing_service` gọi `telegram_service.run_async_forward` để gửi thông tin email đến nhóm hỗ trợ. Sau đó, gọi `gmail_service.mark_as_read` Kết thúc luồng.
11. **Mark as Read:** Sau khi xử lý thành công (gửi reply hoặc forward thành công), `processing_service` gọi `gmail_service.mark_as_read`.
12. **Xác nhận Pub/Sub:** Callback trong `main.py` gọi `message.ack()` nếu xử lý thành công (bao gồm cả gửi reply hoặc forward thành công) hoặc `message.nack()` nếu có lỗi nghiêm trọng trong quá trình xử lý để Pub/Sub có thể thử gửi lại.

2.6. Thiết kế Module

Các module Python chính và trách nhiệm của chúng:

- `config.py`: Quản lý và cung cấp các biến cấu hình từ môi trường/file `.env`.
- `logger_config.py`: Thiết lập cấu hình logging chung cho ứng dụng.
- `main.py`: Điểm vào chính, khởi tạo listener Pub/Sub, điều phối gọi `processing_service` khi có tin nhắn mới.
- `gmail_service.py`: Đóng gói các tương tác với Gmail API (OAuth, fetch, send, modify labels).
- `gemini_service.py`: Đóng gói các tương tác với Gemini API (classify, evaluate, generate response), bao gồm logic retry.
- `ebay_service.py`: Đóng gói các tương tác với eBay API (generate search params, search items).
- `processing_service.py`: Chứa logic nghiệp vụ cốt lõi, điều phối luồng xử lý giữa các service khác.
- `telegram_service.py`: Đóng gói việc gửi tin nhắn đến Telegram Bot API.
- `setup_gmail_watch.py`: Script tiện ích để thiết lập Gmail push notifications đến Pub/Sub.
- `populate_data.py`: Script tiện ích để khởi tạo dữ liệu mẫu.

3. Hướng dẫn Triển khai, Cài đặt và Sử dụng

3.1. Yêu cầu hệ thống

- Python: Phiên bản 3.10 trở lên.
- Google Cloud Platform: Một Project với các API sau đã được bật: Gmail API, Google Cloud Pub/Sub API, AI Studio API cho Gemini.
- Tài khoản Gmail: Tài khoản sẽ được sử dụng để nhận và gửi email hỗ trợ.
- Ebay: Tài khoản Ebay đã tạo dự án với môi trường production, có app id, cert id... (môi trường sandbox không có dữ liệu để tìm kiếm)
- Telegram: Một Bot Token và Chat ID của nhóm/kênh hỗ trợ.
- Hệ điều hành: Linux, macOS, hoặc Windows (với môi trường Python được thiết lập).

3.2. Cài đặt

1. **Lấy mã nguồn:** Clone repository chứa mã nguồn dự án tại https://github.com/nguyencaobach123/automated-email-esi.
2. **Tạo môi trường ảo:**

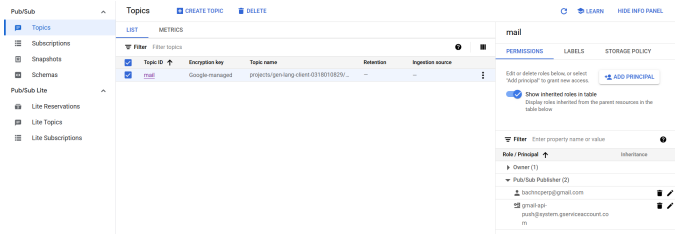
```
python -m venv .venv
# Linux/macOS:
source .venv/bin/activate
# Windows:
.\venv\Scripts\activate
```

3. **Cài đặt thư viện:**

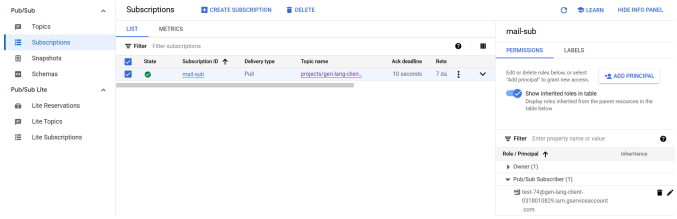
```
pip install -r requirements.txt
```

4. **Thiết lập Google Cloud:**
   - Truy cập Google Cloud Console (https://console.cloud.google.com/).
   - Tạo một Google Cloud Project mới hoặc chọn project hiện có.
   - Bật các API cần thiết: Gmail API, Google Cloud Pub/Sub API, Gemini API).
   - **Service Account (Cho Pub/Sub):**
     * Vào IAM & Admin > Service Accounts.
     * Tạo một Service Account mới.
     * Tạo Key cho Service Account này (chọn JSON), tải về và lưu vào thư mục gốc dự án với tên mặc định đuôi json. File này cần được bảo mật
   - **Gmail API Credentials (Cho User):**
     * Vào APIs & Services > Credentials.
     * Tạo Credentials > OAuth client ID.
     * Chọn Application type là Desktop app.
     * Đặt tên app.
     * Tải file JSON credentials về, đổi tên thành `credentials.json` và đặt vào thư mục gốc dự án.
   - **Gemini API Key:**
     * Truy cập Google AI Studio (https://aistudio.google.com/) hoặc Google Cloud Console để lấy API Key cho Gemini.
5. **Pub/Sub:**
   - Tạo một Pub/Sub Topic (ví dụ trong hình là `mail`). Ghi lại tên đầy đủ (ví dụ trong hình 7 là `projects/gen-lang-client-0318010829/topics/mail`).
   - Tạo một Pub/Sub Subscription liên kết với Topic trên (ví dụ trong hình 11 là `mail-sub`). Chọn kiểu delivery là Pull. Ghi lại tên đầy đủ (ví dụ trong hình 11 là `projects/gen-lang-client-0318010829/subscriptions/mail-sub`).

- Cấp quyền Pub/Sub Publisher cho email `gmail-api-push@system.gserviceaccount.com` (mail cố định phục vụ việc publish vào topic khi có mail mới)
- Cấp quyền Pub/Sub Subscriber cho service account được tạo ở trên.

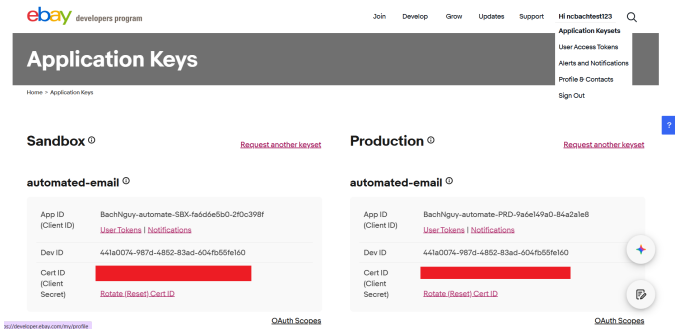


Hình 7: Cài đặt topic và publisher



Hình 8: Cấu hình subscription

6. **Thiết lập eBay:**
   - Truy cập trang web eBay Developer Program và đăng ký tài khoản developer.
   - Sau khi tài khoản được duyệt, đăng nhập vào Developer Account.
   - Chọn Application Keyset, chọn môi trường Production.
   - eBay sẽ cung cấp bộ khóa ứng dụng (Application Keys) bao gồm **App ID**, **Dev ID**, và **Cert ID**. Lưu lại các khóa này.
   - Cấu hình các khóa API này trong file cấu hình hoặc biến môi trường của ứng dụng để sử dụng cho việc gọi các API của eBay.



Hình 9: Tạo project trên ebay

7. **Thiết lập Telegram:**
   - Mở Telegram, tìm BotFather.
   - Tạo bot mới bằng lệnh `/newbot`. Làm theo hướng dẫn để đặt tên và username cho bot.
   - BotFather sẽ cung cấp một **Bot Token**. Lưu lại token này.
   - Tạo một nhóm chat Telegram mới (hoặc dùng nhóm có sẵn) cho đội ngũ hỗ trợ và thêm bot được tạo vào.
   - Lấy **Chat ID** của nhóm bằng cách gửi một tin nhắn mới vào nhóm để có cập nhật, sau đó truy cập đường link `https://api.telegram.org/bot<YourBOTToken>/getUpdates`, tại đây, dữ liệu json sẽ được trả về, trong đó có chat id của nhóm chat như trong Hình 10

Hình 10: Dữ liệu trả về từ api getupdate cho bot có chứa chat id

3.3. Cấu hình

3.4. Khởi chạy và Sử dụng

1. **Cấu hình các thông tin trong file env** Cấu hình các thông tin dựa trên các bước đã chuẩn bị ở trên

```
# Gmail API - Obtain credentials.json, run auth flow once to get token.json
GMAIL_CREDENTIALS_FILE='credentials.json'
GMAIL_TOKEN_FILE='token.json'
GMAIL_SCOPES='https://www.googleapis.com/auth/gmail.modify'
GMAIL_USER_ID='me'
# GMAIL_WATCH_LABEL_ID='YOUR_SUPPORT_LABEL_ID' # Optional: Only process emails with this label
GMAIL_WATCH_TOPIC_NAME='projects/gen-lang-client-0318010829/topics/mail'
SERVICE_ACCOUNT_KEY_FILE = 'gen-lang-client-0318010829-fdece7664895.json'
# Google AI (Gemini) API
GEMINI_API_KEY=

# Telegram Bot API
TELEGRAM_BOT_TOKEN=
TELEGRAM_CHAT_ID=-1002685616047


# Google Cloud Project & Pub/Sub
# Ensure the Pub/Sub API is enabled in your Google Cloud Project
GOOGLE_PROJECT_ID='gen-lang-client-0318010829'
GMAIL_PUBSUB_TOPIC_PATH='projects/gen-lang-client-0318010829/topics/mail'
GMAIL_PUBSUB_SUBSCRIPTION_PATH='projects/gen-lang-client-0318010829/subscriptions/mail-sub'

# eBay API - Replace with your actual Sandbox keys
EBAY_APP_ID='BachNguy-automate-PRD-9a6e149a0-84a2a1e8'
EBAY_DEV_ID='441a0074-987d-4852-83ad-604fb55fe160'
EBAY_CERT_ID=

# eBay Browse API OAuth Credentials
EBAY_OAUTH_CLIENT_ID='BachNguy-automate-PRD-9a6e149a0-84a2a1e8'
EBAY_OAUTH_CLIENT_SECRET=

# eBay API Environment (sandbox or production)
EBAY_ENVIRONMENT=production
```

Hình 11: Cấu hình file .env

2. **Xác thực Gmail lần đầu (Chạy 1 lần):** Chạy ứng dụng chính lần đầu tiên sẽ kích hoạt quy trình OAuth 2.0:

```
python main.py
```

Một URL xác thực sẽ hiện ra trong console. Mở URL này trong trình duyệt, đăng nhập bằng tài khoản Gmail đã cấu hình, và cấp quyền cho ứng dụng. Sau khi xác thực thành công, file

`token.json` sẽ được tạo ra trong thư mục gốc. Có thể dừng ứng dụng (Ctrl+C) sau khi file token được tạo.

3. **Thiết lập Gmail Watch (Chạy 1 lần và định kỳ):** Dảm bảo `GMAIL_PUBSUB_TOPIC_PATH` trong `.env` là chính xác. Chạy script sau:

```
python setup_gmail_watch.py
```

Script này yêu cầu Gmail gửi thông báo đến Pub/Sub topic khi có email mới. Thiết lập này có thời hạn 7 ngày, Cần chạy lại script này trước khi hết hạn (có thể cấu hình cron chạy hằng ngày hoặc chạy tay)

4. **Chạy Listener chính:** Sau khi hoàn tất các bước trên, khởi chạy ứng dụng listener:

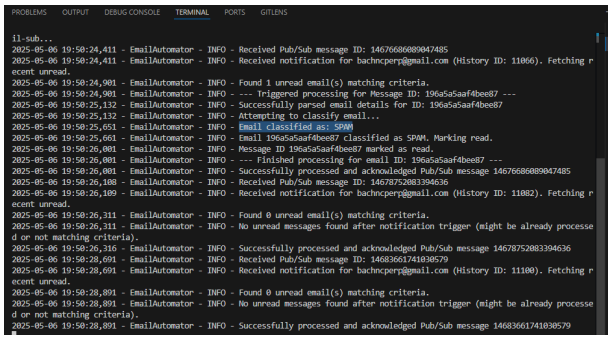
```
python main.py
```

Ứng dụng sẽ bắt đầu lắng nghe tin nhắn từ Pub/Sub. Log sẽ hiển thị trong console.

5. **Sử dụng:**
   - Gửi email đến địa chỉ email đã cấu hình.
   - Quan sát log để theo dõi quá trình xử lý.
   - Kiểm tra email phản hồi tự động (nếu có).
   - Kiểm tra nhóm chat Telegram để xem các email được chuyển tiếp.
   - Dừng ứng dụng bằng cách nhấn Ctrl+C trong cửa sổ console đang chạy `main.py`.

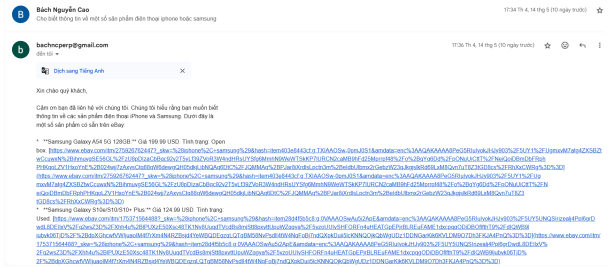
6. **Một số hình ảnh trong quá trình sử dụng thực tế** Phần này minh họa một số kịch bản hoạt động của hệ thống thông qua các hình ảnh chụp màn hình.
   Xử lý Email SPAM: Khi một email được gửi đến và hệ thống phân loại đó là SPAM:

Hình 12: Ví dụ email được xác định là SPAM



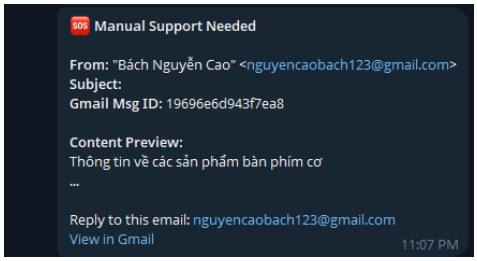
Hình 13: Log hệ thống ghi nhận việc phân loại là SPAM và đánh dấu đã đọc

Như trong log (Hình 13), hệ thống đã xác định email (Hình 12) là SPAM và thực hiện hành động đánh dấu email là đã đọc mà không cần xử lý thêm.
Tự động Phản hồi Email Hợp lệ: Khi một email hợp lệ (PROCESS) được gửi đến và hệ thống tìm thấy thông tin đủ liên quan từ thông tin được trả về trên Ebay để trả lời:



Hình 14: Email phản hồi tự động được tạo và gửi đi bởi hệ thống

Hệ thống sử dụng AI để tạo nội dung phản hồi phù hợp dựa trên yêu cầu của khách hàng và thông tin từ Ebay API trả về, sau đó tự động gửi email trả lời (Hình 14) trong cùng luồng thư.
Chuyển tiếp Email đến Telegram: Khi một email cần xử lý nhưng hệ thống không tìm thấy đủ thông tin liên quan hoặc xác định yêu cầu là phức tạp, thông tin sẽ được chuyển tiếp đến nhóm hỗ trợ qua Telegram:



Hình 15: Tin nhắn Telegram chứa thông tin email được chuyển tiếp đến nhóm hỗ trợ

Tin nhắn trên Telegram (Hình 15) bao gồm các thông tin chính như người gửi, tiêu đề, trích đoạn nội dung và link đến email gốc trên Gmail, giúp nhân viên hỗ trợ nhanh chóng nắm bắt và xử lý.

4. Đánh giá Hiệu suất, Khả năng mở rộng và Bảo mật

4.1. Hiệu suất

- **Độ trễ (Latency):** Thời gian từ lúc email đến cho đến khi gửi phản hồi tự động hoặc forward đến telegram là tương đối thấp, chỉ khoảng 30 giây đổ lại nếu không có lỗi gì xảy ra.
- **Thông lượng (Throughput):** Khả năng xử lý số lượng email đồng thời bị giới hạn bởi:
  - Giới hạn tần suất (rate limits) của Gmail API và Gemini API.
  - Khả năng xử lý của một tiến trình listener `main.py` duy nhất (trong thiết lập cơ bản).
  - Giới hạn của Google Cloud Pub/Sub (thường khá cao).

  Trong cấu hình cơ bản với một listener, thông lượng sẽ tương đối thấp. Nhóm chưa thực hiện thử nghiệm load test thực tế do có các giới hạn của google cho việc gọi api.

4.2. Khả năng mở rộng

- **Điểm mạnh:**
  - **Pub/Sub:** Kiến trúc Pub/Sub vốn hỗ trợ nhiều subscriber cùng đọc từ một subscription. Có thể chạy nhiều instance của `main.py` (ví dụ: trên các máy khác nhau hoặc trong container) để xử lý song song các thông báo đến, giúp tăng thông lượng đáng kể.
  - **Dịch vụ Cloud:** Gmail API, Gemini API, Pub/Sub là các dịch vụ đám mây có khả năng mở rộng cao do Google quản lý.
- **Điểm yếu/Cần cải thiện:**
  - **Listener đơn lẻ:** Triển khai cơ bản trong `main.py` chỉ chạy một listener. Cần có cơ chế để chạy và quản lý nhiều workers (ví dụ: dùng Supervisor, Docker Swarm/Compose, Kubernetes).
  - **Xử lý đồng bộ:** Một số tác vụ trong `processing_service` là đồng bộ (blocking). Sử dụng asyncio hoặc các thư viện xử lý bất đồng bộ có thể cải thiện hiệu năng trên mỗi worker.

4.3. Bảo mật

- **Quản lý Credentials:**
  - `.env`: Lưu trữ credentials (API keys, DB password) trong file `.env` tốt hơn hardcode, nhưng file này vẫn cần được bảo vệ (không commit vào Git, cấp quyền đọc hạn chế).
  - `credentials.json` (OAuth User): Chứa client secret, cần bảo mật.
  - `token.json` (OAuth User Token): Chứa token truy cập và refresh token của người dùng, rất nhạy cảm, cần bảo mật tuyệt đối.
  - Service Account Key JSON: Chứa private key, cực kỳ nhạy cảm, cần bảo mật tuyệt đối.
  - **Cải thiện:** Trong môi trường production, nên sử dụng các dịch vụ quản lý bí mật chuyên dụng như Google Secret Manager, HashiCorp Vault thay vì file `.env` hay lưu trực tiếp file key.
- **Quyền truy cập:**
  - OAuth Scope: `gmail.modify` cấp quyền đọc, gửi, và thay đổi (như xóa label UNREAD). Đây là quyền khá rộng nhưng cần thiết cho chức năng. Cần đảm bảo chỉ ứng dụng này sử dụng token.
  - Service Account Roles: Cần cấp quyền tối thiểu cần thiết (ví dụ: chỉ `Pub/Sub Subscriber`).
- **Bảo mật dữ liệu:**
  - Email Content: Hệ thống xử lý nội dung email có thể chứa thông tin nhạy cảm (PII). Cần có chính sách về lưu trữ log và xử lý dữ liệu tuân thủ quy định. Hiện tại chưa có cơ chế che giấu PII tự động.
  - Giao tiếp: Các lượt gọi API Google (Gmail, Gemini, Pub/Sub), Ebay và Telegram mặc định sử dụng HTTPS, đảm bảo mã hóa trên đường truyền.
- **Input Sanitization:** Hệ thống ưu tiên xử lý plain text từ email, giảm thiểu rủi ro từ HTML/script độc hại. Tuy nhiên, nếu logic phân tích HTML được thêm vào sau này, cần cẩn trọng với XSS.
- **Logging:** Log có thể vô tình ghi lại thông tin nhạy cảm. Cần xem xét việc lọc hoặc che giấu dữ liệu nhạy cảm trước khi ghi log.

5. Kết luận

Dự án đã phát triển thành công một hệ thống prototype minh chứng cho sức mạnh của **Tích hợp được Tăng cường bởi AI (AI-Enhanced Integration)** trong việc tự động hóa quy trình xử lý email hỗ trợ khách hàng. Bằng cách kết hợp chặt chẽ các dịch vụ đám

mây (Gmail API, Google Cloud Pub/Sub) với năng lực tiên tiến của Trí tuệ Nhân tạo từ Google Gemini, hệ thống không chỉ tự động hóa các tác vụ mà còn đưa ra các quyết định thông minh trong suốt luồng tích hợp dữ liệu và quy trình.

Nền tảng của hệ thống là kiến trúc hướng sự kiện, nơi AI đóng vai trò then chốt trong nhiều khâu:

- **Tích hợp thông minh đầu vào:** AI (Gemini) tự động phân loại nội dung và ý định của email đến, cho phép hệ thống định tuyến và xử lý thông tin một cách phù hợp ngay từ khâu tiếp nhận.
- **Tích hợp tạo truy vấn thông minh**: Sau khi có được nội dung từ email, AI tạo ra truy vấn phù hợp (ví dụ thêm khoảng giá, thêm tình trạng sản phẩm...) dựa trên hướng dẫn sử dụng từ chính official document của Ebay, giúp lấy được các thông tin đúng như mong muốn.
- **Tạo phản hồi động và tích hợp ngược:** AI (Gemini LLM) không chỉ tạo ra các phản hồi dựa trên mẫu cứng, mà còn tổng hợp thông tin từ nhiều nguồn (email gốc, dữ liệu trả về từ Ebay) để sinh ra các câu trả lời tự nhiên, phù hợp ngữ cảnh, và sau đó tích hợp ngược kết quả này vào luồng giao tiếp qua Gmail.
- **Tích hợp có điều kiện với con người:** AI quyết định khi nào một yêu cầu vượt quá khả năng xử lý tự động và cần chuyển tiếp thông minh đến nhân viên hỗ trợ qua Telegram, đảm bảo sự liền mạch trong quy trình tích hợp giữa máy và người.

Kiến trúc module hóa và việc đóng gói các tương tác AI thành các service riêng biệt (`Gemini Service`) tạo điều kiện cho một hệ thống linh hoạt, dễ bảo trì và có khả năng mở rộng.

Mặc dù đây là phiên bản prototype, dự án đã cho thấy tiềm năng to lớn của AI trong việc nâng cao hiệu quả và trí thông minh của các giải pháp tích hợp. Các bước phát triển tiếp theo sẽ tập trung vào:

- **Đo lường và tối ưu hóa hiệu suất tích hợp AI:** Đánh giá thực tế độ trễ, thông lượng, và đặc biệt là độ chính xác của các quyết định do AI đưa ra trong quy trình tích hợp.
- **Tinh chỉnh và học hỏi liên tục cho AI:** Tinh chỉnh prompt cho các mô hình Gemini và có thể xem xét các kỹ thuật học tăng cường dựa trên phản hồi để AI ngày càng đưa ra các quyết định tích hợp tốt hơn.
- **Mở rộng khả năng tích hợp thông minh:** Triển khai cơ chế chạy nhiều worker listener và tối ưu hóa các tương tác AI để xử lý tải cao hơn trong môi trường production.
- **Tăng cường bảo mật và quản trị cho các điểm tích hợp AI:** Đảm bảo an toàn cho dữ liệu đầu vào/ra của các mô hình AI và quản lý vòng đời của các mô hình/API AI được sử dụng.
- **Xây dựng cơ chế giám sát và phân tích nâng cao:** Theo dõi hiệu quả của các quyết định tích hợp do AI thực hiện và cung cấp thông tin chi tiết để phục vụ cho việc cải tiến liên tục.
- **Cải thiện khả năng xử lý lỗi:** Hoàn thiện cơ chế xử lý lỗi chi tiết hơn cho từng bước trong quy trình tích hợp, đặc biệt là các lỗi liên quan đến AI.

Tóm lại, dự án không chỉ tự động hóa một quy trình nghiệp vụ cụ thể, mà còn là một minh chứng cho thấy cách AI có thể làm cho bản thân quá trình tích hợp trở nên thông minh hơn, linh hoạt hơn và mang lại giá trị cao hơn. Giải pháp "Tự động hóa Xử lý Email Hỗ trợ Khách hàng bằng AI" này hứa hẹn mang lại lợi ích đáng kể về thời gian, chi phí và chất lượng dịch vụ hỗ trợ khách hàng, đồng thời mở ra hướng tiếp cận mới, được tăng cường bởi AI, cho các bài toán tích hợp hệ thống phức tạp trong tương lai.

Tài liệu tham khảo (Tham khảo Công nghệ)

- Google Cloud Pub/Sub: https://cloud.google.com/pubsub/docs
- Gmail API: https://developers.google.com/gmail/api/
- Google AI Gemini API: https://ai.google.dev/docs
- Ebay Search API: https://developer.ebay.com/api-docs/buy/browse/resources/item_summary/methods/search
- Telegram Bot API: https://core.telegram.org/bots/api
- google-api-python-client: https://github.com/googleapis/google-api-python-client
- google-cloud-pubsub (Python): https://cloud.google.com/python/docs/reference/pubsub/latest
- google-generativeai (Python): https://github.com/google/generative-ai-python