**HỌC VIỆN CÔNG NGHỆ & BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ THÀNH PHỐ HỒ CHÍ MINH**

**BÁO CÁO**

**MÔN HỌC: NHẬP MÔN CÔNG NGHỆ PHẦN MỀM**

**CHỦ ĐỀ: Trợ lý học tập cho sinh viên học viện**

**Lớp E22CQCN02-N**

**SINH VIÊN THỰC HIỆN:**

**Trần Anh Tuấn - N22DCAT066  
Sỳ Hưng - N22DCCN137**

**Nguyễn Vũ Việt Cường - N22DCCN111**

**GIẢNG VIÊN HƯỚNG DẪN: ThS. Châu Văn Vân**

**NĂM HỌC: 2025-2026**

## 

[**1. Tổng quan hệ thống 4**](#_bj2wg3ywuyni)

[1.1 Use case diagram 4](#_vwakaoy81nlm)

[**2. Kiến trúc phân lớp 4**](#_x1gz7ep8y11y)

[2.1. Frontend Components (Các thành phần giao diện người dùng) 4](#_eno5n4qx19m)

[2.2. Backend Services (Các dịch vụ phía máy chủ) 4](#_fg6tsaymh0fq)

[2.3. External Integrations (Tích hợp bên ngoài) 4](#_dtkeddh4tr2c)

[**3. Luồng xử lý dữ liệu 5**](#_4tvusqs8knou)

[**4. Hệ thống phân loại truy vấn 5**](#_jxlz0eutnq9)

[**5. Tạo phản hồi AI 6**](#_3wuyxxfg7dht)

[**6. Tích hợp hệ thống học viện 6**](#_vrmp7djlqfx)

[**7. Lưu trữ dữ liệu 6**](#_xqwmgdz480bu)

[**8. Hệ thống Agent 6**](#_kruva0ltzicm)

[**2. Backend Implementation (Triển khai Backend) 7**](#_m0eqrq7nyzp3)

[2.1 Flask Application Structure (Cấu trúc ứng dụng Flask) 7](#_amrkqqe4q4bg)

[2.2 API Routes (Các định tuyến API) 7](#_ac25ku25llas)

[2.3 Core Services (Các dịch vụ cốt lõi) 8](#_h78sh4fr1khs)

[2.3.1 AI Service (Dịch vụ AI) 8](#_6a0jkrnhcug9)

[2.3.2 Query Classifier Service (Dịch vụ phân loại truy vấn) 8](#_5v8814gnwdb0)

[2.3.3 Schedule Services (Dịch vụ lịch học) 8](#_sq1e0ob0245u)

[2.3.4 File Service and Vector Search (Dịch vụ tệp và tìm kiếm vector) 9](#_430mk8cj5roe)

[2.4 External Integrations (Tích hợp bên ngoài) 9](#_bg6xa8xh86jj)

[2.4.1 LM Studio Integration (Tích hợp LM Studio) 9](#_hn4zdxy6spe5)

[2.4.2 PTIT University Integration (Tích hợp Học viện PTIT) 9](#_kmq9vxseeia6)

[2.4.3 Data Storage with Supabase (Lưu trữ dữ liệu với Supabase) 10](#_yuwep9v86jeh)

[2.4.4 Web Search Integration (Tích hợp tìm kiếm web) 10](#_llyp14tcnpjy)

[**3. Frontend Implementation (Triển khai Frontend) 10**](#_nnvturygtc67)

[3.1 Component Structure (Cấu trúc Component) 10](#_1wfxkbtdfqgj)

[3.2 Chat Interface Implementation (Triển khai Giao diện Trò chuyện) 11](#_cufskl2leykw)

[3.3 Authentication and Settings (Xác thực và Cài đặt) 12](#_k500x1kbohox)

[3.3.1 Xác thực người dùng 12](#_u9lf2j41g8kf)

[3.3.2 Cài đặt thông tin đăng nhập Học viện PTIT 12](#_8lndfxphx8bb)

[**4. Agent System 13**](#_7jm2n2don30n)

[4.1 Tổng quan về Agent System 13](#_e1bnuuavrodg)

[4.2 Cấu trúc Agent 13](#_aoemmq67qyml)

[4.3 Quản lý Agent 13](#_3lvbvgo55oq8)

[4.4 Tương tác với LM Studio 13](#_26aeu6jwhpfw)

[4.4.1 Cấu hình kết nối LM Studio 14](#_1ij95vx274iy)

[4.4.2 Quy trình tương tác với LM Studio 14](#_5fv9na2i7var)

[4.5 Tích hợp Agent trong Frontend 14](#_m4mz9prp49e2)

[4.5.1 AgentSelector Component 14](#_x9irbldsr19t)

[4.5.2 Lưu trữ Agent đã chọn 14](#_neo3you5a9p)

[4.5.3 Gửi Agent ID trong yêu cầu 14](#_uh9dkjbg7n4l)

[4.6 Ví dụ luồng hoạt động đầy đủ 14](#_12ottkjtwuh7)

[4.7 Cấu trúc dữ liệu lưu trữ 14](#_n9pto72fq87t)

[4.8 Ưu điểm của Agent System 15](#_f5arve4me4dh)

[**5. System Operation: End-to-End Flow trong Student Assistant Agent 15**](#_xcgmh4ogdqyo)

[5.1 User Authentication (Xác thực người dùng) 15](#_1q7oeit8q6u)

[5.2 File Upload and Processing (Tải lên và xử lý tệp) 16](#_p1j9029w9l21)

[5.3 Query Processing (Xử lý truy vấn) 17](#_3qjte45gpora)

[**6. Vector Embedding: Technical Deep Dive 18**](#_yfoqllq6rbq)

[6.1 Implementation Logic (Logic triển khai) 19](#_c8cu41vuyov3)

[6.2 Embedding Generation (Tạo Embedding) 19](#_d43ubmb7l06u)

[6.3 Semantic Search Implementation (Triển khai Tìm kiếm Ngữ nghĩa) 20](#_v9vr40lsuucw)

[6.4 Integration with AI Service (Tích hợp với AI Service) 21](#_2y6rovuvdu1q)

[**7. Hiệu năng hệ thống và các giới hạn 22**](#_su9bves6mrmm)

[7.1 Đặc điểm hiệu năng 22](#_onubljb1gl3)

[7.2 Hạn chế 22](#_gaadh059xi6u)

[**8. Backend Frameworks và Libraries 23**](#_dtdiuonfh1y7)

[8.1 Flask 23](#_3s3a8hdqxti6)

[8.2 Sentence Transformers 23](#_dhx3lc7bhugb)

[8.3 HTTPX 23](#_12c10fu8oxs0)

[8.4 BeautifulSoup 23](#_wxn1ixuj5ros)

[8.5 Supabase 23](#_8x6qhez6wf8m)

[**9. Frontend Frameworks và Libraries 23**](#_wxribskmr2b8)

[9.1 React 23](#_1s9s82u4q4li)

[9.3 Lucide React 23](#_nilvqwde6gyn)

[**10. AI và NLP Libraries 24**](#_7s8d2rgg2u72)

[10.1 LM Studio 24](#_9yv6lre7fpa9)

[10.2. Brave Search API 24](#_qj35v4jsfja5)

[**11. Các Thư Viện Hỗ Trợ Khác 24**](#_ur6xq62hg3wy)

[11.1 Unidecode 24](#_kipkcgp2dwpg)

## 

## 

## 

## **1. Tổng quan hệ thống**

Hệ thống Student Assistant Agent là một trợ lý AI được thiết kế để hỗ trợ sinh viên tại học viện PTIT. Hệ thống cung cấp các chức năng chính sau:

1. **Truy xuất thông tin học tập**: Lịch học, lịch thi trực tiếp từ hệ thống trường đại học
2. **Hỗ trợ AI về giáo dục**: Trả lời các câu hỏi học thuật và giáo dục
3. **Nhận thức về ngữ cảnh tài liệu**: Khả năng trả lời câu hỏi về tài liệu đã tải lên
4. **Tích hợp tìm kiếm web**: Cải thiện câu trả lời với thông tin từ tìm kiếm web

### **1.1 Use case diagram**

### 

## **2. Kiến trúc phân lớp**

Hệ thống được tổ chức theo kiến trúc phân lớp với các thành phần chính sau:

### **2.1. Frontend Components (Các thành phần giao diện người dùng)**

Frontend được xây dựng bằng React với các thành phần chính:

* **ChatInterface**: Container UI chính, quản lý phiên chat, xử lý lựa chọn agent
* **ChatInput**: Xử lý đầu vào của người dùng, tải lên tệp, bật/tắt tìm kiếm web
* **MessageItem**: Hiển thị các loại tin nhắn khác nhau (văn bản, mã, sơ đồ)
* **Settings**: Cấu hình người dùng, thông tin đăng nhập trường đại học

### **2.2. Backend Services (Các dịch vụ phía máy chủ)**

Backend được xây dựng bằng Flask với các dịch vụ chính:

* **AI Service**: Quản lý tương tác với AI, xử lý các loại trò chuyện khác nhau
* **Query Classifier**: Phân loại các truy vấn của người dùng
* **Schedule Service**: Truy xuất lịch học
* **Exam Schedule Service**: Quản lý thông tin lịch thi
* **File Service**: Xử lý tài liệu
* **Web Search Service**: Thực hiện tìm kiếm web
* **PTIT Auth Service**: Xác thực với hệ thống trường đại học

### **2.3. External Integrations (Tích hợp bên ngoài)**

Hệ thống tích hợp với nhiều dịch vụ bên ngoài:

* **LM Studio**: Cung cấp khả năng AI, được sử dụng bởi AI Service
* **PTIT University API**: Nguồn dữ liệu của trường đại học, sử dụng cho thông tin lịch học và lịch thi
* **PostgeSQL Database**: Lưu trữ dữ liệu, lưu trữ lịch sử trò chuyện, cài đặt người dùng, vector embedding
* **Brave Search API**: Khả năng tìm kiếm web, cung cấp kết quả tìm kiếm web cho các truy vấn

## 3. Luồng xử lý dữ liệu

Khi người dùng gửi một truy vấn, hệ thống xử lý theo các bước sau:

1. **Nhận truy vấn**: Frontend nhận truy vấn từ người dùng và gửi đến backend
2. **Phân loại truy vấn**: Backend sử dụng Query Classifier để phân loại truy vấn
3. **Xử lý theo loại truy vấn**:
   * **Truy vấn lịch học/lịch thi**: Kết nối với PTIT API để lấy dữ liệu
   * **Truy vấn UML**: Tạo sơ đồ UML thông qua AI
   * **Truy vấn với tệp đính kèm**: Xử lý ngữ cảnh tệp
   * **Truy vấn với tìm kiếm web**: Tìm kiếm web và tích hợp kết quả
   * **Truy vấn giáo dục chung**: Xử lý thông qua AI
4. **Trả về kết quả**: Backend trả về kết quả cho frontend để hiển thị

## **4. Hệ thống phân loại truy vấn**

Thành phần Query Classifier đóng vai trò quan trọng bằng cách xác định loại truy vấn của người dùng, quyết định cách xử lý:

| **STT** | **Loại truy vấn** | **Mô tả** | **Cách xử lý** |
| --- | --- | --- | --- |
| **1** | **schedule** | Câu hỏi về lịch học | Truy xuất dữ liệu lịch học từ API trường đại học |
| **2** | **examschedule** | Câu hỏi về lịch thi | Truy xuất dữ liệu lịch thi từ API trường đại học |
| **3** | **date\_query** | Câu hỏi liên quan đến ngày tháng | Xử lý cả dữ liệu lịch học và lịch thi |
| **4** | **uml** | Yêu cầu UML/sơ đồ | Tạo sơ đồ PlantUML |
| **5** | **general** | Câu hỏi giáo dục | Xử lý bởi AI với ngữ cảnh giáo dụ**c** |
| **6** | **other** | Chủ đề không liên quan đến giáo dục | Trả về phản hồi không giáo dục tiêu chuẩn |

## **5. Tạo phản hồi AI**

Thành phần AI Service (ai\_service.py) xử lý các loại trò chuyện khác nhau:

1. **Chat cơ bản**: Câu hỏi giáo dục chung được xử lý thông qua LM Studio
2. **Chat với ngữ cảnh tệp**: Phản hồi dựa trên nội dung tài liệu đã tải lên
3. **Chat với tìm kiếm web**: Phản hồi nâng cao kết hợp kết quả tìm kiếm web
4. **Tạo UML**: Tạo sơ đồ phần mềm với PlantUML

## **6. Tích hợp hệ thống học viện**

Hệ thống tích hợp với API của học viện PTIT để truy xuất thông tin lịch học và lịch thi theo thời gian thực:

1. Xác thực sinh viên thông qua thông tin đăng nhập PTIT
2. Truy xuất thông tin học kỳ hiện tại
3. Xử lý lịch học
4. Truy xuất và lọc lịch thi

Tích hợp này cho phép sinh viên đặt câu hỏi bằng ngôn ngữ tự nhiên về lịch học của họ như "Tôi có lớp học nào vào ngày mai?" hoặc "Khi nào tôi thi môn Cơ sở dữ liệu?"

## **7. Lưu trữ dữ liệu**

Hệ thống sử dụng Supabase làm cơ sở dữ liệu chính để lưu trữ:

1. **Lịch sử trò chuyện**: Lưu trữ tất cả các tin nhắn và phản hồi
2. **Cài đặt người dùng**: Lưu trữ thông tin đăng nhập và tùy chọn
3. **Vector embbeding**: Lưu trữ tệp đã tải lên và vector nhúng của chúng
4. **Kết quả tìm kiếm web**: Lưu trữ kết quả tìm kiếm để sử dụng lại

## **8. Hệ thống Agent**

Hệ thống hỗ trợ nhiều agent AI khác nhau với các cấu hình khác nhau:

1. Mỗi agent có thể có mô hình, nhiệt độ và cấu hình riêng
2. Người dùng có thể chọn agent khác nhau cho các phiên chat khác nhau
3. Cấu hình agent được lưu trữ với phiên chat

## 2. Backend Implementation (Triển khai Backend)

### **2.1 Flask Application Structure (Cấu trúc ứng dụng Flask)**

Hệ thống Student Assistant Agent được xây dựng trên nền tảng Flask, một framework Python nhẹ và linh hoạt. Cấu trúc ứng dụng được tổ chức theo mô hình module hóa với các thành phần chính sau:

1. **Cấu trúc thư mục**:
   * backend/app: Thư mục chính chứa mã nguồn backend
   * backend/app/routes: Chứa các định tuyến API
   * backend/app/services: Chứa các dịch vụ xử lý logic nghiệp vụ
   * backend/app/lib: Chứa các thư viện và kết nối bên ngoài
   * backend/app/utils: Chứa các tiện ích hỗ trợ
   * backend/app/config: Chứa cấu hình hệ thống
2. **Khởi tạo ứng dụng**:
   * Ứng dụng Flask được khởi tạo với các Blueprint để tổ chức các nhóm route
   * Các dịch vụ chính được khởi tạo khi ứng dụng khởi động
3. **Dependency Injection**:
   * Các dịch vụ được kết nối với nhau thông qua dependency injection
   * Ví dụ: schedule\_service được cung cấp auth\_service và ai\_service

### **2.2 API Routes (Các định tuyến API)**

Backend cung cấp các API endpoint chính sau:

1. **Chat API**:
   * /chat: Endpoint chính xử lý các truy vấn chat từ người dùng
   * Hỗ trợ nhiều loại truy vấn: truy vấn thông thường, truy vấn với tìm kiếm web, truy vấn với ngữ cảnh tệp
2. **Agent API**:
   * /agents: Trả về danh sách các agent AI có sẵn trong hệ thống
   * Cho phép người dùng chọn agent phù hợp với nhu cầu
3. **Xử lý truy vấn theo loại**:
   * Hệ thống phân loại truy vấn và xử lý theo các đường dẫn khác nhau
   * Truy vấn lịch học/lịch thi được xử lý bằng cách kết nối với API của Học viện PTIT
   * Truy vấn UML được xử lý bằng cách tạo sơ đồ PlantUML
   * Truy vấn giáo dục chung được xử lý thông qua AI
4. **Luồng xử lý truy vấn**:
   * Nhận truy vấn từ người dùng
   * Phân loại truy vấn bằng Query Classifier
   * Xử lý truy vấn theo loại tương ứng
   * Trả về kết quả cho frontend

### **2.3 Core Services (Các dịch vụ cốt lõi)**

#### **2.3.1 AI Service (Dịch vụ AI)**

AiService là thành phần trung tâm xử lý tương tác với mô hình AI:

1. **Chức năng chính**:
   * chat\_with\_ai: Xử lý trò chuyện cơ bản với AI
   * chat\_with\_file\_context: Xử lý trò chuyện với ngữ cảnh tệp
   * chat\_with\_web\_search: Xử lý trò chuyện với kết quả tìm kiếm web
   * \_render\_plantuml: Xử lý và hiển thị sơ đồ PlantUML
2. **Tương tác với LM Studio**:
   * Gửi yêu cầu đến LM Studio API
   * Xử lý phản hồi và định dạng kết quả
   * Hỗ trợ nhiều loại agent với cấu hình khác nhau
3. **Xử lý lỗi và logging**:
   * Ghi log chi tiết về quá trình tương tác với AI
   * Xử lý các trường hợp lỗi và trả về thông báo phù hợp

#### **2.3.2 Query Classifier Service (Dịch vụ phân loại truy vấn)**

Hệ thống Student Assistant Agent sử dụng một phương pháp phân loại truy vấn dựa trên mô hình ngôn ngữ lớn (LLM) thông qua LM Studio. Phương pháp này được triển khai trong lớp QueryClassifier và sử dụng hàm classify\_query\_lmstudio để phân loại các truy vấn của người dùng thành các loại khác nhau.

1. **Các loại truy vấn được hỗ trợ**:

Hệ thống phân loại truy vấn thành các loại sau:

1. **schedule**: Các câu hỏi về lịch học, thời khóa biểu
2. **examschedule**: Các câu hỏi về lịch thi
3. **date\_query**: Các câu hỏi liên quan đến ngày tháng không cụ thể về lịch học/lịch thi
4. **uml**: Các câu hỏi về sơ đồ, UML, PlantUML
5. **general**: Các câu hỏi giáo dục chung
6. **other**: Các câu hỏi không liên quan đến giáo dục
7. **Xử lý kết quả phân loại**:
   * Trả về kết quả phân loại với độ tin cậy
   * Xử lý các trường hợp đặc biệt như truy vấn trống
8. **Tích hợp với LM Studio**:
   * Sử dụng prompt đặc biệt để hướng dẫn mô hình phân loại
   * Xử lý kết quả JSON từ LM Studio

#### **2.3.3 Schedule Services (Dịch vụ lịch học)**

ScheduleService và ExamScheduleService quản lý việc truy xuất và xử lý thông tin lịch học và lịch thi:

1. **Kết nối với API PTIT**:
   * Xác thực với hệ thống PTIT thông qua PTITAuthService
   * Truy xuất dữ liệu lịch học và lịch thi từ API của Học viện
2. **Xử lý dữ liệu lịch học**:
   * Phân tích và định dạng dữ liệu lịch học
   * Lọc lịch học theo ngày, tuần, học kỳ
3. **Định dạng kết quả**:
   * Chuyển đổi dữ liệu lịch học thành định dạng dễ đọc
   * Tạo phản hồi có cấu trúc cho người dùng

#### **2.3.4 File Service and Vector Search (Dịch vụ tệp và tìm kiếm vector)**

Hệ thống hỗ trợ tải lên và truy vấn tài liệu:

1. **Xử lý tệp**:
   * Tải lên và lưu trữ tệp trong Supabase
   * Trích xuất và vector hóa nội dung tệp
2. **Tìm kiếm ngữ nghĩa**:
   * Tìm kiếm các đoạn văn bản liên quan đến truy vấn
   * Cung cấp ngữ cảnh tài liệu cho AI

### **2.4 External Integrations (Tích hợp bên ngoài)**

#### **2.4.1 LM Studio Integration (Tích hợp LM Studio)**

Hệ thống tích hợp với LM Studio để cung cấp khả năng AI:

1. **Cấu hình kết nối**:
   * Kết nối với LM Studio thông qua API
   * Xác thực bằng token ai\_service.py:12-17
2. **Các mô hình được sử dụng**:
   * CLASSIFIER\_MODEL: Mô hình phân loại truy vấn (qwen3-1.7b)
   * TIME\_PARSER\_MODEL: Mô hình phân tích thời gian (qwen3-4b)
3. **Định dạng yêu cầu và phản hồi**:
   * Gửi yêu cầu với prompt và tham số
   * Xử lý phản hồi JSON từ LM Studio

#### **2.4.2 PTIT University Integration (Tích hợp Học viện PTIT)**

Hệ thống tích hợp với API của Học viện Công nghệ Bưu chính Viễn thông (PTIT):

1. **Xác thực và đăng nhập**:
   * Đăng nhập vào hệ thống PTIT bằng thông tin đăng nhập của sinh viên
   * Lấy token xác thực để truy cập API
2. **Truy xuất dữ liệu học kỳ**:
   * Lấy thông tin học kỳ hiện tại
   * Sử dụng thông tin học kỳ để truy xuất lịch học và lịch thi
3. **API lịch học và lịch thi**:
   * Truy xuất lịch học theo học kỳ
   * Truy xuất lịch thi theo học kỳ
   * Lọc và định dạng dữ liệu cho người dùng

#### **2.4.3 Data Storage with Supabase (Lưu trữ dữ liệu với Supabase)**

Hệ thống sử dụng Supabase làm cơ sở dữ liệu chính:

1. **Lưu trữ tin nhắn**:
   * Lưu trữ lịch sử trò chuyện
   * Lưu trữ kết quả tìm kiếm web
2. **Quản lý phiên chat**:
   * Lưu trữ thông tin phiên chat
   * Lưu trữ cài đặt agent cho mỗi phiên
3. **Vector database**:
   * Lưu trữ vector nhúng của tài liệu
   * Hỗ trợ tìm kiếm ngữ nghĩa

#### **2.4.4 Web Search Integration (Tích hợp tìm kiếm web)**

Web Search Integration là một tính năng quan trọng của Student Assistant Agent, cho phép hệ thống tìm kiếm thông tin từ internet để cung cấp câu trả lời chính xác và cập nhật cho người dùng. Tính năng này được triển khai thông qua WebSearchService và tích hợp với Brave Search API

**Quá trình xử lý**:

1. Nhận truy vấn từ người dùng với web\_search\_enabled=true
2. Gọi WebSearchService.search() với query "Các phương pháp học tập hiệu quả cho sinh viên PTIT?"
3. Gửi request đến Brave Search API
4. Nhận và định dạng kết quả tìm kiếm (giới hạn 5 kết quả)
5. Tạo prompt cho AI với kết quả tìm

## 3. Frontend Implementation (Triển khai Frontend)

### **3.1 Component Structure (Cấu trúc Component)**

Hệ thống Student Assistant Agent được xây dựng trên nền tảng React với cấu trúc component được tổ chức theo mô hình phân cấp rõ ràng. Các component chính bao gồm:

1. **ChatInterface**: Component chính quản lý giao diện trò chuyện, là container bao quát toàn bộ ứng dụng.
2. **ChatInput**: Xử lý đầu vào của người dùng, bao gồm nhập văn bản, tải lên tệp và bật/tắt tìm kiếm web.
3. **MessageItem**: Hiển thị các tin nhắn trong cuộc trò chuyện, hỗ trợ nhiều định dạng như văn bản, mã nguồn và sơ đồ.
4. **Settings**: Quản lý cài đặt người dùng, đặc biệt là thông tin đăng nhập vào hệ thống Học viện PTIT.
5. **AgentSelector**: Cho phép người dùng chọn agent AI khác nhau để tương tác.

Các component này được tổ chức theo cấu trúc phân cấp, với ChatInterface là component cha chứa các component con khác. Mỗi component có trách nhiệm riêng biệt, tuân theo nguyên tắc thiết kế "Single Responsibility" (Trách nhiệm đơn lẻ).

### **3.2 Chat Interface Implementation (Triển khai Giao diện Trò chuyện)**

Giao diện trò chuyện là thành phần trung tâm của ứng dụng, được triển khai trong file ChatInterface.jsx. Giao diện này bao gồm các phần chính sau:

1. **Quản lý trạng thái (State Management)**:
   * Sử dụng React Hooks để quản lý trạng thái của ứng dụng
   * Các trạng thái chính bao gồm: người dùng hiện tại, phiên chat, tin nhắn, trạng thái tải, cài đặt, ngữ cảnh tệp và agent được chọn
2. **Xác thực và tải dữ liệu**:
   * Kiểm tra xác thực người dùng khi component được tải
   * Tải các phiên chat từ Supabase
   * Tải tin nhắn của phiên chat hiện tại
3. **Tải tin nhắn**:
   * Phương thức loadMessages tải tin nhắn từ Supabase dựa trên ID phiên chat
   * Cập nhật trạng thái tin nhắn và agent được chọn
4. **Quản lý phiên chat**:
   * Tạo phiên chat mới
   * Xóa phiên chat
   * Chuyển đổi giữa các phiên chat
5. **Xử lý tin nhắn**:
   * Gửi tin nhắn đến backend
   * Xử lý phản hồi từ backend
   * Cập nhật giao diện người dùng với tin nhắn mới
6. **Giao diện người dùng (UI)**:
   * Thanh bên (Sidebar) hiển thị danh sách phiên chat
   * Khu vực hiển thị tin nhắn
   * Thanh nhập tin nhắn ở dưới cùng
   * Các nút điều khiển và cài đặt
7. **Tích hợp tìm kiếm web**:
   * Bật/tắt tìm kiếm web cho các truy vấn
   * Hiển thị kết quả tìm kiếm web trong phản hồi
8. **Xử lý tệp**:
   * Tải lên tệp
   * Xóa ngữ cảnh tệp
   * Hiển thị thông tin tệp đang hoạt động

Ví dụ luồng hoạt động khi người dùng gửi tin nhắn:

1. Người dùng nhập tin nhắn và nhấn gửi
2. handleSendMessage được gọi với nội dung tin nhắn
3. Tin nhắn được gửi đến backend thông qua API
4. Backend xử lý tin nhắn và trả về phản hồi
5. Phản hồi được hiển thị trong giao diện người dùng
6. Tin nhắn và phản hồi được lưu vào Supabase

### **3.3 Authentication and Settings (Xác thực và Cài đặt)**

Hệ thống xác thực và cài đặt của Student Assistant Agent bao gồm hai phần chính: xác thực người dùng và cài đặt thông tin đăng nhập Học viện PTIT.

#### **3.3.1 Xác thực người dùng**

Xác thực người dùng được triển khai trong component Login.jsx:

1. **Giao diện đăng nhập**:
   * Form đăng nhập với các trường email và mật khẩu
   * Hỗ trợ đăng nhập bằng email/mật khẩu
2. **Xử lý đăng nhập**:
   * Sử dụng Supabase Auth để xác thực người dùng
   * Chuyển hướng đến trang chat sau khi đăng nhập thành công
3. **Giao diện thân thiện với người dùng**:
   * Hỗ trợ chế độ sáng/tối
   * Thông báo lỗi rõ ràng

#### **3.3.2 Cài đặt thông tin đăng nhập Học viện PTIT**

Cài đặt thông tin đăng nhập Học viện PTIT được triển khai trong component Settings.jsx:

1. **Giao diện cài đặt**:
   * Dialog hiển thị khi người dùng nhấn vào nút cài đặt
   * Các tab cho các loại cài đặt khác nhau
   * Form nhập thông tin đăng nhập Học viện PTIT
2. **Quản lý thông tin đăng nhập**:
   * Tải thông tin đăng nhập hiện có từ Supabase
   * Cập nhật thông tin đăng nhập
   * Xác thực thông tin đăng nhập với hệ thống Học viện PTIT
3. **Lưu trữ thông tin đăng nhập**:
   * Thông tin đăng nhập được lưu trong bảng university\_credentials trong Supabase
   * Bao gồm tên người dùng, mật khẩu, token truy cập và thông tin khác
4. **Xác thực với hệ thống Học viện PTIT**:
   * Gửi thông tin đăng nhập đến endpoint /ptit-login
   * Xử lý phản hồi từ hệ thống Học viện PTIT
   * Lưu token xác thực để sử dụng trong các yêu cầu sau này Settings.jsx:62-90
5. **Giao diện người dùng thân thiện**:
   * Hiển thị thông báo thành công/lỗi
   * Hỗ trợ chế độ sáng/tối
   * Trạng thái tải khi đang xác thực

Ví dụ luồng hoạt động khi người dùng cập nhật thông tin đăng nhập Học viện PTIT:

1. Người dùng mở dialog cài đặt
2. Nhập mã số sinh viên và mật khẩu
3. Nhấn nút "Lưu thông tin"
4. Hệ thống gửi thông tin đến backend để xác thực với hệ thống Học viện PTIT
5. Nếu xác thực thành công, thông tin được lưu vào Supabase
6. Hiển thị thông báo thành công
7. Thông tin đăng nhập này sẽ được sử dụng khi người dùng truy vấn về lịch học hoặc lịch thi

## 4. Agent System

Dựa trên mã nguồn được cung cấp, tôi sẽ trình bày chi tiết về hệ thống Agent trong Student Assistant Agent, bao gồm cách tương tác với LM Studio.

### **4.1 Tổng quan về Agent System**

Hệ thống Agent trong Student Assistant Agent là một cơ chế cho phép người dùng chọn và sử dụng các mô hình AI khác nhau (agents) để tương tác. Mỗi agent có cấu hình riêng, bao gồm mô hình, nhiệt độ (temperature) và các tham số khác.

Hệ thống này được triển khai chủ yếu trong file backend/app/config/agents.py, định nghĩa các agent có sẵn và cung cấp các hàm tiện ích để truy xuất cấu hình agent.

### **4.2 Cấu trúc Agent**

Mỗi agent được định nghĩa với các thuộc tính sau:

1. **id**: Định danh duy nhất của agent
2. **model**: Tên mô hình AI được sử dụng (ví dụ: qwen3-1.7b, qwen3-4b)
3. **display\_name**: Tên hiển thị trong giao diện người dùng
4. **temperature**: Tham số nhiệt độ ảnh hưởng đến tính ngẫu nhiên của phản hồi
5. **is\_default**: Xác định agent mặc định khi không có agent nào được chỉ định

Ví dụ cấu hình agent: agents.py:14-26

### **4.3 Quản lý Agent**

Hệ thống cung cấp hai hàm tiện ích chính để quản lý agent:

1. **get\_agent**: Truy xuất cấu hình agent theo ID, với cơ chế fallback nếu ID không tồn tại
2. **get\_all\_agents**: Trả về danh sách tất cả các agent có sẵn

### **4.4 Tương tác với LM Studio**

#### **4.4.1 Cấu hình kết nối LM Studio**

Hệ thống kết nối với LM Studio thông qua API REST. Cấu hình kết nối được định nghĩa trong file backend/app/services/ai\_service.py và backend/app/services/lmstudio\_service.py:

#### **4.4.2 Quy trình tương tác với LM Studio**

Quy trình tương tác với LM Studio diễn ra như sau:

1. **Chọn Agent**: Người dùng chọn agent từ giao diện hoặc hệ thống sử dụng agent mặc định
2. **Lấy cấu hình Agent**: Hệ thống truy xuất cấu hình agent từ get\_agent(agent\_id)
3. **Chuẩn bị Payload**: Tạo payload JSON với thông tin mô hình, tin nhắn và tham số
4. **Gửi yêu cầu đến LM Studio**: Sử dụng HTTP POST để gửi yêu cầu đến LM Studio API
5. **Xử lý phản hồi**: Nhận và xử lý phản hồi từ LM Studio

### **4.5 Tích hợp Agent trong Frontend**

#### **4.5.1 AgentSelector Component**

Frontend cung cấp component AgentSelector cho phép người dùng chọn agent: ChatInterface.jsx:506-509

#### **4.5.2 Lưu trữ Agent đã chọn**

Agent được chọn được lưu trữ trong state của component ChatInterface và trong cơ sở dữ liệu Supabase: ChatInterface.jsx:91-102

#### **4.5.3 Gửi Agent ID trong yêu cầu**

Khi gửi tin nhắn, agent\_id được đưa vào payload: ChatInterface.jsx:270-279

### **4.6 Ví dụ luồng hoạt động đầy đủ**

Dưới đây là ví dụ về luồng hoạt động đầy đủ khi người dùng tương tác với hệ thống:

1. **Người dùng chọn agent** từ dropdown AgentSelector
2. **Frontend lưu agent\_id** vào state và cơ sở dữ liệu
3. **Người dùng gửi tin nhắn**
4. **Frontend gửi yêu cầu** đến backend với agent\_id
5. **Backend truy xuất cấu hình agent** bằng get\_agent(agent\_id)
6. **Backend chuẩn bị payload** với thông tin mô hình, nhiệt độ từ cấu hình agent
7. **Backend gửi yêu cầu đến LM Studio API**
8. **LM Studio xử lý yêu cầu** và trả về phản hồi
9. **Backend xử lý phản hồi** và trả về cho frontend
10. **Frontend hiển thị phản hồi** cho người dùng

### **4.7 Cấu trúc dữ liệu lưu trữ**

Agent được lưu trữ trong hai nơi:

1. **Cấu hình tĩnh** trong backend/app/config/agents.py
2. **Cơ sở dữ liệu Supabase** trong bảng chat\_sessions với cột agent\_id

Cấu trúc bảng chat\_sessions:

* id: UUID của phiên chat
* user\_id: ID người dùng
* created\_at: Thời gian tạo
* agent\_id: ID của agent được sử dụng

### **4.8 Ưu điểm của Agent System**

1. **Linh hoạt**: Người dùng có thể chọn agent phù hợp với nhu cầu
2. **Mở rộng dễ dàng**: Có thể thêm agent mới bằng cách cập nhật cấu hình
3. **Tùy chỉnh**: Mỗi agent có thể có cấu hình riêng (mô hình, nhiệt độ)
4. **Trải nghiệm nhất quán**: Agent được lưu trữ với phiên chat, đảm bảo trải nghiệm nhất quán

## 5. System Operation: End-to-End Flow trong Student Assistant Agent

Dựa trên mã nguồn được cung cấp, tôi sẽ trình bày chi tiết về luồng hoạt động end-to-end của hệ thống Student Assistant Agent, bao gồm các phần xác thực người dùng, tải lên và xử lý tệp, xử lý truy vấn, cùng với các ví dụ input/output.

### **5.1 User Authentication (Xác thực người dùng)**

**Tổng quan**

Hệ thống Student Assistant Agent sử dụng Supabase để quản lý xác thực người dùng. Quá trình xác thực bao gồm đăng nhập thông thường và lưu trữ thông tin đăng nhập vào hệ thống Học viện Công nghệ Bưu chính Viễn thông (PTIT).

**Luồng hoạt động**

1. **Đăng nhập người dùng**:
   * Người dùng nhập email và mật khẩu vào form đăng nhập
   * Frontend gửi thông tin đến Supabase Auth
   * Nếu thành công, người dùng được chuyển hướng đến trang chat
2. **Lưu trữ thông tin xác thực**:
   * Thông tin người dùng được lưu trong bảng auth.users của Supabase
   * Token xác thực được lưu trong localStorage của trình duyệt
3. **Kiểm tra xác thực khi khởi động ứng dụng**:
   * Khi component ChatInterface được tải, hệ thống kiểm tra trạng thái xác thực
   * Nếu người dùng đã đăng nhập, tải phiên chat và tin nhắn
4. **Lưu trữ thông tin đăng nhập PTIT**:
   * Người dùng nhập thông tin đăng nhập PTIT trong phần Settings
   * Hệ thống gửi thông tin đến endpoint /ptit-login để xác thực
   * Nếu thành công, thông tin được lưu vào bảng university\_credentials

**Cấu trúc dữ liệu**

1. **Bảng auth.users** (Supabase Auth):
   * id: UUID của người dùng
   * email: Email đăng nhập
   * encrypted\_password: Mật khẩu đã mã hóa
   * Các trường khác liên quan đến xác thực
2. **Bảng university\_credentials**:
   * user\_id: Khóa ngoại liên kết với auth.users
   * university\_username: Tên đăng nhập vào hệ thống PTIT
   * university\_password: Mật khẩu PTIT
   * access\_token: Token truy cập PTIT
   * refresh\_token: Token làm mới PTIT
   * token\_expiry: Thời gian hết hạn token
   * name: Tên người dùng từ hệ thống PTIT

### **5.2 File Upload and Processing (Tải lên và xử lý tệp)**

**Tổng quan**

Hệ thống cho phép người dùng tải lên tệp PDF để AI có thể trả lời các câu hỏi về nội dung tệp. Quá trình này bao gồm tải lên tệp, xử lý và vector hóa nội dung, và sử dụng vector để tìm kiếm ngữ nghĩa.

**Luồng hoạt động**

1. **Tải lên tệp**:
   * Người dùng chọn tệp PDF từ thiết bị
   * Frontend kiểm tra loại tệp (chỉ chấp nhận PDF) và kích thước (tối đa 10MB)
   * Tệp được gửi đến endpoint /file/upload dưới dạng FormData
2. **Xử lý tệp trên server**:
   * Backend nhận tệp và lưu vào thư mục tạm thời
   * Trích xuất văn bản từ PDF
   * Chia văn bản thành các đoạn nhỏ (chunks)
   * Vector hóa các đoạn và lưu vào Supabase
3. **Thiết lập ngữ cảnh tệp**:
   * Backend trả về file\_id cho frontend
   * Frontend lưu file\_id vào state activeFileContext
   * Hiển thị thông báo xác nhận tải lên thành công
4. **Xóa ngữ cảnh tệp**:
   * Người dùng có thể xóa ngữ cảnh tệp bằng cách nhấn nút X
   * Frontend gửi yêu cầu xóa đến endpoint /api/files/{file\_id}
   * Backend xóa vector và metadata của tệp

**Cấu trúc dữ liệu**

1. **Bảng files**:
   * id: UUID của tệp
   * user\_id: ID người dùng đã tải lên
   * name: Tên tệp
   * created\_at: Thời gian tải lên
2. **Bảng file\_chunks**:
   * id: UUID của đoạn
   * file\_id: Khóa ngoại liên kết với files
   * content: Nội dung văn bản của đoạn
   * embedding: Vector nhúng của đoạn (dùng cho tìm kiếm ngữ nghĩa)
   * chunk\_index: Vị trí của đoạn trong tệp

### **5.3 Query Processing (Xử lý truy vấn)**

**Tổng quan**

Xử lý truy vấn là quá trình cốt lõi của hệ thống, bao gồm phân loại truy vấn, xử lý theo loại và tạo phản hồi phù hợp. Hệ thống hỗ trợ nhiều loại truy vấn khác nhau như lịch học, lịch thi, truy vấn tệp và truy vấn giáo dục chung.

**Luồng hoạt động**

1. **Gửi truy vấn**:  
   * Người dùng nhập truy vấn vào ChatInput
   * Frontend tạo payload với truy vấn, thông tin người dùng, ngữ cảnh tệp (nếu có), và cài đặt tìm kiếm web
   * Payload được gửi đến endpoint /chat
2. **Phân loại truy vấn**:  
   * Backend sử dụng QueryClassifier để phân loại truy vấn
   * Truy vấn được phân loại thành các loại: schedule, examschedule, date\_query, uml, general, other
3. **Xử lý theo loại truy vấn**:  
     
    a. **Truy vấn lịch học/lịch thi**:  
   * Lấy thông tin đăng nhập PTIT từ request
   * Đăng nhập vào hệ thống PTIT
   * Lấy thông tin học kỳ hiện tại
   * Truy xuất lịch học/lịch thi từ API PTIT
   * Định dạng kết quả và trả về
4. b. **Truy vấn với ngữ cảnh tệp**:  
   * Tìm kiếm các đoạn văn bản liên quan trong tệp
   * Tạo prompt với ngữ cảnh tệp
   * Gửi prompt đến LM Studio
   * Xử lý phản hồi và trả về
5. c. **Truy vấn với tìm kiếm web**:  
   * Thực hiện tìm kiếm web với Brave Search API
   * Tạo prompt với kết quả tìm kiếm
   * Gửi prompt đến LM Studio
   * Xử lý phản hồi và trả về kèm nguồn
6. d. **Truy vấn giáo dục chung**:  
   * Tạo prompt giáo dục chung
   * Gửi prompt đến LM Studio
   * Xử lý phản hồi và trả về
7. **Lưu trữ tin nhắn**:  
   * Backend lưu tin nhắn người dùng và phản hồi vào bảng messages
   * Frontend cập nhật giao diện với tin nhắn mới

**Cấu trúc dữ liệu**

1. **Bảng chat\_sessions**:
   * id: UUID của phiên chat
   * user\_id: ID người dùng
   * created\_at: Thời gian tạo
   * agent\_id: ID của agent được sử dụng
2. **Bảng messages**:
   * id: UUID của tin nhắn
   * chat\_id: Khóa ngoại liên kết với chat\_sessions
   * role: Vai trò (user, assistant, system)
   * content: Nội dung tin nhắn
   * created\_at: Thời gian tạo
   * sources: Mảng JSON chứa nguồn tìm kiếm web (nếu có)

## **6.** Vector **Embedding: Technical Deep Dive**

### **6.1 Implementation Logic (Logic triển khai)**

Hệ thống Vector Embedding trong Student Assistant Agent được triển khai chủ yếu trong lớp FileService. Lớp này sử dụng mô hình Sentence Transformer để tạo vector nhúng (embeddings) cho các đoạn văn bản, cho phép tìm kiếm ngữ nghĩa trên nội dung tài liệu đã tải lên.

**Khởi tạo và cấu hình**

Trong phương thức khởi tạo của FileService, hệ thống tải mô hình Sentence Transformer và thiết lập các thông số cần thiết:

Các thông số quan trọng:

* **Mô hình**: all-MiniLM-L6-v2 - Một mô hình nhẹ nhưng hiệu quả từ thư viện sentence-transformers
* **Kích thước embedding**: 384 - Số chiều của vector nhúng, phù hợp với mô hình được chọn

**Ý nghĩa của mô hình all-MiniLM-L6-v2**:

* Đây là một mô hình được distill (nén) từ mô hình lớn hơn, giúp giảm kích thước và tăng tốc độ xử lý
* L6 chỉ số lớp transformer (6 lớp)
* v2 là phiên bản cải tiến
* Mô hình này được huấn luyện để tạo ra các vector nhúng có khả năng biểu diễn ngữ nghĩa của văn bản tốt

### **6.2 Embedding Generation (Tạo Embedding)**

**Quá trình tạo Embedding**

Quá trình tạo embedding được thực hiện thông qua phương thức \_create\_embedding:

**Các bước xử lý**:

1. **Kiểm tra văn bản đầu vào**: Nếu văn bản trống hoặc quá ngắn (dưới 3 ký tự), sử dụng văn bản mặc định "empty document"
2. **Tạo embedding**: Sử dụng phương thức encode() của mô hình để chuyển đổi văn bản thành vector nhúng
3. **Chuyển đổi định dạng**: Chuyển vector numpy thành danh sách Python để tương thích với Supabase
4. **Xử lý lỗi**: Nếu có lỗi, trả về vector rỗng (toàn số 0) với kích thước đúng

**Xử lý tài liệu và tạo Embedding**

Khi người dùng tải lên tệp, nội dung tệp được chia thành các đoạn nhỏ (chunks) và vector hóa:

**Giải thích quy trình**:

1. Chia văn bản thành các đoạn 500 ký tự, với bước nhảy 400 ký tự (có chồng lấp 100 ký tự)
2. Tạo vector nhúng cho mỗi đoạn bằng cách gọi phương thức \_create\_embedding

Sau đó, các đoạn và vector nhúng tương ứng được lưu vào bảng file\_chunks trong Supabase

**Cấu trúc dữ liệu lưu trữ**:

* Mỗi bản ghi chứa file\_id, chunk\_index, content và embedding
* Các bản ghi được chèn theo lô (batch) để tối ưu hiệu suất, mỗi lô tối đa 100 bản ghi

### **6.3 Semantic Search Implementation (Triển khai Tìm kiếm Ngữ nghĩa)**

Hệ thống sử dụng tìm kiếm ngữ nghĩa để tìm các đoạn văn bản liên quan đến truy vấn của người dùng. Quá trình này được thực hiện thông qua phương thức search\_relevant\_chunks\_in\_supabase:

**def** **search\_relevant\_chunks\_in\_supabase**(self, query, file\_id, top\_k=10):

*# Tạo embedding cho truy vấn*

emb = self.\_create\_embedding(query)

*# Thiết lập ngưỡng tương đồng cosine*

match\_threshold = 0.5 *# Giá trị cao hơn cho kết quả chất lượng tốt hơn*

*# Gọi RPC để tìm kiếm vector*

response = supabase.rpc('match\_file\_chunks', {

'query\_embedding': emb,

'p\_file\_id': file\_id,

'match\_threshold': match\_threshold,

'match\_count': top\_k

}).execute()

*# Xử lý kết quả và trả về các đoạn văn bản*

*# ...*

**Các thông số quan trọng**:

* **top\_k=10**: Số lượng kết quả tối đa trả về
* **match\_threshold=0.5**: Ngưỡng tương đồng cosine, các đoạn có độ tương đồng dưới ngưỡng này sẽ bị loại bỏ
* **match\_count**: Số lượng kết quả tối đa cần tìm, được đặt bằng top\_k

**Tính toán Cosine Similarity**

Cosine Similarity là phương pháp đo lường độ tương đồng giữa hai vector bằng cách tính cosine của góc giữa chúng:

cosine\_similarity(A, B) = (A · B) / (||A|| \* ||B||)

Trong đó:

* A · B là tích vô hướng của hai vector
* ||A|| và ||B|| là độ dài Euclidean của hai vector

Giá trị cosine similarity nằm trong khoảng [-1, 1]:

* 1: Hai vector hoàn toàn giống nhau
* 0: Hai vector vuông góc (không liên quan)
* -1: Hai vector hoàn toàn đối lập

Trong hệ thống này, match\_threshold=0.5 có nghĩa là chỉ chấp nhận các đoạn có độ tương đồng với truy vấn từ 0.5 trở lên.

**Ví dụ tính toán Cosine Similarity**

Giả sử chúng ta có:

* Vector truy vấn: [0.1, 0.2, 0.3, 0.4]
* Vector đoạn 1: [0.2, 0.3, 0.4, 0.5] (tương đồng cao)
* Vector đoạn 2: [0.5, -0.4, 0.3, -0.2] (tương đồng thấp)

Tính cosine similarity:

* cos(truy vấn, đoạn 1) = (0.1×0.2 + 0.2×0.3 + 0.3×0.4 + 0.4×0.5) / (√(0.1²+0.2²+0.3²+0.4²) × √(0.2²+0.3²+0.4²+0.5²)) ≈ 0.97
* cos(truy vấn, đoạn 2) = (0.1×0.5 + 0.2×(-0.4) + 0.3×0.3 + 0.4×(-0.2)) / (√(0.1²+0.2²+0.3²+0.4²) × √(0.5²+(-0.4)²+0.3²+(-0.2)²)) ≈ 0.08

Kết quả:

* Đoạn 1: 0.97 > 0.5 (match\_threshold) → Được chọn
* Đoạn 2: 0.08 < 0.5 (match\_threshold) → Bị loại bỏ

### **6.4 Integration with AI Service (Tích hợp với AI Service)**

Hệ thống Vector Embedding được tích hợp với AI Service thông qua phương thức chat\_with\_file\_context. Phương thức này sử dụng kết quả tìm kiếm ngữ nghĩa để tạo ngữ cảnh cho AI:

**Quy trình tích hợp**:

1. Nhận truy vấn từ người dùng và ID tệp
2. Lấy cấu hình agent (mô hình, temparature)
3. Tìm kiếm các đoạn văn bản liên quan trong tệp bằng search\_relevant\_chunks\_in\_supabase
4. Tạo ngữ cảnh từ các đoạn văn bản tìm được
5. Tạo prompt cho AI với ngữ cảnh tệp
6. Gửi prompt đến LM Studio
7. Xử lý phản hồi và trả về kết quả

Đoạn mã sau cho thấy cách hệ thống tìm kiếm các đoạn văn bản liên quan:

Sau khi tìm được các đoạn văn bản liên quan, hệ thống tạo prompt cho AI:

**Ví dụ luồng hoạt động end-to-end**

**Ví dụ 1: Tải lên tệp và tạo embedding**

1. Người dùng tải lên tệp PDF "Phương pháp học tập hiệu quả.pdf"

Backend trích xuất văn bản từ PDF, ví dụ:Phương pháp học tập hiệu quả cho sinh viên PTIT bao gồm: 1. Quản lý thời gian hợp lý 2. Ghi chú có hệ thống 3. Ôn tập thường xuyên 4. Học nhóm hiệu quả 5. Sử dụng công nghệ hỗ trợ

1. Chia văn bản thành các đoạn 500 ký tự (trong ví dụ này chỉ có 1 đoạn)
2. Tạo vector nhúng cho đoạn văn bản bằng mô hình all-MiniLM-L6-v2
3. Lưu đoạn văn bản và vector nhúng vào bảng file\_chunks

**Ví dụ 2: Truy vấn với ngữ cảnh tệp**

1. Người dùng gửi truy vấn: "Làm thế nào để quản lý thời gian học tập?"
2. Backend tạo vector nhúng cho truy vấn bằng cùng mô hình all-MiniLM-L6-v2
3. Tính cosine similarity giữa vector truy vấn và các vector đoạn văn bản
4. Chọn các đoạn có cosine similarity > 0.5 và sắp xếp theo thứ tự giảm dần
5. Lấy tối đa 10 đoạn (top\_k=10) có độ tương đồng cao nhất
6. Tạo prompt cho AI với các đoạn văn bản tìm được và trả lời

## **7.** Hiệu năng hệ thống và các giới hạn

### **7.1 Đặc điểm hiệu năng**

* Thời gian phản hồi: Thông thường từ 2–5 giây đối với các truy vấn cơ bản, và từ 5–10 giây đối với các truy vấn yêu cầu ngữ cảnh từ tài liệu.
* Xử lý tài liệu: Mất khoảng 1–2 phút để xử lý một tài liệu PDF gồm 20 trang.
* Người dùng đồng thời: Hệ thống được thiết kế để xử lý nhiều người dùng truy cập cùng lúc.

### **7.2 Hạn chế**

* Loại tài liệu: Hiện tại chỉ hỗ trợ các tập tin định dạng PDF.
* Kích thước tập tin: Giới hạn tối đa 10MB cho mỗi tập tin.
* Hạn chế của mô hình: Phụ thuộc vào khả năng của các mô hình LM Studio được tích hợp.
* Dữ liệu lịch học: Bị giới hạn trong phạm vi thông tin được cung cấp bởi API của Học viện PTIT**.**

## 8. Backend Frameworks và Libraries

### **8.1 Flask**

Flask là một micro web framework cho Python, được sử dụng để xây dựng API backend của ứng dụng.

* **Tài liệu chính thức**: [Flask Documentation](https://flask.palletsprojects.com/)

### **8.2 Sentence Transformers**

Thư viện này được sử dụng để tạo vector embeddings cho tài liệu, cho phép tìm kiếm ngữ nghĩa.

* **Tài liệu chính thức**: [Sentence Transformers Documentation](https://www.sbert.net/)

### **8.3 HTTPX**

HTTPX là một client HTTP đầy đủ tính năng cho Python, hỗ trợ cả đồng bộ và bất đồng bộ.

* **Tài liệu chính thức**: [HTTPX Documentation](https://www.python-httpx.org/)

### **8.4 BeautifulSoup**

Thư viện phân tích cú pháp HTML/XML, được sử dụng trong web scraping.

* **Tài liệu chính thức**: [BeautifulSoup Documentatio](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

### **8.5 Supabase**

Supabase là một nền tảng backend-as-a-service mã nguồn mở, cung cấp cơ sở dữ liệu, xác thực và lưu trữ.

* **Tài liệu chính thức**: [Supabase Documentation](https://supabase.io/docs)

## 9. Frontend Frameworks và Libraries

### **9.1 React**

React là một thư viện JavaScript để xây dựng giao diện người dùng.

* **Tài liệu chính thức**: [React Documentation](https://reactjs.org/docs/getting-started.html)

### **9.3 Lucide React**

Thư viện biểu tượng cho React.

* **Tài liệu chính thức**: [Lucide React Documentation](https://lucide.dev/docs/lucide-react)

## 10. AI và NLP Libraries

### **10.1 LM Studio**

LM Studio được sử dụng để tương tác với các mô hình ngôn ngữ lớn.

* **Tài liệu chính thức**: [LM Studio](https://lmstudio.ai/)

### **10.2. Brave Search API**

API tìm kiếm web được sử dụng để tìm kiếm thông tin trên internet.

* **Tài liệu chính thức**: [Brave Search API Documentation](https://brave.com/search/api/)

## 11. Các Thư Viện Hỗ Trợ Khác

### **11.1 Unidecode**

Thư viện chuyển đổi văn bản có dấu thành không dấu.

* **Tài liệu chính thức**: [Unidecode PyPI](https://pypi.org/project/Unidecode/)