

AEYEPRO

MAT3508 - NHẬP MÔN AI



Nhóm 12:
Lê Tiến Mạnh
Hoàng Văn Phú
Trần Minh Đức

MỤC LỤC

1. Giới thiệu vấn đề
2. Phương pháp tiếp cận
3. Triển khai và phân tích
4. Kết luận và hướng phát triển

• •

1. GIỚI THIỆU VẤN ĐỀ

- Rất nhiều người đang làm việc trước màn hình máy tính hàng giờ liền, sai tư thế và không đảm bảo sức khỏe cho mắt.



1. GIỚI THIỆU VẤN ĐỀ

- Khô mắt, rát mắt, cận thị, đau đầu.
- Các vấn đề về cột sống và cơ xương (đặc biệt là cổ – vai – lưng).



2. PHƯƠNG PHÁP TIẾP CẬN

2.1 Thị giác máy tính

- Cân bằng giữa hiệu năng và độ chính xác
- Độ tin cậy của dữ liệu
- Khả năng xử lý thời gian thực
- Khả năng triển khai thực tế



Sự kết hợp giữa **OpenCV** và **Mediapipe** là lựa chọn tối ưu.

2.1 THỊ GIÁC MÁY TÍNH

a) Mediapipe

- **Sự ổn định theo thời gian**

Giảm nhiễu và tăng tính nhất quán qua các khung hình

- **Tích hợp đa nền tảng**

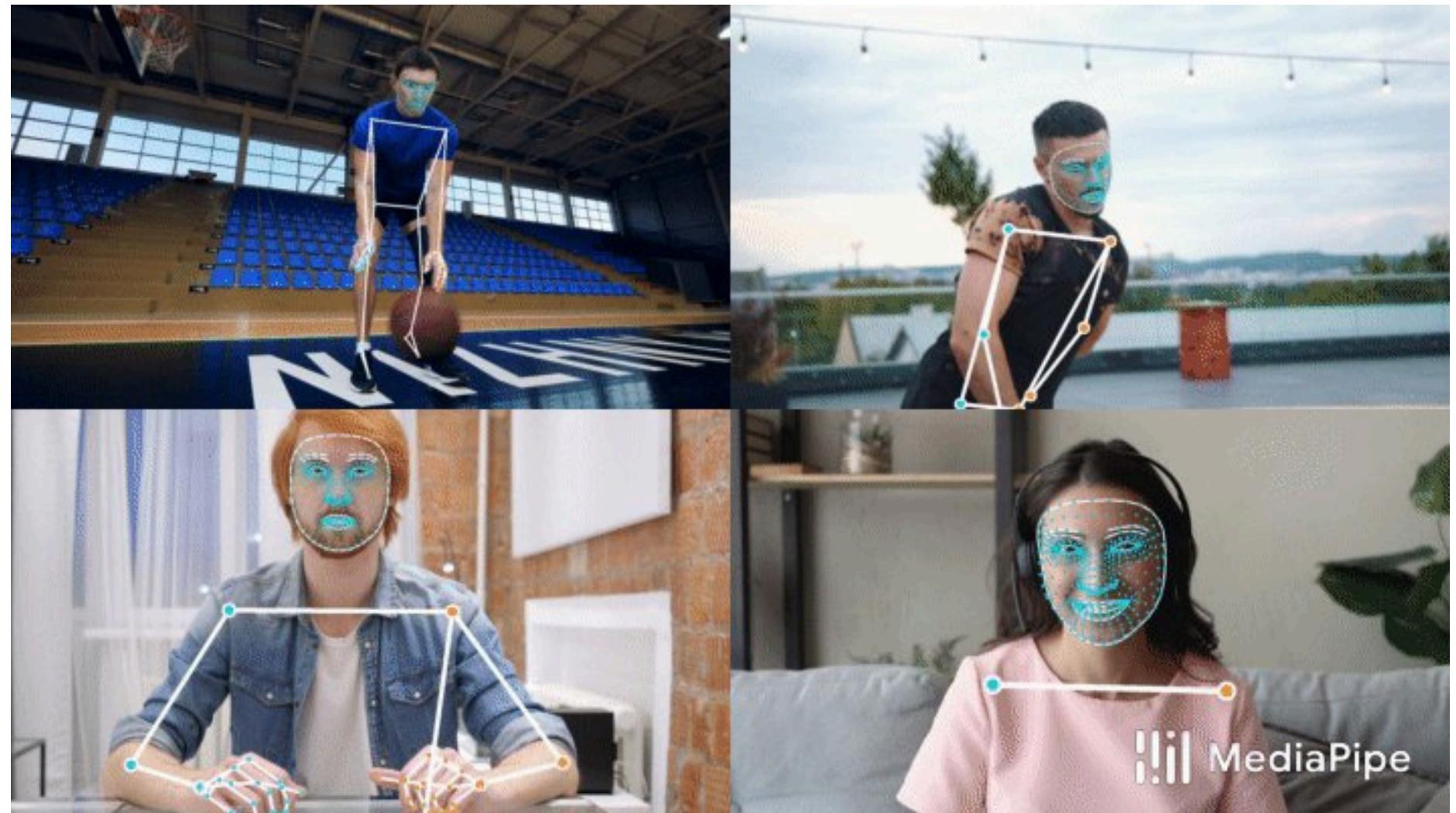
Hỗ trợ trên desktop, web và thiết bị di động

- **Tính năng phù hợp**

Cung cấp World Landmarks cho phân tích không gian

- **Khả năng triển khai dễ dàng**

Tối ưu cho CPU thông thường, không yêu cầu GPU cao cấp



2.1 THỊ GIÁC MÁY TÍNH

a) Mediapipe

Tiêu chí	MediaPipe	YOLOv5-Pose	MoveNet	OpenPose
Giám sát mắt (Gaze/Iris)	Có sẵn, chính xác cao	Không hỗ trợ	Không hỗ trợ	Không chuyên dụng
Giám sát tư thế	Ôn định cho 1 người	Tốt cho nhiều người	Tốt cho fitness	Rất chính xác
Tốc độ Real-time	30+ FPS (CPU)	90+ FPS (GPU)	60+ FPS (CPU/Mobile)	Chậm trên CPU
Tài nguyên phần cứng	CPU, Web, nhẹ	GPU là bắt buộc	Rất nhẹ	GPU mạnh, VRAM lớn
Phù hợp Laptop cá nhân	Rất phù hợp	Không phù hợp	Phù hợp	Không phù hợp

2.1 THỊ GIÁC MÁY TÍNH

b) OpenCV

- Là tầng trung gian lấy dữ liệu cho Mediapipe

Cấu trúc dữ liệu: Xử lý ảnh dưới dạng ma trận đa chiều (Multidimensional Dense Array).

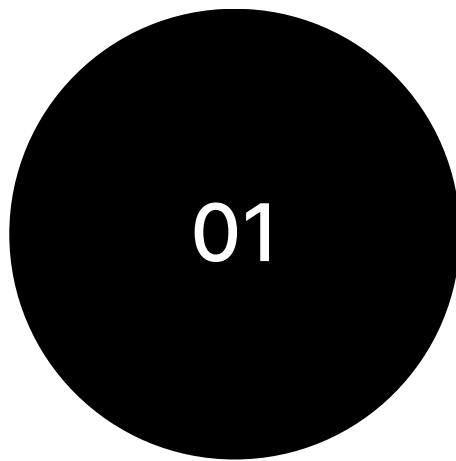
- Mỗi khung hình là một Tensor 3 chiều ($H \times W \times 3$).
- Giá trị pixel: Số nguyên 8-bit [0, 255].
- Mô hình màu: B-G-R (Blue-Green-Red).

Hiệu năng:

- Triển khai trên Python nhưng gọi xuống C++ (Low-level optimization).
- Tốc độ tính toán ma trận cao, quản lý bộ nhớ hiệu quả.
- Tương thích hoàn toàn với NumPy.

2.1 THỊ GIÁC MÁY TÍNH

b) OpenCV



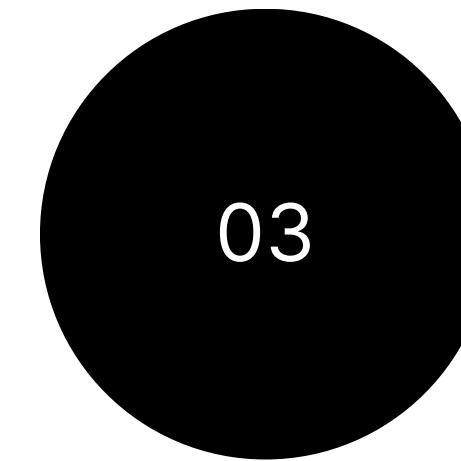
THU THẬP DỮ LIỆU

`cv2.videoCapture()`



CHUYỂN ĐỔI KHÔNG GIAN
MÀU

BGR → RGB



CHUẨN HÓA HÌNH HỌC

Lật ảnh qua trục dọc

2.1 THỊ GIÁC MÁY TÍNH

b) Cơ chế kết hợp

- **Input:** OpenCV thu nhận & xử lý ảnh BGR -> RGB.
- **Inference:** MediaPipe tính toán Landmarks.
- **Analysis:** Phân tích chỉ số hình học từ Landmarks.
- **Decision:** So sánh ngưỡng an toàn -> Cảnh báo.



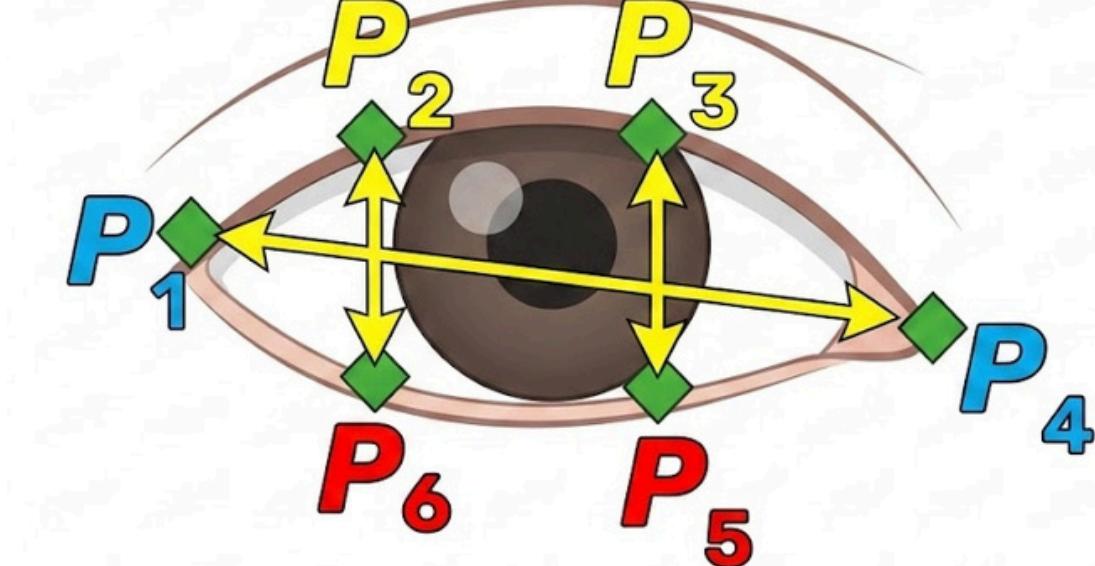
2.1 THỊ GIÁC MÁY TÍNH

c) Các tiêu chí đánh giá

EAR (Eye Aspect Ratio):

- Nhắm mắt: Tử số $\rightarrow 0 \Rightarrow$ EAR giảm đột ngột.
- Buồn ngủ (Drowsiness): EAR duy trì mức thấp trong khoảng 3 - 5 giây.

\Rightarrow Hệ thống kích hoạt cảnh báo người dùng cần nghỉ ngơi.



$$\text{EAR} = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2 \|P_1 - P_4\|}$$

2.1 THỊ GIÁC MÁY TÍNH

c) Các tiêu chí đánh giá

- Độ lệch vai
- Góc nghiêng đầu trái/phải
- Góc cúi/ngửa đầu
- Tiêu chí khác: Khoảng cách mắt đến màn hình, số lần buồn ngủ, thông số chớp mắt

2.2 RAG

a) Tổng quan: RAG truyền thống

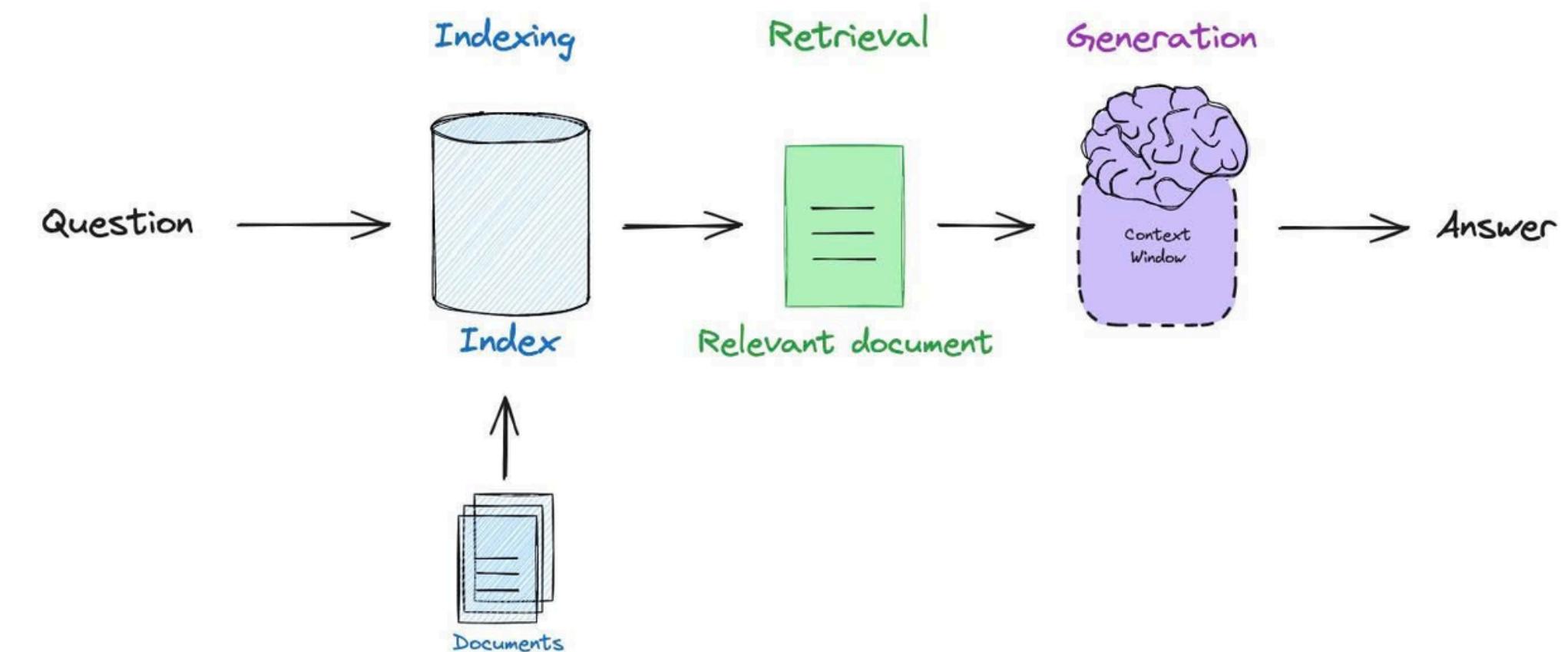
Mục tiêu: Bổ sung bộ nhớ ngoài để cung cấp thêm thông tin ngữ cảnh cho LLM.

Quy trình đơn hướng (One-way):

- Retrieve (Truy xuất) → Read (Đọc)
→ Generate (Sinh)

Công thức:

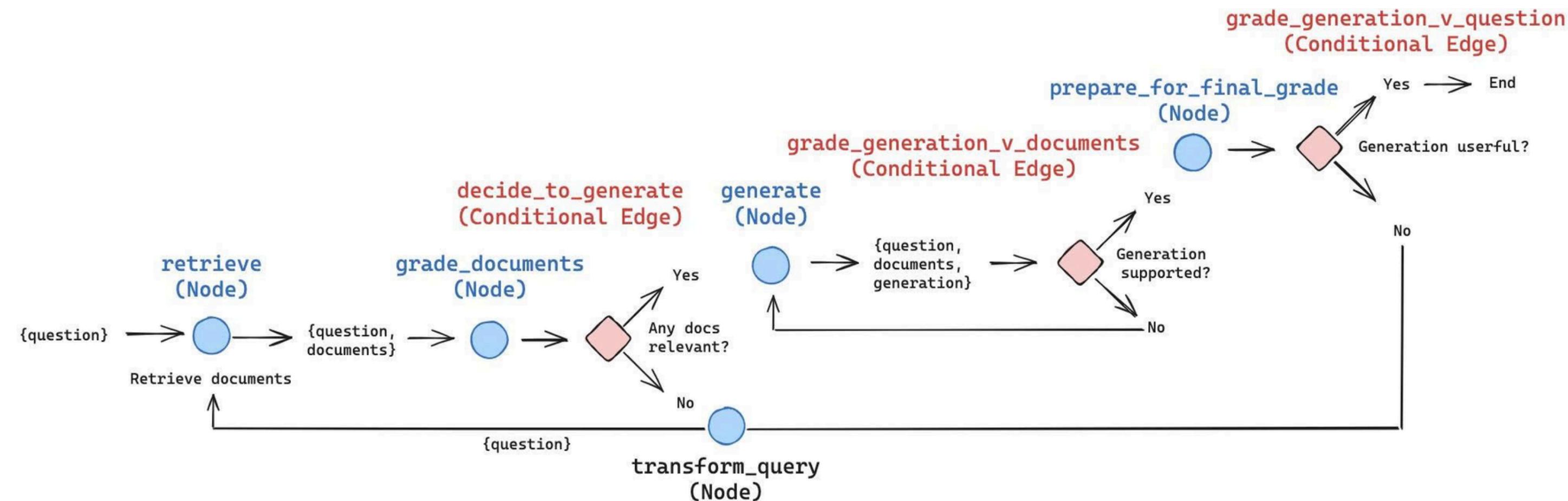
- Truy xuất: Tìm Top-K văn bản D tương đồng nhất với câu hỏi q .
- Sinh câu trả lời



2.2 RAG

b) Adaptive RAG

- Chuyển từ mô hình "Dây chuyền sản xuất" (Linear) sang "Tác nhân nhận thức" (Cognitive Agent).
- Thay đổi cách hoạt động: Đánh giá → Lập kế hoạch → Thực thi → Kiểm chứng.



2.2 RAG

b) Adaptive RAG

Cơ chế định tuyến Động (Dynamic Query Routing):

- Hỏi lý thuyết → Vector Store (ChromaDB).
- Hỏi số liệu → Pandas Agent.
- Hỏi xã giao → LLM trực tiếp (Bỏ qua truy xuất).

Cơ chế tự phản ứng & sửa lỗi (Self-Reflection):

- Retrieval Grader: Tài liệu có liên quan không? → Nếu không: Viết lại câu hỏi (Rewrite Query).
- Hallucination Grader: Trả lời có bịa đặt không?
- Answer Grader: Đã giải quyết vấn đề người dùng chưa?

Kiến trúc không chu trình sang đồ thị có chu trình (Cyclic Graph):

- Chuyển đổi cấu trúc: DAG (Tuyến tính) → Cyclic Graph (Vòng lặp).
- Duy trì trạng thái (Statefulness): Hệ thống "nhớ" lỗi sai để điều chỉnh hướng đi mới.

2.2 RAG

c) Các công nghệ được sử dụng

LangGraph: Quản lý luồng hoạt động hệ thống

- Dựa trên mô hình Pregel (Bulk Synchronous Parallel).
- Quản lý GraphState (Bộ nhớ chia sẻ) để điều khiển định tuyến và cơ chế Retry tự động.

ChromaDB: Lưu trữ dữ liệu

- Tài liệu được chunking và embedding vào đây.
- Thuật toán tìm kiếm láng giềng gần nhất xấp xỉ (ANN) độ phức tạp $O(\log(N))$.

Hugging Face Sentence Transformers: Embedding

- Bảo mật tuyệt đối: Dữ liệu bệnh nhân không rời khỏi hệ thống (Tuân thủ HIPAA).
- Tối ưu: Loại bỏ chi phí tính toán theo token, giảm độ trễ mạng.

2.2 RAG

c) Các công nghệ được sử dụng

LangChain & Pandas Agent: Xử lý dữ liệu

- Cơ chế: Không ép LLM "học thuộc" số liệu, mà dùng LLM để sinh mã Python.
- Thực thi: Chạy mã trong môi trường Sandbox → Đảm bảo tính toán chính xác tuyệt đối (trung bình, tổng hợp, lọc) mà Vector Search không làm được.

LangSmith: Quan sát & đánh giá

- Chức năng:
 - Tracing: Theo dõi input/output từng node và thời gian thực thi.
 - Debugging: Phân tích kết quả của các bước chấm điểm (Grading).
- Mục đích: Tinh chỉnh Prompt và tối ưu hóa ngưỡng quyết định (Decision Thresholds).

3. TRIỂN KHAI

3.1 Kiến trúc tổng thể

1. Mô hình:

- Client-Server với cơ chế tách biệt hoàn toàn giữa giao diện (Frontend) và xử lý logic (Backend).

2. Frontend (Client):

- HTML/CSS/JS để làm giao diện người dùng.
- Nhiệm vụ: Hiển thị Video Stream, Bảng chỉ số sức khỏe, Chatbot.

3. Backend (Server):

- Ngôn ngữ: Python + Flask Framework.
- Nhiệm vụ: Xử lý thị giác máy tính, quản lý mô hình RAG, điều phối API.

3. TRIỂN KHAI

3.1 Kiến trúc tổng thể

1. Giao thức HTTP (RESTful API):

- Đặc điểm: Tin cậy cao, phi trạng thái.
- Dùng cho: Các lệnh **điều khiển** (Bật/Tắt Camera), **cài đặt** (Settings), gửi tin nhắn Chat.

2. Giao thức WebSocket (Socket.IO):

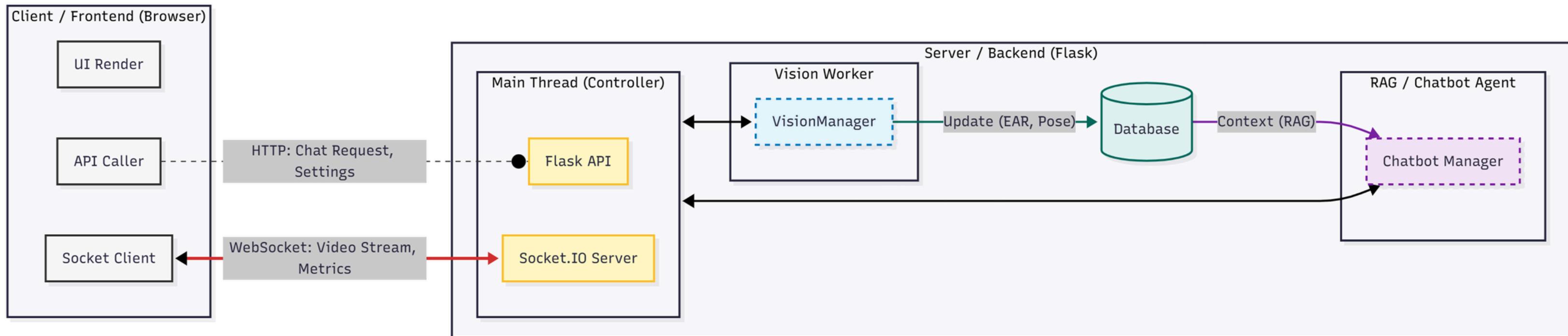
- Đặc điểm: Kết nối 2 chiều liên tục, độ trễ cực thấp.
- Dùng cho: Stream video và cập nhật chỉ số sức khỏe.

3. Xử lý Đa luồng & Singleton Pattern:

- Tách biệt luồng: Tách biệt luồng chính (Flask server) và phụ (Xử lý ảnh).
- Singleton Design Pattern: Đảm bảo chỉ có một instance quản lý Camera và GPU, tránh xung đột tài nguyên phần cứng

3. TRIỂN KHAI

3.1 Kiến trúc tổng thể



3. TRIỂN KHAI

3.2 Giao diện người dùng

1. Công nghệ:

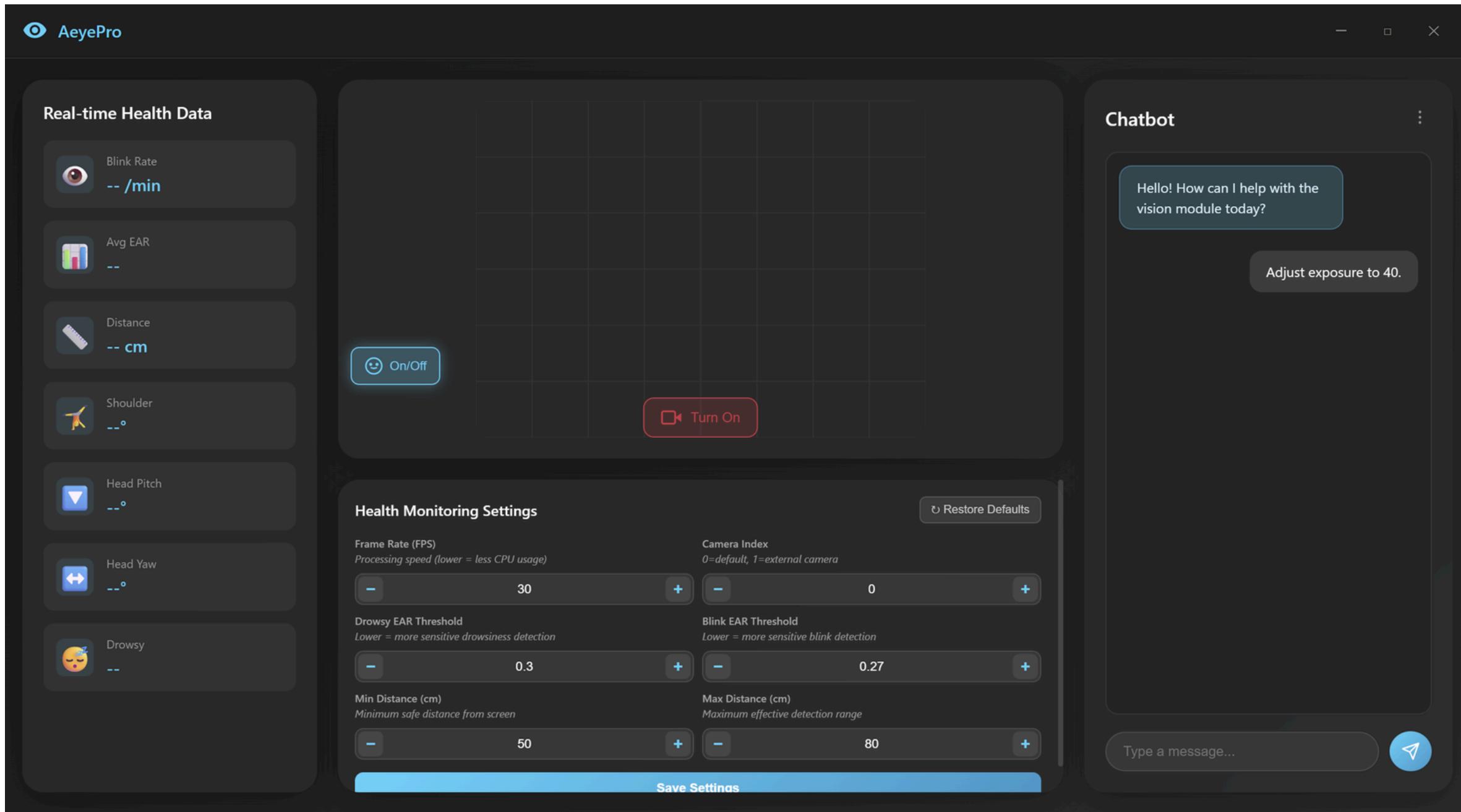
- HTML: Cố định layout.
- CSS: Theme và tương tác với giao diện.
- JavaScript: gửi API Requests và giao tiếp với Backend.

2. Bố cục (Layout): Chia làm 4 vùng chức năng:

- Hiển thị số liệu Real-time (EAR, Blink Rate).
- Viewport: Hiển thị Camera.
- Tùy chỉnh cài đặt.
- Chatbot: Cửa sổ hội thoại thông minh.

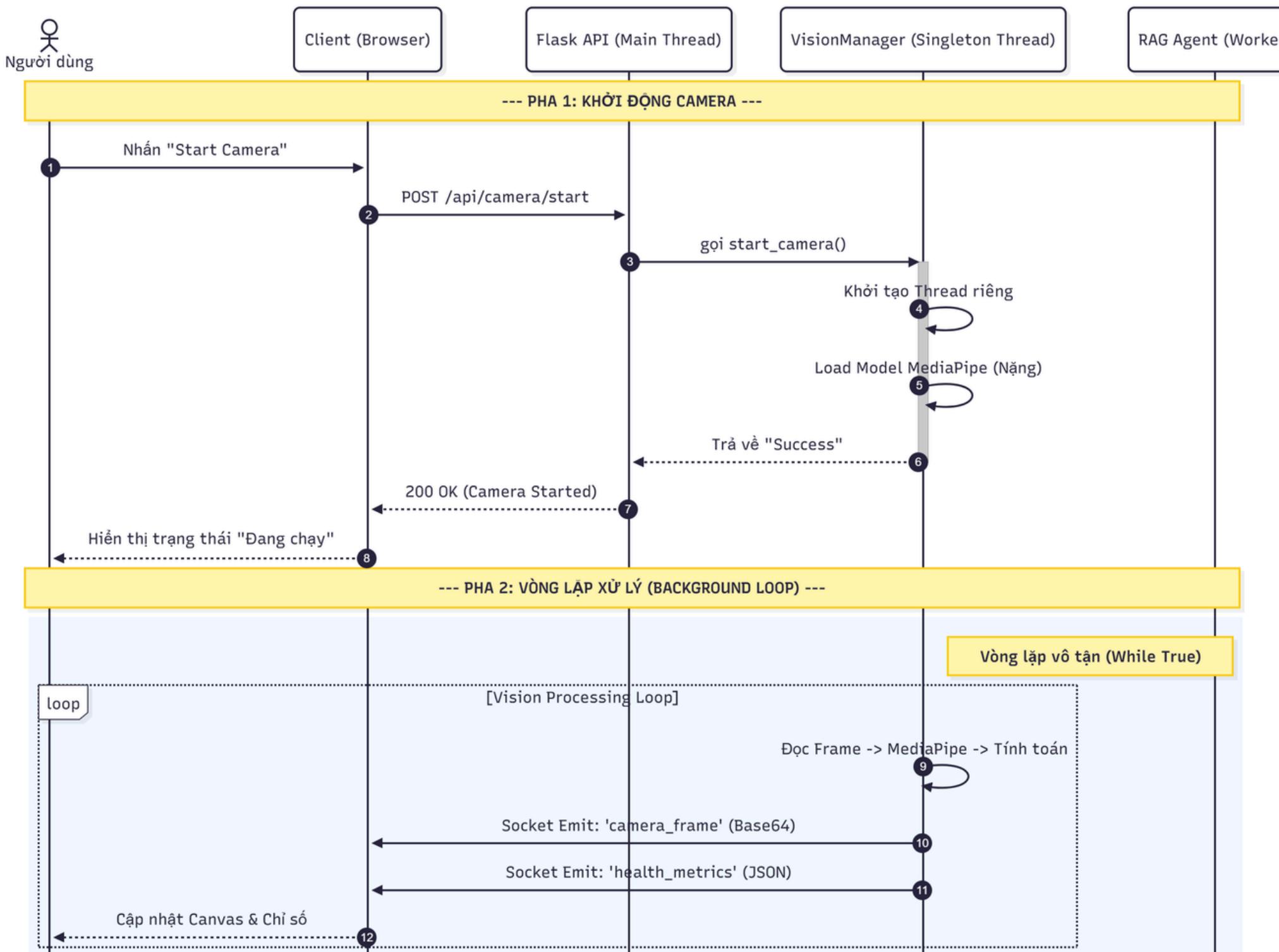
3. TRIỂN KHAI

3.2 Giao diện người dùng



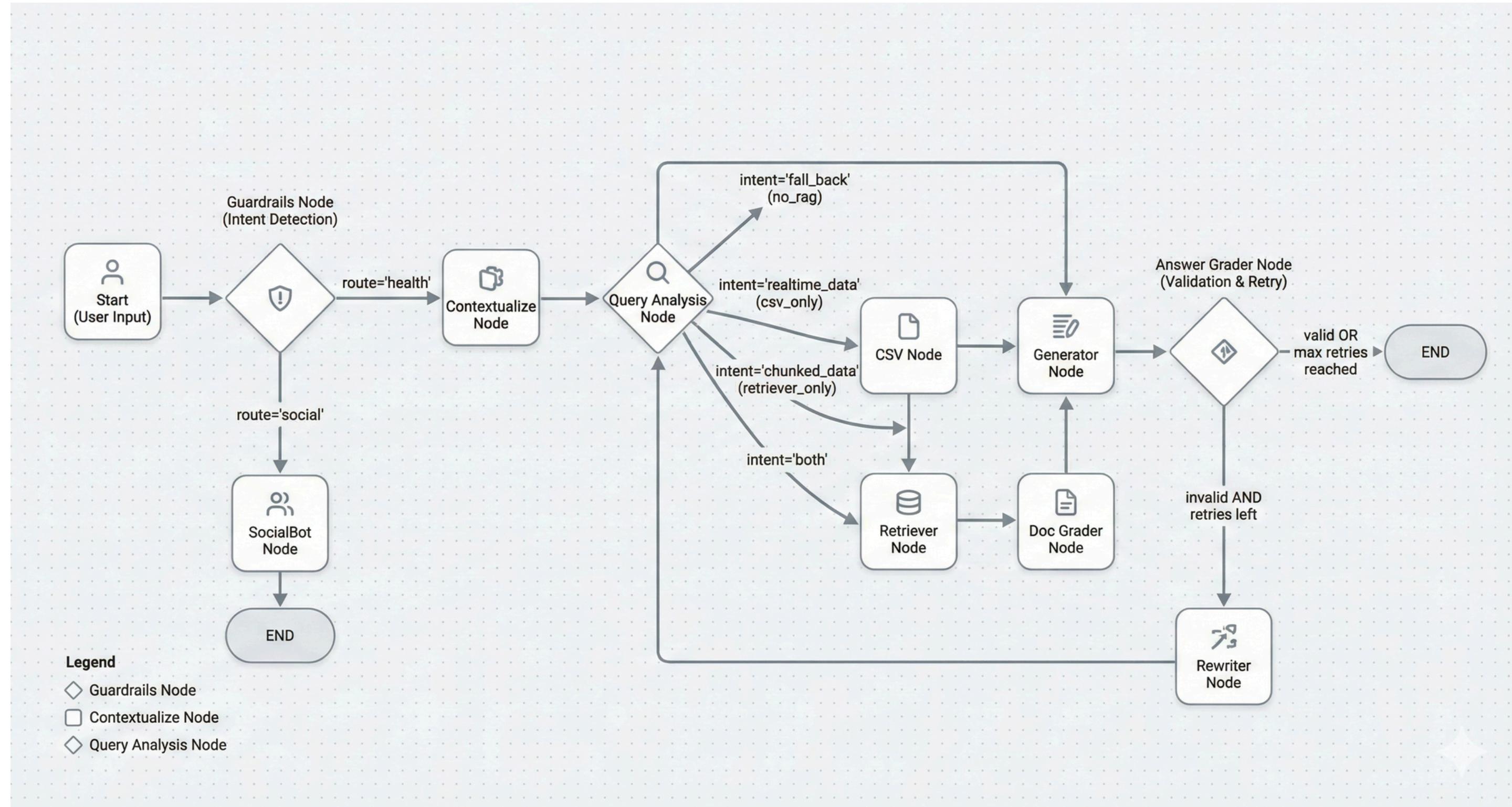
3. TRIỂN KHAI

3.3.1 Tổng quan về luồng xử lý thị giác



3. TRIỂN KHAI

3.4.2 Tổng quan về luồng RAG



4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết luận

1. Về kỹ thuật

- Xây dựng kiến trúc đa luồng kết hợp Computer Vision thời gian thực và Generative AI.
- Hệ thống MediaPipe + OpenCV ổn định 15–30 FPS; trích xuất chính xác EAR, góc đầu, vai.
- Phát triển Chatbot Adaptive RAG + LangGraph, xử lý dữ liệu đa phương thức.

2. Về sản phẩm

- Giao diện thân thiện, đầy đủ chức năng.
- Cảnh báo thời gian thực + báo cáo lịch sử sức khỏe.
- Chatbot cá nhân hóa dựa trên dữ liệu người dùng.

3. Ứng dụng thực tế

- Hỗ trợ giảm nguy cơ mắc hội chứng thị giác máy tính, đau cổ vai gáy cho học sinh, sinh viên, dân văn phòng.
- Triển khai được trên máy tính phổ thông nhờ MediaPipe CPU và Gemini Flash tối ưu chi phí.

4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.2 Hướng phát triển

1. Mục tiêu ngắn hạn

- Tăng trải nghiệm người dùng: thêm biểu đồ sức khỏe theo tuần/tháng.
- Cải thiện thuật toán cảnh báo và giảm cảnh báo sai.
- Xây dựng bộ cài đặt/.exe tiện dụng.

2. Mục tiêu dài hạn

- Phát triển phiên bản Mobile / Cross-platform.
- Xây dựng hệ thống Cloud để đồng bộ dữ liệu và đưa ra lời khuyên chính xác.
- Nâng cấp hệ thống từ Adaptive RAG sang kiến trúc Agentic AI để tự động thực hiện các tác vụ phức tạp hơn.

4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.3 Phương pháp thực hiện

- Tích hợp Small Language Models (Gemma 2) để hỗ trợ hoạt động ngoại tuyến và bảo mật dữ liệu.
- Chuyển sang kiến trúc Microservices (Vision, Chatbot, User/Data) giúp dễ dàng mở rộng và bảo trì.
- Hợp tác chuyên gia y tế để chuẩn hóa Knowledge Base.
- Tập trung mở rộng kho công cụ (Tools) cho Agent thông qua việc tích hợp các API ngoại vi (như Google Calendar API, Hospital RESTful API).

THE END