FaceCam – Tài Liệu Thiết Kế Hệ Thống

Contents

[1. Từ điển thuật ngữ 2](#_Toc206169096)

[2. Ngôn ngữ thiết kế 2](#_Toc206169097)

[3. Các bên liên quan 2](#_Toc206169098)

[4. Các mối quan tâm 3](#_Toc206169099)

[4.1 Yêu cầu chức năng 3](#_Toc206169100)

[4.2 Yêu cầu phi chức năng 3](#_Toc206169101)

[4.3 Mô tả Use Case 3](#_Toc206169102)

[5. Các lát cắt 4](#_Toc206169103)

[5.1 Context View 4](#_Toc206169104)

[5.2 Logical View 5](#_Toc206169105)

[5.3 Process / Sequence View 5](#_Toc206169106)

[5.4 Deployment View 5](#_Toc206169107)

[5.5 Data View 5](#_Toc206169108)

[6. Các thành phần 5](#_Toc206169109)

[7. Lý do các quyết định thiết kế 6](#_Toc206169110)

[8. Cấu hình & Triển khai 6](#_Toc206169111)

[9. API chính 6](#_Toc206169112)

[10. Kiểm thử 6](#_Toc206169113)

[11. Kế hoạch phát triển tương lai 7](#_Toc206169114)

# 1. Từ điển thuật ngữ

- FaceCam: Tên hệ thống nhận diện khuôn mặt.

- Camera HTTP/RTSP: Nguồn video stream từ IP camera, webcam qua HTTP/MJPEG hoặc RTSP.

- DeepFace: Thư viện nhận diện khuôn mặt (sử dụng mô hình Facenet).

- Embedding: Vector đặc trưng trích xuất từ ảnh khuôn mặt.

- Cosine distance: Thước đo độ tương đồng giữa các embedding.

- Snapshot: Ảnh cắt từ khung hình khi có nhận diện thành công.

- Frontend Admin: Giao diện quản trị (quản lý người, camera, lịch sử).

- Frontend User: Giao diện xem luồng và lịch sử nhận diện.

- Thread: Luồng xử lý riêng đọc và phân tích video.

# 2. Ngôn ngữ thiết kế

- Backend: Python 3.11, FastAPI, SQLAlchemy, PostgreSQL, OpenCV, DeepFace.

- Frontend: ReactJS, Ant Design.

- Triển khai: Docker, Docker Compose.

- Giao tiếp: REST API, MJPEG stream.

# 3. Các bên liên quan

|  |  |
| --- | --- |
| **Bên liên quan** | **Vai trò** |
| Quản trị hệ thống | Quản lý camera, danh sách người |
| Người dùng quan sát | Xem luồng camera, lịch sử nhận diện |
| Bộ phận phát triển | Triển khai, bảo trì hệ thống, thay đổi cấu hình. |

# 4. Các mối quan tâm

## 4.1 Yêu cầu chức năng

- Quản lý danh sách người với ảnh mẫu.

- Quản lý nhiều camera HTTP/RTSP.

- Nhận diện khuôn mặt thời gian thực.

- Lưu snapshot + thông tin nhận diện.

- Xem lịch sử nhận diện.

## 4.2 Yêu cầu phi chức năng

- Độ trễ nhận diện thấp, nhận diện và cập nhật sự kiện trong vòng 5 giây.

- Khả năng mở rộng với đa camera.

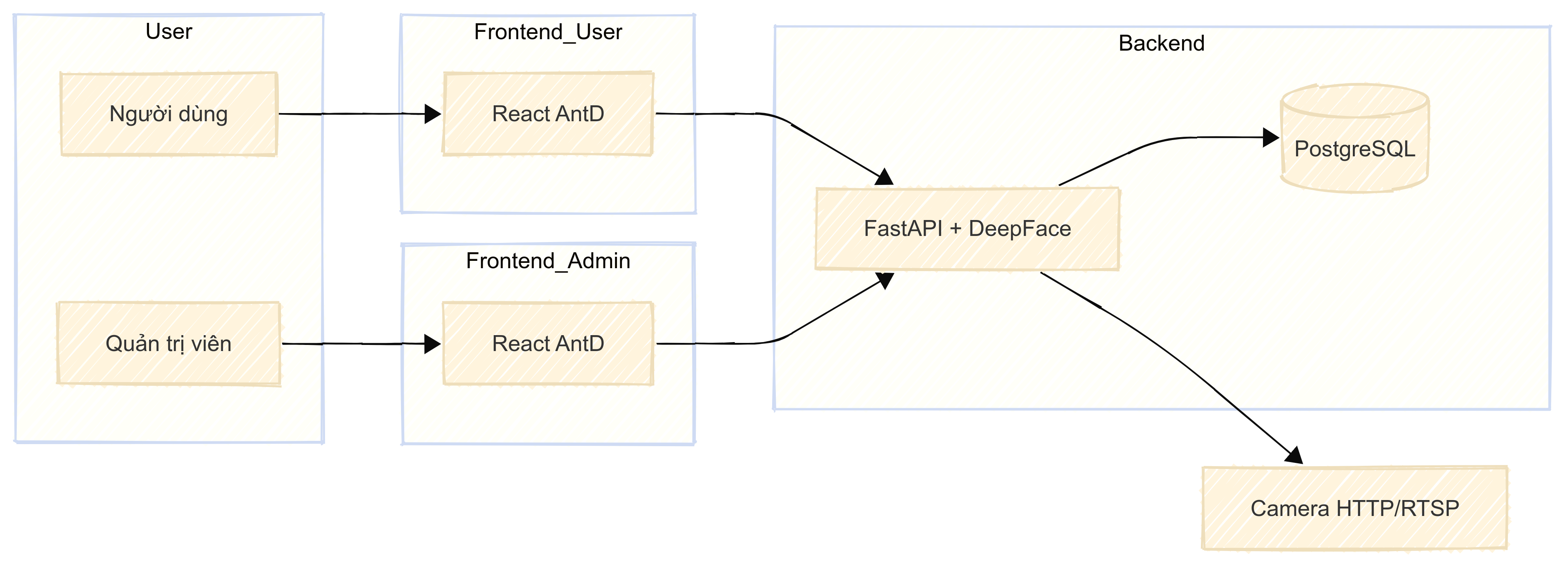
- Bảo mật API (tương lai có auth).

## C:\Users\ACER\Downloads\Untitled diagram _ Mermaid Chart-2025-08-15-092907.png4.3 Mô tả Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã UC** | **Tên Use Case** | **Mô tả** | **Tác nhân** |
| UC01 | Xem video trực tiếp | |  | | --- | |  |  |  | | --- | | Người dùng truy cập giao diện web để xem luồng video trực tiếp từ camera. Hệ thống kết nối tới camera qua giao thức RTSP/HTTP và hiển thị hình ảnh theo thời gian thực, hỗ trợ nhiều camera cùng lúc, người dùng chọn camera để theo dõi. | | User |
| UC02 | Xem lịch sử nhận diện | |  | | --- | |  |  |  | | --- | | Hiển thị danh sách các sự kiện nhận diện đã lưu trong cơ sở dữ liệu, bao gồm ảnh chụp khuôn mặt, ID người, thời gian, ID Camera. | | User |
| UC03 | Quản lý camera | |  | | --- | |  |  |  | | --- | | Quản trị viên thêm mới hoặc xóa camera trong hệ thống. Bao gồm thông tin tên camera, địa chỉ stream, vị trí. Việc thay đổi cập nhật ngay lập tức tới backend để kết nối hoặc ngắt kết nối camera. | | Admin |
| UC04 | Quản lý danh sách nhận diện | Quản trị viên thêm hoặc xóa người trong danh sách nhận diện. Khi thêm mới, hệ thống nhận ảnh khuôn mặt và lưu trữ dữ liệu đặc trưng để phục vụ nhận diện. Khi xóa, dữ liệu liên quan được gỡ bỏ khỏi hệ thống. | Admin |
| UC05 | Nhận diện thời gian thực | Hệ thống backend nhận luồng video từ camera, sử dụng mô hình DeepFace để phát hiện và nhận diện gương mặt. Nếu trùng khớp với danh sách đã lưu, sẽ gắn nhãn và gửi thông tin nhận diện cho frontend hiển thị ngay lập tức. | Camera |
| UC06 | Lưu trữ dữ liệu nhận diện | Sau khi nhận diện thành công, hệ thống lưu thông tin vào cơ sở dữ liệu, gồm ảnh khuôn mặt, tên, thời gian, ID camera. Dữ liệu sẽ phục vụ tra cứu sau này. | Camera |

# 5. Các lát cắt

## 5.1 Context View



## 5.2 Logical View

- models.py: Định nghĩa bảng Person, Camera, History.

- utils.py: Xử lý embedding, so sánh cosine.

- camera\_manager.py: Thread đọc camera, nhận diện, lưu snapshot & lịch sử.

- routers/: API REST cho people, cameras, history, stream.

## 5.3 Process / Sequence View

- Người quản trị thêm camera → Backend tạo thread đọc luồng → Mỗi frame → DeepFace trích xuất embedding → So khớp → Nếu match → Lưu snapshot & history.

## C:\Users\ACER\Downloads\Untitled diagram _ Mermaid Chart-2025-08-15-090503.png5.4 Deployment View

## 5.5 Data View

- Person: id, name, embedding, image\_path.

- Camera: id, name, url.

- History: id, person\_id, camera\_id, snapshot\_path, score, timestamp.

# 6. Các thành phần

- FastAPI App: main.py, đăng ký router, cấu hình CORS.

- Database Layer: database.py, models.py, schemas.py.

- Recognition Engine: utils.py + DeepFace.

- Camera Manager: camera\_manager.py.

- Routers: people.py, cameras.py, history.py, stream.py.

- Frontend: React (AntD), API calls.

- Docker Config: docker-compose.yml, Dockerfile.backend, Dockerfile.

# 7. Lý do các quyết định thiết kế

- FastAPI: Hiệu năng cao, hỗ trợ xử lý bất đồng bộ (async) giúp phục vụ nhiều kết nối camera và người dùng cùng lúc. Cú pháp hiện đại, dễ bảo trì và mở rộng.

- DeepFace: Tích hợp nhiều mô hình (Facenet, ArcFace, VGG-Face, …) cho phép linh hoạt lựa chọn. API đơn giản, dễ huấn luyện và triển khai.

- Chạy mỗi camera trong một thread: Giúp xử lý độc lập, tránh ảnh hưởng giữa các camera khi một luồng gặp sự cố. Dễ dàng quản lý kết nối và khởi động/tắt camera khi cần.

- Lưu embedding: Giảm tải tính toán khi so khớp gương mặt (không phải tái trích xuất đặc trưng). Tối ưu dung lượng lưu trữ so với việc giữ toàn bộ ảnh gốc..

- Dùng MJPEG stream: Giao thức đơn giản, dễ hiển thị trên web bằng <img> hoặc <video>. Không yêu cầu thiết lập phức tạp như WebRTC. Độ trễ thấp, phù hợp cho giám sát thời gian thực.

- PostgreSQL: phù hợp khi triển khai production với dữ liệu lớn nhiều kết nối đồng thời.

# 8. Cấu hình & Triển khai

- Docker Compose orchestrate: backend, frontend-user, frontend-admin, db.

- Mount volume cho uploads/ và db.

- Env: POSTGRES\_USER, POSTGRES\_PASSWORD, POSTGRES\_DB, POSTGRES\_HOST.

# 9. API chính

- POST /api/people/add

- GET /api/people

- DELETE /api/people/{id}

- POST /api/cameras

- GET /api/cameras

- DELETE /api/cameras/{id}

- GET /api/history

- GET /api/stream/{cam\_id}

# 10. Kiểm thử

- Unit test utils (embedding, so khớp).

- Test API CRUD.

- Test stream và nhận diện.

# 11. Kế hoạch phát triển tương lai

- Thêm auth, phân quyền.

- Tích hợp GPU, WebSocket.

- Cảnh báo qua email/Zalo khi nhận diện.