ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
Khoa Toán – Cơ – Tin học

BÁO CÁO THỰC TẬP

Đề tài:

Xây dựng hệ thống chấm công sử dụng phương pháp nhận diện khuôn mặt

Học phần: MAT3374 Thực tập thực tế phát triển phần mềm

Sinh viên thực hiện: Hoàng Anh Minh – 22001616

Lời cảm ơn

Trước hết, em xin bày tỏ lòng biết ơn chân thành tới Ban lãnh đạo Đài Phát thanh và Truyền hình Hà Nội, cùng Phòng Tổ chức đã tạo mọi điều kiện thuận lợi để em có cơ hội tham gia thực tập tại Trung tâm Kỹ thuật – Phòng Công nghệ thông tin.

Bên cạnh đó, em xin gửi lời cảm ơn sâu sắc đến cô Nguyễn Thị Bích Thuỷ và thầy Vũ Tiến Dũng – giảng viên phụ trách học phần thực tập – đã nhiệt tình hướng dẫn các thủ tục và yêu cầu trong suốt quá trình rèn luyện tại doanh nghiệp. Sự hỗ trợ và định hướng của thầy cô đã giúp em hoàn thành tốt quá trình thực tập cũng như chuẩn bị cho báo cáo này.

Trong kỳ thực tập tại Phòng CNTT, em là người duy nhất đảm nhận mảng phát triển phần mềm, trong khi các anh, chị trong phòng chủ yếu phụ trách về quản trị mạng và quản trị hệ thống server. Em được giao nhiệm vụ nghiên cứu và xây dựng **hệ thống chấm công bằng nhận diện khuôn mặt** để hướng tới ứng dụng trong công tác quản lý nhân sự của Đài trong tương lai. Trong quá trình đó, mặc dù không trực tiếp tham gia lập trình, nhưng các anh, chị đã luôn quan tâm, góp ý về giao diện, tính năng cũng như cách thức triển khai để hệ thống phù hợp với thực tế tại đơn vị. Đây là sự hỗ trợ vô cùng quý báu, giúp em vừa phát huy được kiến thức chuyên môn, vừa tiếp cận thêm những kinh nghiệm trong việc triển khai một dự án thực tế tại doanh nghiệp.

Thông qua kỳ thực tập này, em đã học hỏi được nhiều kinh nghiệm thực tế, từ đó rèn luyện và phát triển bản thân. Đây là động lực lớn để em tiếp tục nỗ lực, không ngừng nâng cao năng lực chuyên môn, nhằm vận dụng công nghệ thông tin vào giải quyết các bài toán thực tiễn của doanh nghiệp như quản lý nhân sự, tối ưu quy trình làm việc hay triển khai các hệ thống thông minh ứng dụng trí tuệ nhân tạo. Em tin rằng những trải nghiệm quý báu tại Phòng Công nghệ thông tin – Đài Hà Nội sẽ là nền tảng vững chắc để em tiếp tục phát triển và đóng góp nhiều hơn trong tương lai.

Mục lục

Danh mục hình ảnh

Danh mục bảng

Danh mục từ viết tắt

AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
GPU	Graphics Processing Unit
IoU	Intersection over Union
JSON	JavaScript Object Notation
ORDBMS	Object-Relational Database Management System
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
YOLO	You Only Look Once

Chương 1

Giới thiệu

1.1 Về dự án

Trong thời gian thực tập tại Phòng Công nghệ thông tin – Trung tâm Kỹ thuật, Đài Phát thanh và Truyền hình Hà Nội, em được giao nhiệm vụ nghiên cứu và phát triển hệ thống chấm công bằng nhận diện khuôn mặt nhằm hướng tới việc ứng dụng và triển khai tại Đài trong tương lai. Trong quá trình thực hiện, em luôn nhận được sự quan tâm, chỉ bảo và tạo điều kiện từ các anh, chị trong phòng, đặc biệt là những góp ý thiết thực về giao diện, tính năng của hệ thống.

Đây là một dự án có tính thực tiễn cao, xuất phát từ nhu cầu quản lý nhân sự và chấm công hiệu quả tại các cơ quan, doanh nghiệp. Mặc dù hệ thống hiện tại vẫn chưa đạt yêu cầu để áp dụng chính thức tại Đài, nhưng quá trình xây dựng và thử nghiệm đã giúp em trau dồi kỹ năng chuyên môn, vận dụng kiến thức đã học, tiếp cận thực tế công việc và học hỏi cách ứng dụng CNTT vào giải quyết các vấn đề quản lý trong doanh nghiệp.

1.1.1 Mục tiêu dự án

Dự án được triển khai với các mục tiêu cụ thể sau:

- **Độ chính xác và hiệu năng:** Xây dựng lõi AI có khả năng nhận diện khuôn mặt với độ chính xác cao, hoạt động ổn định và cho thời gian phản hồi nhanh.
- **Tính năng quản trị:** Phát triển một trang quản trị (Admin Dashboard) toàn diện, cho phép quản lý thông tin nhân viên, theo dõi trạng thái thiết bị Kiosk, và trích xuất báo cáo chấm công một cách trực quan, linh hoạt.
- **Trải nghiệm người dùng:** Thiết kế ứng dụng trên thiết bị đầu cuối (Kiosk App) với giao diện thân thiện, dễ sử dụng, đảm bảo quy trình chấm công diễn

ra nhanh chóng và liền mạch.

- **Khả năng triển khai và mở rộng:** Đóng gói toàn bộ hệ thống bằng công nghệ container hóa, đảm bảo việc cài đặt, vận hành và mở rộng hệ thống trong tương lai được thực hiện một cách dễ dàng và nhất quán.

1.1.2 Phạm vi công việc

Các công việc chính em đảm nhiệm bao gồm:

- Thiết kế và quản lý cơ sở dữ liệu (Database)
- Xây dựng hệ thống Backend và lõi AI
- Phát triển ứng dụng quản trị (Admin Dashboard)
- Phát triển ứng dụng chạy trên thiết bị ngoại vi (Kiosk App)
- Kiểm thử và đánh giá hệ thống.

Chi tiết các công việc và giải pháp kỹ thuật áp dụng sẽ được trình bày trong các phần tiếp theo.

1.2 Các công nghệ được sử dụng và lý do lựa chọn

1.2.1 Database

PostgreSQL

PostgreSQL (thường được gọi là Postgres) là một hệ quản trị cơ sở dữ liệu quan hệ đối tượng (ORDBMS) mã nguồn mở. Nó không chỉ hỗ trợ các tính năng SQL tiêu chuẩn mà còn cung cấp nhiều tính năng nâng cao để xử lý các loại dữ liệu phức tạp và các workload lớn, nhấn mạnh vào khả năng mở rộng và tuân thủ tiêu chuẩn kỹ thuật.

- **Xử lý dữ liệu đa dạng:** PostgreSQL hỗ trợ nhiều kiểu dữ liệu nâng cao như JSONB, dữ liệu hình học (PostGIS), mảng và cho phép định nghĩa kiểu dữ liệu riêng, rất phù hợp cho các hệ thống phức tạp.
- **Độ tin cậy cao:** Tuân thủ nghiêm ngặt tiêu chuẩn ACID, đảm bảo dữ liệu luôn chính xác và an toàn ngay cả khi hệ thống gặp sự cố.
- **Khả năng mở rộng:** Hỗ trợ index phức tạp, thủ tục lưu trữ và extension, cho phép tùy biến linh hoạt theo nhu cầu.
- **Xử lý đồng thời hiệu quả:** Sử dụng cơ chế MVCC, cho phép nhiều người

dùng truy cập cùng lúc mà vẫn duy trì hiệu năng cao.

Chính vì khả năng xử lý dữ liệu phức tạp và độ tin cậy cao, PostgreSQL đã được lựa chọn để lưu trữ các thông tin quan trọng như vector đặc trưng khuôn mặt (embedding vector) và lịch sử chấm công, đảm bảo tính toàn vẹn dữ liệu cốt lõi cho hệ thống.

1.2.2 Backend và AI/ML

FastAPI

FastAPI là một web framework hiện đại, hiệu suất cao của Python được thiết kế để xây dựng các API một cách nhanh chóng. Nó được xây dựng dựa trên Starlette (cho hiệu năng) và Pydantic (cho xác thực dữ liệu).

- **Hiệu năng cao:** Hỗ trợ lập trình bất đồng bộ (async), cho tốc độ xử lý ngang ngửa NodeJS và Go, phù hợp cho các hệ thống AI thời gian thực.
- **Tự động sinh tài liệu:** Hỗ trợ Swagger UI và ReDoc giúp kiểm thử và chia sẻ API dễ dàng.
- **Xác thực dữ liệu mạnh:** Tích hợp Pydantic đảm bảo dữ liệu vào/ra được kiểm soát chặt chẽ.
- **Phát triển nhanh:** Cú pháp hiện đại, dễ viết và giảm đáng kể thời gian triển khai.

Tốc độ xử lý vượt trội của FastAPI là yếu tố then chốt, cho phép backend phục vụ các mô hình AI và phản hồi yêu cầu chấm công từ nhiều thiết bị Kiosk.

YOLOv11s Detection

Đây là một mô hình thuộc họ YOLO (You Only Look Once), được huấn luyện để thực hiện tác vụ phát hiện khuôn mặt trong ảnh. Phiên bản 's' (small) được tối ưu để có kích thước nhỏ và tốc độ nhanh.

- **Thời gian thực:** Có khả năng phát hiện khuôn mặt cực nhanh, phù hợp xử lý video trực tiếp và các hệ thống yêu cầu phản hồi tức thì.
- **Nhẹ và dễ triển khai:** Phiên bản 's' có kích thước nhỏ, tiêu thụ ít tài nguyên, phù hợp cả trên GPU hạn chế hoặc trong môi trường container.
- **Độ chính xác ổn định:** Dù đánh đổi một phần độ chính xác để tối ưu tốc độ, YOLOv11s vẫn đạt kết quả tốt trong điều kiện thực tế.

Mô hình detection này đóng vai trò lớp đầu tiên trong pipeline AI, nhanh chóng

xác định vị trí khuôn mặt để xử lý tiếp theo.

YOLOv11s Classification

Đây là một mô hình phân loại (classifier - cls), sử dụng kiến trúc YOLO ('s' - small) để xác định xem một khuôn mặt đã được phát hiện là thật (live) hay là giả mạo (spoof), ví dụ như ảnh chụp từ màn hình, ảnh in trên giấy.

- **Tốc độ và hiệu quả:** Kiến trúc nhỏ gọn đảm bảo xử lý nhanh, không làm chậm toàn bộ hệ thống.
- **Nhất quán kiến trúc:** Dùng chung họ YOLO cho cả phát hiện và phân loại giúp đơn giản hóa triển khai và tối ưu tài nguyên.
- **Tăng cường bảo mật:** Ngăn chặn các hình thức tấn công đơn giản như ảnh in hoặc màn hình hiển thị.

Nhược điểm, mô hình này chủ yếu dựa trên phân tích hình ảnh 2D, do đó có thể bị đánh lừa bởi các kỹ thuật giả mạo phức tạp hơn như mặt nạ 3D hoặc video deepfake. Hiệu quả của mô hình phụ thuộc rất lớn vào sự đa dạng của các loại tấn công giả mạo có trong tập dữ liệu huấn luyện.

InsightFace buffalo-l

InsightFace là một bộ công cụ mã nguồn mở hàng đầu cho phân tích và nhận dạng khuôn mặt. buffalo-l là một trong những mô hình mạnh nhất của họ, dùng để chuyển đổi một hình ảnh khuôn mặt thành một vector đặc trưng (embedding) dạng số.

- **Độ chính xác hàng đầu:** buffalo-l đạt kết quả state-of-the-art trên nhiều bộ benchmark, có khả năng phân biệt cả những khuôn mặt rất giống nhau.
- **Kháng biến thể mạnh:** Giữ hiệu quả cao ngay cả khi thay đổi góc nhìn, ánh sáng, biểu cảm hoặc có vật thể che khuất.
- **Được tin dùng rộng rãi:** Sở hữu cộng đồng phát triển lớn, ứng dụng trong nghiên cứu lẫn sản phẩm thương mại, đảm bảo tính ổn định và độ tin cậy.

Nhờ đặc tính này, buffalo-l phù hợp để nhận diện nhân viên trong điều kiện thực tế phức tạp (ánh sáng yếu, góc mặt thay đổi).

Nhược điểm, buffalo-l ('l' - large) có kích lớn và yêu cầu nhiều sức mạnh tính toán hơn so với các phiên bản nhỏ hơn (buffalo-s). Cần có GPU để đạt được tốc độ tốt nhất.

SQLAlchemy

SQLAlchemy là một bộ công cụ SQL và là một Object Relational Mapper (ORM) cho Python. Nó cho phép các lập trình viên tương tác với cơ sở dữ liệu bằng các đối tượng Python thay vì viết các câu lệnh SQL thuần.

- **Linh hoạt và mạnh mẽ:** Cung cấp cả ORM cấp cao cho các thao tác đơn giản và Core Expression Language cấp thấp cho phép viết các truy vấn phức tạp gần như SQL thuần.
- **Hỗ trợ đa dạng cơ sở dữ liệu:** Có thể làm việc với nhiều hệ quản trị CSDL khác nhau (PostgreSQL, MySQL, SQLite,...) chỉ bằng cách thay đổi chuỗi kết nối.
- **Quản lý session và transaction hiệu quả:** Cung cấp một hệ thống quản lý session mạnh mẽ, đảm bảo tính toàn vẹn dữ liệu (ACID).
- **Tích hợp tốt với FastAPI:** FastAPI có tài liệu hướng dẫn và hỗ trợ rất tốt cho việc tích hợp với SQLAlchemy.

SQLAlchemy giúp chuẩn hóa và đơn giản hóa việc tương tác với cơ sở dữ liệu PostgreSQL từ backend FastAPI, tăng tốc độ phát triển và giảm thiểu lỗi.

1.2.3 Admin-Dashboard

React.js

React.js là một thư viện JavaScript mã nguồn mở do Facebook phát triển, được sử dụng để xây dựng giao diện người dùng (UI), đặc biệt là cho các ứng dụng một trang (Single điểm, React chỉ tập trung vào lớp giao diện (View layer). Để xây dựng một ứng dụng hoàn chỉnh, bạn phải tự lựa chọn và kết hợp nó với các thư viện khác để xử lý định tuyến (routing), quản lý trạng thái (state management), v.v. Điều này có thể gây khó khăn cho người mới bắt đầu.

Material-UI (MUI)

Material-UI (nay được gọi chính thức là MUI) là một bộ thư viện chứa các component React được xây dựng sẵn, tuân theo hệ thống thiết kế Material Design của Google. Nó cung cấp cho bạn mọi thứ từ nút bấm, form, cho đến các layout phức tạp.

- **Tăng tốc độ phát triển:** Cung cấp sẵn bộ component hiện đại, hoạt động tốt, giúp rút ngắn đáng kể thời gian xây dựng giao diện.
- **Thiết kế chuyên nghiệp và nhất quán:** Ứng dụng mặc định mang phong

cách Material Design, quen thuộc với người dùng, đồng thời tạo cảm giác hiện đại và đồng bộ.

- **Khả năng tùy biến mạnh mẽ:** Hệ thống chủ đề (theme) cho phép điều chỉnh màu sắc, phông chữ, kích thước... theo đặc thù thương hiệu của hệ thống.
- **Tài liệu phong phú:** Có hướng dẫn chi tiết, nhiều ví dụ minh họa, hỗ trợ lập trình viên triển khai nhanh và đúng chuẩn.

Sự kết hợp giữa React và MUI cho phép xây dựng một trang quản trị hiện đại, có tính tương tác cao và chuyên nghiệp một cách nhanh chóng, giúp người quản trị dễ dàng thao tác và giám sát hệ thống.

1.2.4 Kiosk-App

Flutter

Flutter là một bộ công cụ phát triển giao diện người dùng (UI toolkit) mã nguồn mở do Google tạo ra. Nó cho phép các nhà phát triển xây dựng các ứng dụng đẹp mắt, được biên dịch trực tiếp ra mã máy (natively compiled) cho thiết bị di động (iOS, Android), web, và máy tính (Windows, macOS, Linux) chỉ từ một cơ sở mã nguồn duy nhất. Ngôn ngữ lập trình được sử dụng là Dart.

- **Một code-base, đa nền tảng:** Viết một lần, triển khai trên cả Android, iOS và máy tính giúp tiết kiệm thời gian, chi phí và công sức so với việc phát triển riêng biệt từng nền tảng.
- **Phát triển nhanh với Hot Reload:** Tính năng "Hot Reload" cho phép hiển thị ngay lập tức các thay đổi trong code mà không cần khởi động lại ứng dụng, tăng tốc độ thử nghiệm và phát triển giao diện.
- **Giao diện đẹp và linh hoạt:** Flutter không phụ thuộc vào các thành phần gốc (native widgets) mà tự vẽ toàn bộ giao diện, đảm bảo tính đồng nhất, mượt mà và khả năng tùy biến cao trên mọi thiết bị.
- **Hiệu năng gần như native:** Do được biên dịch trực tiếp xuống mã máy ARM/x86, ứng dụng Flutter có hiệu năng cao, gần tương đương với các ứng dụng native.

Với mục tiêu có thể triển khai ứng dụng Kiosk trên nhiều loại thiết bị (máy tính bảng Android, iPad,...) trong tương lai mà không tốn công phát triển lại, Flutter và triết lý "một codebase" của nó là giải pháp hoàn hảo cho dự án.

Nhược điểm, một ứng dụng Flutter cơ bản thường có dung lượng lớn hơn so với ứng dụng native tương đương, do phải đóng gói cả engine của Flutter và bộ thư viện

đi kèm. Kho thư viện và công cụ hỗ trợ của Flutter vẫn chưa thể sánh bằng hệ sinh thái khổng lồ và lâu đời của Android (Java/Kotlin) và iOS (Swift/Objective-C).

1.2.5 Hạ tầng và triển khai

Docker

Docker là một nền tảng mã nguồn mở giúp tự động hóa việc triển khai, chạy và quản lý ứng dụng bằng cách sử dụng công nghệ containerization. Hiểu đơn giản, Docker "đóng gói" một ứng dụng cùng với tất cả các thứ nó cần để chạy (thư viện, tệp cấu hình, môi trường runtime) vào một đơn vị duy nhất gọi là container. Container này có thể chạy một cách nhất quán trên mọi máy tính, từ laptop của lập trình viên đến máy chủ trên cloud.

- **Môi trường nhất quán:** Giải quyết triệt để vấn đề *"It works on my machine"* bằng cách đóng gói ứng dụng cùng toàn bộ môi trường, đảm bảo chạy giống hệt nhau từ máy phát triển đến máy chủ sản phẩm.
- **Triển khai nhanh chóng:** Container khởi động trong vài giây, nhanh hơn nhiều so với máy ảo truyền thống. Việc triển khai trở nên đơn giản, chỉ cần `docker run`.
- **Hiệu quả tài nguyên:** Container chia sẻ kernel của hệ điều hành, nhẹ hơn và tốn ít CPU/RAM hơn so với máy ảo, cho phép chạy nhiều dịch vụ trên cùng một máy chủ.
- **Cô lập và an toàn:** Mỗi container hoạt động trong một môi trường riêng, hạn chế ảnh hưởng lẫn nhau, tăng tính ổn định và bảo mật.
- **Khả năng mở rộng:** Hỗ trợ dễ dàng nhân bản hoặc xóa container theo nhu cầu. Các công cụ điều phối như Docker Swarm hay Kubernetes cho phép tự động hóa việc mở rộng và quản lý hệ thống.

Trong hệ thống này, Docker được sử dụng để đóng gói backend (FastAPI), các mô hình AI (YOLOv11, InsightFace) và dịch vụ cơ sở dữ liệu, giúp triển khai nhanh chóng, dễ dàng mở rộng, đồng thời đảm bảo tính ổn định khi vận hành trên nhiều môi trường khác nhau. Nhược điểm, Docker không phải là máy ảo, nó không thể dùng để chạy một hệ điều hành hoàn toàn khác.

Chương 2

Công việc triển khai

2.1 Thiết kế hệ thống

2.1.1 Lựa chọn hướng tiếp cận

Trong các hệ thống ứng dụng AI, đặc biệt là xử lý ảnh và video, có hai hướng tiếp cận kiến trúc chính: Xử lý trên thiết bị (On-device) và Xử lý theo mô hình Client-Server. Việc lựa chọn hướng đi phù hợp là một quyết định nền tảng, ảnh hưởng đến hiệu năng, chi phí, khả năng bảo trì và mở rộng của toàn bộ hệ thống.

Phân tích hướng tiếp cận On-device

Đây là phương pháp mà toàn bộ quá trình xử lý AI (phát hiện khuôn mặt, chống giả mạo, trích xuất và so khớp embedding) đều được thực hiện trực tiếp trên thiết bị đầu cuối (Kiosk App).

- **Ưu điểm:**
  - **Tốc độ phản hồi cực nhanh:** Do không có độ trễ mạng, kết quả chấm công gần như tức thì.
  - **Hoạt động ngoại tuyến:** Hệ thống có thể tiếp tục chấm công ngay cả khi mất kết nối mạng, dữ liệu sẽ được đồng bộ sau.
  - **Bảo mật dữ liệu:** Dữ liệu hình ảnh khuôn mặt không cần phải gửi ra khỏi thiết bị, tăng cường quyền riêng tư.
- **Nhược điểm:**
  - **Yêu cầu phần cứng mạnh:** Thiết bị Kiosk phải có cấu hình đủ mạnh (CPU, GPU, RAM) để chạy các mô hình AI.
  - **Khó khăn trong bảo trì và cập nhật:** Mỗi khi có phiên bản mô hình AI mới hoặc cần cập nhật danh sách nhân viên, phải triển khai cập nhật phần mềm trên TẤT CẢ các thiết bị.

- **Hạn chế về độ chính xác:** Thường phải sử dụng các mô hình AI phiên bản nhỏ (lightweight) để đảm bảo hiệu năng, có thể làm giảm độ chính xác.

Phân tích hướng tiếp cận Client-Server

Đây là phương pháp mà thiết bị Kiosk (Client) chỉ đóng vai trò thu thập dữ liệu (chụp ảnh) và gửi về một máy chủ trung tâm (Server) để thực hiện toàn bộ quá trình xử lý AI. Đây chính là hướng tiếp cận đã được lựa chọn trong dự án.

- **Ưu điểm:**
  - **Quản lý tập trung:** Toàn bộ dữ liệu nhân viên, embedding khuôn mặt, và lịch sử chấm công được quản lý tại một nơi duy nhất. Việc cập nhật mô hình AI hay logic nghiệp vụ chỉ cần thực hiện một lần trên server.
  - **Sử dụng mô hình AI mạnh mẽ:** Server có thể được trang bị GPU chuyên dụng, cho phép triển khai các mô hình AI lớn và có độ chính xác cao nhất (như InsightFace buffalo-l) mà không bị giới hạn bởi cấu hình thiết bị đầu cuối.
  - **Chi phí thiết bị thấp:** Các máy Kiosk không cần cấu hình mạnh, có thể là các máy tính bảng thông thường, giúp giảm đáng kể chi phí phần cứng.
  - **Dễ dàng mở rộng:** Việc thêm một điểm chấm công mới chỉ đơn giản là triển khai thêm một Kiosk App và kết nối nó tới server.
- **Nhược điểm:**
  - **Phụ thuộc vào mạng:** Yêu cầu phải có kết nối mạng ổn định giữa Kiosk và Server để hệ thống hoạt động.
  - **Độ trễ mạng:** Luôn có một khoảng trễ nhất định do quá trình gửi và nhận dữ liệu qua mạng.

Biện luận và Quyết định

Sau khi cân nhắc các ưu và nhược điểm, Client-Server đã được lựa chọn cho dự án vì những lý do sau:

- **Ưu tiên độ chính xác và khả năng quản lý:** Yêu cầu của bài toán là cần một hệ thống có độ chính xác cao và dễ dàng quản lý tập trung toàn bộ dữ liệu. Client-Server cho phép sử dụng các mô hình AI mạnh mẽ và quản trị toàn bộ hệ thống thông qua Admin Dashboard một cách hiệu quả.
- **Tối ưu hóa chi phí và bảo trì:** Việc sử dụng các thiết bị Kiosk gọn nhẹ giúp giảm chi phí đầu tư ban đầu. Quan trọng, khả năng cập nhật và bảo trì hệ thống từ một máy chủ giúp tiết kiệm thời gian và công sức trong dài hạn.

- **Phù hợp với môi trường doanh nghiệp:** Nhược điểm về sự phụ thuộc vào mạng có thể chấp nhận được, vì hầu hết doanh nghiệp đều có sẵn hạ tầng mạng nội bộ ổn định.

2.1.2 Xây dựng kiến trúc tổng thể

Hệ thống được thiết kế theo kiến trúc phân tán Multi-Component với Container Orchestration. Kiến trúc này chia hệ thống thành các khối chức năng độc lập nhưng có khả năng tương tác chặt chẽ với nhau, bao gồm:

- **Backend:** "Bộ não" trung tâm xử lý logic và AI.
- **Admin-Dashboard :** Giao diện web quản trị.
- **Kiosk-App:** Ứng dụng chấm công trên thiết bị đầu cuối.
- **Database:** Hệ quản trị cơ sở dữ liệu.
- **Docker:** Container hóa và triển khai

Kiến trúc này mang lại sự linh hoạt cao, cho phép phát triển, nâng cấp hoặc thay thế từng thành phần mà không ảnh hưởng đến toàn bộ hệ thống. Đồng thời, nó cũng tạo điều kiện thuận lợi cho việc mở rộng quy mô khi cần thiết.

2.1.3 Lựa chọn công nghệ

Việc lựa chọn công nghệ được cân nhắc kỹ lưỡng dựa trên hiệu năng, hệ sinh thái hỗ trợ và sự phù hợp với từng thành phần:

- **Backend:** FastAPI (Python) được chọn nhờ hiệu năng vượt trội, khả năng xử lý bất đồng bộ và tính năng tự động sinh tài liệu API, rất phù hợp cho một hệ thống cần xử lý nhiều yêu cầu đồng thời từ các Kiosk.
- **Admin-Dashboard:** React.js được sử dụng để xây dựng một giao diện người dùng hiện đại, có tính tương tác cao. Kiến trúc dựa trên component của React giúp việc quản lý và tái sử dụng mã nguồn trở nên hiệu quả.
- **Kiosk-App:** Flutter là lựa chọn tối ưu cho việc phát triển ứng dụng đa nền tảng, cho phép chỉ với một codebase duy nhất có thể triển khai ứng dụng trên nhiều loại thiết bị khác nhau, giúp tiết kiệm thời gian và chi phí.
- **Database:** PostgreSQL phù hợp vì: mạnh mẽ, hỗ trợ dữ liệu JSON, mở rộng tốt.
- **Triển khai:** Docker và Docker Compose được chọn làm nền tảng triển khai để đóng gói toàn bộ hệ thống, đảm bảo tính nhất quán giữa môi trường phát triển và sản phẩm, đồng thời đơn giản hóa tối đa quá trình cài đặt.

2.1.4 Thiết kế cơ sở dữ liệu

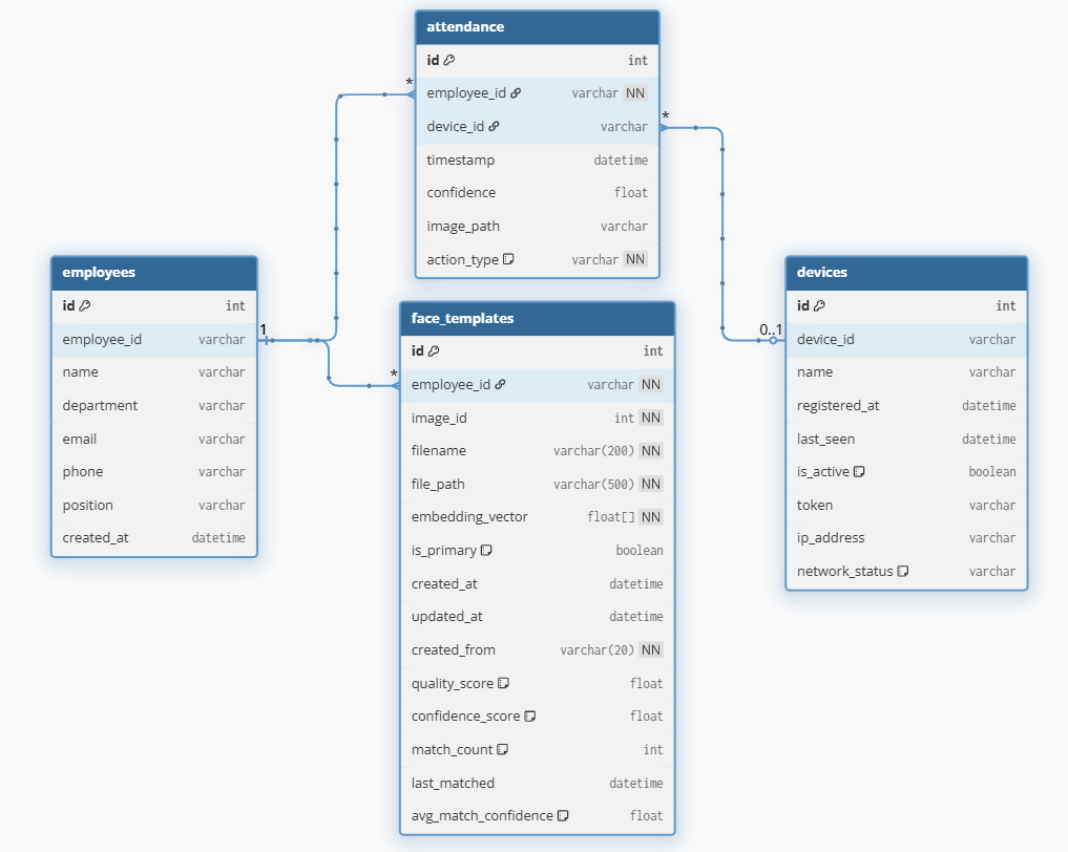
Cấu trúc cơ sở dữ liệu được thiết kế để lưu trữ thông tin một cách có tổ chức, bao gồm các bảng chính:

- **Employees:** Lưu thông tin nhân viên (employee-id, name, email, phone, department, position, created-at).

- **Face-Templates:** Lưu trữ các vector đặc trưng khuôn mặt đã được mã hóa của từng nhân viên (employee-id, filename, file-path, embedding-vector, confidence-score, quality-score, created-at, last-matched).

- **Devices:** Quản lý thông tin và trạng thái của các máy Kiosk (device-id, name, ip-address, network-status, registered-at, last-seen, is-active, token).

- **Attendance:** Ghi lại lịch sử các lần chấm công (employee-id, device-id, timestamp, confidence, image-path, action-type).



Hình 2.1: Sơ đồ quan hệ giữa các bảng trong Database

2.2 Phát triển và tích hợp

2.2.1 Huấn luyện mô hình AI

Huấn luyện YOLO11s cho phát hiện khuôn mặt

- Dữ liệu huấn luyện và kiểm định

  - **Tập huấn luyện (Train set):** Gồm 13,386 ảnh kèm nhãn (bounding box) đã được chuẩn hóa.

  - **Tập kiểm định (Validation set):** Gồm 3,347 ảnh kèm nhãn (bounding box) đã được chuẩn hóa, dùng để theo dõi quá trình huấn luyện.

  - **Tập kiểm tra (Test set):** Sau khi huấn luyện, mô hình được đánh giá trên tập gồm 3,226 ảnh kèm nhãn chuẩn.

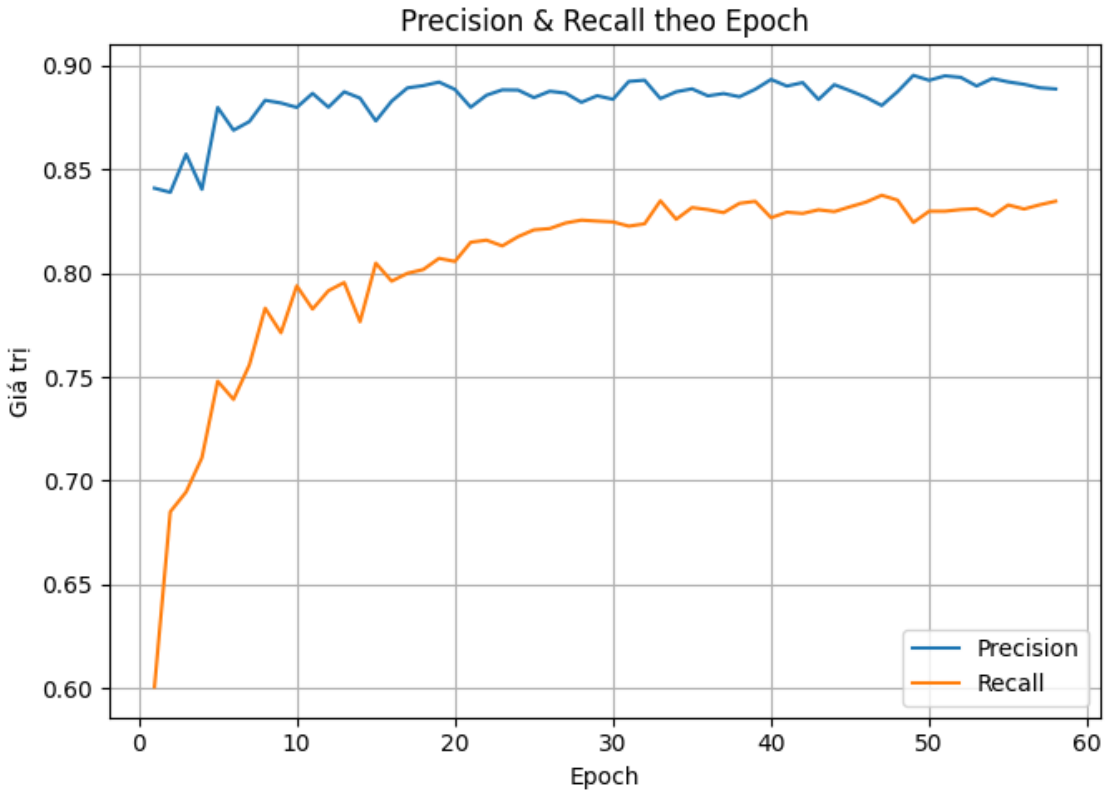
- Quá trình huấn luyện

  - Mô hình được huấn luyện bằng YOLOv11s (phiên bản nhỏ, ưu tiên tốc độ và gọn nhẹ), với 70 epoch và batch size = 64.

  - Dữ liệu huấn luyện gồm 13,386 ảnh, trong khi tập validation gồm 3,347 ảnh và được sử dụng để đánh giá sau mỗi epoch.

  - Các chỉ số theo dõi trong quá trình huấn luyện bao gồm: Training loss, Validation loss, Precision (P), Recall (R), mAP50 và mAP50-95.

- Kết quả huấn luyện

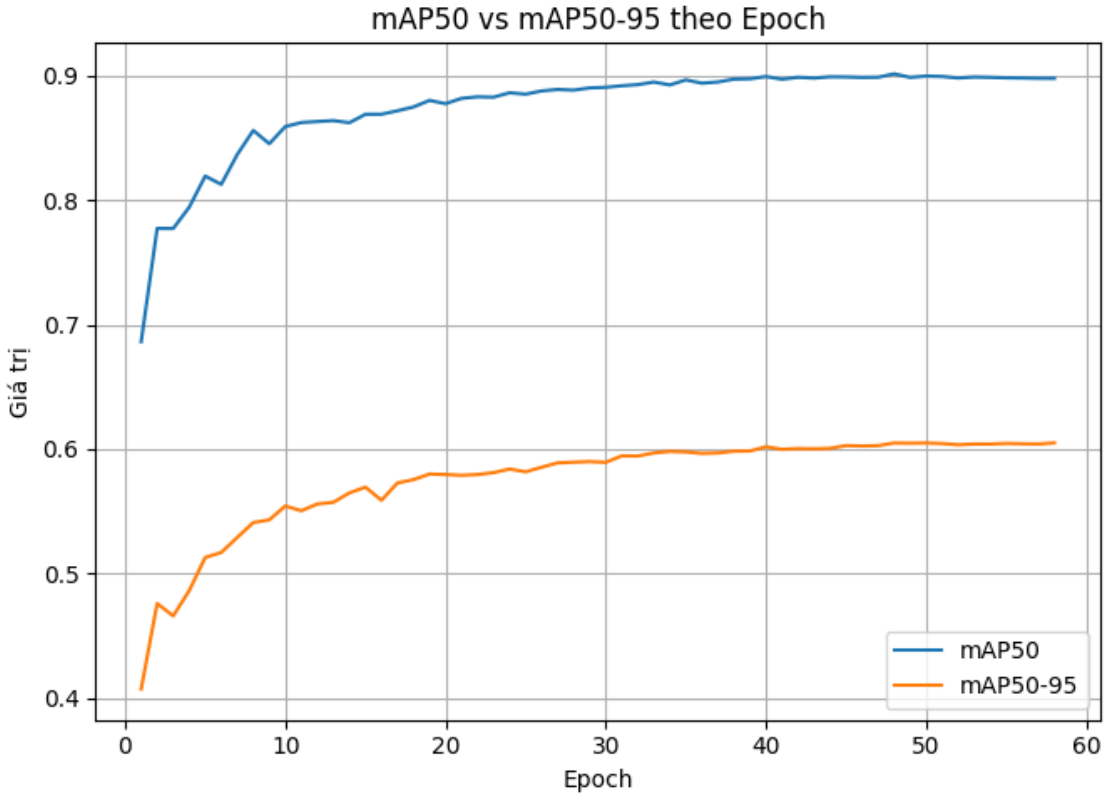


Hình 2.2: Biểu đồ Precision và Recall

– Precision đạt trung bình 0.88, chứng tỏ mô hình hiếm khi dự đoán sai một đối tượng không phải khuôn mặt là khuôn mặt (giảm false positives).

– Recall đạt trung bình 0.83, nghĩa là mô hình bắt được hầu hết các khuôn mặt có trong ảnh, chỉ bỏ sót một tỷ lệ nhỏ (false negatives).

– Với đặc thù ứng dụng nhận diện khuôn mặt, Recall cao là rất quan trọng để không bỏ sót khuôn mặt cần phát hiện, và kết quả hiện tại cho thấy mô hình đủ tin cậy để triển khai thực tế.



Hình 2.3: Biểu đồ các chỉ số mAP

– mAP50 đạt 0.9, chứng tỏ mô hình phát hiện khuôn mặt rất tốt khi yêu cầu độ khớp hộp (IoU) ở mức cơ bản (0.5).

– mAP50-95 đạt 0.6, thấp hơn đáng kể so với mAP50, phản ánh rằng hộp dự đoán vẫn còn sai lệch nhẹ so với vị trí thật của khuôn mặt, nhất là ở các ngưỡng IoU cao.

– Trong thực tế, với bài toán nhận diện khuôn mặt (đầu vào cho bước embedding/recognition sau này), việc phát hiện đúng vị trí khuôn mặt là đủ quan trọng, nên mAP50 cao đã là một kết quả khả quan. Tuy nhiên, để cải thiện độ chính xác hộp, có thể cân nhắc huấn luyện thêm hoặc tăng cường dữ liệu.

Hình 2.4: Biểu đồ Total Loss

- Cả train loss và val loss đều giảm dần, hội tụ ở mức 2.7 và không có dấu hiệu overfitting.
- Việc dừng sớm tại epoch 58 nhờ early stopping cho thấy mô hình đã hội tụ sớm và không cần thiết phải huấn luyện thêm đến 70 epoch.
- Đây là dấu hiệu tích cực, đảm bảo mô hình tổng quát hóa tốt trên dữ liệu mới.

Huấn luyện YOLOv11s-cls cho phân loại ảnh thật – giả

- Dữ liệu huấn luyện và kiểm định
  - **Tập huấn luyện (Train set):** 4,832 ảnh nhãn "real" và 6,929 ảnh nhãn "fake", được sử dụng để huấn luyện mô hình.
  - **Tập kiểm định (Validation set):** 839 ảnh nhãn "real" và 1432 ảnh nhãn "fake", dùng để theo dõi và điều chỉnh trong quá trình huấn luyện.
  - **Tập kiểm tra (Test set):** 119 ảnh nhãn "real" và 304 ảnh nhãn "fake", được sử dụng để đánh giá cuối cùng sau huấn luyện.
- Quá trình huấn luyện
  - Mô hình được huấn luyện bằng YOLOv11s-cls (phiên bản nhỏ, ưu tiên tốc

độ và gọn nhẹ), với 3 epoch và batch size = 32.

- Dữ liệu huấn luyện gồm 11761 ảnh, trong khi tập validation gồm 2271 ảnh và được sử dụng để đánh giá sau mỗi epoch.
- Các chỉ số theo dõi trong quá trình huấn luyện bao gồm: Training loss và Top1-acc.

- Kết quả huấn luyện
  - Mô hình YOLOv11s-cls cho tác vụ phân loại ảnh thật – giả đã nhanh chóng hội tụ chỉ sau 3 epoch. Giá trị loss giảm mạnh từ 0.04 ở epoch 1 xuống còn 0.0024 ở epoch 3. Độ chính xác top1-acc đạt 98.9% ngay từ epoch 2 và đạt 99.7% khi kết thúc huấn luyện, cho thấy sự khác biệt giữa hai lớp dữ liệu "real" và "fake" là tương đối rõ ràng, mô hình dễ dàng phân biệt.
  - Tuy nhiên, kết quả đạt mức gần như hoàn hảo trên tập huấn luyện và kiểm định có thể phản ánh hiện tượng dữ liệu chưa đủ đa dạng. Cần tiếp tục đánh giá trên tập kiểm tra độc lập và mở rộng thêm dữ liệu giả mạo với nhiều kịch bản khác nhau (in ảnh, quay màn hình, deepfake, v.v.) để đảm bảo tính tổng quát và độ tin cậy của mô hình trong môi trường thực tế.

2.2.2 Phát triển Backend

Xây dựng và phát triển API

Hệ thống backend được tổ chức thành 5 Router (bộ điều tuyến) chính với hơn 40 endpoints, bao phủ toàn bộ các chức năng nghiệp vụ:

- **Employee Router:** Quản lý thông tin nhân viên, bao gồm các API để tạo, sửa, xóa, lấy danh sách, và đặc biệt là xử lý ảnh để tạo face template cho AI.
- **Device Router:** Quản lý thiết bị kiosk, bao gồm đăng ký, cập nhật tín hiệu sống (heartbeat), và cơ chế xóa mềm (soft delete) nhằm bảo toàn dữ liệu chấm công.
- **Attendance Router:** Xử lý nghiệp vụ chấm công, nổi bật với endpoint `/check` – trái tim của hệ thống. API này đảm nhận nhận diện khuôn mặt, xác thực chống giả mạo, ghi nhận kết quả, và sử dụng cơ chế lock để tránh xung đột khi nhiều thiết bị cùng gửi yêu cầu.
- **Recognition Router:** Cung cấp các API chuyên biệt để tương tác trực tiếp với pipeline AI, cho phép kiểm tra và xác minh khuôn mặt độc lập.
- **Device Management Router:** Dành cho quản trị viên, cung cấp dashboard tổng quan về trạng thái hệ thống, thiết bị hoạt động và tình trạng các mô hình AI.

Việc tổ chức hệ thống API thành các Router riêng biệt giúp đảm bảo tính mô-đun, dễ dàng mở rộng, bảo trì và tích hợp thêm chức năng mới trong tương lai.

Phát triển tầng Service

Để tách biệt logic nghiệp vụ khỏi tầng API, hệ thống được xây dựng với nhiều service chuyên biệt:

- **Real AI Service:** Tích hợp pipeline AI 3 bước:
  1. **Anti-Spoofing:** Chống giả mạo bằng ảnh hoặc video.
  2. **Face Detection:** Phát hiện khuôn mặt trong ảnh.
  3. **Face Recognition:** Trích xuất vector đặc trưng 512 chiều.

- **Enhanced Recognition Service:** Kết hợp AI Service và Template Manager, thực hiện nhận diện nâng cao bằng cách so sánh embedding khuôn mặt với cơ sở dữ liệu sử dụng công thức Cosine Similarity và áp dụng ngưỡng để phân loại.

$$\text{similarity}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \, \|\mathbf{v}\|}$$

- **Employee Service:** Đảm nhận toàn bộ nghiệp vụ liên quan đến nhân viên, bao gồm CRUD và kết nối trực tiếp với AI để quản lý dữ liệu khuôn mặt.

Tầng Service đóng vai trò trung gian quan trọng, đảm bảo tính tách biệt và tái sử dụng.

Triển khai và vận hành

- **Containerization:** Toàn bộ backend và cơ sở dữ liệu PostgreSQL được đóng gói bằng Docker và quản lý qua `docker-compose.yml`.

- **Cấu hình môi trường:** Sử dụng Pydantic Settings kết hợp file `.env` để dễ dàng quản lý tham số triển khai.

- **Giám sát hệ thống:** Endpoint `/health` được xây dựng để kiểm tra trạng thái kết nối với database và phát hiện sớm sự cố.

Triển khai bằng Docker giúp hệ thống ổn định, dễ di chuyển và đảm bảo tính nhất quán giữa các môi trường. Cơ chế giám sát chủ động giúp nâng cao độ tin cậy trong vận hành.

2.2.3 Phát triển Admin Dashboard

Xây dựng giao diện (UI) và trải nghiệm người dùng (UX)

Giao diện Admin Dashboard được thiết kế theo tiêu chí **Mobile-first** và trực quan:

- **Thiết kế Responsive:** Sử dụng Grid của Material-UI để đảm bảo hiển thị tốt trên nhiều kích thước màn hình.

- **Phản hồi trực quan:**

  - Loading States bằng `CircularProgress` và `LinearProgress`.

  - Thông báo thành công/lỗi bằng `Alert` với mã màu rõ ràng.

  - Hộp thoại xác nhận để ngăn chặn thao tác nguy hiểm.

- **Điều hướng thông minh:** Menu và AppBar được thiết kế đồng bộ, tự động thu gọn và làm nổi bật trang đang hoạt động.

Thiết kế ưu tiên trải nghiệm người dùng, vừa thân thiện vừa giảm thiểu sai sót trong quá trình thao tác.

Phát triển các Module chức năng chính

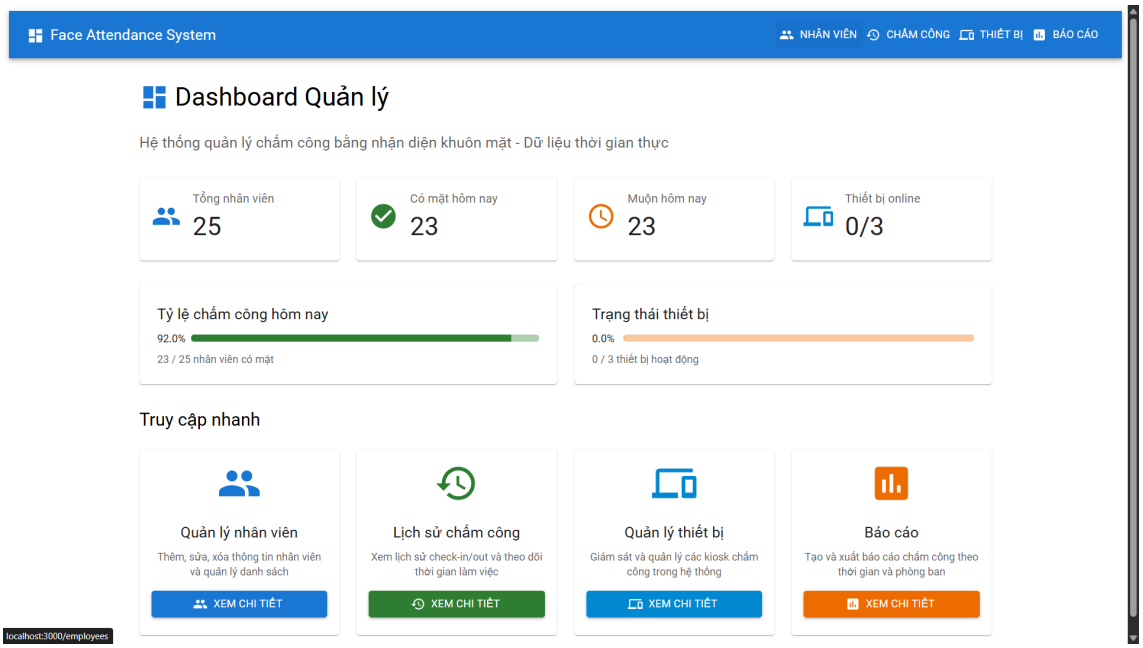
Hệ thống được phát triển với nhiều module quản lý và giám sát, đảm bảo tính toàn diện trong vận hành. Các module chính bao gồm:

- **Dashboard (Trang tổng quan)**

  - Hiển thị các chỉ số theo thời gian thực: số nhân viên, số người có mặt, trạng thái thiết bị.

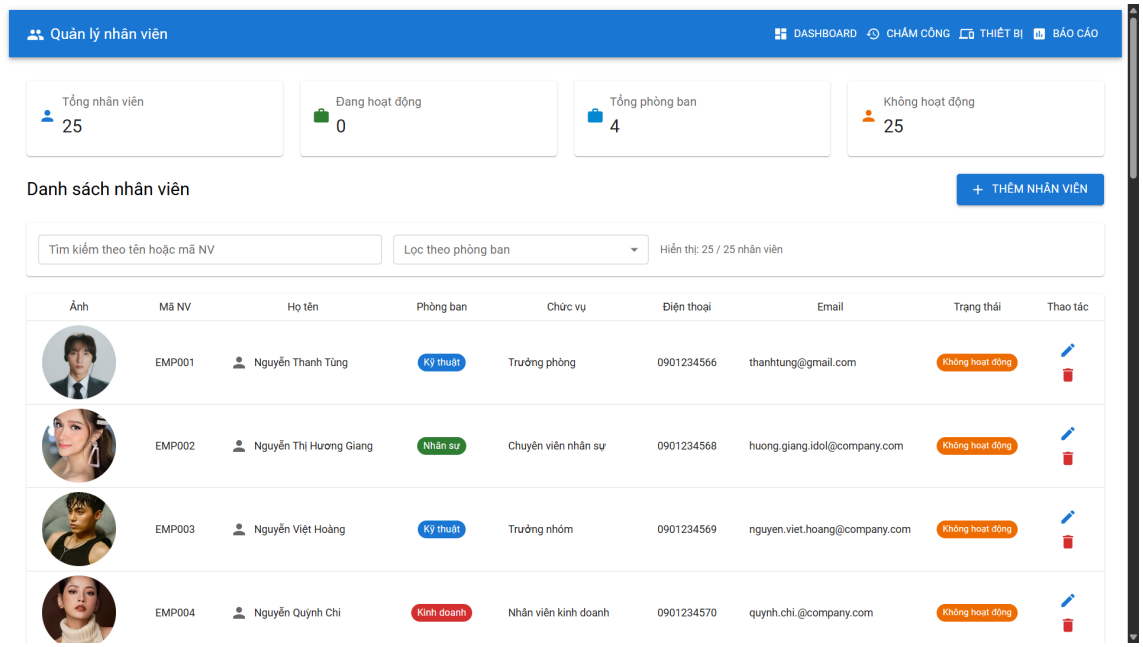
  - Tự động làm mới dữ liệu để đảm bảo thông tin luôn cập nhật.

  - Trực quan hóa tỷ lệ chuyên cần và trạng thái thiết bị bằng progress bar.



Hình 2.5: Giao diện chính (Dashboard)

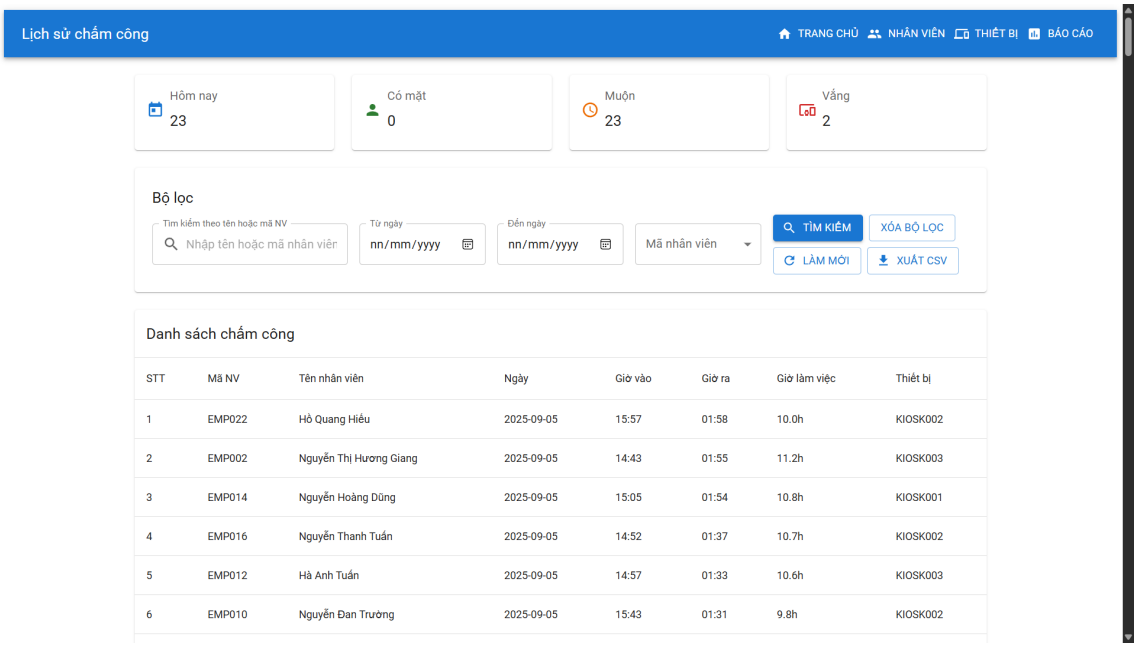
- **Quản lý nhân viên**
  - Cung cấp đầy đủ chức năng CRUD (Tạo, Đọc, Cập nhật, Xóa).
  - Hỗ trợ tải nhiều ảnh đại diện, có cơ chế chống cache đảm bảo ảnh luôn được cập nhật.
  - Tích hợp tìm kiếm và lọc đa tiêu chí (theo tên, phòng ban, trạng thái).



Hình 2.6: Giao diện Quản lý nhân viên

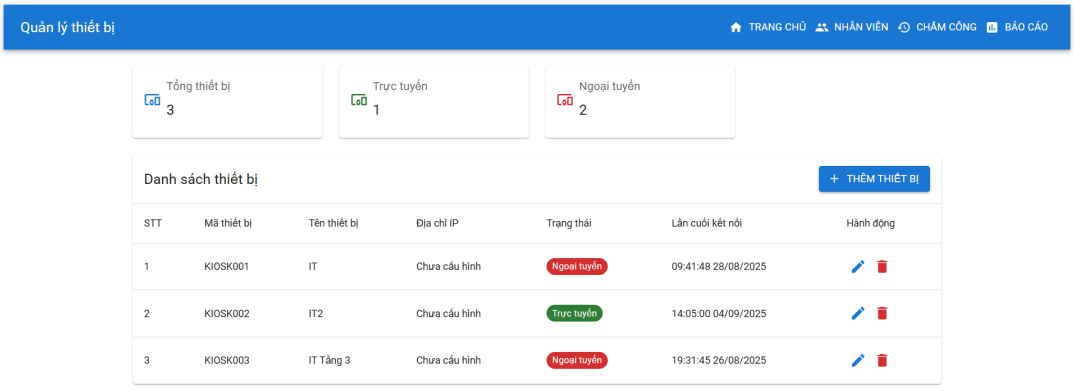
- **Lịch sử chấm công**

– Xây dựng pipeline xử lý dữ liệu để nhóm bản ghi theo ngày và nhân viên.
– Tự động xác định thời gian check-in đầu tiên và check-out cuối cùng để tính tổng giờ làm việc.



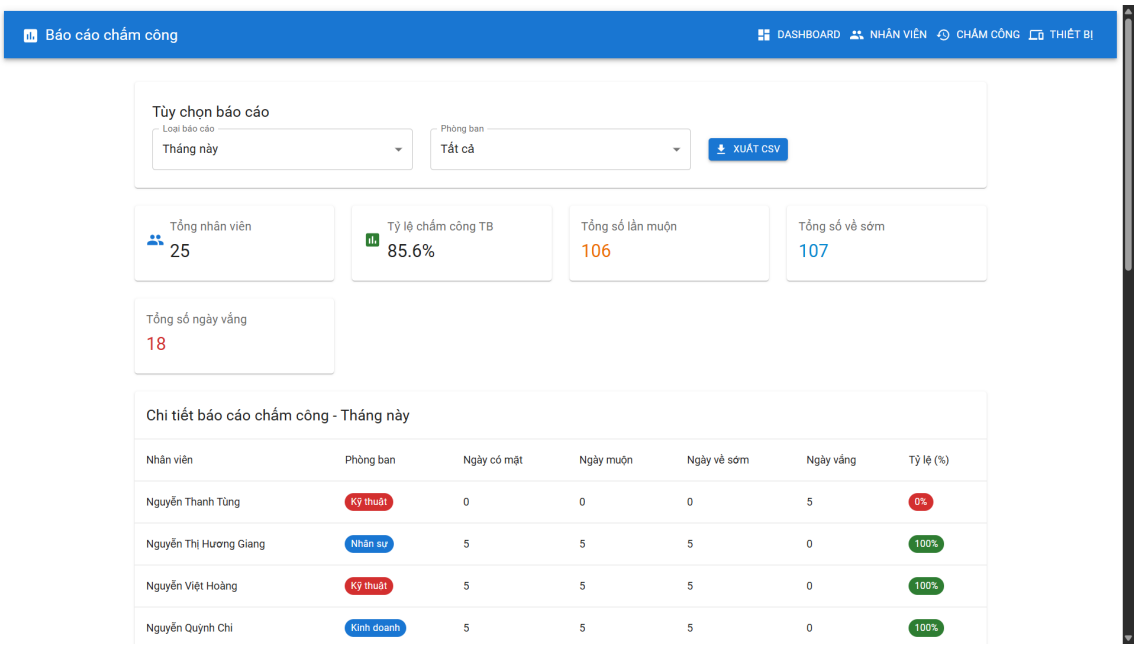
Hình 2.7: Giao diện Lịch sử chấm công

- **Quản lý thiết bị Kiosk**
  – Tự động tạo mã thiết bị thông minh (ví dụ: KIOSK001, KIOSK002).
  – Giám sát trạng thái online/offline theo thời gian thực, hiển thị "last seen" với cơ chế làm mới mỗi 2 phút.



Hình 2.8: Giao diện Quản lý thiết bị

- **Báo cáo thống kê**
  - Hỗ trợ tạo báo cáo động theo khoảng thời gian (ngày, tuần, tháng).
  - Xuất báo cáo sang định dạng Excel, PDF để lưu trữ và phân tích.



Hình 2.9: Giao diện Báo cáo thống kê

Sự kết hợp giữa **Backend mạnh mẽ** (API, Service, Docker) và **Admin Dashboard trực quan** (UI/UX hiện đại, dữ liệu thời gian thực) đã tạo nên một hệ thống nhận diện và chấm công hoàn chỉnh, vừa đảm bảo tính kỹ thuật, vừa thân thiện với người dùng cuối.

2.2.4 Phát triển Kiosk

Xây dựng giao diện (UI) và trải nghiệm người dùng (UX)

Ứng dụng Kiosk được thiết kế nhằm mang lại trải nghiệm liền mạch, đơn giản và chuyên nghiệp cho người dùng cuối. Các yếu tố chính bao gồm:

- **Chế độ toàn màn hình (Immersive Full-Screen):** Ứng dụng chạy ở chế độ toàn màn hình, ẩn thanh trạng thái và thanh điều hướng để đảm bảo người dùng không thể thoát ra ngoài ngoài ý muốn.
- **Giao diện ngang (Landscape Mode):** Bố cục tối ưu hóa cho màn hình ngang với tỉ lệ 60/40 (60% dành cho camera, 40% cho khu vực điều khiển), phù hợp với tablet tại quầy lễ tân.
- **Hệ thống phản hồi và hoạt ảnh:**

- **Phản hồi trực quan:** Viền camera và biểu tượng thay đổi màu sắc theo trạng thái (sẵn sàng, xử lý, thành công, lỗi).

- **Hoạt ảnh mượt mà:** Nút bấm và màn hình kết quả sử dụng hoạt ảnh (scale, slide, fade) để tạo cảm giác chuyên nghiệp.

- **Tự động reset:** Sau khi chấm công, màn hình kết quả hiển thị 5 giây kèm đồng hồ đếm ngược trước khi trở về trạng thái sẵn sàng.

Phát triển các Service chức năng cốt lõi

Để đảm bảo hiệu năng và khả năng mở rộng, hệ thống Kiosk được chia thành nhiều service độc lập:

- **Enhanced Camera Service:** Tự động lựa chọn camera trước, chụp ảnh ở độ phân giải cao, tắt thu âm để bảo vệ quyền riêng tư, và quản lý vòng đời camera hiệu quả (tự động giải phóng và khởi tạo lại tài nguyên).

- **API Service:** Quản lý việc gửi ảnh chấm công bằng Multipart Upload, đồng thời phân tích phản hồi từ backend để trả về thông tin nhân viên đầy đủ.

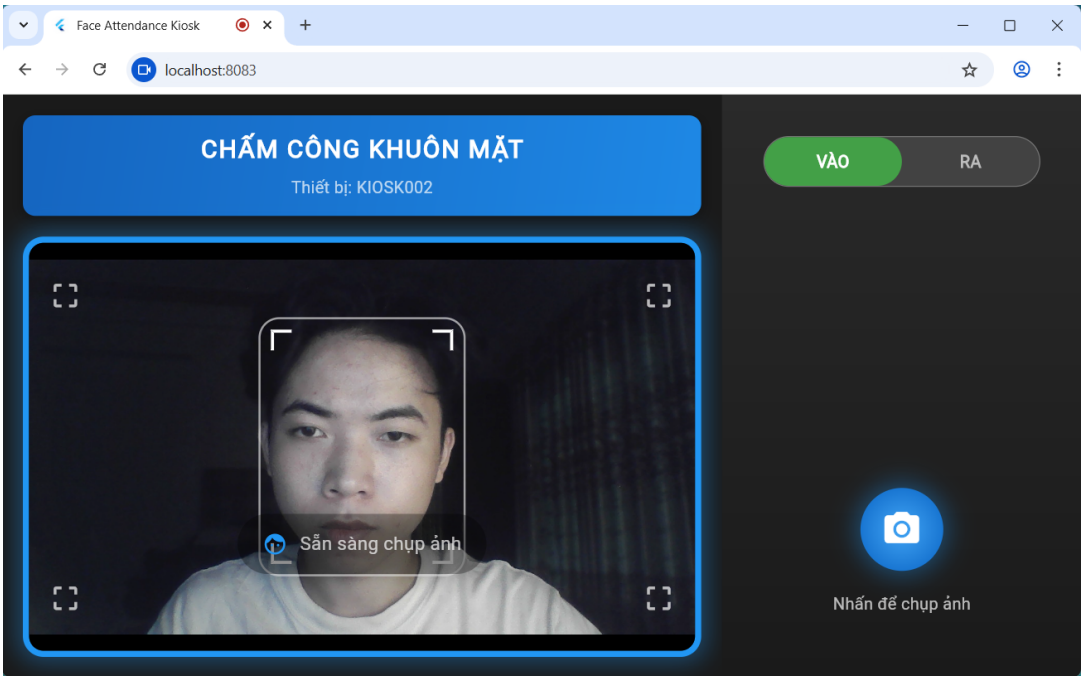
- **Discovery Service:** Xây dựng pipeline tìm kiếm server đa tầng, ưu tiên sử dụng địa chỉ IP đã lưu trong cache và xác thực kết nối bằng endpoint `/health`.

- **Device Configuration Service:** Quản lý định danh thiết bị (Device ID) trên nhiều nền tảng: đọc từ URL (Web), biến môi trường (Mobile), hoặc tự động sinh ID duy nhất. Đồng thời, duy trì trạng thái hoạt động bằng cơ chế "last seen" gửi về backend.

Giao diện và trạng thái hoạt động

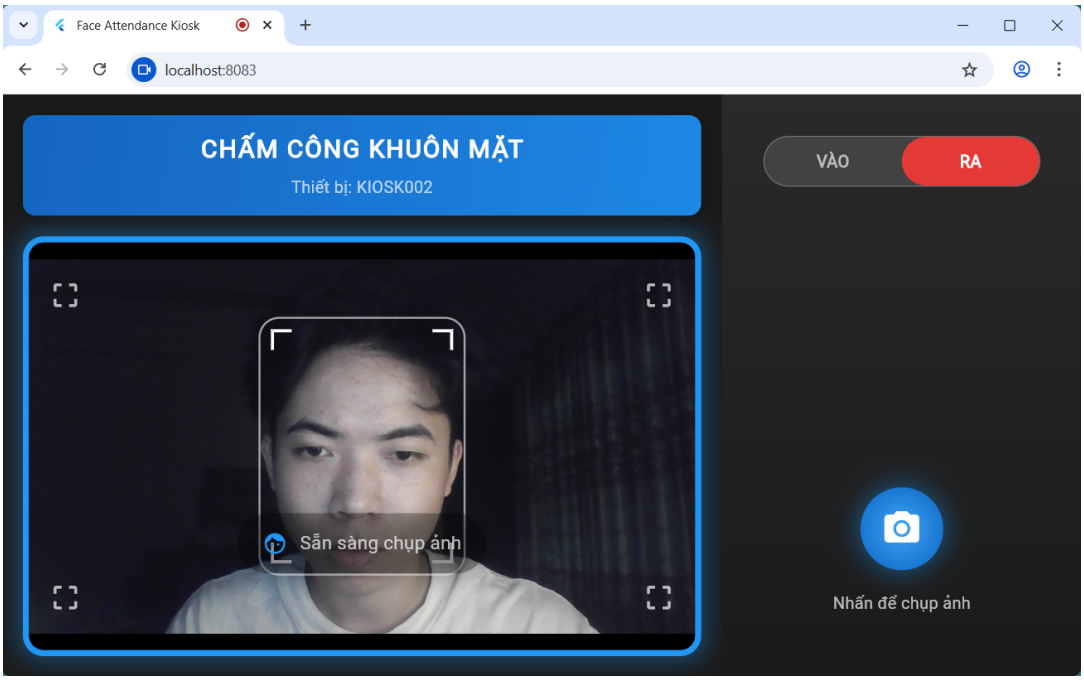
Ứng dụng Kiosk hỗ trợ nhiều trạng thái hiển thị khác nhau, giúp người dùng dễ dàng nhận biết kết quả chấm công cũng như tình trạng hệ thống. Mỗi trạng thái được thiết kế với màu sắc, biểu tượng và thông điệp trực quan nhằm giảm thiểu nhầm lẫn trong quá trình thao tác. Các trạng thái chính bao gồm:

- **Chụp ảnh vào (Check-in):** Hiển thị khung camera và nút xác nhận để ghi nhận thời điểm vào.



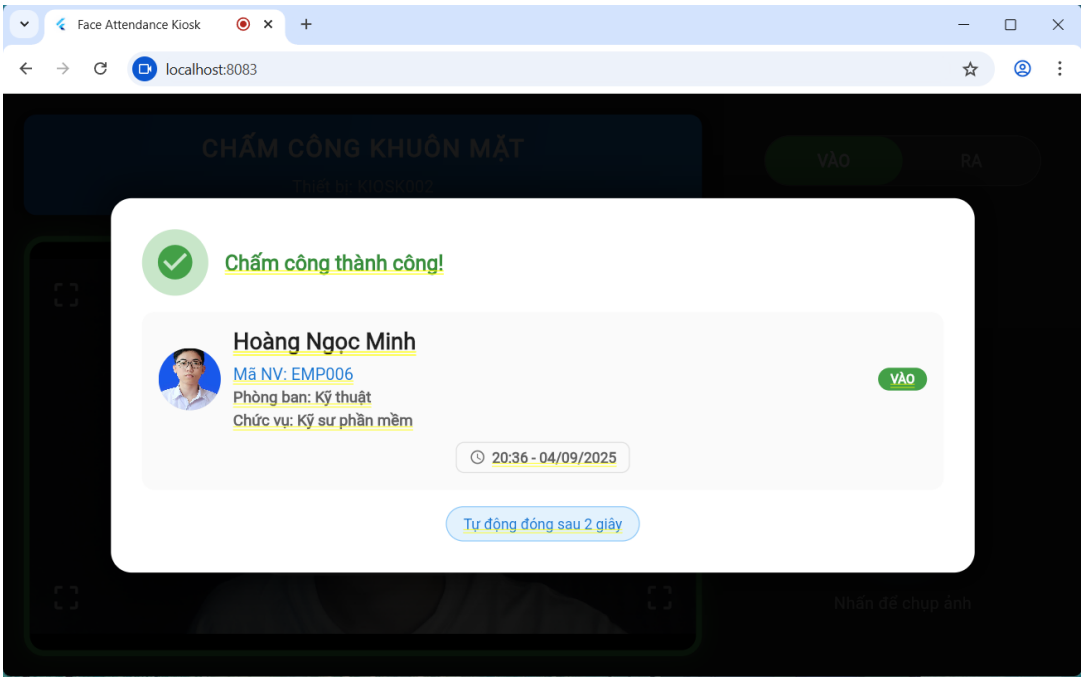
Hình 2.10: Giao diện chụp ảnh Vào

- **Chụp ảnh ra (Check-out):** Giao diện tương tự nhưng được đánh dấu rõ ràng để phân biệt với check-in.

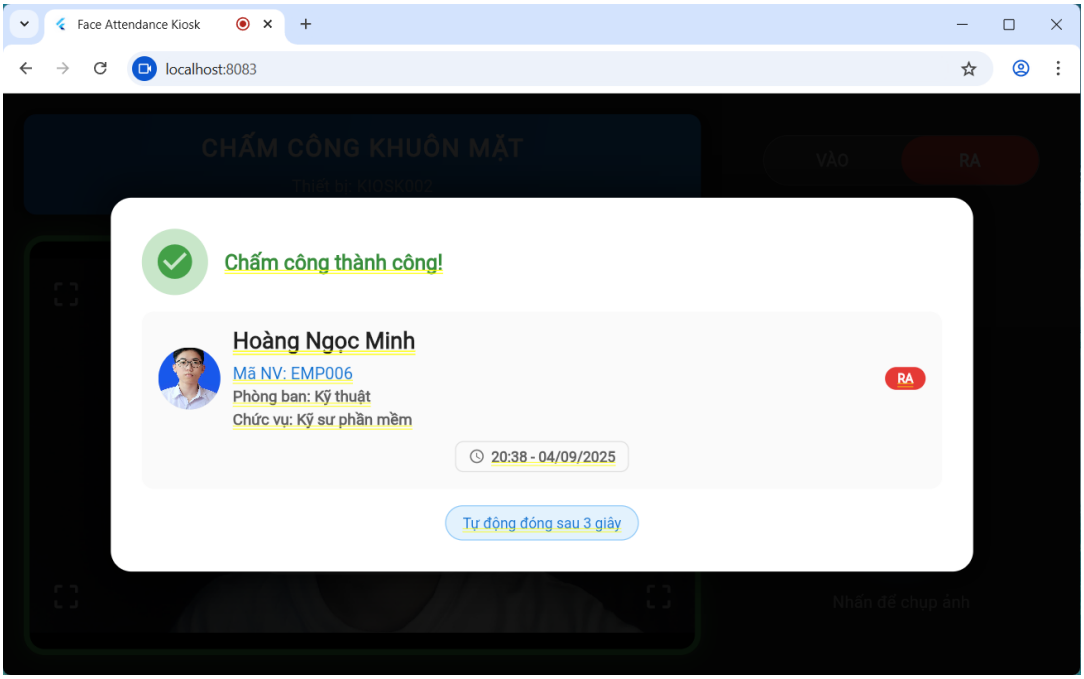


Hình 2.11: Giao diện chụp ảnh Ra

- **Kết quả thành công:** Thông báo xác nhận khi nhân viên vào/ra thành công, kèm ảnh, họ tên, mã nhân viên, phòng ban, chức vụ và thời gian.

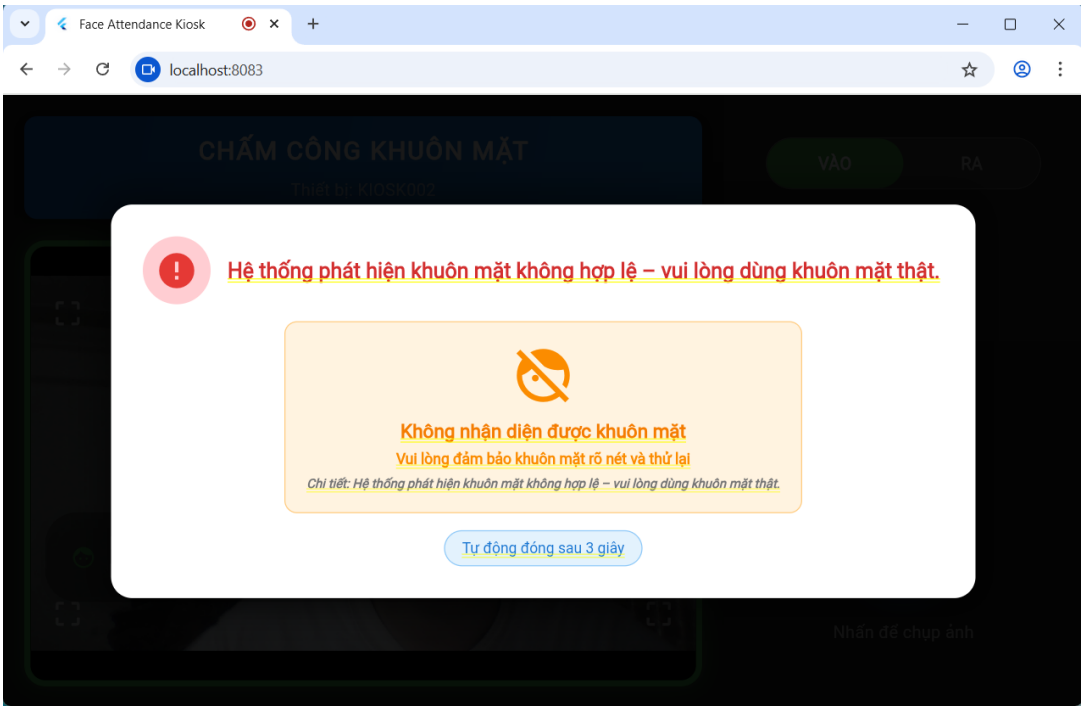


Hình 2.12: Thông báo vào thành công



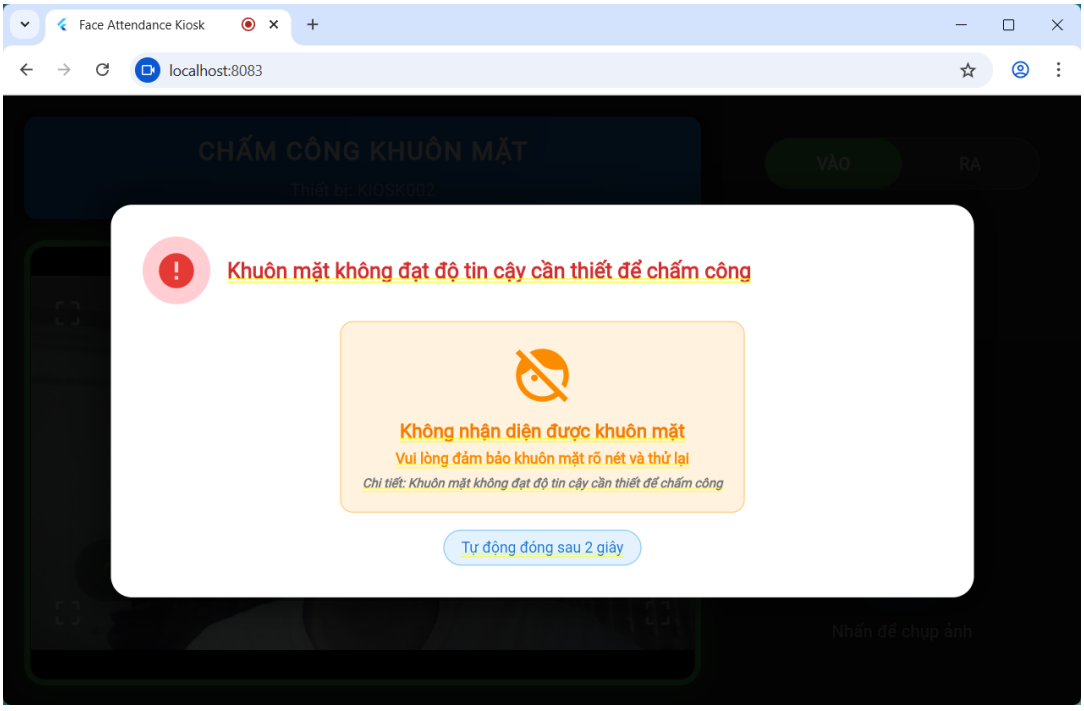
Hình 2.13: Thông báo ra thành công

- **Cảnh báo giả mạo (Spoof Alert):** Hệ thống phát hiện hành vi giả mạo như ảnh in, màn hình hoặc video, hiển thị cảnh báo và từ chối ghi nhận chấm công.



Hình 2.14: Cảnh báo giả mạo (Fake face)

- **Khuôn mặt không đủ tin cậy (Low Confidence Match):** Nhận diện được khuôn mặt thật nhưng điểm so khớp không vượt ngưỡng tin cậy. Nguyên nhân có thể do (i) nhân viên chưa có dữ liệu, (ii) điều kiện ánh sáng/góc nhìn không chuẩn, hoặc (iii) khác biệt lớn so với dữ liệu lưu trữ.



Hình 2.15: Cảnh báo khuôn mặt không đủ tin cậy

Chương 3

Kết quả

3.1 Đánh giá kết quả lõi AI

Để đánh giá hiệu năng của lõi AI trong một kịch bản gần với thực tế, một thử nghiệm định lượng đã được thực hiện với cơ chế sử dụng ngưỡng xác thực (authentication threshold).

3.1.1 Quy trình thử nghiệm nâng cao

Thử nghiệm được tiến hành dựa trên bộ dữ liệu bao gồm 2562 hình ảnh chân dung của 31 người khác nhau, với các đặc điểm như sau:

- **Tập tham chiếu (Reference Set):** Đối với mỗi người, 3 hình ảnh được lựa chọn ngẫu nhiên để xây dựng cơ sở dữ liệu tham chiếu. Các hình ảnh này được đưa qua pipeline phát hiện khuôn mặt (YOLOv11s detection) và trích xuất đặc trưng (InsightFace buffalo-l) để tạo ra các vector embedding. Kết quả là một cơ sở dữ liệu tham chiếu chứa $31 \times 3 = 93$ vector embedding.
- **Tập kiểm tra (Test Set):** 2469 hình ảnh còn lại được sử dụng làm tập dữ liệu kiểm tra để đánh giá độ chính xác của hệ thống.
- **Quy trình đánh giá:** Với mỗi ảnh trong tập kiểm tra, hệ thống thực hiện pipeline: (1) Phát hiện khuôn mặt, (2) Trích xuất vector embedding, (3) Tính toán độ tương đồng Cosine (Cosine Similarity) với tất cả 93 vector trong tập tham chiếu, (4) Gán nhãn của vector tham chiếu có độ tương đồng cao nhất cho ảnh đang kiểm tra.

Tuy nhiên, ta áp dụng một logic quan trọng để biến mô hình từ một bài toán phân loại lý thuyết ("Đây là ai trong 31 người?") thành một hệ thống định danh thực tế ("Đây có phải là một người quen không, và nếu có thì là ai?"):

- Một hình ảnh trong tập kiểm tra chỉ được coi là định danh thành công nếu điểm tương đồng Cosine (Cosine Similarity) cao nhất của nó với một trong các ảnh tham chiếu **lớn hơn ngưỡng 0.6**.
- Nếu điểm tương đồng cao nhất không vượt qua ngưỡng này, hình ảnh sẽ được coi là "không nhận diện được" (unrecognized), tương ứng với một lần Từ chối Sai (False Rejection).

Cách tiếp cận này mô phỏng chính xác cách hệ thống hoạt động trong thực tế, ưu tiên độ tin cậy hơn là cố gắng nhận diện mọi khuôn mặt.

3.1.2 Kết quả và phân tích

Với việc áp dụng ngưỡng xác thực, hệ thống đạt độ chính xác tổng thể là **94.7%**. Báo cáo phân loại chi tiết được trình bày trong Bảng 3.1.

Table 3.1: Bảng báo cáo phân loại chi tiết với ngưỡng similarity > 0.6

Tên lớp (Class)	**Precision**	**Recall**	**F1-score**	**Support**
Akshay Kumar	1.00	0.79	0.88	47
Alexandra Daddario	1.00	0.98	0.99	89
Billie Eilish	1.00	0.60	0.75	95
Courtney Cox	1.00	0.73	0.84	77
...	...	...	...	...
Zac Efron	1.00	0.92	0.96	88
Accuracy			**0.95**	**2469**
Macro Avg	**0.97**	**0.92**	**0.94**	**2469**
Weighted Avg	**1.00**	**0.95**	**0.97**	**2469**

Phân tích kết quả:

- **Độ tin cậy gần như tuyệt đối (Precision $\approx$ 1.0):** Chỉ số Precision cho thấy khi hệ thống đã đưa ra một định danh, quyết định đó gần như luôn luôn đúng. Tỷ lệ nhận diện nhầm người này thành người khác (False Acceptance) là cực kỳ thấp, đảm bảo tính bảo mật và minh bạch cho hệ thống chấm công.
- **Sự đánh đổi giữa Precision và Recall:** Để đạt được độ tin cậy cao, hệ thống đã chấp nhận bỏ qua một số trường hợp khó. Chỉ số Recall có sự biến động (ví dụ, Billie Eilish chỉ đạt 0.60) cho thấy khoảng 40% ảnh của người này không đủ "giống" với các ảnh tham chiếu để vượt qua ngưỡng 0.6. Điều này thể hiện triết lý thiết kế "thà bỏ sót còn hơn nhận sai", phù hợp với một hệ thống an ninh.
- **Nguyên nhân của Recall thấp:** Các trường hợp có Recall thấp có thể do các ảnh tham chiếu ban đầu chưa đủ đa dạng để bao quát hết các biến thể về ngoại hình (góc mặt, biểu cảm, kiểu tóc) của một người trong tập kiểm tra.

3.1.3 Thảo luận về việc lựa chọn ngưỡng

Việc lựa chọn ngưỡng 0.6 là một sự cân bằng giữa an toàn và tiện lợi.

- **Nếu tăng ngưỡng (≥ 0.7):** Hệ thống sẽ trở nên "khó tính" hơn, Precision có thể tăng nhưng Recall sẽ giảm mạnh, khiến người dùng phải thử lại nhiều lần.

- **Nếu giảm ngưỡng (≤ 0.5):** Hệ thống sẽ "dễ tính" hơn, Recall tăng lên giúp người dùng chấm công dễ hơn, nhưng Precision có thể giảm, làm tăng nguy cơ nhận diện sai.

Do đó, giá trị 0.6 được xem là phù hợp cho bài toán này, tối ưu hóa giữa việc giảm thiểu sai sót và đảm bảo trải nghiệm người dùng.

3.1.4 Phân tích và thảo luận kết quả

Kết quả 100% phản ánh hiệu năng vượt trội của pipeline AI (YOLOv11s detection + InsightFace) khi hoạt động trên một bộ dữ liệu có chất lượng cao và các đặc điểm được kiểm soát (controlled environment). Các hình ảnh trong bộ dữ liệu thử nghiệm có độ phân giải tốt, điều kiện ánh sáng đồng đều và ít sự thay đổi về góc mặt, giúp mô hình dễ dàng trích xuất các đặc trưng và so khớp với độ tin cậy tối đa.

Tuy nhiên, cần phải nhận định rằng kết quả này có thể không phản ánh chính xác hiệu năng của hệ thống trong môi trường vận hành thực tế. Trong thực tế, dữ liệu đầu vào từ camera Kiosk sẽ phức tạp hơn rất nhiều, bao gồm các thách thức như: ánh sáng yếu, ngược sáng, khuôn mặt bị che một phần (bởi khẩu trang, tóc, tay), góc mặt nghiêng, và chuyển động mờ (motion blur). Do đó, mặc dù thử nghiệm này xác nhận lõi AI hoạt động cực kỳ hiệu quả về mặt lý thuyết, việc đánh giá trong các điều kiện thực tế khắc nghiệt hơn là cần thiết để có cái nhìn toàn diện về độ tin cậy của hệ thống.

3.2 Đánh giá giao diện quản trị (Admin Dashboard)

Bên cạnh lõi AI, giao diện quản trị là một thành phần quan trọng, ảnh hưởng trực tiếp đến khả năng vận hành và giám sát hệ thống của người quản lý. Do đó, việc kiểm tra tính năng và đo lường hiệu năng của Admin Dashboard là rất cần thiết.

3.2.1 Kiểm tra chức năng

Một loạt các kịch bản kiểm thử đã được thực hiện để xác minh rằng tất cả các chức năng nghiệp vụ chính của hệ thống đều hoạt động đúng như thiết kế. Kết quả được tổng hợp trong Bảng 3.2.

Table 3.2: Bảng kết quả kiểm tra chức năng Admin Dashboard

STT	Chức năng	Kết quả
1	Hiển thị thông số tổng quan theo thời gian thực	Đạt
2	Quản lý nhân viên (Thêm, sửa, xóa, tìm kiếm, lọc)	Đạt
3	Quản lý lịch sử chấm công (Lọc theo ngày, mã nhân viên, xuất CSV)	Đạt
4	Quản lý thiết bị Kiosk (Thêm, sửa, xóa, xem trạng thái)	Đạt
5	Tạo và xuất báo cáo chấm công (Tuần/Tháng này) ra file CSV	Đạt

Kết quả kiểm tra cho thấy 100% các chức năng nghiệp vụ cốt lõi của Admin Dashboard đều hoạt động ổn định, chính xác và xử lý đúng các trường hợp đầu vào.

3.2.2 Đánh giá hiệu năng

Để định lượng hóa trải nghiệm "mượt mà" của người dùng, thời gian phản hồi của các tác vụ chính đã được đo lường. Phép đo được thực hiện bằng công cụ Developer Tools của trình duyệt trên một máy tính có kết nối mạng ổn định, cơ sở dữ liệu chứa 25 nhân viên và 1000 bản ghi chấm công.

Table 3.3: Bảng kết quả đo lường hiệu năng giao diện

STT	Hành động	Thời gian phản hồi TB (s)
1	Tải trang Dashboard (có thống kê)	~ 1
2	Tải trang danh sách Nhân viên (25 bản ghi)	~ 1
3	Mở form chỉnh sửa thông tin một nhân viên	~ 0.3
4	Chỉnh sửa thông tin một nhân viên	~ 0.5
5	Thực hiện thao tác xóa một nhân viên	~ 0.5
6	Lọc lịch sử chấm công trong phạm vi 1 tuần	~ 0.3
7	Xuất CSV lịch sử chấm công	~ 0.5
8	Mở form chỉnh sửa thông tin thiết bị Kiosk	~ 0.3
9	Chỉnh sửa thông tin một thiết bị Kiosk	~ 0.5
10	Xuất CSV báo cáo chấm công (Tuần/Tháng này)	~ 0.5

Các số liệu cho thấy thời gian phản hồi của hầu hết các tác vụ quan trọng đều dưới 1 giây. Ngay cả với tác vụ phức tạp nhất là lọc dữ liệu, thời gian phản hồi vẫn ở mức chấp nhận được. Điều này khẳng định hệ thống có hiệu năng tốt, đảm bảo trải nghiệm người dùng nhanh và không có độ trễ đáng kể trong quá trình thao tác.

3.3 Đánh giá giao diện chấm công (Kiosk App)

Kiosk App là giao diện tương tác trực tiếp với nhân viên, do đó được thiết kế với ưu tiên hàng đầu là sự đơn giản, tốc độ và rõ ràng. Việc kiểm thử chức năng và hiệu năng của ứng dụng Kiosk đảm bảo rằng trải nghiệm chấm công diễn ra nhanh chóng, chính xác và dễ sử dụng.

3.3.1 Kiểm tra chức năng

Các kịch bản kiểm thử tập trung vào các tình huống thực tế mà nhân viên sẽ gặp khi sử dụng Kiosk App. Kết quả được tổng hợp trong Bảng 3.4.

Table 3.4: Bảng kết quả kiểm tra chức năng Kiosk App

STT	**Chức năng**	**Kết quả**
1	Nhận diện khuôn mặt để chấm công	Đạt
2	Thông báo trạng thái chấm công (Thành công / Thất bại)	Đạt
3	Hiển thị tên và ảnh nhân viên sau khi xác thực	Đạt
4	Xử lý trường hợp không nhận diện được (yêu cầu thử lại)	Đạt
5	Đồng bộ dữ liệu chấm công với máy chủ	Đạt

3.3.2 Đánh giá hiệu năng

Để đánh giá mức độ "mượt mà" của quá trình chấm công, thời gian phản hồi đo lường trong điều kiện mạng ổn định và dữ liệu chứa 25 nhân viên.

Kết quả cho thấy, toàn bộ quy trình Check-in/Check-out từ lúc nhân viên đứng trước thiết bị, hệ thống nhận diện khuôn mặt, gửi dữ liệu lên máy chủ, phân tích bằng mô hình AI, đến khi trả kết quả hiển thị trên màn hình chỉ mất trung bình khoảng 3 giây. Thời gian này bao gồm cả các bước xử lý quan trọng như phát hiện khuôn mặt, so khớp embedding và đồng bộ dữ liệu chấm công lên server. Đây là một thông số quan trọng, bởi thời gian chờ đợi lâu có thể ảnh hưởng trực tiếp đến trải nghiệm người dùng cũng như tính hiệu quả khi triển khai ở quy mô lớn.

3.4 Kết luận

Dự án "Xây dựng hệ thống chấm công bằng nhận diện khuôn mặt" đã gần hoàn thiện, đáp ứng đa phần các mục tiêu cốt lõi đã đề ra. Hệ thống đã chứng minh được hiệu quả qua các bài kiểm tra thực nghiệm, với lõi AI đạt độ chính xác cao, các ứng dụng Admin Dashboard và Kiosk App hoạt động ổn định, cung cấp trải

nghiệm người dùng tốt và đáp ứng đầy đủ các nhu cầu nghiệp vụ về quản lý nhân sự và chấm công.

Những kết quả đạt được không chỉ là một giải pháp công nghệ cụ thể mà còn là minh chứng cho khả năng ứng dụng thành công các công nghệ tiên tiến như Trí tuệ nhân tạo, container hóa và phát triển đa nền tảng để giải quyết các bài toán thực tiễn trong doanh nghiệp. Quá trình thực hiện dự án đã mang lại nhiều kiến thức và kinh nghiệm quý báu, từ việc thiết kế kiến trúc hệ thống, huấn luyện và tối ưu mô hình AI, cho đến việc phát triển và triển khai một sản phẩm hoàn chỉnh.

3.5 Hướng phát triển trong tương lai

Mặc dù hệ thống hiện tại đã gần hoàn thiện, vẫn còn nhiều tiềm năng để cải tiến và mở rộng nhằm nâng cao hiệu quả, bảo mật và khả năng thích ứng. Một số hướng phát triển chính:

- **Nâng cao bảo mật:** Áp dụng mã hóa đầu cuối (TLS/HTTPS) và xác thực bằng token cho từng thiết bị, ngăn chặn tấn công và đảm bảo an toàn dữ liệu.
- **Cơ chế tự học (Rolling Embeddings):** Hệ thống tự động cập nhật vector embedding từ các lần chấm công thành công, giúp thích ứng với thay đổi ngoại hình nhân viên mà không cần tái huấn luyện thủ công.
- **Tối ưu hóa hiệu năng:** Chuyển đổi mô hình sang ONNX/TensorRT, tối ưu truy vấn cơ sở dữ liệu và dùng caching để giảm độ trễ.
- **Mở rộng đa thiết bị:** Nâng cấp backend để quản lý đồng thời nhiều Kiosk, dễ dàng triển khai trên diện rộng.
- **Hoàn thiện giao diện Kiosk App:** Cải thiện giao diện dựa trên phản hồi người dùng, bổ sung trạng thái xử lý và thông báo lỗi rõ ràng hơn.

Tài liệu tham khảo

- **Ultralytics:** https://docs.ultralytics.com/vi/models/yolo11/
- **DeepInsight:** https://github.com/deepinsight/insightface
- **YOLOv11 Detection:** https://docs.ultralytics.com/vi/tasks/detect/
- **YOLOv11 Classification:** https://docs.ultralytics.com/vi/tasks/classify/