# BÁO CÁO GIỮA KỲ
# ĐỒ ÁN TỐT NGHIỆP – CAPSTONE PROJECT

## **Đề tài**: "Giải pháp chấm công cho nhà máy ứng dụng công nghệ nhận diện khuôn mặt"


**Sinh viên thực hiện:**

| Họ tên | MSSV | Lớp |
|---|---|---|
| Nguyễn Đình Khánh | 106200056 | 20DT2 |
| Hà Phước Phúc | 106200065 | 20DT2 |

**Hướng dẫn:** (TS.) Nguyễn Duy Nhật Viễn

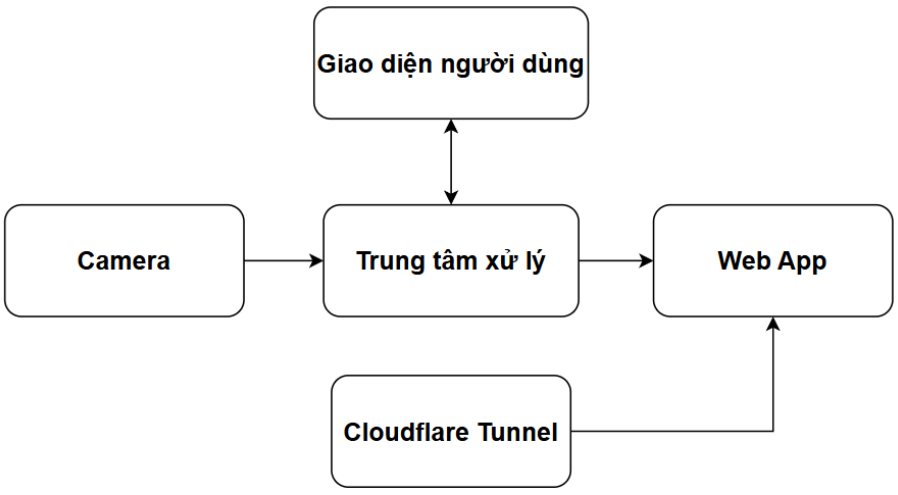

*Đà Nẵng, tháng 05 năm 2025*

## 2.1. Giải pháp

Đề tài xây dựng một quy trình chấm công tự động dựa trên công nghệ nhận diện khuôn mặt, nhằm tăng hiệu quả quản lý nhân sự, giảm thiểu gian lận và tối ưu hóa việc ghi nhận thời gian làm việc trong môi trường nhà máy. Hệ thống tích hợp mô hình học sâu để xử lý khuôn mặt và có giao diện trực quan, dễ sử dụng. Nội dung của những giải pháp này đã được nhóm trình bày ở báo cáo M1.

Các giải pháp mà nhóm áp dụng trong đồ án lần này tập trung vào việc xây dựng các thành phần cốt lõi của hệ thống chấm công: từ việc phát hiện khuôn mặt, trích xuất đặc trưng, phân loại danh tính đến lưu trữ dữ liệu và hiển thị giao diện. Tất cả các thành phần này được triển khai và tích hợp trên nền tảng phần cứng là Raspberry Pi, đóng vai trò như một bộ xử lý trung tâm.

- **Phát hiện khuôn mặt**: Sử dụng Mediapipe để phát hiện khuôn mặt thời gian thực từ camera USB.
- **Trích xuất đặc trưng**: Ảnh khuôn mặt được chuẩn hóa và ánh xạ thành vector đặc trưng bằng mô hình FaceNet đã tải sẵn.
- **Nhận diện khuôn mặt**: Vector đặc trưng được phân loại bằng mô hình SVM (SVC), giúp xác định chính xác danh tính nhân viên đã đăng ký.
- **Hiển thị**: Giao diện desktop được xây dựng bằng Tkinter trên Raspberry Pi, cho phép nhập thông tin thêm và xóa nhân viên, hiển thị trực tiếp kết quả nhận diện.
- **Lưu trữ**: Dữ liệu nhân viên và chấm công được lưu song song vào MySQL (qua XAMPP) và file CSV để đảm bảo dự phòng và đồng bộ hóa.
- **Web App**: Xây dựng bằng PHP kết nối với MySQL, cho phép người quản trị xem lịch sử chấm công, danh sách nhân viên và xuất dữ liệu từ xa. Hệ thống được công khai an toàn qua Internet thông qua dịch vụ Cloudflare Tunnel.




***Hình 2.1.*** *Sơ đồ khối hệ thống*

**Hình 2.1.** Mô tả sơ đồ khối của hệ thống chấm công sử dụng công nghệ nhận diện khuôn mặt. Camera có nhiệm vụ ghi lại hình ảnh khuôn mặt của nhân viên và truyền về trung tâm xử lý (Raspberry Pi), nơi thực hiện nhận diện và lưu trữ dữ liệu chấm công. Giao diện đồ họa được giúp người quản lý dễ dàng thao tác như thêm, xóa nhân viên hoặc theo dõi kết quả nhận diện. Dữ liệu sau khi xử lý cũng được đồng bộ lên Web App, cho phép truy cập từ xa thông qua kết nối bảo mật Cloudflare Tunnel.

***Bảng 2.1.*** *Phân công công việc*

| STT | Nội dung | Khánh | Phúc |
|---|---|---|---|
| **1** | Tìm hiểu đề tài, phân tích yêu cầu và xác định kiến trúc hệ thống tổng thể. | ✓ | ✓ |
| **2** | Thiết kế sơ đồ khối chức năng và luồng dữ liệu hệ thống. | ✓ | ✓ |
| **3** | Nghiên cứu Raspberry Pi, camera và các thư viện nhận diện khuôn mặt. | ✓ | ✓ |
| **4** | Xây dựng chức năng nhận diện khuôn mặt cơ bản trên Raspberry Pi. | | ✓ |
| **5** | Phát triển chức năng thêm/xóa nhân viên trên hệ thống. | | ✓ |
| **6** | Thiết kế và xây dựng giao diện hiển thị trên Raspberry Pi. | ✓ | |
| **7** | Kiểm thử giao diện hiển thị và kết nối với chức năng nhận diện. | ✓ | ✓ |
| **8** | Thiết kế và lập trình trang web hiển thị dữ liệu chấm công. | ✓ | ✓ |
| **9** | Kết nối web với cơ sở dữ liệu MySQL, xây dựng chức năng tải file chấm công. | | ✓ |

| | | | |
|---|---|---|---|
| **10** | Thêm tính năng lưu dữ liệu vào CSV cục bộ (sao lưu dự phòng). | | ✓ |
| **11** | Đồng bộ dữ liệu từ CSV sang MySQL khi mất kết nối Internet. | | ✓ |
| **12** | Cấu hình database bằng dotenv tách riêng trong file .env_file_mysql. | | ✓ |
| **13** | Xây dựng tính năng đăng nhập cho phép Admin thao tác thêm/xóa/chỉnh sửa trên giao diện Tkinter. | ✓ | |
| **14** | Kiểm thử hệ thống giao tiếp giữa Raspberry Pi và giao diện web. | ✓ | ✓ |
| **15** | Triển khai truy cập từ xa trang web bằng Cloudflare Tunnel. | ✓ | |
| **16** | Thử nghiệm toàn hệ thống trong điều kiện thực tế. | ✓ | ✓ |
| **17** | Kiểm tra, tối ưu, sửa lỗi và cải tiến nếu cần. | ✓ | ✓ |
| **18** | Viết báo cáo tổng kết. | ✓ | ✓ |
| **19** | Chạy thử nghiệm chính thức, hoàn tất sản phẩm. | ✓ | ✓ |
| **20** | Hoàn thiện mô hình trình bày sản phẩm. | ✓ | ✓ |

### 2.1.1. Webcam nhận diện

**a. Chức năng:**

Khối webcam nhận diện có chức năng ghi nhận hình ảnh khuôn mặt người dùng theo thời gian thực và truyền hình ảnh đến bộ xử lý trung tâm để thực hiện quá trình nhận diện. Đây là điểm khởi đầu của toàn bộ hệ thống, đóng vai trò thu thập dữ liệu gốc.

Trong quá trình vận hành, webcam được ví như "con mắt" của hệ thống, quan sát và phát hiện khi nhân viên đứng trước thiết bị. Nếu thiết bị này không hoạt động, toàn bộ quá trình chấm công sẽ bị gián đoạn.

**b. Xử lý tín hiệu:**

**- Tín hiệu vào:** Không có tín hiệu điều khiển chủ động, webcam hoạt động liên tục khi được cấp nguồn và được khởi tạo bởi phần mềm.

**- Tín hiệu ra:** Chuỗi khung hình thời gian thực (video frames), được truyền trực tiếp đến bộ xử lý trung tâm để phân tích khuôn mặt.

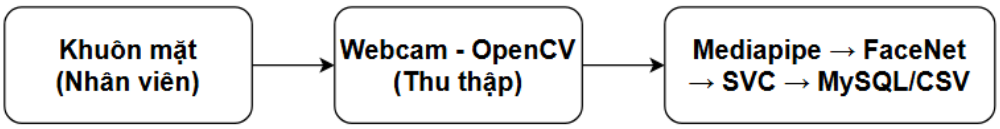
- Các khung hình này tiếp tục được đưa vào pipeline xử lý gồm: phát hiện khuôn mặt (Mediapipe), trích xuất đặc trưng (FaceNet) và phân loại (SVM).

- Tốc độ xử lý tín hiệu phụ thuộc vào độ phân giải và hiệu năng phần cứng, dao động ổn định ở mức 25 - 30 FPS.

**c. Cách thức triển khai:**

Thiết bị sử dụng là webcam USB rời, độ phân giải 1080p, kết nối qua USB 3.0 và được Raspberry Pi nhận diện tự động. Trong chương trình Python, webcam được khởi tạo bằng thư viện OpenCV thông qua lệnh `cv2.VideoCapture(0)`, cho phép thu nhận khung hình thời gian thực để đưa vào chuỗi xử lý nhận diện khuôn mặt.

Sau khi kết nối thành công, OpenCV sẽ liên tục lấy khung hình và chuyển chúng sang bước phát hiện khuôn mặt. Quá trình này diễn ra hoàn toàn tự động trong suốt thời gian hệ thống hoạt động.




***Hình 2.2** Sơ đồ luồng dữ liệu chính từ Webcam đến các thành phần xử lý*

**Hình 2.2** Thể hiện luồng dữ liệu chi tiết hơn, bắt đầu từ webcam (được truy cập qua OpenCV), sau đó đến các mô hình và thư viện xử lý (Mediapipe, FaceNet, SVC), và cuối cùng là lưu trữ dữ liệu vào MySQL và CSV.

### 2.1.2. Bộ xử lý trung tâm

**a. Chức năng của khối:**

Bộ xử lý trung tâm chịu trách nhiệm toàn bộ quy trình nhận diện khuôn mặt, ghi nhận chấm công, lưu trữ dữ liệu và điều khiển hiển thị giao diện. Đây là thành phần chính

của hệ thống, đảm nhận việc xử lý tín hiệu đầu vào từ webcam, phân tích khuôn mặt và cập nhật kết quả vào cơ sở dữ liệu.

**b. Xử lý tín hiệu:**

**- Tín hiệu vào:**

+ Chuỗi khung hình thời gian thực từ webcam.

+ Tín hiệu điều khiển từ người dùng nhập trên giao diện (thêm/xóa nhân viên).

**- Tín hiệu ra:**

+ Kết quả nhận diện khuôn mặt: mã nhân viên, họ tên, vị trí, thời gian.

+ Dữ liệu lưu vào MySQL và file CSV.

+ Tín hiệu hiển thị thông tin và ảnh khuôn mặt trên giao diện Tkinter.

**c. Cách thức triển khai:** Bộ xử lý trung tâm được triển khai trên Raspberry Pi, sự kết hợp của nhiều thư viện xử lý ảnh và học máy

**- Phát hiện khuôn mặt:** Sử dụng MediaPipe Face Detection với *`model_selection=0`*, giúp phát hiện khuôn mặt nhanh và chính xác trong khung hình thời gian thực.

**- Trích xuất đặc trưng:** Hình ảnh khuôn mặt được chuẩn hóa về *`160×160pixel`* rồi đưa vào mô hình FaceNet *`(20180402-114759.pb)`* Mô hình ánh xạ khuôn mặt thành vector 128 chiều, đại diện duy nhất cho mỗi cá nhân.

**- Nhận diện khuôn mặt:** Vector đặc trưng đầu ra từ FaceNet được phân loại bằng mô hình SVM (SVC) với kernel RBF (C=20.0, gamma=0.01). Nếu xác suất dự đoán vượt ngưỡng 0.8, hệ thống xác nhận là nhận diện thành công; ngược lại sẽ hiển thị là "Unknown".

**- Ghi nhận & lưu trữ:**

+ Dữ liệu chấm công được ghi vào file checkin.csv.

+ Đồng thời cập nhật vào bảng checkin của MySQL.

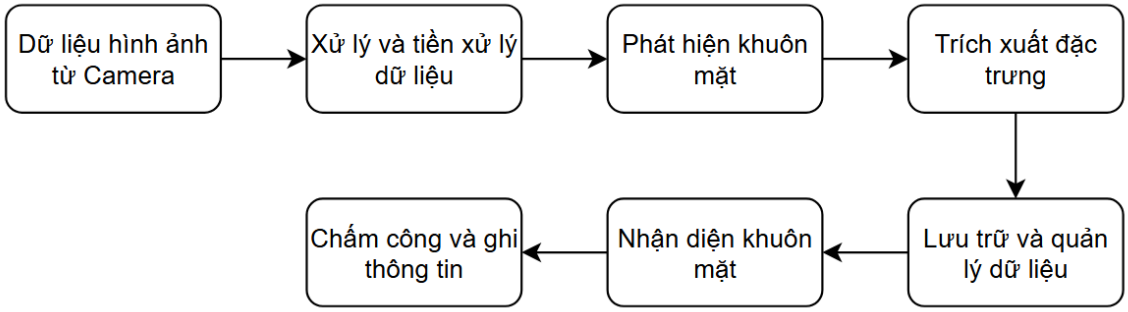
**- Đồng bộ dữ liệu CSV – MySQL**: Hệ thống có chức năng đồng bộ hai chiều giữa CSV và MySQL, giúp đảm bảo dữ liệu nhất quán khi mất kết nối tạm thời với database.

**- Cấu hình database bằng dotenv:** Thông tin kết nối MySQL được tách riêng trong file .env_file_mysql, tải bằng thư viện dotenv. Điều này giúp bảo mật thông tin nhạy cảm (host, user, password) và dễ dàng thay đổi cấu hình.

**- Xử lý song song với threading:** Để tránh gián đoạn giao diện hoặc lag hệ thống, các tác vụ như huấn luyện mô hình, xóa dữ liệu nhân viên hoặc đồng bộ dữ liệu được thực hiện trong các luồng riêng biệt. Điều này giúp đảm bảo giao diện luôn mượt và hệ thống không bị treo.

**- Xử lý lỗi & ngoại lệ:** Code sử dụng nhiều khối try...except để xử lý lỗi đọc/ghi file, lỗi kết nối CSDL, lỗi khi không nhận diện được ảnh. Nhờ đó, hệ thống có tính ổn định cao, có thể hoạt động lâu dài trong điều kiện không hoàn hảo.

**- Tự động huấn luyện lại SVM:** Khi thêm nhân viên mới, hệ thống thu thập dữ liệu và huấn luyện lại mô hình SVM để cập nhật danh sách nhận diện. Quá trình huấn luyện được xử lý tự động trong luồng riêng, giúp người dùng không cần can thiệp kỹ thuật.




***Hình 2.3*** *Quy trình xử lý dữ liệu của bộ xử lý trung tâm*

**Hình 2.3** Mô tả quy trình xử lý tại bộ xử lý trung tâm. Dữ liệu hình ảnh từ camera được tiền xử lý và đưa vào mô-đun phát hiện khuôn mặt. Khuôn mặt sau đó được trích xuất đặc trưng và lưu vào cơ sở dữ liệu. Khi nhận diện, hệ thống so sánh với dữ liệu đã lưu, xác định danh tính và ghi nhận chấm công nếu hợp lệ.

### 2.1.3. Giao diện Tkinter

**a. Chức năng:**

Giao diện Tkinter là điểm tương tác giữa người dùng và hệ thống. Giao diện hiển thị trực tiếp trên màn hình HDMI của Raspberry Pi, cho phép người dùng thao tác toàn bộ hệ thống mà không cần dòng lệnh. Các chức năng chính bao gồm:

- Hiển thị luồng video thời gian thực từ camera.
- Hiển thị ảnh khuôn mặt đã nhận diện kèm thông tin cá nhân (tên, mã NV, vị trí, thời gian, trạng thái).
- Cung cấp các nút chức năng: bắt đầu nhận diện, thêm nhân viên, xóa nhân viên.
- Gọi các hàm xử lý tương ứng để lưu dữ liệu vào cơ sở dữ liệu hoặc file CSV.

**b. Xử lý tín hiệu:**

**- Dữ liệu đầu vào:**

- Hình ảnh thời gian thực từ camera.
- Thao tác của người dùng (click nút, nhập tên/mã nhân viên...).
- Kết quả xử lý khuôn mặt từ mô hình (tên, mã NV, vị trí, ảnh đã cắt).

**- Dữ liệu đầu ra:**

- Giao diện cập nhật thông tin chấm công trực tiếp.
- Giao diện hiển thị hình ảnh và dữ liệu nhân viên hiển thị rõ ràng.

**c. Cách thức triển khai:**

Giao diện được xây dựng bằng thư viện Tkinter trong Python, chia thành các tab chức năng. Mỗi tab tương ứng với một tác vụ cụ thể (chấm công, thêm, xóa nhân viên...).

Luồng video từ webcam được hiển thị bằng cách sử dụng OpenCV kết hợp với PIL để cập nhật ảnh liên tục trong widget.

Các nút bấm được gán hàm xử lý để gọi tiến trình như: nhận diện, huấn luyện lại mô hình, xóa dữ liệu...

Để tránh treo giao diện, các tiến trình nặng (như huấn luyện mô hình) được thực hiện bằng threading trong nền.

Thông tin kết quả được cập nhật lại liên tục lên GUI bằng cơ chế gọi lặp


***Hình 2.4*** *Sơ đồ quy trình xử lý của giao diện Tkinter*

**Hình 2.4** Mô tả quy trình xử lý:

- Khởi động giao diện và chọn tab: Giao diện được xây dựng bằng Tkinter, hiển thị trên màn hình HDMI của Raspberry Pi. Các tab chức năng như chấm công, thêm, xóa nhân viên được quản lý bằng ttk.Notebook.

- Hiển thị thông tin và chức năng: Mỗi tab hiển thị nội dung riêng,

    + Tab chấm công: luồng video thời gian thực (OpenCV + PIL).

    + Tab thêm/xóa: vùng nhập dữ liệu, nút thao tác và ảnh mẫu.

    + Giao diện cập nhật liên tục

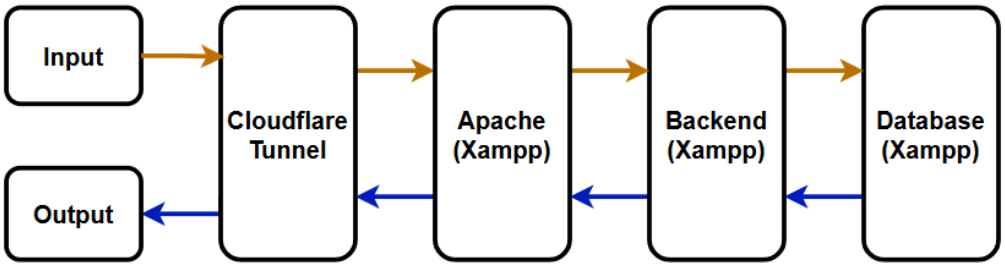
- Xác định thao tác người dùng: Giao diện lắng nghe thao tác như nhấn nút hoặc nhập thông tin, sau đó gọi đúng hàm xử lý.

- Gọi tiến trình xử lý: Các thao tác như nhận diện, huấn luyện SVM hoặc xóa dữ liệu sẽ được gọi tùy theo yêu cầu. Các tiến trình nặng sử dụng threading để không làm treo giao diện.

- Cập nhật kết quả: Sau khi xử lý, kết quả như ảnh khuôn mặt, thông tin nhận diện, thời gian... sẽ được hiển thị trực tiếp trên giao diện.

### 2.1.4. Hệ thống Web App (XAMPP)

Web App với vai trò là cầu nối trực quan giữa người quản lý và toàn bộ hệ thống chấm. Nó cho phép theo dõi, tra cứu, và xuất dữ liệu chấm công từ xa một cách thuận tiện, đồng thời đảm bảo dữ liệu được hiển thị rõ ràng và dễ sử dụng. Khối này được xây dựng với PHP, sử dụng dịch vụ Cloudflare Tunnel để hỗ trợ truy cập từ xa qua Internet mà không cần cấu hình mạng phức tạp.




***Hình 2.5*** *Sơ đồ quy trình xử lý của Web App*

Chức năng chính của Web App:

- Hiển thị thông tin nhân viên, thời gian chấm công.
- Hiển thị danh sách nhân viên đã đăng ký trong hệ thống nhận diện khuôn mặt.

- Cung cấp tính năng lọc và tìm kiếm theo tên nhân viên hoặc theo tháng, năm.
- Cho phép xuất dữ liệu chấm công ra tệp định dạng .csv hoặc .xlsx.
- Truy cập từ xa thông qua Cloudflare Tunnel.

Dữ liệu đầu vào chính của khối bao gồm:

- Thao tác nhấn vào các nút chức năng như lọc theo thời gian, tìm kiếm nhân viên, hoặc xuất dữ liệu.
- Cơ sở dữ liệu đã lưu.

**Hình 2.5** Mô tả quá trình xử lý của Web App:

- Người dùng truy cập website và gửi yêu cầu (HTTP/HTTPS request) từ trình duyệt.
- Cloudflare Tunnel nhận yêu cầu từ Internet, tạo đường hầm đến máy chủ cục bộ chạy XAMPP.
- Apache (XAMPP) tiếp nhận yêu cầu, nếu là tệp PHP thì chuyển cho backend xử lý.
- Backend (PHP trong XAMPP) xử lý logic ứng dụng, truy vấn cơ sở dữ liệu nếu cần.
- MySQL Database (XAMPP) thực hiện truy vấn (SELECT/INSERT/UPDATE...) và trả kết quả về cho backend.
- Backend xử lý kết quả, tạo nội dung HTML/JSON và gửi về cho Apache.
- Apache chuyển phản hồi ngược qua Cloudflare Tunnel ra Internet.
- Trình duyệt nhận và hiển thị nội dung kết quả.

Kết quả xử lý được phản ánh qua hai dạng chính:

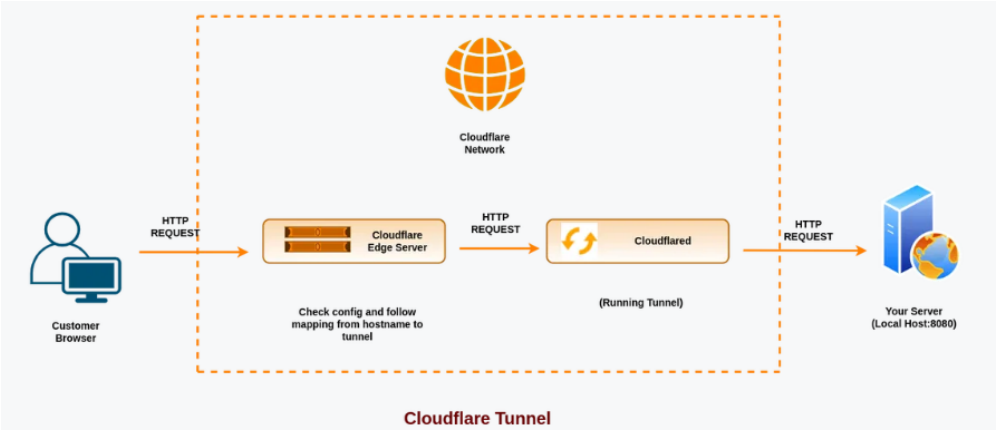
- Các bảng chấm công luôn cập nhật theo thông tin mới nhất từ hệ thống.
- Người dùng có thể tải về dữ liệu lịch sử chấm công dưới định dạng .csv hoặc .xlsx tùy theo nhu cầu.

### 2.1.5. Cloudflare Tunnel

Cloudflare Tunnel là một giải pháp hiện đại giúp kết nối các ứng dụng chạy trên máy cục bộ hoặc mạng nội bộ với Internet một cách an toàn, bảo mật và dễ dàng mà không cần mở cổng trên router hay cấu hình firewall phức tạp.

Chức năng chính:

- Kết nối ứng dụng nội bộ ra internet mà không cần mở cổng mạng hay cấu hình firewall.
- Bảo vệ máy chủ nội bộ khỏi tấn công trực tiếp.
- Mã hóa toàn bộ dữ liệu truyền qua tunnel.
- Tạo subdomain truy cập nhanh, kể cả dùng domain riêng.




***Hình 2.6*** *Sơ đồ quy trình hoạt động của Cloudflare Tunnel*

Dữ liệu đầu vào bao gồm:

- Yêu cầu HTTP từ người dùng.
- Dữ liệu trong yêu cầu.
- Yêu cầu được kiểm tra bảo mật bởi Cloudflare.

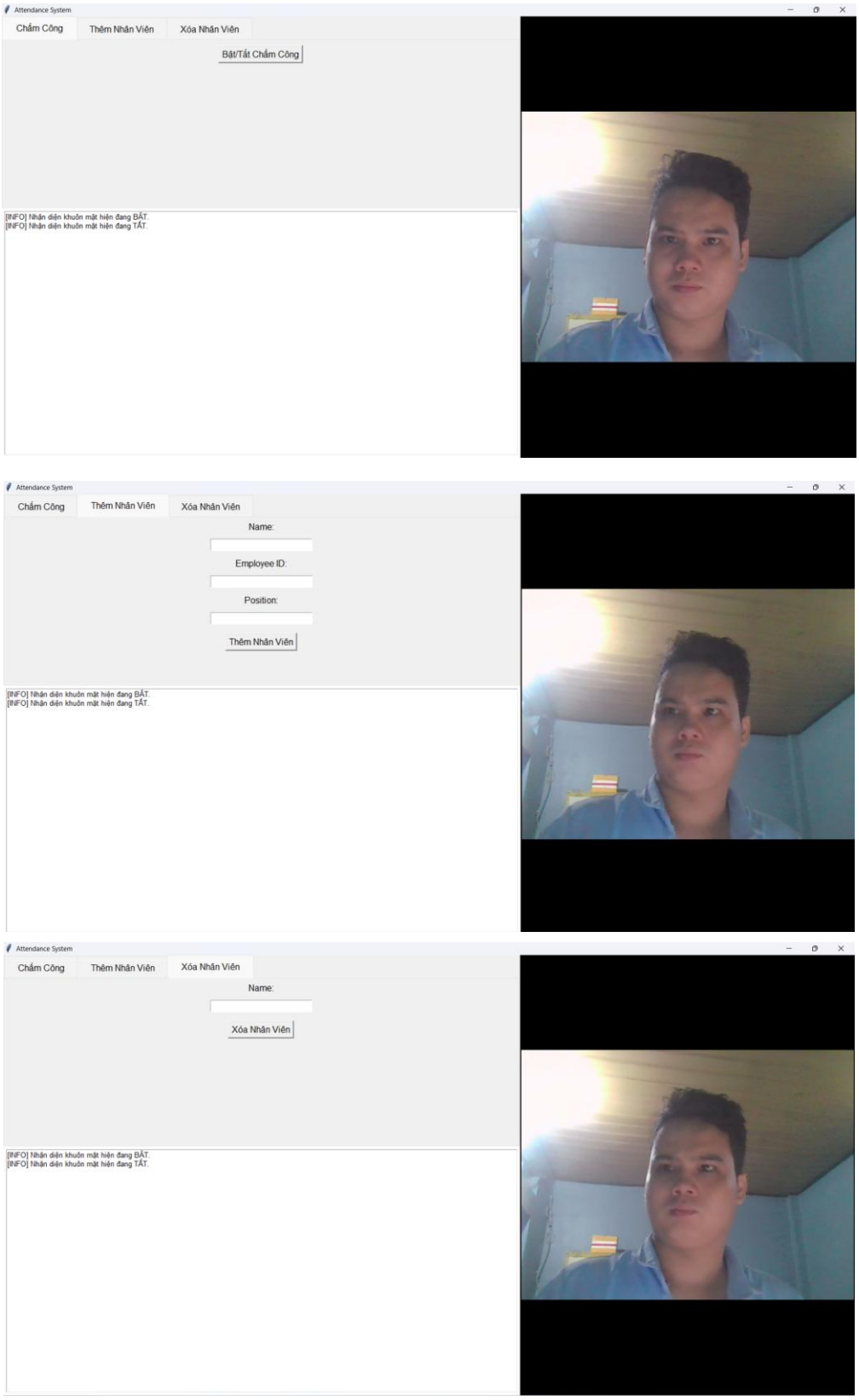
**Hình 2.6** mô tả quy trình hoạt động:

- Người dùng truy cập vào ứng dụng qua một địa chỉ.
- Cloudflare Edge Server kiểm tra bảo mật yêu cầu từ người dùng và xác thực, lọc các nguy cơ tấn công hoặc truy cập trái phép.
- Cloudflare chuyển tiếp yêu cầu qua một kết nối an toàn (Tunnel) đến máy chủ nội bộ nơi ứng dụng đang chạy.
- Máy chủ nội bộ xử lý yêu cầu và gửi phản hồi ngược lại qua Tunnel, và Cloudflare chuyển phản hồi đến người dùng.

Dữ liệu đầu ra:

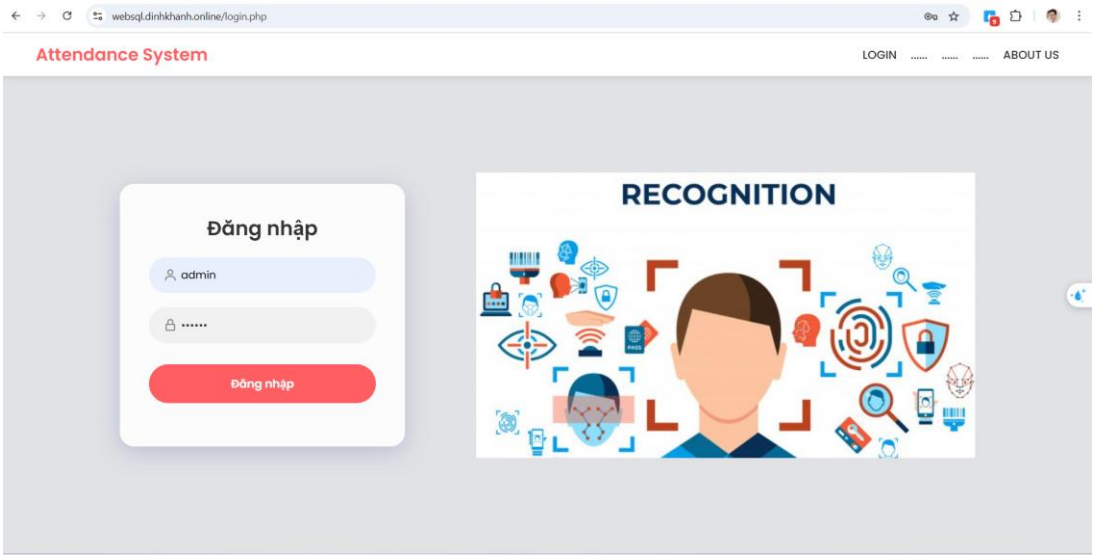
- Yêu cầu HTTP được chuyển tiếp tới máy chủ nội bộ qua Tunnel.
- Phản hồi HTTP từ máy chủ nội bộ (mã trạng thái, dữ liệu phản hồi).
- Phản hồi được chuyển lại từ Cloudflare đến người dùng.

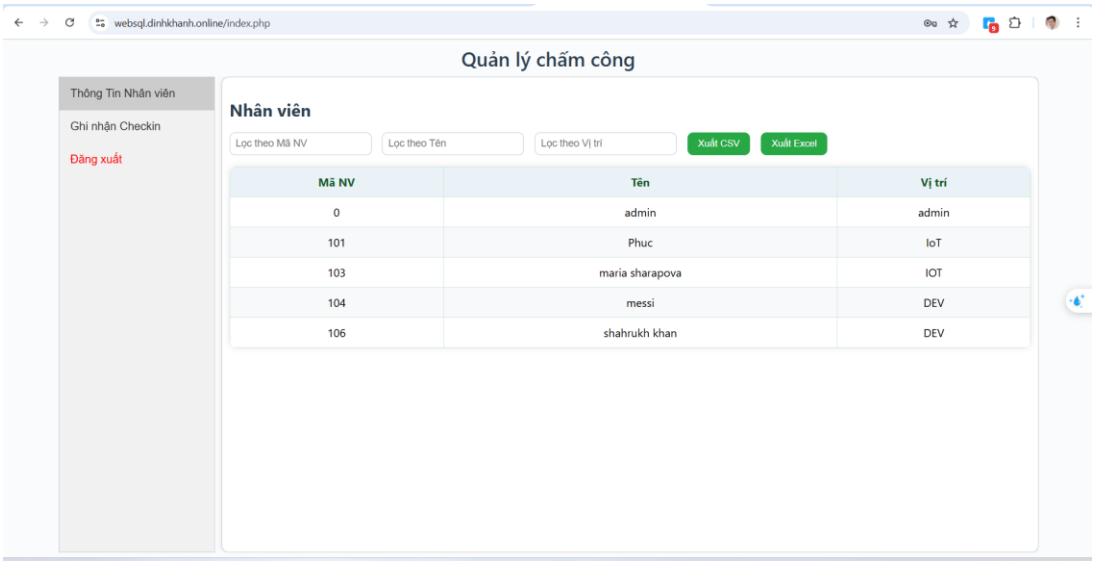
## 2.2. Kết quả (tính đến thời điểm hiện tại)

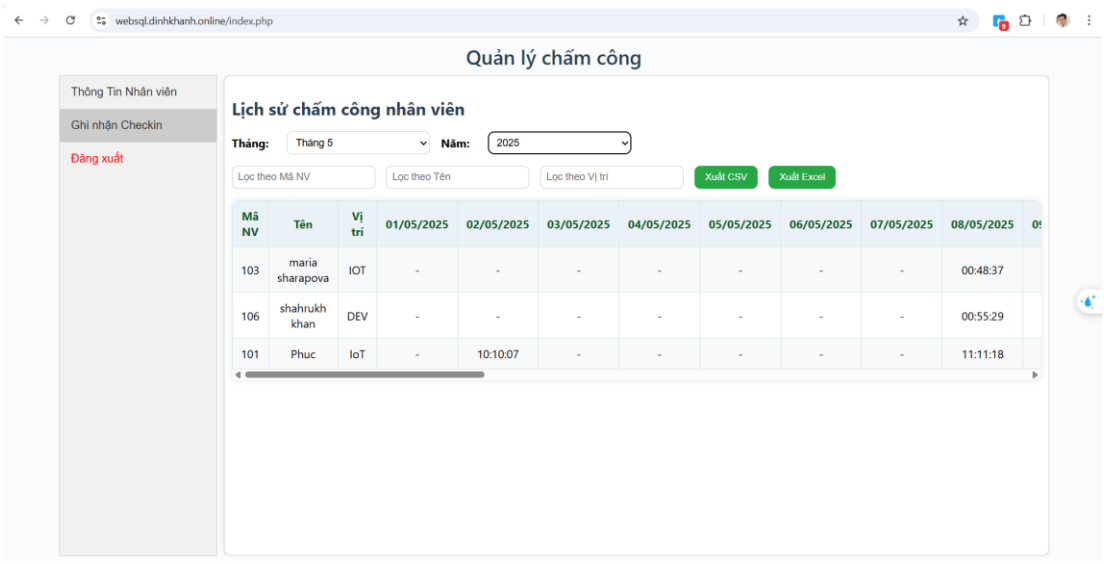



***Hình 2.7*** *Giao diện hiển thị màn hình chấm công*

**Hình 2.7** cho thấy giao diện hiển thị GUI ở thời điểm hiện tại. Phía bên trái màn hình là bảng điều khiển bao gồm các tab chức năng chính như: "Bật/Tắt Chấm Công", "Thêm Nhân Viên" và "Xóa Nhân Viên". Khi nhấn vào tab "Thêm Nhân Viên", hệ thống sẽ hiển thị thêm các ô nhập thông tin như tên, mã nhân viên và vị trí công việc cùng với nút "Xác Nhận Thêm". Tương tự, khi chọn "Xóa Nhân Viên", người dùng sẽ thấy ô nhập tên và nút "Xác Nhận Xóa". Bên phải giao diện là khung hiển thị camera, nơi người dùng có thể thấy hình ảnh khuôn mặt của mình theo thời gian thực. Thiết kế này chưa đạt yêu cầu đề ra và đang trong quá trình chỉnh sửa.

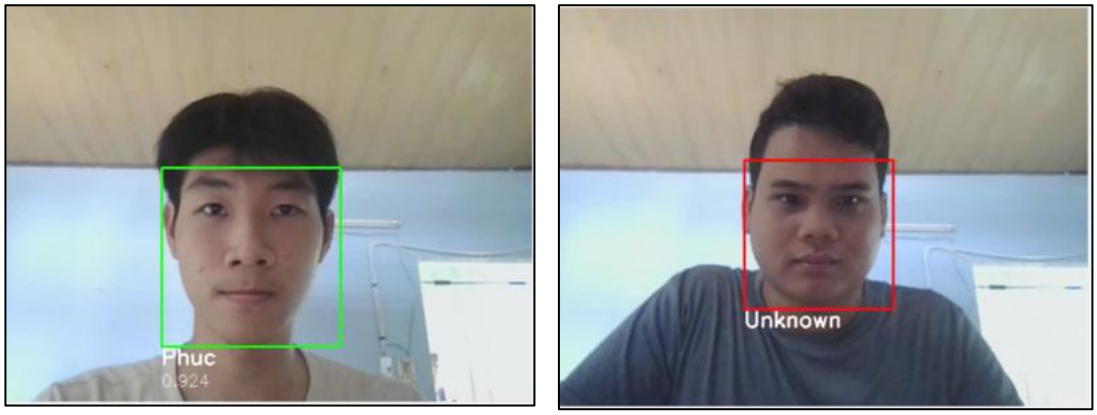





***Hình 2.8*** *Giao diện web trên máy tính*

**Hình 2.8**cho thấy các phần chính của trang web quản lý chấm công. Người dùng đăng nhập bằng tài khoản quản trị để truy cập hệ thống. Trong trang quản lý nhân viên, danh sách nhân viên được hiển thị rõ ràng theo mã, tên và vị trí, kèm theo các chức năng lọc và xuất dữ liệu (.csv, .xlsx). Trang lịch sử chấm công cho phép tra cứu theo tháng, năm, mã nhân viên, tên hoặc vị trí, đồng thời hỗ trợ xuất file dữ liệu tùy chọn.




***Hình 2.9*** *Kết quả nhận diện khuôn mặt*

**Hình 2.9** cho thấy kết quả nhận diện của hệ thống chấm công. Ở ảnh đầu, hệ thống nhận diện thành công và xác định người dùng là "Phuc" với độ chính xác 0.924. Trong ảnh thứ hai, hệ thống không nhận diện được khuôn mặt, hiển thị "Unknown", cho thấy người này chưa được đăng ký trong cơ sở dữ liệu.

### 2.3. Kế hoạch

| STT | Nội dung | Thời gian (ngày) | Ngày bắt đầu | Đã thực hiện |
|---|---|---|---|---|
| **1** | Tìm hiểu đề tài, phân tích yêu cầu và xác định kiến trúc hệ thống tổng thể. | 7 | 3/3/2025 | ✓ |
| **2** | Thiết kế sơ đồ khối chức năng và luồng dữ liệu hệ thống. | 4 | 10/3/2025 | ✓ |
| **3** | Nghiên cứu Raspberry Pi, camera và các thư viện nhận diện khuôn mặt. | 5 | 10/3/2025 | ✓ |
| **4** | Xây dựng chức năng nhận diện khuôn mặt cơ bản trên Raspberry Pi. | 10 | 15/03/2025 | ✓ |
| **5** | Phát triển chức năng thêm/xóa nhân viên trên hệ thống. | 7 | 25/03/2025 | ✓ |
| **6** | Thiết kế và xây dựng giao diện hiển thị trên Raspberry Pi. | 8 | 1/4/2025 | ✓ |
| **7** | Kiểm thử giao diện hiển thị và kết nối với chức năng nhận diện. | 5 | 9/4/2025 | ✓ |
| **8** | Thiết kế và lập trình trang web hiển thị dữ liệu chấm công. | 9 | 14/04/2025 | ✓ |
| **9** | Kết nối web với cơ sở dữ liệu MySQL, xây dựng chức năng tải file chấm công. | 5 | 23/04/2025 | ✓ |
| **10** | Kiểm thử hệ thống giao tiếp giữa Raspberry Pi và giao diện web. | 5 | 28/04/2025 | ✓ |
| **11** | Triển khai truy cập từ xa trang web bằng Cloudflare Tunnel. | 3 | 3/5/2025 | ✓ |

| | | | | |
|---|---|---|---|---|
| **12** | Thêm tính năng lưu dữ liệu vào CSV cục bộ (sao lưu dự phòng). | 2 | 4/5/2025 | ✓ |
| **13** | Đồng bộ dữ liệu từ CSV sang MySQL khi mất kết nối Internet. | 2 | 4/5/2025 | ✓ |
| **14** | Cấu hình database bằng dotenv tách riêng trong file .env_file_mysql. | 1 | 5/5/2025 | ✓ |
| **15** | Xây dựng tính năng đăng nhập cho phép Admin thao tác thêm/xóa/chỉnh sửa trên giao diện Tkinter. | 3 | 5/5/2025 | ✓ |
| **16** | Thử nghiệm toàn hệ thống trong điều kiện thực tế. | 7 | 8/5/2025 | |
| **17** | Kiểm tra, tối ưu, sửa lỗi và cải tiến nếu cần. | 5 | 14/05/2025 | |
| **18** | Viết báo cáo tổng kết. | 10 | 19/05/2025 | |
| **19** | Chạy thử nghiệm chính thức, hoàn tất sản phẩm. | 4 | 25/05/2025 | |
| **20** | Hoàn thiện mô hình trình bày sản phẩm. | 5 | 29/05/2025 | |