**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

BÁO CÁO

PBL5 - ĐỒ ÁN KỸ THUẬT MÁY TÍNH

# ĐỀ TÀI : XÂY DỰNG HỆ THỐNG MỞ CỬA SỬ DỤNG NHẬN DIỆN KHUÔN MẶT

**Cán bộ doanh nghiệp hướng dẫn: ThS. Lê Văn Khanh-Paradox**

**Giảng viên hướng dẫn: TS. Trần Thế Vũ**

| NHÓM: 01TT<br>HỌ VÀ TÊN SINH VIÊN | LỚP HỌC PHẦN ĐỒ ÁN |
|---|---|
| Nguyễn Đức Duy | 20.Nh10A |
| Nguyễn Quốc Tĩnh | 20.Nh10A |
| Trần Văn Luýt | 20.Nh10A |
| Phan Phú Quý | 20.Nh10A |

ĐÀ NẴNG, 05/2023

# TÓM TẮT ĐỒ ÁN

Hiện nay, trên thị trường có nhiều sản phẩm về mở khoá cửa sử dụng nhận diện khuôn mặt nhưng điểm chung hầu hết các sản phẩm như vậy đều có giá thành khá cao. Do đó, nhóm chúng em đã đề xuất và tiến hành xây dựng đồ án với đề tài "Hệ thống mở cửa sử dụng nhận diện khuôn mặt" với mong muốn tạo ra sản phẩm có thể phát hiện khuôn mặt, phân biệt mặt thật mặt giả và nhận diện để mở cửa với giá thành rẻ hơn. Nhóm đã sử dụng các công nghệ AI trong việc phát hiện, phân biệt thật giả và nhận diện khuôn mặt,Restful API bằng Python để thực hiện việc truyền nhận dữ liệu, ESP-32CAM trong việc chụp ảnh và React-Native để xây dựng một ứng dụng di dộng nhằm quản lý việc mở cửa. Tuy nhiên vẫn còn vài điểm thiếu sót nên hệ thống vẫn chưa hoàn hảo. Nhóm chúng em sẽ tiếp tục phát triển và hoàn thiện trong tương lai.

# BẢNG PHÂN CÔNG NHIỆM VỤ

| Sinh viên thực hiện | Các nhiệm vụ | Kết quả |
|---|---|---|
| Nguyễn Đức Duy | - Lắp đặt, thiết kế phần cứng của mô hình.<br>- Lập trình vi điều khiển esp32-cam và giao tiếp với web server.<br>- Sử dụng mô hình nhận dạng<br>- Đánh giá, kiểm thử. | Hoàn thành |
| Nguyễn Quốc Tĩnh | - Lập trình Web Server.<br>- Lập trình ứng dụng di động.<br>- Chụp ảnh, thu thập dữ liệu khuôn mặt.<br>- Đánh giá, kiểm thử. | Hoàn thành |
| Trần Văn Luýt | - Tìm hiểu và nghiên cứu các mô hình trí tuệ nhân tạo, học máy, học sâu để áp dụng vào hệ thống.<br>- Chụp ảnh, thu thập dữ liệu khuôn mặt.<br>- Đánh giá, kiểm thử. | Hoàn thành |
| Phan Phú Quý | - Chụp ảnh, thu thập dữ liệu khuôn mặt.<br>- Lập trình Web Server.<br>- Lập trình ứng dụng di động.<br>- Đánh giá, kiểm thử. | Hoàn thành |

**Bảng 1. Bảng phân công nhiệm vụ**

# MỤC LỤC

# DANH SÁCH HÌNH VẼ

# DANH SÁCH BẢNG BIỂU

# 1. GIỚI THIỆU

## 1.1. Thực trạng sản phẩm

Hiện nay, sản phẩm hệ thống mở cửa sử dụng nhận diện khuôn mặt đang được ứng dụng rộng rãi trong các tòa nhà, cơ quan, công ty, khách sạn và các khu vực công cộng khác. Một số thực trạng về sản phẩm này có thể được đề cập như sau:

- Độ chính xác: Công nghệ nhận diện khuôn mặt ngày càng được cải tiến để đạt được độ chính xác cao hơn. Tuy nhiên, công nghệ vẫn chưa phân biệt được giữa mặt thật và mặt giả mạo, lỗi này có thể được lợi dụng và gây nguy hiểm đến sự an toàn và bảo mật của người sử dung.
- Ứng dụng kèm theo: Chưa có nhiều hệ thống mở cửa nhận diện khuôn mặt hiện nay có kèm theo một ứng dụng di động giúp người dùng có khả năng kiểm soát việc đóng mở cửa, thống kê, gửi cảnh báo cho người dùng cũng như để thêm khuôn mặt.
- Giá thành: Giá thành cho sản phẩm còn tương đối đắt đỏ đối với một số cá nhân và tổ chức.

Với những thực trạng trên, chúng em quyết định phát triển hệ thống mở khoá cửa bằng nhận diện khuôn mặt có giá thành rẻ hơn, có thể phân biệt mặt thật so với mặt giả và thống kê số lần nhận diện thông qua ứng dụng kèm theo.

## 1.2. Các vấn đề cần giải quyết

Để làm ra được sản phẩm, cần phải giải quyết những vấn đề cũng như đạt được các yêu cầu sau đây:

- Cần các thiết bị phần cứng để thu nhận dữ liệu.
- Phát triển thuật toán để phát hiện, phân biệt được mặt thật và mặt giả, nhận diện khuôn mặt.
- Cần server để lưu trữ các ảnh cũng như để xử lý việc nhận diện khuôn mặt đồng thời xử lý các yêu cầu được gửi đến.
- Cần xây dựng một ứng dụng di động để quản lý hệ thống.

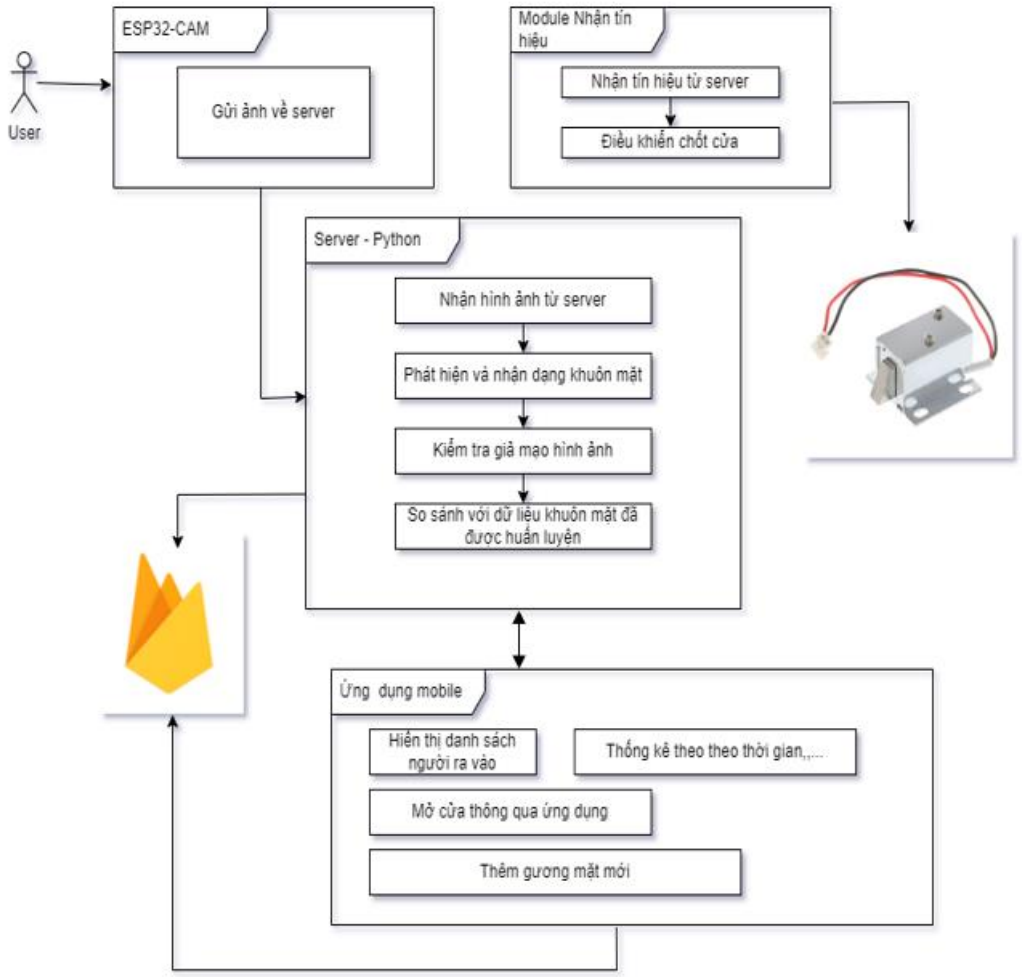
## 1.3. Đề xuất giải pháp tổng quan

| Vấn đề | Giải pháp |
|---|---|
| Phần cứng | - ESP-32CAM<br>- Khoá cửa điện từ<br>- Relay hw-803<br>- Điện thoại thông minh<br>- Cảm biến siêu âm |
| Phát hiện và nhận diện khuôn mặt | - Thư viện openCV, MTCNN, Facenet |
| Server | - Xây dựng server dùng Flask Python, lưu trữ bằng Firebase và SQLite.<br>- Thu thập dữ liệu từ các thiết bị.<br>- Xử lý, phân tích dữ liệu, gửi lại kết quả cho người dùng. |
| Ứng dụng hỗ trợ | - Xây dựng ứng dụng điện thoại bằng React-Native<br>- Cho phép quản lí hệ thống từ xa<br>- Nhận thông báo khi có sự xâm nhập không được phép |

**Bảng 2. Bảng giải pháp tổng quan**

# 2. GIẢI PHÁP

## 2.1. Tổng quan hệ thống




***Hình 1. Tổng quan hệ thống***

Hệ thống sử dụng ESP32-CAM để lấy hình ảnh, ứng dụng trên điện thoại để thực hiện việc quản lý đóng mở cửa cũng như để thêm khuôn mặt. Các thông tin về người dùng được lưu trữ trên Firebase và việc xử lý ảnh, lưu trữ ảnh được thực hiện trên server viết bằng Python. Thông qua kết nối mạng, điện thoại và ESP32-CAM có thể kết nối tới Server thông qua API.

## 2.2. Giải pháp phần cứng

Gồm có các linh kiện sau:

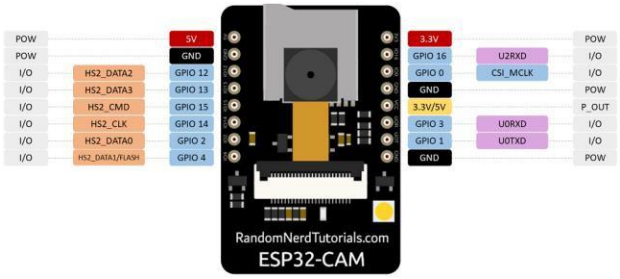
- ESP32-CAM:


***Hình 2. ESP32-CAM***

| Tên | Mô tả |
|---|---|
| Điện áp cung cấp | 5V |
| SPI Flash | Mặc định 32MB |
| RAM | 520KB SRAM + 4MB PSRAM |
| Bộ nhớ ngoài | Khe cắm thẻ micro SD lên đến 4GB |
| Bluetooth | Chuẩn Bluetooth 4.2 BR/EDR và BLE |
| WiFi | 802.11 b/g/n |
| Interface | UART, SPI, I2C, PWM |
| IO Port | 9 |
| Tốc độ truyền UART | 115200bps(Mặc định) |
| Camera | + Đầu nối FPC<br>+ Hỗ trợ camera OV2640(bán kèm theo board) hoặc camera OV7670<br>+ JPEG(chỉ hỗ trợ OV2640), BMP, GRAYSCALE<br>+ Đèn led |
| Dải quang phổ | 2412 ~2484MHz |
| Antenna | Onboard PCB antenna, gain 2dBi |
| Transmit Power | 802.11b: 17±2 dBm (@11Mbps)<br>802.11g: 14±2 dBm (@54Mbps)<br>802.11n: 13±2 dBm (@MCS7) |
| Receiving Sensitivity | CCK, 1 Mbps : -90dBm<br>CCK, 11 Mbps: -85dBm<br>6 Mbps (1/2 BPSK): -88dBm |

| | |
|---|---|
| | 54 Mbps (3/4 64-QAM): -70dBm<br>MCS7 (65 Mbps, 72.2 Mbps): -<br>67dBm |
| Tiêu thụ điện năng | + Tắt đèn flash: 180mA@5V<br>+ Bật đèn flash và bật độ sáng tối đa: 310mA@5V<br>+ Deep-sleep: 6mA@5V<br>+ Moderm-sleep: 20mA@5V<br>+ Light-sleep: 6.7mA@5V |
| Bảo mật | WPA/WPA2/WPA2-Enterprise/WPS |
| Nhiệt độ hoạt động | -20 ℃ ~ 85 ℃ |
| Môi trường bảo quản | -40 ℃ ~ 90 ℃ , < 90%RH |

**Bảng 3. Thông số kỹ thuật của ESP32-CAM**



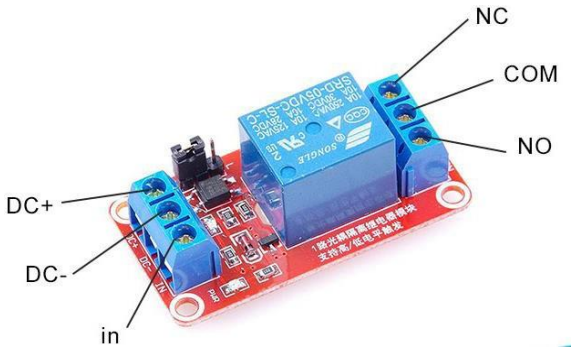

***Hình 3. Sơ đồ chân ESP32-CAM***

- Khóa cửa điện tử:

***Hình 4. Khóa cửa điện tử***

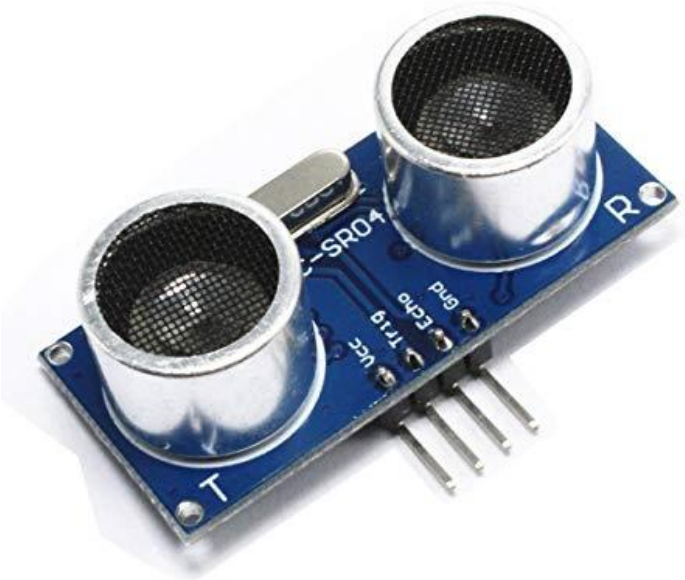
- Relay hw-803


*Hình 5. Relay hw-803*



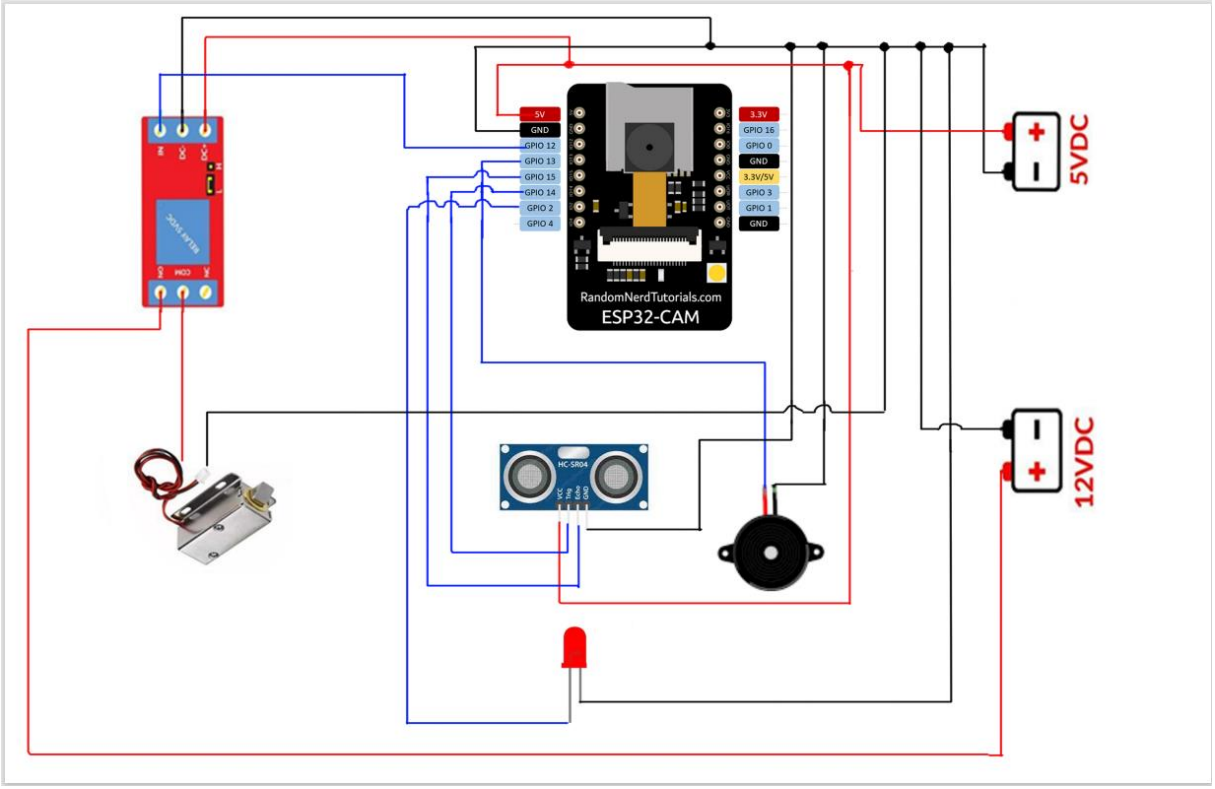

*Hình 6. Sơ đồ chân của relay hw-803*

- Cảm biến siêu âm HC-SR04:




*Hình 7. Cảm biến siêu âm*

Sơ đồ lắp mạch:



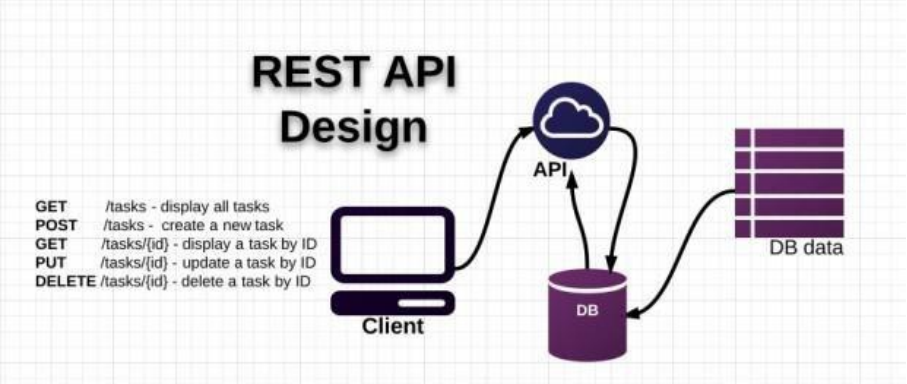

***Hình 8. Sơ đồ lắp mạch***

| STT | Tên | Giá |
|---|---|---|
| 1 | Esp32-Cam và đế nạp | 200,000 |
| 2 | Cảm biến siêu âm HC-SR04 | 20,000 |
| 3 | Chốt điện | 60,000 |
| 4 | Module Relay 12V-10A | 18,000 |
| 5 | Dây điện | 20,000 |
| 6 | Đèn led, loa mini | 10,000 |
| 7 | Tấm fomex trắng, keo dán | 110,000 |
| 8 | Dây micro USB | 50,000 |
| 9 | 3 pin sạc 3.7V | 45,000 |
| 10 | Hộp đế pin | 10,000 |
| 11 | Bộ sạc pin 1 cell 18500 | 19,000 |
| | | Tổng: 562,000 |

**Bảng 4. Bảng kê chi phí đồ án**

## 2.3. Giải pháp truyền thông

### 2.3.1. Restful API




***Hình 9. Mô hình RESTful API***

**API** (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà người dùng cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.

**RESTful API** là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau. **REST** hoạt động chủ yếu dựa vào giao thức HTTP :

- GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
- POST (CREATE): Tạo mới một Resource.
- PUT (UPDATE): Cập nhật thông tin cho Resource.
- DELETE (DELETE): Xoá một Resource.

### 2.3.2. Flask framework

Flask là một web framework nhỏ gọn và linh hoạt cho Python được sử dụng để xây dựng các ứng dụng web và RESTful API. Flask cung cấp các tính năng cơ bản để xử lý các yêu cầu HTTP, định nghĩa các routes (đường dẫn), xử lý các lỗi và trả về kết quả dưới dạng HTML, JSON, hoặc các định dạng khác.

Một số đặc điểm nổi bật của Flask:

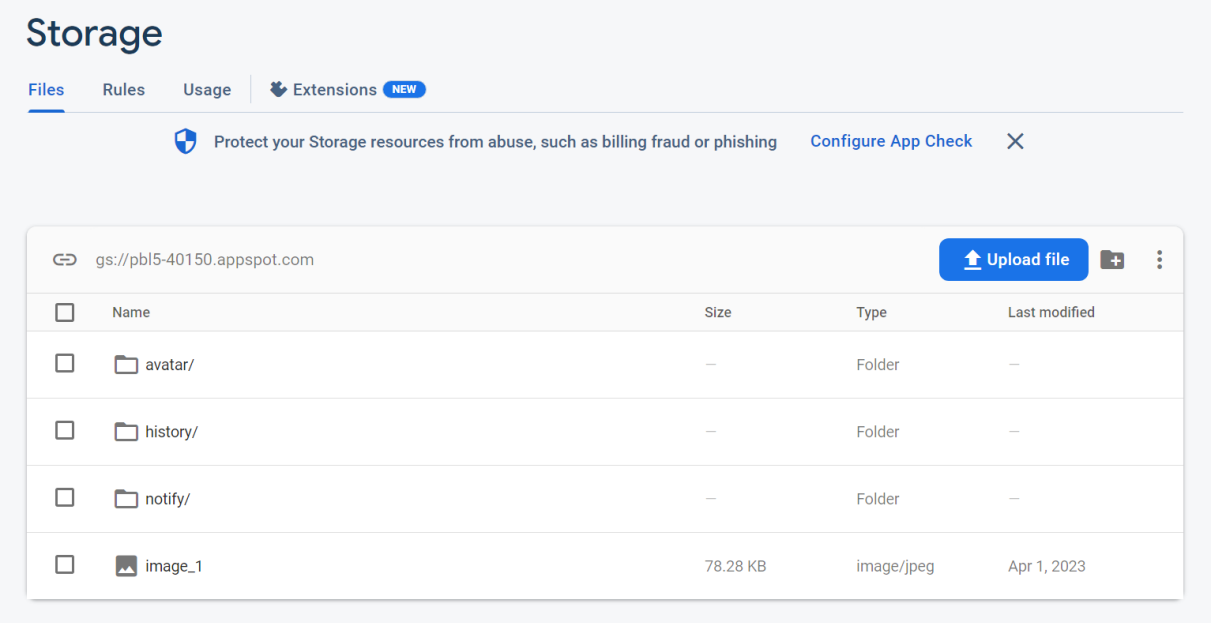
- Flask được thiết kế để linh hoạt và dễ dàng mở rộng. Nó không yêu cầu sử dụng các component hoặc thư viện cụ thể, giúp bạn tùy chỉnh ứng dụng theo nhu cầu của mình.
- Flask có cấu trúc code đơn giản, dễ đọc và dễ hiểu, giúp các lập trình viên tập trung vào phát triển các tính năng chính của ứng dụng.
- Flask có thư viện đa dạng và phong phú, cung cấp nhiều tính năng hữu ích như validation, authentication, caching, và nhiều thứ khác.
- Flask hỗ trợ nhiều kiểu kết nối cơ sở dữ liệu.

### 2.3.3. Lưu trữ dữ liệu sử dụng Firebase

Firebase là một nền tảng Backend-as-a-Service (BaaS) được phát triển bởi Google để giúp các nhà phát triển xây dựng các ứng dụng web và di động nhanh chóng và dễ dàng hơn. Firebase cung cấp các tính năng và dịch vụ như cơ sở dữ liệu thời gian thực, lưu trữ đám mây, xác thực người dùng, thông báo đẩy, phân tích và nhiều hơn nữa. Firebase cũng cung cấp các SDK cho các nền tảng phổ biến như Android, iOS, JavaScript, Unity và C++.

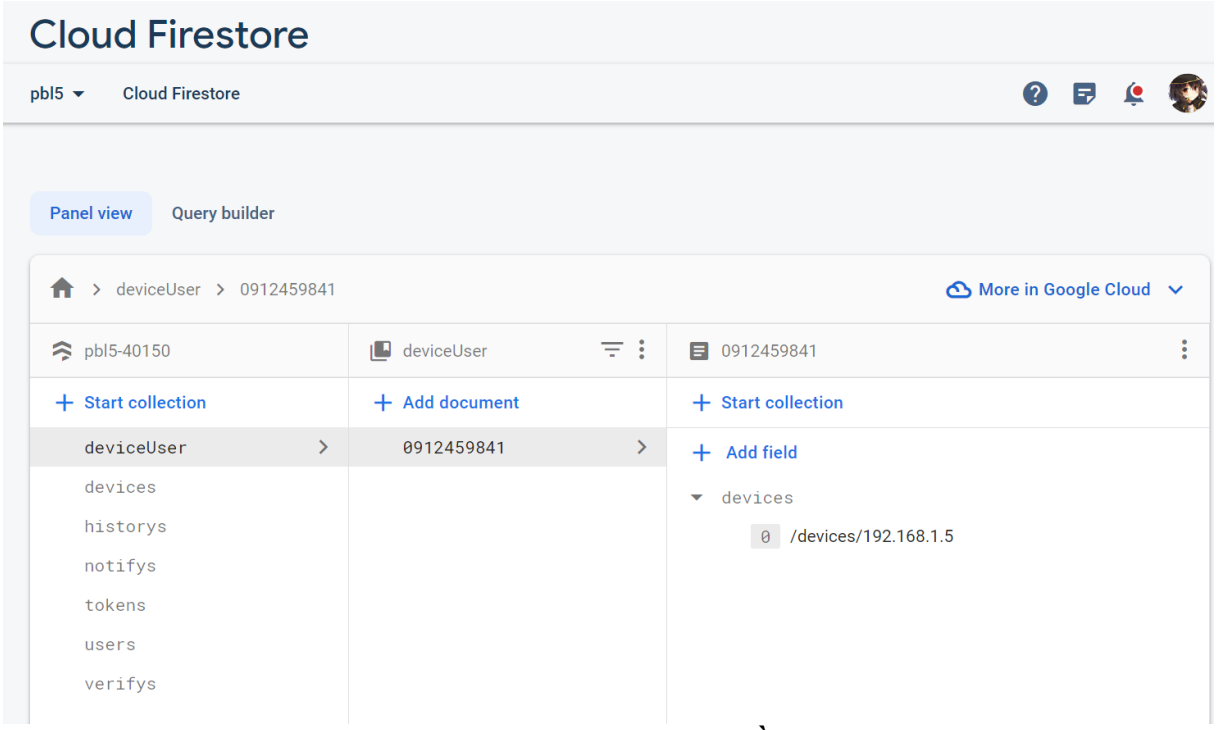
Các tính năng của Firebase được sử dụng trong đồ án:

- Firebase Storage: dịch vụ lưu trữ đám mây cho phép các ứng dụng lưu trữ và quản lý các tệp tin, hình ảnh, video và âm thanh.




***Hình 10. Firebase storage của đồ án***

- Firestore Database: là một cơ sở dữ liệu đám mây có cấu trúc tài liệu được phát triển bởi Google. Firestore được xây dựng trên cơ sở dữ liệu NoSQL và cho phép các nhà phát triển lưu trữ và truy vấn dữ liệu trong thời gian thực.




*Hình 11. Firestore Database của đồ án.*

## 2.4. Hệ thống nhận dạng và xử lí khuôn mặt

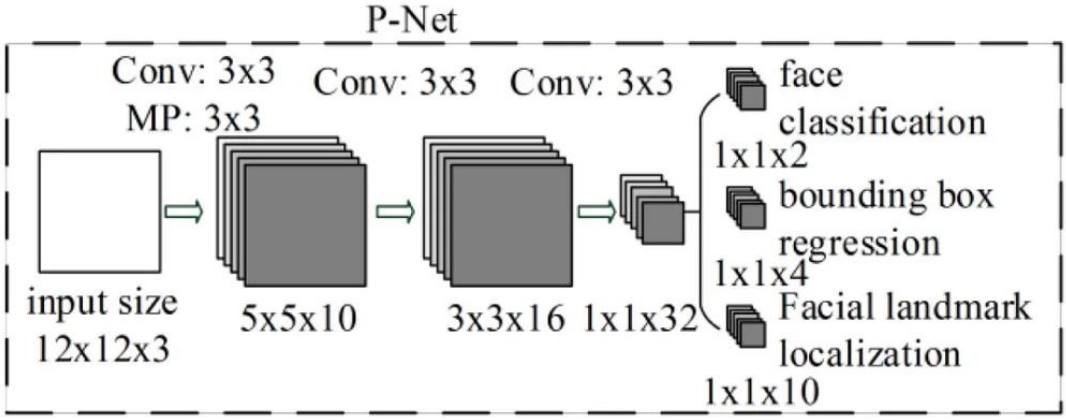
### 2.4.1. Phát hiện khuôn mặt sử dụng MTCNN

Về mặt cấu trúc MTCNN bao gồm 3 mạng CNN (Convolutional Neural Networks) xếp chồng và đồng thời hoạt động khi phát hiện và xác định khuôn mặt. Mỗi mạng CNN trong MTCNN có cấu trúc và vai trò khác nhau trong việc phát hiện khuôn mật. Kết quả dữ liệu đầu ra của MTCNN là véc-tơ đặc trưng biểu diễn cho vị trí khuôn mặt được xác định trong bức ảnh (mắt, mũi, miệng,...).

MTCNN hoạt động theo 3 bước với 3 mạng nơ-ron riêng cho mỗi bước (P-Net, R-Net và O-Net). Khi sử dụng, MTCNN sẽ cho phép tạo ra nhiều bản sao của hình ảnh đầu vào, với các kích thước khác nhau để làm dữ liệu đầu vào.

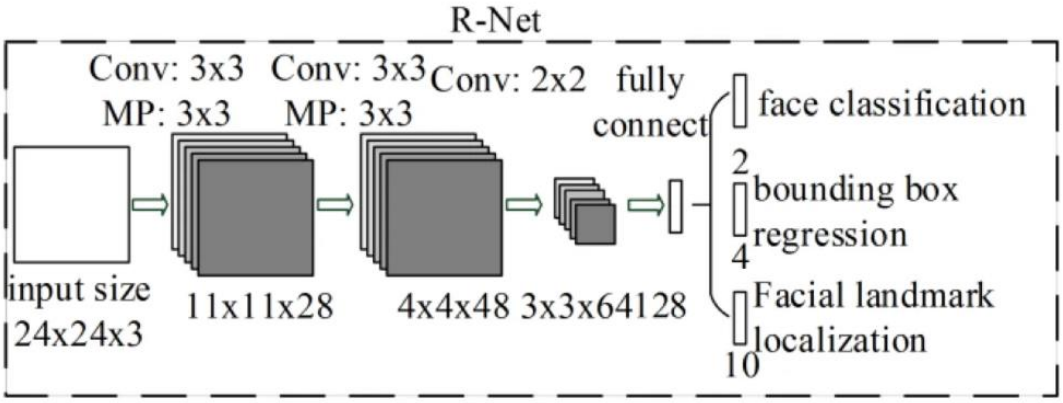
**Tầng 1 (P-Net):** Sử dụng mạng CNN, gọi là Mạng đề xuất (P-Net), để thu được các cửa sổ chứa khuôn mặt và các vectơ hồi quy trong các cửa sổ đó. Tiếp theo, các cửa sổ chứa khuôn mặt được hiệu chuẩn dựa trên các vector hồi quy. Cuối cùng, những cửa

sổ xếp chồng nhau tại một vùng được hợp nhất thành một cửa sổ. Kết quả đầu ra là các cửa sổ có thể chứa khuôn mặt.



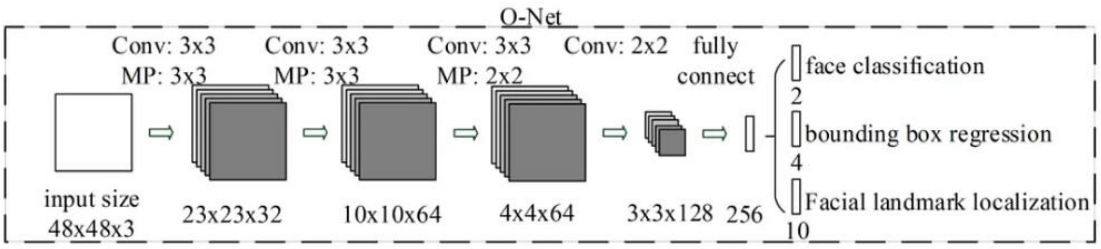

***Hình 12. Lớp P-Net của mạng MTCNN***

**Tầng 2 (R-Net):** Tất cả các cửa sổ chứa khuôn mặt từ tầng 1 sẽ được sàng lọc bằng cách đưa vào một CNN khác gọi là Mạng lọc (R-Net) để tiếp tục loại bỏ một số lượng lớn các cửa sổ không chứa khuôn mặt. Sau đó, thực hiện hiệu chuẩn với véc-tơ hồi quy và thực hiện hợp nhất các cửa sổ xếp chồng nhau tại một vùng.




***Hình 13. Lớp R-Net của MTCNN***

**Tầng 3 (O-Net):** Tầng này tương tự như tầng 2, sử dụng CNN chi tiết nhất được gọi là Mạng đầu ra (O-Net) để lọc kết quả một lần nữa và đánh dấu vị trí năm điểm chính trên khuôn mặt, bao gồm 2 mắt, 1 mũi, 2 bên cánh môi và điểm confident của mỗi box.




***Hình 14. Lớp O-Net của mạng MTCNN***

## 2.4.2. Nhận diện khuôn mặt sử dụng Facenet

FaceNet là một thuật toán hỗ trợ cho việc nhận dạng và phân cụm khuôn mặt FaceNet sử dụng một mạng CNN và cho phép giảm số chiều dữ liệu của véc-tơ đặc trưng (thường sử dụng là 128 chiều). Do đó, cho phép tăng tốc độ huấn luyện và xử lý mà độ chính xác vẫn được đảm bảo. Đối với thuật toán FaceNet, hàm loss function sử dụng hàm triplet loss cho phép khắc phục hạn chế của các phương pháp nhận dạng trước đây, quá trình huấn luyện cho phép học được đồng thời: Sự giống nhau giữa hai bức ảnh (nếu hai bức ảnh cùng một lớp) và sự khác nhau giữa hai bức ảnh (nếu chúng không cùng một lớp).

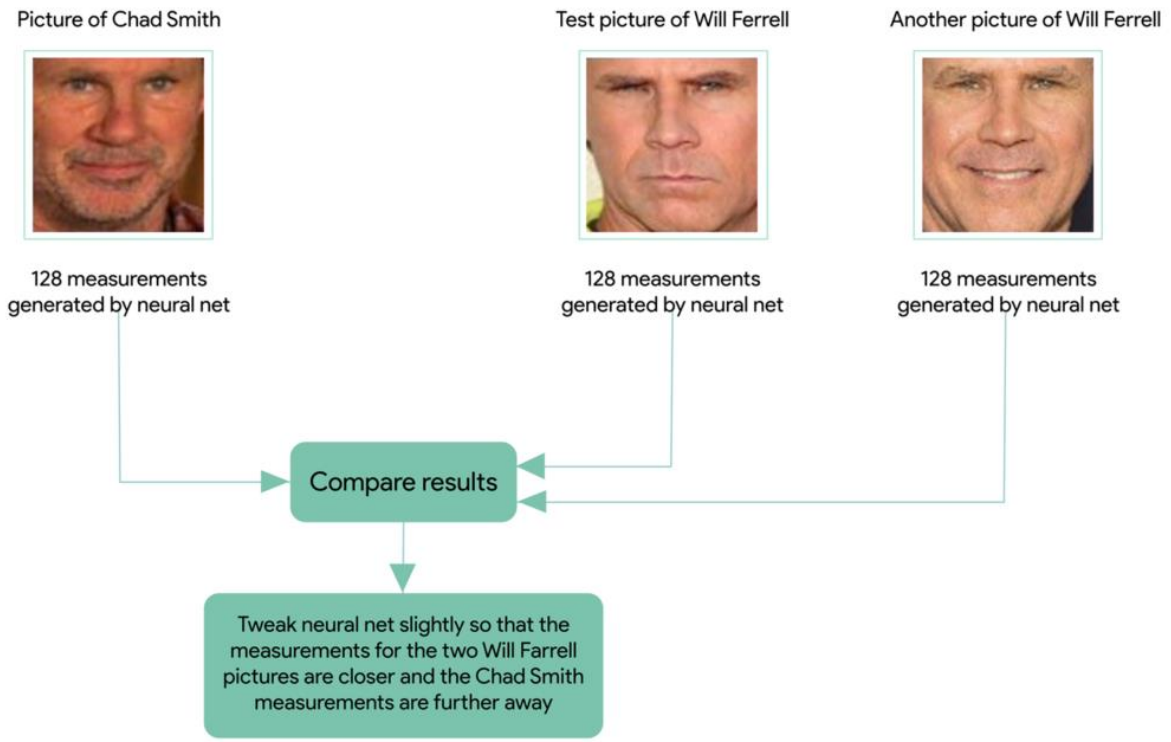
FaceNet chính là một dạng Siam network thường biểu diễn véc-tơ đặc trưng của các bức ảnh trong một không gian Euclidean n chiều (thường là 128 chiều). Việc biểu diễn thường tuân theo quy tắc: Nếu khoảng cách giữa các véc-tơ embedding càng nhỏ, thi mức độ tương đồng giữa chúng càng lớn và ngược lại. Tập hợp véc-tơ này sẽ là dữ liệu đầu vào cho hàm loss function để đánh giá chỉ số khoảng cách giữa các véc-tơ.

FaceNet sử dụng CNN bằng cách dùng hàm f(x) và nhúng hình ảnh X vào không gian Euclidean d chiều sao cho khoảng cách giữa các hình ảnh của 1 người không phụ thuộc vào điều kiện bên ngoài, khoảng cách giữa các khuôn mặt giống nhau (cùa cùng một người là nhỏ) trong khi khoảng cách giữa các ảnh khác nhau sẽ có khoảng cách lớn.

Hàm $f(x) \in R^d$ có chức năng biểu diễn ảnh x vào không gian Euclidean d chiều. Tại đây, sẽ có 3 bức ảnh là: Anchor (ảnh gốc), Positive (ảnh gần giống với ảnh gốc) và Negative (ảnh khác với ảnh gốc). Sau khi biểu diễn vào không gian Euclide thì tương ứng với 3 bức ảnh trên là $X_a$, $X_p$ và $X_n$. Để nhận dạng tốt thì khoảng cách từ $X_a$ tới $X_p$ sẽ phải nhỏ hơn khoảng cách từ $X_a$ tới $X_n$ .

Hàm triplet loss sẽ có dạng như sau:

$$L(Xa, Xp, Xn) = \sum ||f(Xa) - f(Xp)||_2^2 - ||f(Xa) - f(Xn)||_2^2 + g$$




***Hình 15. Minh họa bộ ba sai số***

Khi huấn luyện mô hình Siam network với triplet loss cần phải xác định trước cặp ảnh ($X_a$, $X_p$) thuộc về cùng một người. Ảnh $X_n$ là ảnh khác với ảnh gốc của người đó thường sẽ được lựa chọn ngẫu nhiên từ các bức ảnh thuộc các lớp còn lại. Do đó, tập hợp ảnh $X_n$ thường được thu thập nhiều hơn 1 bức ảnh/1 người để có thể chuẩn bị được tập dữ liệu huấn luyện. Nếu 1 người chỉ có 1 ảnh thì có thể đưa những tập dữ liệu như vậy làm bộ ảnh $X_n$ khi huấn luyện.

Như đã nêu trên có thể thấy, khi sử dụng triplet loss vào các mô hình CNN có thể tạo ra các véc-tơ đặc trưng tốt nhất cho mỗi một bức ảnh. Các véc-tơ đặc trưng này sẽ cho phép phân biệt rõ các ảnh Negative (ảnh khác với ảnh gốc) rất giống ảnh Positive (ảnh gần giống với ảnh gốc). Hơn nữa, khoảng cách giữa các bức ảnh thuộc cùng một lớp sẽ trở nên gần nhau hơn trong không gian chiếu Euclidean.

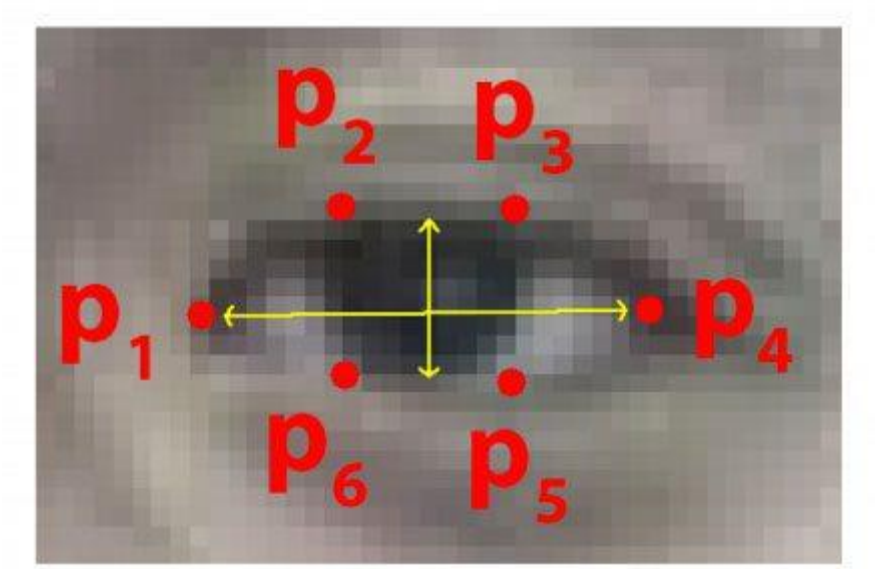
Tuy vậy, việc sử dụng bộ ba như trên sẽ khiến cho quá trình hội tụ chậm. Do đó, cần chọn bộ ba thích hợp trong quá trình huấn luyện để cải thiện được hiệu suất và độ chính xác của mô hình. Để khắc phục được việc hội tụ chậm, thường sẽ chọn bộ ba sai số sao cho khoảng cách giữa ảnh gốc và ảnh gần với ảnh gốc (ảnh của cùng 1 người) là lớn nhất và khoảng cách giữa ảnh gốc và ảnh của người khác là gần nhất.

### 2.4.3. Phát hiện giả mạo bằng nhận diện nháy mắt

Mặc dù đã có thể phát hiện và nhận dạng khuôn mặt, hệ thống vẫn có thể bị đánh lừa bằng cách sử dụng những hình ảnh đã được chụp sẵn của người dùng. Vì vậy, nhóm quyết định khắc phục điểm yếu này bằng cách sử dụng nhận diện nháy mắt để phát hiện giả mạo.

Để xây dựng bộ nhận diện nháy mắt, chúng ta cần tính toán một đại lượng gọi là eye aspect ratio (EAR), được giới thiệu bởi Soukupová và Čech trong bài báo cáo khoa học năm 2016 của họ, "Phát hiện nháy mắt thời gian thực bằng cách sử dụng các điểm đánh dấu khuôn mặt". Phương pháp này để phát hiện nháy mắt nhanh chóng, hiệu quả và dễ dàng triển khai.

Mỗi con mắt được thể hiện bởi 6 tọa độ (x,y), bắt đầu từ khóe bên trái của mắt, chạy dần theo chiều kim đồng hồ, được thể hiện dưới hình sau:




***Hình 16. 6 tọa độ đánh dấu của mắt***

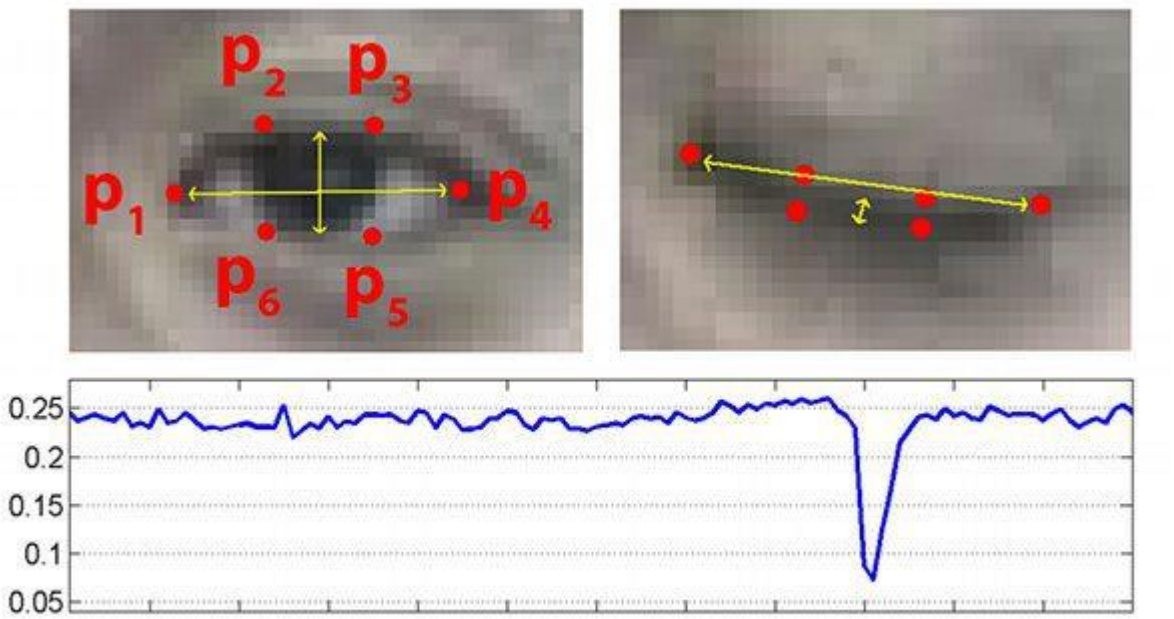
Dựa trên những điểm này, ta có phương trình EAR thể hiện mối quan hệ giữa chiều rộng và chiều cao của các tọa độ này:

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

*Hình 17. Phương trình EAR*

Tử số của phương trình này tính khoảng cách giữa các tọa độ mắt theo chiều dọc, trong khi mẫu số tính khoảng cách giữa các tọa độ mắt theo chiều ngang, đồng thời định trọng số cho mẫu một cách phù hợp vì chỉ có một bộ điểm ngang nhưng có hai bộ điểm dọc.

Theo như bài báo chỉ ra, giá trị của EAR gần như không đổi khi mắt còn đang mở, nhưng sẽ nhanh chóng giảm xuống 0 khi xảy ra việc nháy mắt. Đồ thị sau sẽ trực quan giá trị EAR khi mở mắt và nháy mắt:




*Hình 18. Đồ thị trực quan giá trị EAR*

Sử dụng phương trình đơn giản này, chúng ta có thể tránh các kỹ thuật xử lý hình ảnh và chỉ cần dựa vào tỷ lệ khoảng cách giữa các tọa độ mắt để xác định xem một người có nháy mắt hay không.

## 2.5. Giải pháp ứng dụng di động

### 2.5.1. Ý tưởng

Xây dựng ứng dụng di động để để tương tác với server để quản lí việc đóng mở cửa. Ứng dụng sẽ có 2 tác nhân chính là "Người dùng chủ" và "Người dùng".

Người dùng có quyền xem lịch sử đóng mở cửa, xem tên những người mình đã thêm khuôn mặt, thêm, xóa khuôn mặt, xem trạng thái cửa, đóng, mở của từ xa thông qua Internet, đóng, mở của ở gần thông qua Bluetooth, có quyền xem thông báo về các hành vị xâm nhập bất hợp pháp, có quyền thêm người dùng mới vào hệ thống.

Người dùng chủ là người có tất cả tính năng của người dùng, tuy nhiên lại có thêm đặc quyền là có thể xem những người dùng phụ thuộc và xóa người dùng phụ thuộc họ.

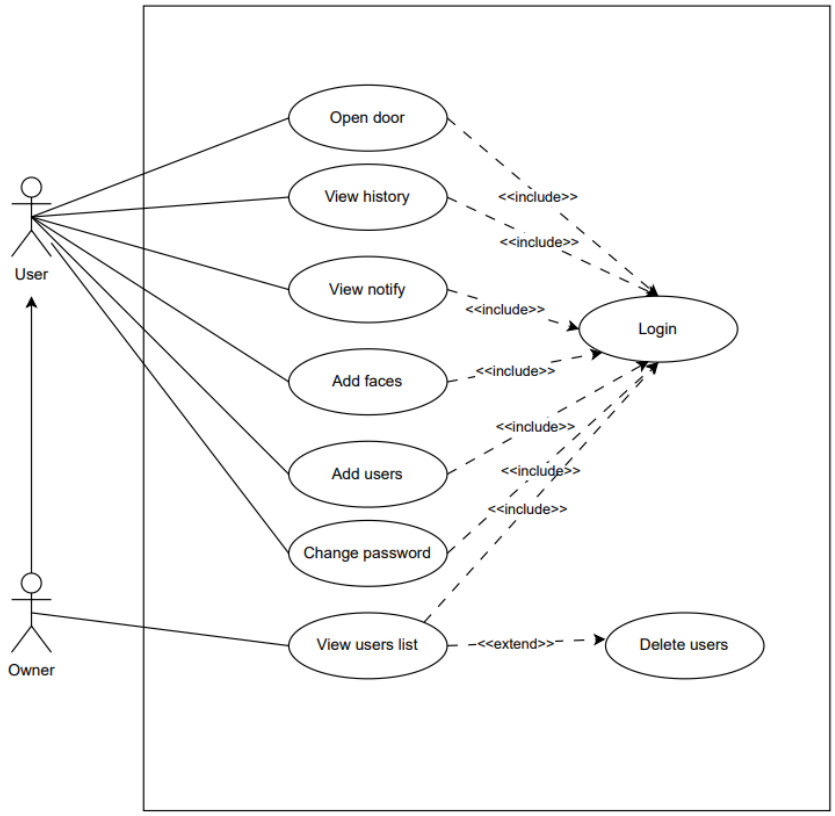
### 2.5.2. React Native

React Native là một framework được phát triển bởi Facebook để xây dựng các ứng dụng di động cho Android và iOS bằng việc sử dụng các công nghệ web như React, JavaScript và CSS.

Một số đặc điểm nổi bật của React Native:

- React Native cho phép viết một lần và chạy được trên cả hai hệ điều hành Android và iOS, giúp tiết kiệm thời gian và chi phí phát triển ứng dụng.
- React Native sử dụng kiến trúc dựa trên components để xây dựng giao diện người dùng. Các components có thể được tái sử dụng và kết hợp với nhau để tạo ra các giao diện phức tạp.
- React Native có thư viện cộng đồng phong phú và đa dạng, cung cấp nhiều module hữu ích như Redux, React Navigation, React Native Elements, và nhiều thứ khác.

### 2.5.3. Biểu đồ usecase ứng dụng




***Hình 19. Biểu đồ usecase của ứng dụng***

# 3. KẾT QUẢ

## 3.1. Kết quả nhận diện

### 3.1.1. Phát hiện khuôn mặt

**a. Tập dữ liệu**

- Trong phần huấn luyện mô hình phát hiện khuôn mặt, nhóm sử dụng tập dữ liệu được thu thập từ nhiều nguồn. Thông tin tập dữ liệu:

https://drive.google.com/drive/folders/1G7fkdsXkYo_hM53g8p6EhXGhjPwDVcf9

+ Mô tả: Tập dữ liệu có 1486 mẫu là có khuôn mặt, 4948 mẫu là ảnh tự nhiên không có mặt người.

**b. Huấn luyện**

- Sau khi có được dữ liệu và phân chia dữ liệu thích hợp, nhóm đã thực hiện cài đặt mô hình MTCNN. Sau thực hiện đánh giá kết quả trên tập dữ liệu, nhóm thấy được độ chính xác là cao và không cần thiết thực hiện quá trình huấn luyện lại.

- Tuy nhiên, nhóm có tinh chỉnh một số tham số về kích thước khuôn mặt nhỏ nhất, ngưỡng và hệ số trượt để đảm bảo mô hình phù hợp với đề tài.

**c. Kết quả giải pháp phát hiện khuôn mặt**

- Kết quả định tính:

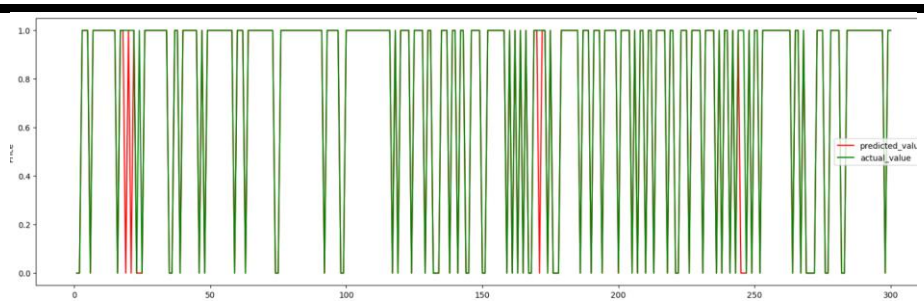
+ Khi camera có độ phân giải thấp, rất khó phát hiện ở khoảng cách lớn.

+ Vẫn còn phát hiện nhầm từ không có khuôn mặt thành có khuôn mặt.

- Kết quả định lượng:

+ Nhóm đã sử dụng hàm accuracy_score của thư viện scikit-learn để đo lường độ chính xác của mô hình phân loại khuôn mặt.

+ Độ chính xác phát hiện khuôn mặt trên tập dữ liệu là 97,20%.




*Hình 20. Đồ thị biểu diễn chính xác phát hiện khuôn mặt*

### 3.1.2. Nhận diện khuôn mặt

**a. Tập dữ liệu**

- Tập dữ liệu hình ảnh thu được từ nhiều nguồn, thông tin tập dữ liệu https://drive.google.com/drive/folders/15E6y28VC_d37Ee94pkZEvr5ZGNksMM5P

+ Mô tả: Gồm 166 người khác nhau về giới tính và có tổng cộng 4424 ảnh được phân thành 2 tập train và test

**b. Huấn luyện**

- Nhóm thực hiện crop và resize ảnh về kích thước thích hợp cho mô hình mtcnn.
- Nhóm thực hiện cài đặt mô hình FaceNet, sử dụng kiến trúc mạng neural network từ tập dữ liệu pretrain để trích xuất đặc trưng khuôn mặt sau đó sử dụng đặc trưng này để huấn luyện một mô hình phân loại khuôn mặt với tham số batch_size được sử dụng ở đây là 1000
- Quá trình huấn luyện được thực hiện trên tập dữ liệu đã xử lí ở trên.
- Để thực hiện việc huấn luyện dữ liệu cần nhận dạng, ta dùng facenet tải mạng neural được lưu ở Models/20180402-114759.pb để cài đặt mô hình trích xuất đặc trưng. Sau đó lưu dữ liệu sau khi trích xuất đặc trưng vào Models/facemodel.pkl.
- Để nhận khuôn mặt ta cần dùng facenet tải mạng neural được lưu ở Models/-20180402-114759.pb, sau đó sử dụng dữ liệu sau khi train được lưu ở Models/- facemodel.pkl để nhận dạng khuôn mặt.

**c. Kết quả giải pháp nhận hiện khuôn mặt**

- Kết quả của mô hình phân loại khuôn mặt

```
Accuracy_train: 100%
Accuracy_test: 88%
```

*Hình 21. Độ chính xác của mô hình khi đánh giá*

### 3.1.3. Nhận dạng nháy mắt

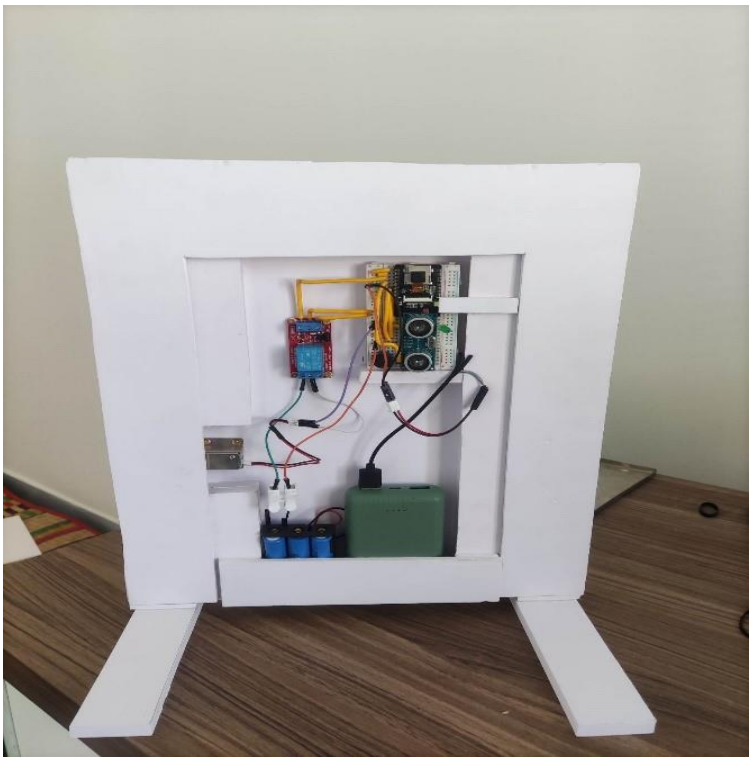
- Nhận hình ảnh từ camera , sử dụng model shape_predictor_68_face_landmarks.dat để xác định 68 điểm đặc trưng khuôn mặt, sau đó xác định điểm từ 36-42 là mắt trái, từ 42-48 là mắt phải, cắt vùng chứa mắt dựa vào các điểm này.

- Sử dụng model 2018_12_17_22_58_35.h5 để nhận dạng xem mắt đang nhắm mắt hay mở mắt, nếu ngưỡng xác định >0.1 thì mắt đang mở.

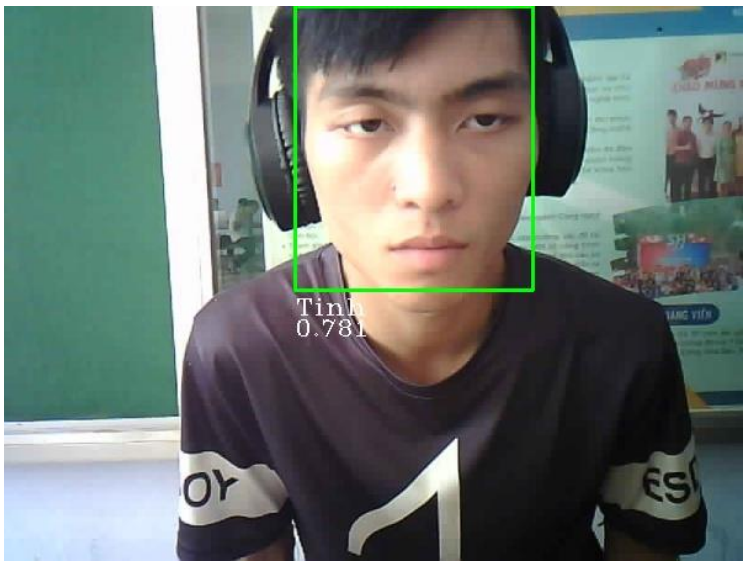
- Sau đó, chương trình tính toán xác suất trung bình của các dự đoán trong lịch sử và lưu trữ lịch sử dự đoán. Nếu xác suất trung bình nhỏ hơn một ngưỡng, chương trình sẽ tăng biến đếm số lần chớp mắt lên 1 và lưu thời gian của lần chớp mắt cuối cùng vào biến.

- Nếu số lần chớp mắt từ 2 trở và thời gian giữa hai lần chớp mắt là nhỏ hơn hoặc bằng 2 giây, chương trình sẽ xác định đó là một mặt thật. Nếu không có chớp mắt nào được phát hiện trong 5 giây thì chương trình xác định đó là mặt giả và reset lại biến đếm.

## 3.2. Kết quả mô hình



*Hình 22. Mô hình cửa*




*Hình 23. Kết quả nhận dạng khuôn mặt*

## 3.3. Server API

| | |
|---|---|
| **Tên chức năng** | Thêm người dùng |
| **URL** | /users/addUser |
| **Method** | POST |
| **Parameters** | “phone”, “name”, “phoneOwner” |

**Bảng 5. Bảng mô tả API thêm người dùng**

| | |
|---|---|
| **Tên chức năng** | Xóa người dùng |
| **URL** | /users/deleteUser |
| **Method** | POST |
| **Parameters** | “phone” |

**Bảng 6. Bảng mô tả API xóa người dùng**

| | |
|---|---|
| **Tên chức năng** | Thêm người dùng đã tồn tại |
| **URL** | /users/addUserExists |
| **Method** | POST |
| **Parameters** | “phone”, “name”, “phoneOwner”, “verification” |

**Bảng 7. Bảng mô tả API thêm người dùng đã tồn tại**

| | |
|---|---|
| **Tên chức năng** | Cập nhật mật khẩu |
| **URL** | /users/updatePassword |
| **Method** | POST |
| **Parameters** | “phone”, “password”, “newPassword” |

**Bảng 8. Bảng mô tả API cập nhật mật khẩu**

| | |
|---|---|
| **Tên chức năng** | Gửi mã xác thực số điện thoại |
| **URL** | /users/resendVerifyCode |
| **Method** | POST |
| **Parameters** | “phone” |

**Bảng 9. Bảng mô tả API gửi mã xác thực SĐT**

| | |
|---|---|
| **Tên chức năng** | Xem danh sách khuôn mặt |
| **URL** | /face/<phone> |
| **Method** | GET |
| **Parameters** | “phone” |

**Bảng 10. Bảng mô tả API xem danh sách mặt**

| | |
|---|---|
| **Tên chức năng** | Thêm khuôn mặt |
| **URL** | /api/upload |
| **Method** | POST |
| **Parameters** | “phone”, “name”, “image”,”count” |

**Bảng 11. Bảng mô tả API thêm khuôn mặt**

| | |
|---|---|
| **Tên chức năng** | Xóa khuôn mặt |
| **URL** | /faces/<id> |
| **Method** | DELETE |
| **Parameters** | “id” |

**Bảng 12. Bảng mô tả API xóa mặt**

| | |
|---|---|
| **Tên chức năng** | Mở cửa |

| | |
|---|---|
| **URL** | /unlockDoor |
| **Method** | POST |
| **Parameters** | “phone”, “addressDoor” |

**Bảng 13. Bảng mô tả API mở cửa qua internet**

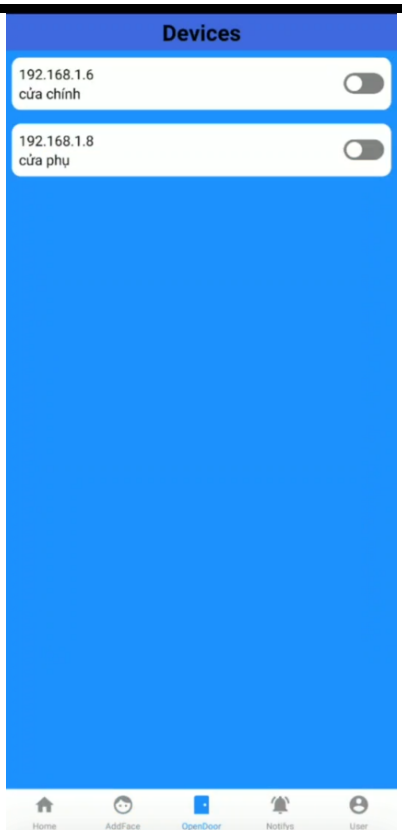
| | |
|---|---|
| **Tên chức năng** | Khóa cửa |
| **URL** | /lockDoor |
| **Method** | POST |
| **Parameters** | “phone”, “addressDoor” |

**Bảng 14. Bảng mô tả API khóa cửa**

## 3.4. Kết quả ứng dụng di động

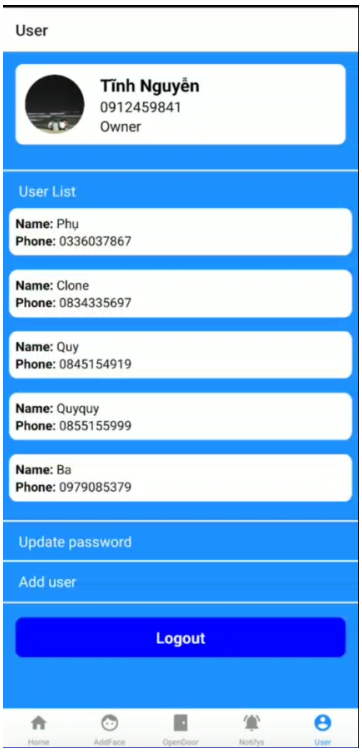

*Hình 24. Màn hình lịch sử mở cửa*


*Hình 25. Màn hình quản lý khuôn mặt*

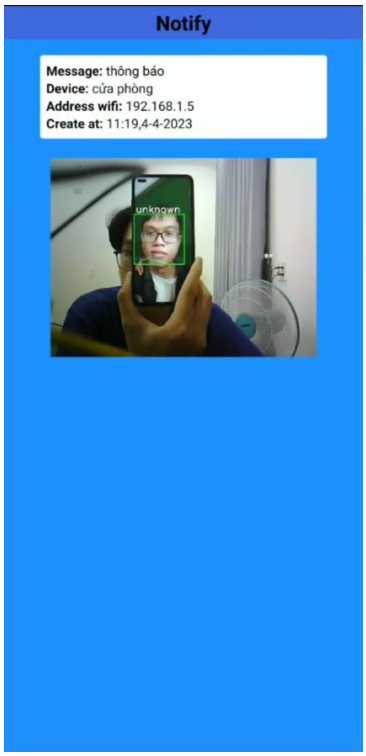



*Hình 26. Màn hình xem và mở cửa từ xa*

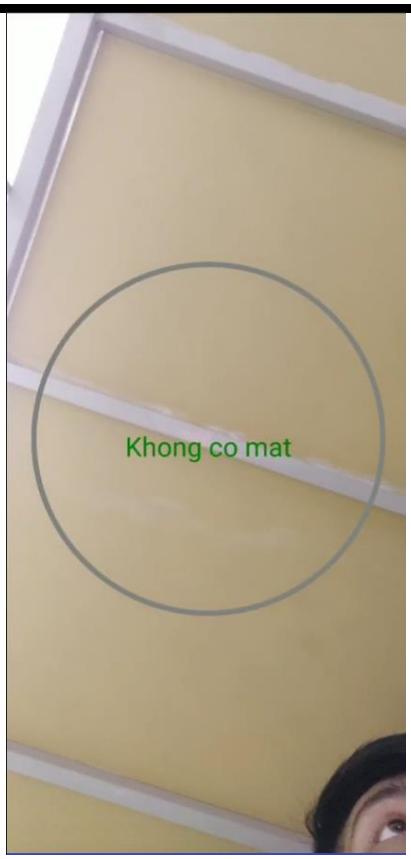

*Hình 27. Màn hình danh sách thông báo*



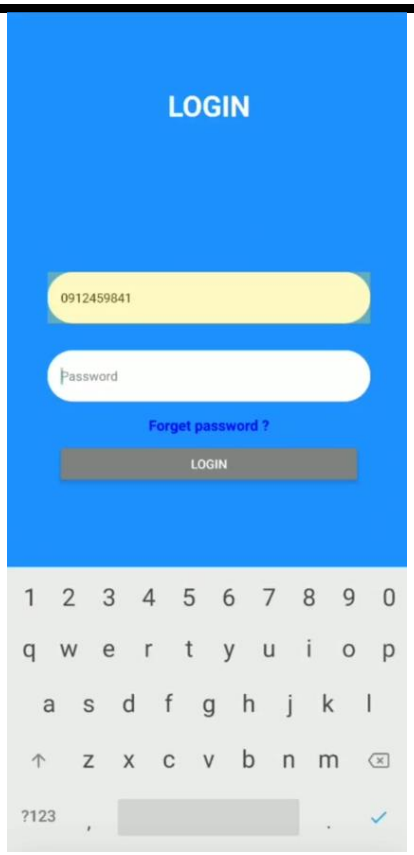

*Hình 28. Màn hình quản lý tài khoản*




*Hình 29. Màn hình thông báo*




*Hình 30. Màn hình thêm khuôn mặt*




*Hình 31. Màn hình đăng nhập*

# 4. KẾT LUẬN

## 4.1. Đánh giá

**Thiết kế:** Sản phẩm có thiết kế gọn gàng, trọng lượng nhẹ, có thể di chuyển được và có thể cải tiến trong tương lai về phần cứng và phần mềm.

**Chức năng:**

- **Phát hiện khuôn mặt:** Phát hiện khuôn mặt đúng, không nhầm vị trí
- **Nhận diện khuôn mặt:** Nhận diện với độ chính xác cao, ít xảy ra nhầm lẫn, tốc độ xử lí khá chậm
- **Nhận diện nháy mắt :** Độ chính xác không quá cao, đôi khi phải nháy nhiều hơn so với yêu cầu ban đầu mới có thể xác thực
- **Server:** Server hệ thống xử lý ổn định, gần như không xảy ra lỗi.
- **Ứng dụng di động:** Ứng dụng có giao diện bắt mắt, dễ sử dụng, đầy đủ các chức năng, phù hợp với yêu cầu đồ án

## 4.2. Hướng phát triển

**Nhận diện khuôn mặt:** Nhóm sẽ tiếp tục cải tiến mô hình để tăng tốc độ nhận dạng, đồng thời cần nghiên cứu và lựa chọn mô hình mang lại độ chính xác cao với tốc độ nhận dạng nhanh chóng

**Server:** Server mới chỉ được sử dụng trên số lượng request thấp, nhóm sẽ tìm hiểu cách để cải thiện khả năng xử lí của server, chế độ đa luồng. Đồng thời, tăng khả năng bảo mật cho hệ thống. Cải thiện hệ thống cơ sở dữ liệu để hệ thống vận hành tốt hơn, đáp ứng nhu cầu người dùng hơn

**Phần cứng:** Có thể thay đổi camera có độ phân giải cao hơn để cho phép lấy hình ảnh khuôn mặt tốt hơn. Thêm bộ nguồn có thể thông báo trình trạng pin của hệ thống để người sử dụng có thể theo dõi tình trạng pin.

**Ứng dụng quản lí:** Phát triển tính năng điều khiển thông qua Bluetooth phòng trường hợp vi điều khiển không thể kết nối mạng (mất điện, đổi wifi) hoàn thiện hơn.

# 5. TÀI LIỆU THAM KHẢO

**[1].** Soukupova, T., & Jan, C. (2016, May 3). Real-time eye blink detection using facial landmarks. https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

**[2].** Ly, N. T. X., & Hà, M. V. (2021, July 15). Ứng dụng Thuật Toán FACENET xây dựng hệ thống nhận dạng khuôn mặt. https://sti.vista.gov.vn/tw/Lists/TaiLieu-KHCN/Attachments/327760/CVv465V19S72021059.pdf

**[3].** Chiến Thắng, N. (2019, September 11). Nhận diện Khuôn Mặt Trong Video bằng MTCNN VÀ facenet. https://www.miai.vn/2019/09/11/face-recog-2-0-nhan-dien-khuon-mat-trong-video-bang-mtcnn-va-facenet/

**[4].** Santos, S. (2020, March 10). *ESP32-cam video streaming and face recognition with Arduino Ide*. Random Nerd Tutorials. https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/

**[5].** Introduction · REACT NATIVE. React Native RSS. (2023, March 17). https://reactnative.dev/docs/getting-started

**[6].** LFW - People (Face Recognition). https://www.kaggle.com/datasets/atulanandjha/lfwpeople?select=lfw-funneled.tgz