ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Đồ án Đa ngành - hướng trí tuệ nhân tạo (CO3107)

Project

HỆ THỐNG HỖ TRỢ MỞ KHÓA CỬA THÔNG MINH

GVHD: Trần Huy & Vũ Văn Tiến
Nhóm sinh viện thực hiện: Nguyễn Mậu Minh Đức - 2010230
Phạm Hoàng Đức Huy - 2011286
Quách Minh Đức - 2010231
Trần Tuấn Anh - 2010878

TP. HỒ CHÍ MINH, THÁNG 5 / 2023

Mục lục

1 Thành Viên và Khối lượng công việc

Full Name	**Student ID**	**Percentage of work**	**Note**
Nguyễn Mậu Minh Đức	2010230	100%	
Phạm Hoàng Đức Huy	2011286	100%	
Quách Minh Đức	2010231	100%	Leader
Trần Tuấn Anh	2010878	100%	

2 Document history

Date	Version	Changes	Person in charge
17/02	0.0	Giới thiệu tổng quan, danh sách các yêu cầu chức năng và phi chức năng, các thiết bị cần thiết.	All
23/02	0.1	Thêm mô tả kiến trúc hệ thống	Nguyễn Mậu Minh Đức, Quách Minh Đức
30/02	0.2	Thêm Use-case diagram	Phạm Hoàng Đức Huy, Trần Tuấn Anh
07/03	0.2	Thêm Use-case scenario, miêu tả chi tiết use case	Phạm Hoàng Đức Huy, Trần Tuấn Anh
13/03	0.3	Chỉnh sửa use-case scenario, functional requirements	Quách Minh Đức
16/03	1.0	Thêm hiện thực phần cứng và thiết kế Dashboard	Quách Minh Đức
17/03	1.1	Thêm thiết kế Entity relationship diagram và lược đồ cơ sở dữ liệu	Nguyễn Mậu Minh Đức, Phạm Hoàng Đức Huy, Trần Tuấn Anh
18/03	1.2	Thêm các thiết kế UI	Nguyễn Mậu Minh Đức, Phạm Hoàng Đức Huy, Trần Tuấn Anh
21/03	1.3	Thêm phần hiện thực cơ sở dữ liệu	Nguyễn Mậu Minh Đức, Phạm Hoàng Đức Huy, Trần Tuấn Anh
23/03	1.4	Thêm phần hiện thực Mobile App	Phạm Hoàng Đức Huy
24/03	1.5	Thêm phần Web API và kết nối với Cơ sở dữ liệu	Nguyễn Mậu Minh Đức
30/03	1.6	Bổ sung hiện thực phần cứng và Adafruit.io	Quách Minh Đức
08/04	1.7	Bổ sung hiện thực kết nối Adafruit.io với phía người dùng	All
25/04	1.8	Bổ sung hiện thực phần cứng và hiện thực module AI	Quách Minh Đức
06/05	2.0	Chỉnh sửa chi tiết lần cuối trước khi nộp bài	All

3 Giới thiệu tổng quan đồ án

Trong thời đại kỹ thuật số hiện nay, hệ thống IoT (Internet of Things - Mạng lưới internet vạn vật) đã trở thành một phần quan trọng và không thể thiếu trong cuộc sống của con người. IoT có thể giúp chúng ta kết nối, thu thập và phân tích dữ liệu từ nhiều thiết bị thông minh khác nhau, giúp chúng ta đưa ra quyết định thông minh và tối ưu hóa hoạt động của các hệ thống.

Trong cuộc sống hiện đại, các hệ thống IoT có mặt ở khắp mọi nơi, từ nhà ở cho đến các ngành công nghiệp và giao thông vận tải. Trong nhà, các hệ thống IoT giúp cho chúng ta quản lý năng lượng, ánh sáng, nhiệt độ, an ninh và các thiết bị gia đình khác. Ví dụ, hệ thống nhà thông minh có thể điều khiển việc mở và tắt đèn, máy lạnh, máy giặt và các thiết bị khác từ xa thông qua điện thoại di động hoặc máy tính bảng.

Nhận thấy tầm quan trọng của IoT trong đời sống, và kết hợp với sự cần thiết của việc tự động hóa của việc cho phép ra vào một khu vực hoặc một căn hộ. Trong đồ án này của nhóm ĐHA, nhóm quyết định lựa chọn "Hệ thống hỗ trợ mở khóa cửa thông minh" làm đề tài nghiên cứu.

Hình 1: Khóa cửa thông minh

Hệ thống hỗ trợ mở khóa cửa thông minh là một giải pháp công nghệ giúp quản lý ra vào một khu vực hoặc một căn hộ một cách an toàn, thuận tiện. Hệ thống này sử dụng công nghệ IoT để liên kết các thiết bị cảm biến, bộ điều khiển, phần mềm, camera, khóa cửa và các thiết bị ngoại vi khác để tạo ra một hệ thống cửa thông minh hiệu quả.

Ngoài ra, hệ thống này còn tích hợp nhiều tính năng thông minh như nhận diện khuôn mặt, quản lý thông tin người dùng và lịch sử ra vào, từ đó hỗ trợ giải quyết các vấn đề an ninh tại khu vực hoặc căn hộ một cách tốt hơn. Đây cũng là mục đích của nhóm đang muốn hướng đến.

4 Danh sách thiết bị

Sử dụng dây tín hiệu làm phương tiện truyền tải tín hiệu giữa các thiết bị đầu vào và đầu ra.

1. **Yolo:Bit**
   - Đặc điểm: Board điều khiển nhỏ gọn, hỗ trợ lập trình các dự án IoT quy mô nhỏ.
   - Ứng dụng: Xử lý logic cho toàn bộ hệ thống. Do board có tích hợp cả còi nên đảm nhiệm thêm việc phát âm thanh thông báo người dùng xác thực khuôn mặt thành công/thất bại hoặc cửa mở/đóng.
   - Số lượng thiết bị: 1
2. **Cảm biến ánh sáng**
   - Đặc điểm: Thiết bị đo lường được sử dụng để đo mức độ ánh sáng trong môi trường.
   - Ứng dụng: Đo cường độ ánh sáng môi trường từ đó quyết định bật/tắt bóng đèn hỗ trợ camera.
   - Số lượng thiết bị: 1
3. **Công tắc Relay**
   - Đặc điểm: Thiết bị điện tử hỗ trợ điều khiển mạch tự động.
   - Ứng dụng: Dùng để mô phỏng cho chốt khóa cửa
   - Số lượng thiết bị: 1
4. **Camera**
   - Đặc điểm: Thiết bị quang học có khả năng thu lại ánh sáng và ghi lại hình ảnh một cách chính xác.
   - Ứng dụng: Kiểm tra và xác nhận danh tính người được thu hình từ camera.
   - Số lượng thiết bị: 1
5. **Đèn 4 LED RGB**
   - Đặc điểm: loại đèn sử dụng bốn chip LED để phát sáng, mỗi chip LED có thể tạo ra ba màu sắc đỏ, xanh lá cây và xanh lam.
   - Ứng dụng: Làm bóng đèn flash cho camera trong trường hợp thiếu sáng.
   - Số lượng thiết bị: 1

5 Yêu cầu người dùng

Để viết user requirement cho hệ thống tự động thông minh, ta cần lấy ý kiến từ người dùng về các chức năng, tính năng mà họ mong muốn hệ thống phải đáp ứng được.

- Hệ thống phải cung cấp chế độ mở khóa tự động thông qua nhận diện khuôn mặt.
- Hệ thống phải cung cấp một cơ chế để người quản lí có thể mở từ xa trong trường hợp khuôn mặt không có trong dữ liệu.
- Hệ thống phải cung cấp chức năng quản lí hệ thống cho phép người quản lí theo dõi thông tin về những người có thể tự động ra vào cũng như về các hoạt động đóng/mở cửa trong ngày.
- Hệ thống đơn giản, điều khiển dễ dàng, trực quan.

6 Yêu cầu hệ thống

6.1 Functional Requirements

Hệ thống gồm 5 module chính như sau:

#	Module	Mô tả
1	Nhận và hiển thị dữ liệu từ thiết bị	Thu thập, theo dõi cường độ ánh sáng môi trường theo thời gian
2	Kiểm tra dữ liệu nhận được vượt quá ngưỡng cho phép	Bật/tắt đèn tự động dựa vào ngưỡng cường độ sáng thiết đặt trước
3	Điều khiển thiết bị	Điều khiển chốt cửa và còi báo mở/đóng cửa
4	Ghi nhận hoạt động	Ghi nhận lịch sử mở cửa (thời gian, người đi vào) và lịch sử nhận diện khuôn mặt
5	Ứng dụng Machine Learning	Nhận diện khuôn mặt

6.2 Non-Functional Requirements

Các vấn đề cần được thỏa mãn của hệ thống như sau:

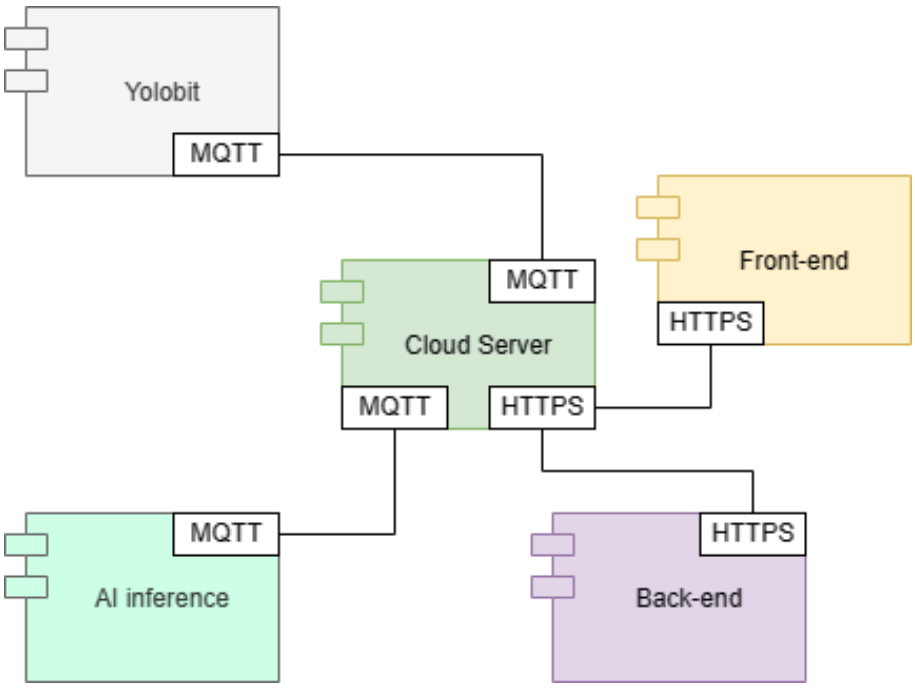
- Hiệu suất: Hệ thống nhận diện, hệ thống thống phản hồi hiệu quả, độ trễ thông báo không quá 2s.
- Tính dễ sử dụng: người dùng có thể sử dụng thành thục ứng dụng sau 30 phút luyện tập (ít hơn 4 lỗi).
- Kích thước: có thể nhận diện được tối đa 100 người, dữ liệu người dùng cũng được lưu trữ tương đương.
- Độ tin cậy: ít nhất là 95%.
- Bảo mật: có hệ thống xác thực, ủy quyền. Bên cạnh đó những dữ liệu phải được bảo mật tuyệt đối, phòng ngừa rò rỉ dữ liệu.
- Khả năng bảo trì: hiện thực dự án theo quy trình chuyên nghiệp, đảm bảo mã nguồn dễ dàng duy trì và nâng cấp trong tương lai.

7 Kiến trúc hệ thống

Phần kiến trúc hệ thống là rất quan trọng trong thiết kế hệ thống khóa cửa thông minh sử dụng nhận diện khuôn mặt. Phần này định nghĩa các thành phần chính của hệ thống, cách chúng tương tác với nhau để đáp ứng yêu cầu của người dùng. Với kiến trúc được xây dựng tốt, hệ thống có thể hoạt động ổn định, đảm bảo tính bảo mật và tăng hiệu suất.

Trong hệ thống khóa cửa thông minh này, việc sử dụng nhận diện khuôn mặt sẽ mang đến nhiều tiện ích cho người dùng, giúp tiết kiệm thời gian, tăng tính bảo mật và thuận tiện cho việc quản lý quyền truy cập. Tuy nhiên, để thực hiện điều này cần phải có một kiến trúc hệ thống phù hợp, đáp ứng các yêu cầu về tính năng, hiệu suất và bảo mật.

Hệ thống được thiết kế với các thành phần chính như Cloud server, Frontend, Backend và thiết bị xử lý Yolobit, tương tác với nhau một cách linh hoạt, đảm bảo tính bảo mật cao và tăng hiệu suất hoạt động. Kiến trúc hệ thống cũng đảm bảo khả năng mở rộng trong tương lai, giúp cho việc phát triển trở nên dễ dàng hơn..



Hình 2: Kiến trúc hệ thống

Các thành phần có thể được mô tả đơn giản như sau:

- Cloud Server: Chịu trách nhiệm lưu trữ, xử lý và quản lý dữ liệu về nhận dạng khuôn mặt và dữ liệu từ cảm biến. Sử dụng phương thức MQTT để kết nối với vi điều khiển Yolobit và HTTPS để kết nối với backend. Kiến trúc bên trong của module này bao gồm các thành phần chính sau:

  - Hệ thống quản lí lưu trữ: Dùng để quản lí việc lưu trữ thông tin về người dùng và các sự kiện như việc mở/khóa cửa, nắm vai trò điều hướng cho luồng dữ liệu đi xuống module Back-end.
  - Hệ thống xử lý dữ liệu: Dùng để xử lý dữ liệu từ các thiết bị IoT và cung cấp các API để truy cập dữ liệu.
  - Hệ thống giao tiếp với vi điều khiển: Dùng để kết nối với Yolobit thông qua giao thức MQTT để nhận dữ liệu từ các cảm biến và gửi lệnh điều khiển đến thiết bị đó.
  - Hệ thống giao tiếp với Back-end/ Front-end: Dùng để kết nối với Backend thông qua giao thức HTTPS để lấy dữ liệu và cập nhật thông tin người dùng.

- AI inference: Bộ phận chịu trách nhiệm thực hiện nhận diện khuôn mặt của người muốn đi vào và gửi thông tin về lần nhận diện lên server thông qua giao thức MQTT.
- Back-end: Phần mềm quản lý các thông tin về người dùng, quyền truy cập và các cửa cần được kiểm soát. Backend nhận dữ liệu từ Cloud Server và gửi thông tin cho Frontend.Phần này được xây dựng theo kiến trúc MVC (quy chuẩn trong quy trình phát triển phần mềm) nhằm giúp module đảo bảo tính bảo mật thống tin người dùng tốt nhất.
- Front-end: Giao diện người dùng của hệ thống, chịu trách nhiệm hiển thị các thông tin cần thiết và cho phép người dùng tương tác với hệ thống. Tương tự module Back-end trong việc sử dụng kiến trúc MVC. Thêm nữa module cho phép hiển thị trên nhiều nền tảng (Web/ App) để dễ dàng truy cập vào sử dụng, tăng tính dễ sử dụng của hệ thống.
- Hệ thống mạch phần cứng dùng Yolobit: Bộ phận xử lý logic cho toàn bộ thiết bị. Bộ phận này được kết nối với Cloud Server thông qua kết nối MQTT.

8 Use-case

8.1 Đặc tả

8.1.1 Nhận và hiển thị dữ liệu từ thiết bị

Use case name	Nhận và hiển thị dữ liệu từ thiết bị
Description	Thu thập, theo dõi cường độ ánh sáng môi trường theo thời gian
Actor	YoloBit
Trigger	2 giây trôi qua kể từ lần cập nhật lên server cuối cùng
Pre-condition	Cảm biến ánh sáng được kết nối với YoloBit, YoloBit đã subscribed server
Post-condition	Dữ liệu được thu thập và hiển thị trên dashboard
Normal flow	1. Cảm biến ánh sáng gửi dữ liệu về YoloBit. 2. YoloBit nhận dữ liệu từ cảm biến ánh sáng và gửi lên hệ thống. 3. Hệ thống nhận dữ liệu từ YoloBit và hiển thị dưới dạng đồ thị đường.
Alternative flow	None
Exception flow	Tại bước 1: YoloBit không nhận được dữ liệu từ cảm biến 1a. YoloBit về trạng thái đợi dữ liệu trong 2 giây 1b. Quay về bước 1 ở normal flow

8.1.2 Kiểm tra dữ liệu nhận được vượt quá ngưỡng cho phép

Use case name	Kiểm tra dữ liệu nhận được vượt quá ngưỡng cho phép
Description	Bật/tắt đèn tự động dựa vào ngưỡng cường độ sáng thiết đặt trước
Actor	YoloBit
Trigger	Cường độ ánh sáng nhận được từ cảm biến nhỏ hơn/lớn hơn hoặc bằng ngưỡng đặt trước
Pre-condition	Cảm biến ánh sáng được kết nối với YoloBit, YoloBit đã subscribed server
Post-condition	Đèn 4 LED RGB bật/tắt
Normal flow	1. YoloBit nhận dữ liệu từ cảm biến ánh sáng. 2. YoloBit so sánh cường độ nhận được với ngưỡng ánh sáng đặt trước. 3. Nếu cường độ sáng nhỏ hơn ngưỡng, đèn mở.
Alternative flow	Tại bước 3: Nếu cường độ sáng lớn hơn ngưỡng, đèn tắt.
Exception flow	None

8.1.3 Điều khiển thiết bị

Use case name	Điều khiển thiết bị
Description	Điều khiển chốt cửa và còi báo mở/đóng cửa
Actor	Người dùng, YoloBit, AI inference
Trigger	Người dùng nhấn mở/đóng cửa thông qua webapp hoặc khuôn mặt được nhận diện thành công/thất bại
Pre-condition	Chốt cửa (relay) được kết nối với YoloBit, YoloBit đã subscribed server
Post-condition	Chốt cửa mở/đóng theo yêu cầu
Normal flow	1. AI inference gửi dữ liệu về lần nhận diện lên server. 2. Hệ thống nhận và gửi dữ liệu về cho YoloBit. 3. YoloBit nhận dữ liệu nhận diện từ server. 4. YoloBit kiểm tra chế độ hoạt động đang là AUTO hay MANUAL. 5. Nếu chế độ đang là AUTO, khuôn mặt được nhận diện thành công thì chốt cửa sẽ được mở ra. 6. YoloBit phát âm thanh báo mở cửa. 7. Người dùng nhấn đóng cửa thông qua app. 8. YoloBit phát âm thanh báo đóng cửa.
Alternative flow	Tại bước 5: Nếu chế độ là MANUAL 5.1.a. Người dùng kiểm tra dữ liệu khuôn mặt và nhấn mở cửa để mở chốt. 5.1.b. Quay lại bước 5. Tại bước 5: Nếu chế độ là AUTO, khuôn mặt được nhận diện không thành công. 5.2.a. YoloBit phát âm thanh báo không thành công.
Exception flow	None

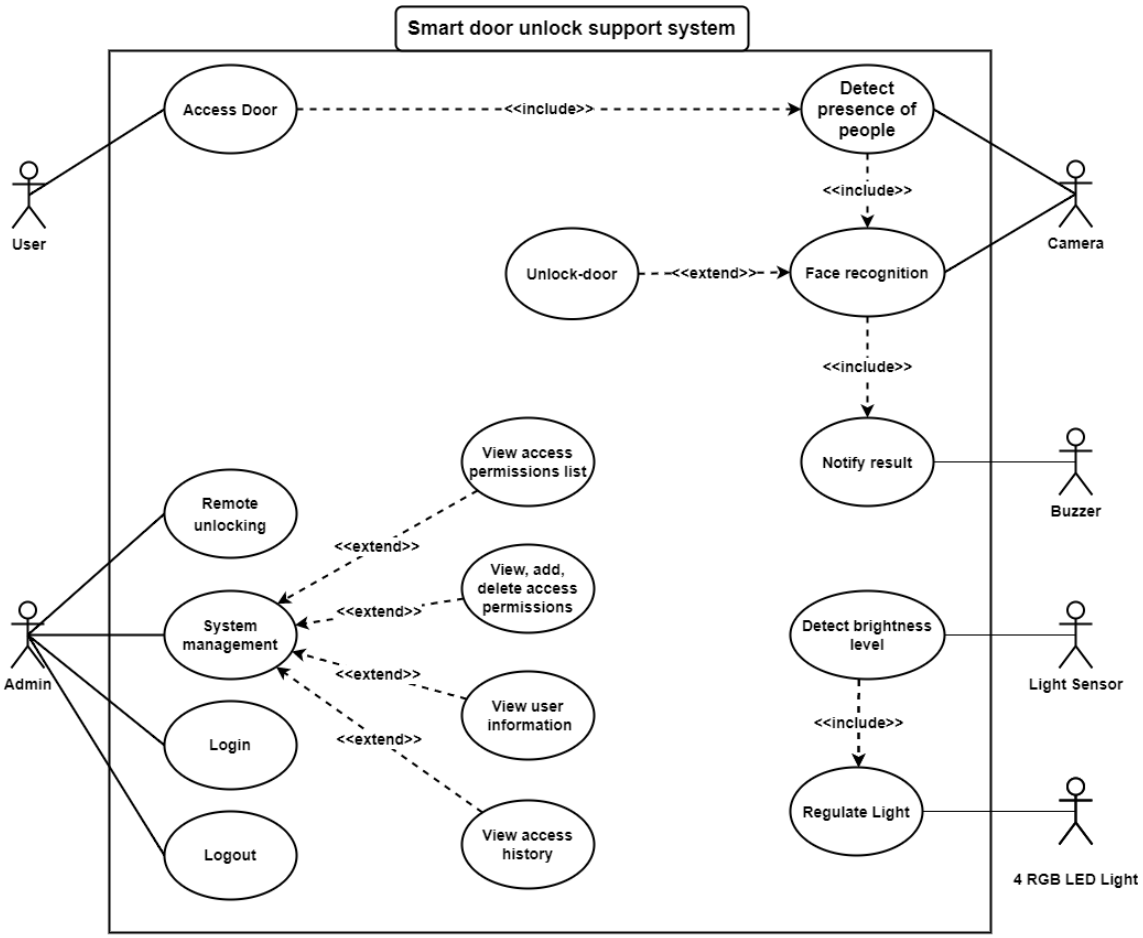
8.1.4 Ghi nhận hoạt động

Use case name	Ghi nhận hoạt động
Description	Ghi nhận lịch sử mở cửa (thời gian, người đi vào) và lịch sử nhận diện khuôn mặt
Actor	YoloBit, AI inference
Trigger	Hoạt động nhận diện khuôn mặt hoặc hoạt động mở cửa diễn ra
Pre-condition	AI inference được kết nối với YoloBit, YoloBit đã subscribed server
Post-condition	Thông tin hoạt động được cập nhật trên dashboard và app
Normal flow	1. AI inference gửi thông tin về lần nhận diện mới nhất lên server (thành công/thất bại, thông tin của người đi vào). 2. Server nhận và gửi thông tin về YoloBit. 3. YoloBit nhận thông tin nhận diện từ server, xử lí và gửi lại thông tin lên dashboard và app. 4. Nếu cửa được mở, YoloBit gửi thông tin về người đi vào, thời gian mở cửa lên dashboard và app.
Alternative flow	None
Exception flow	None

8.1.5 Ứng dụng Machine Learning

Use case name	Ứng dụng Machine Learning
Description	Nhận diện khuôn mặt
Actor	Người muốn đi vào, AI inference
Trigger	Khi có người đứng ở trước cửa
Pre-condition	AI inference được kết nối với YoloBit, YoloBit đã subscribed server
Post-condition	Dữ liệu nhận diện khuôn mặt được gửi lên server
Normal flow	1. Người dùng nhấn nút để bắt đầu nhận diện khuôn mặt 2. AI inference thực hiện phân loại khuôn mặt của người muốn đi vào sử dụng mô hình đã được huấn luyện từ trước trong khoảng 5 giây. 3. AI inference gửi kết quả nhận diện cùng với tên của người muốn đi vào lên server với cú pháp "[Tên];[kết quả nhận diện dưới dạng 0 hoặc 1]". 4. Nếu nhận diện không thành công, quay lại bước 1.
Alternative flow	None
Exception flow	None

8.2 Use case Diagram



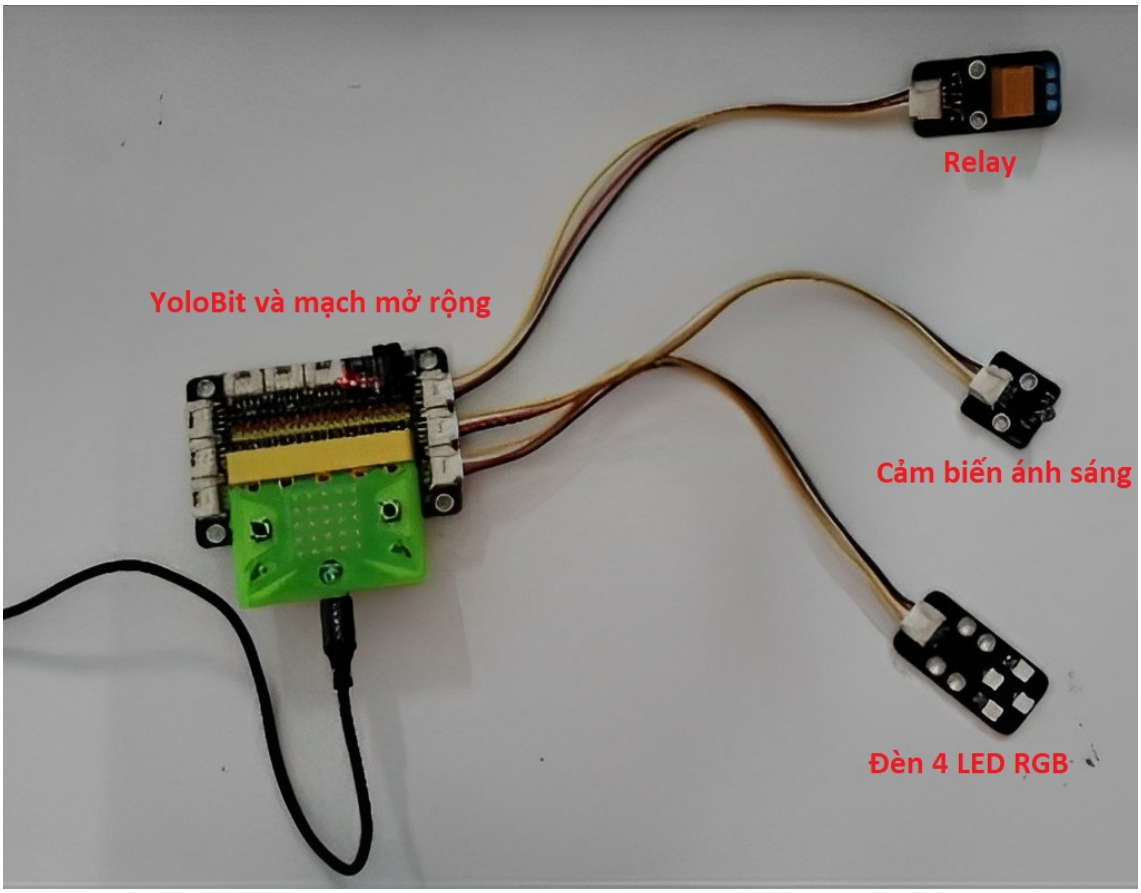
Hình 3: Use case Diagram

9 Hiện thực hệ thống

Link dẫn đến repo của dự án: https://github.com/MauDucKG/DADN_TTNT_HK222_DHA

9.1 Hiện thực phần cứng

Hệ thống có hai thành phần cần sử dụng tới phần cứng: hệ thống mạch YoloBit và AI inference. Phần AI inference chỉ bao gồm camera nhận diện khuôn mặt nên không được đề cập ở đây. Các thiết bị còn lại sẽ được kết nối với nhau tạo thành hệ thống mạch YoloBit như hình dưới đây:



Hình 4: Hệ thống mạch YoloBit

Hệ thống mạch trên sẽ được cấp nguồn từ cổng USB 3.0 của máy tính. Cổng này cũng đóng vai trò là cổng kết nối Serial giữa máy tính và YoloBit, hỗ trợ việc nạp code xử lí cho toàn bộ mạch.

Phần code xử lý được xây dựng sử dụng MicroPython. Luồng xử lý được tóm tắt như sau:

- Đầu tiên, thiết bị sau khi được khởi động sẽ kết nối tới WiFi và server Adafruit để tiến hành gửi và nhận dữ liệu.

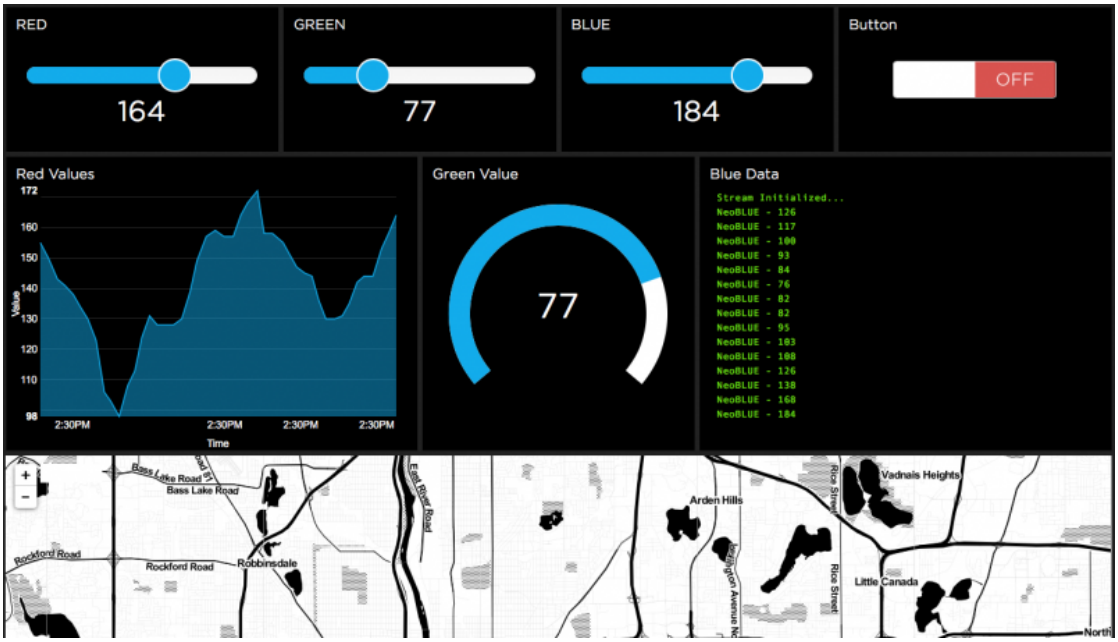
- Thiết bị sẽ luôn ở trạng thái nhận tín hiệu nhận diện khuôn mặt từ server. Nếu như chế độ hoạt động đang là AUTO và nhận diện khuôn mặt thành công thì tín hiệu mở cửa sẽ được gửi tới thiết bị, cửa sẽ được mở và âm thanh báo thành công sẽ vang lên. Còn nếu nhận diện khuôn mặt thất bại thì tín hiệu khóa cửa sẽ được gửi tới thiết bị và cửa sẽ vẫn tiếp tục khóa, đi kèm với âm thanh báo thât bại. Thông tin về lịch sử ra vào cũng như nhận diện khuôn mặt sẽ được cập nhật lên dashboard.
- Ở chế độ hoạt động MANUAL, thông tin nhận diện khuôn mặt sẽ không có tác dụng trong việc mở cửa. Người dùng lúc này phải thực hiện đóng mở cửa thủ công thông qua app. Cơ chế mở cửa giống như ở chế độ AUTO.
- Cứ mỗi 5 giây một lần, thiết bị sẽ gửi thông tin về cường độ ánh sáng của môi trường lên server. Nếu như cường độ ánh sáng trên nhỏ hơn ngưỡng ánh sáng (người dùng tự thiết đặt) thì đèn sẽ sáng lên và ngược lại.

9.2 Thiết kế Dashboard và tạo Feed nhận dữ liệu từ thiết bị

Dashboard và Feed là hai bộ phận ở trung tâm hệ thống, đóng vai trò tiếp nhận, xử lý và hiển thị các dữ liệu từ phía phần cứng. Ngoài ra, hai bộ phận này cũng là trung gian giữa Web app và phần cứng, hỗ trợ giao tiếp giữa thiết bị và người dùng.

9.2.1 Giới thiệu về Adafruit IO

Adafruit.io là một nền tảng trực tuyến được phát triển bởi Adafruit Industries, cho phép người dùng tạo ra một môi trường thân thiện, dễ sử dụng để tương tác với các phần cứng và phần mềm trong các dự án Internet of Things. Thông qua nền tảng này, người dùng có thể điều khiển các thiệt bị IoT của mình từ xa, đồng thời thu thập và phân tích dữ liệu từ các thiết bị trên theo thời gian nhằm hỗ trợ trong việc tự động hóa hệ thống. Ngoài ra, Adafruit.io còn cung cấp khả năng tích hợp với các nền tảng khác như Twitter, Facebook, Telegram,... tùy vào mục đích của người sử dụng.

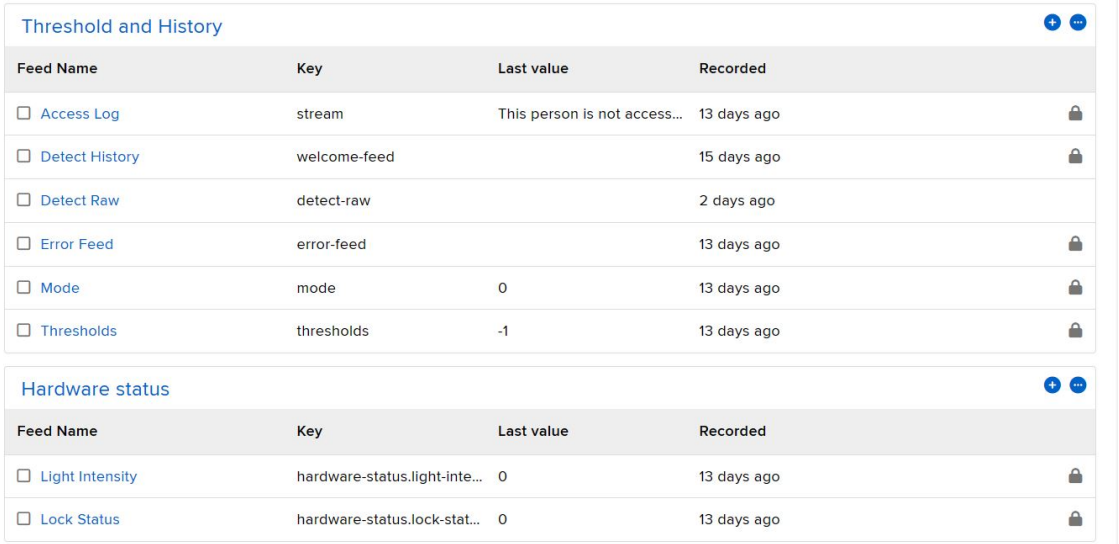


Hình 5: Giao diện mẫu Dashboard Adafruit.io

Trong đồ án lần này, nhóm sử dụng Adafruit.io để thu thập dữ liệu từ cảm biến cũng như quản lý dữ liệu về các lần đóng/mở cửa.

9.2.2 Thiết kế Feed

Dựa vào đặc tả hệ thống, nhóm xây dựng các feed sau trên server Adafruit để tiếp nhận dữ liệu từ thiết bị:



Threshold and History

Feed Name	Key	Last value	Recorded
Access Log	stream	This person is not access...	13 days ago
Detect History	welcome-feed		15 days ago
Detect Raw	detect-raw		2 days ago
Error Feed	error-feed		13 days ago
Mode	mode	0	13 days ago
Thresholds	thresholds	-1	13 days ago

Hardware status

Feed Name	Key	Last value	Recorded
Light Intensity	hardware-status.light-inte...	0	13 days ago
Lock Status	hardware-status.lock-stat...	0	13 days ago

Hình 6: Các Feed nhận dữ liệu trên server Adafruit

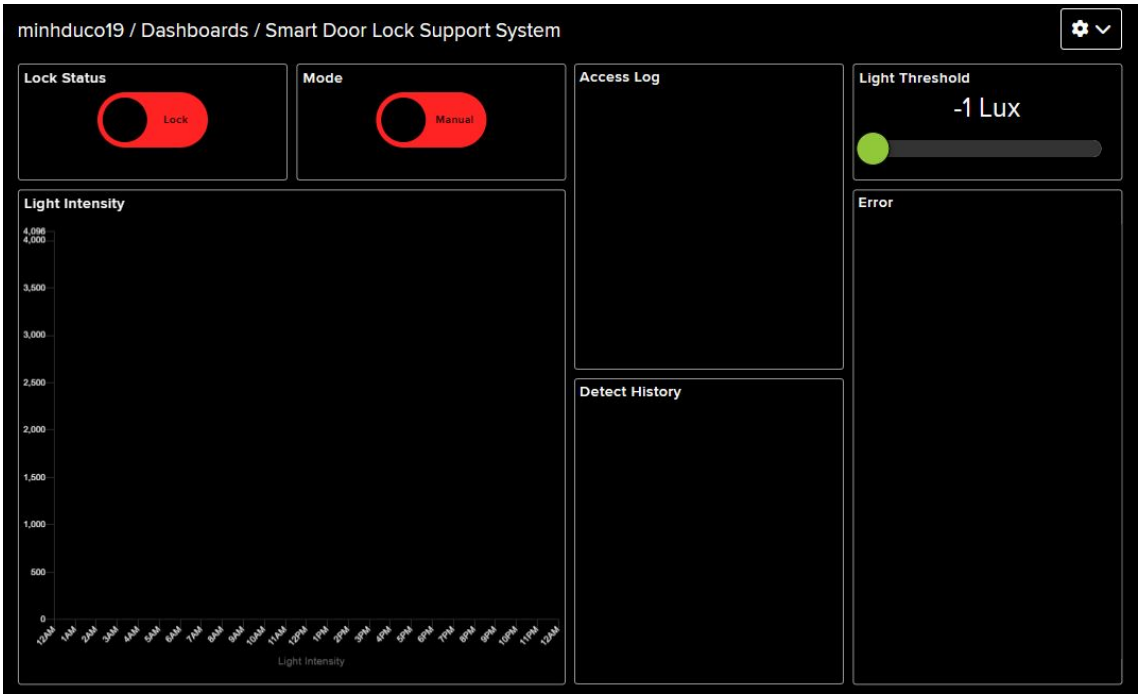
Các feed trên được chia làm hai nhóm: *Threshold and History* và *Hardware status*. Chức năng từng feed như sau:

- Access Log: Feed này dùng để lưu lại lịch sử mở cửa.
- Detect History: Dùng để lưu và hiển thị lịch sử nhận diện khuôn mặt.
- Detect Raw: Dùng để tiếp nhận thông tin nhận diện khuôn mặt từ AI inference. Thông tin này sau đó sẽ được gửi về thiết bị để xử lí. Thông tin của feed này sẽ không được hiển thị trên Dashboard.
- Error feed: Dùng để hiển thị các lỗi trên Adafruit.
- Mode: Ghi nhận mode hiện tại của hệ thống (Automatic hoặc Manual).
- Thresholds: Lưu threshold về độ sáng để điều khiển bật/tắt đèn.
- Light Intensity: Ghi nhận các mức độ sáng môi trường do thiết bị gửi về.
- Lock Status: Ghi nhận trạng thái đóng/mở cửa hiện tại.

Thông tin từ các feed sẽ được gửi tới phần Backend của Web thông qua các HTTP API được cung cấp bởi Adafruit để tiếp tục xử lí.

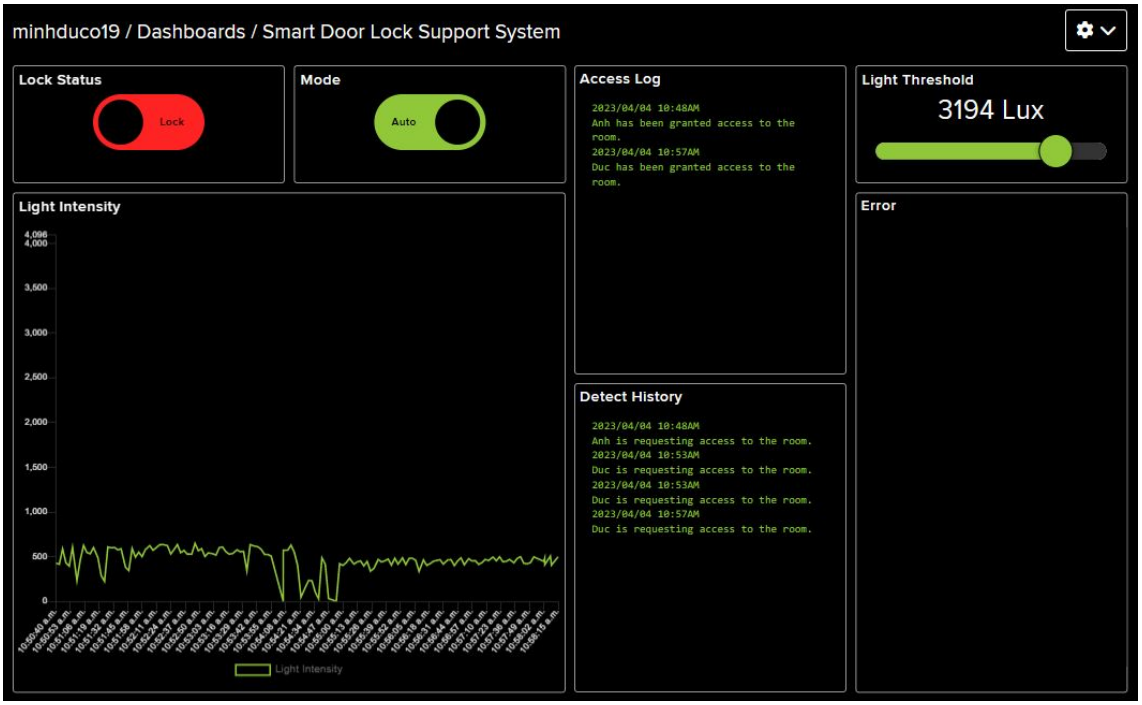
9.2.3 Thiết kế Dashboard

Để hiển thị các thông tin từ Feed một cách trực quan, nhóm thiết kế Dashboard có bố cục như sau (tên các block trong Dashboard ứng với các feed ở phần trước):



Hình 7: Bố cục Dashboard trên server Adafruit

Dưới đây là giao diện Dashboard khi hệ thống hoạt động:



Hình 8: Giao diện Dashboard khi hệ thống hoạt động

9.3 Thiết kế UI

Thiết kế giao diện người dùng (UI) là một phần quan trọng trong quá trình phát triển hệ thống. Nó đảm bảo rằng người dùng có thể tương tác với hệ thống một cách dễ dàng và hiệu quả. Một thiết kế UI tốt cần phải cân nhắc đến nhu cầu của người dùng, sự dễ sử dụng, tính thẩm mỹ và độ linh hoạt. Để đạt được điều này, phát triển UI thường bao gồm quá trình thiết kế, thử nghiệm và cải tiến. Một số công cụ phổ biến để thiết kế UI bao gồm các phần mềm thiết kế đồ họa, các framework UI và các nguyên tắc thiết kế giao diện người dùng được chấp nhận.

Ở đây nhóm sử dụng công cụ ***Figma*** *làm môi trường phát triển và sử dụng Design system* ***Material X*** *làm Kit chính*

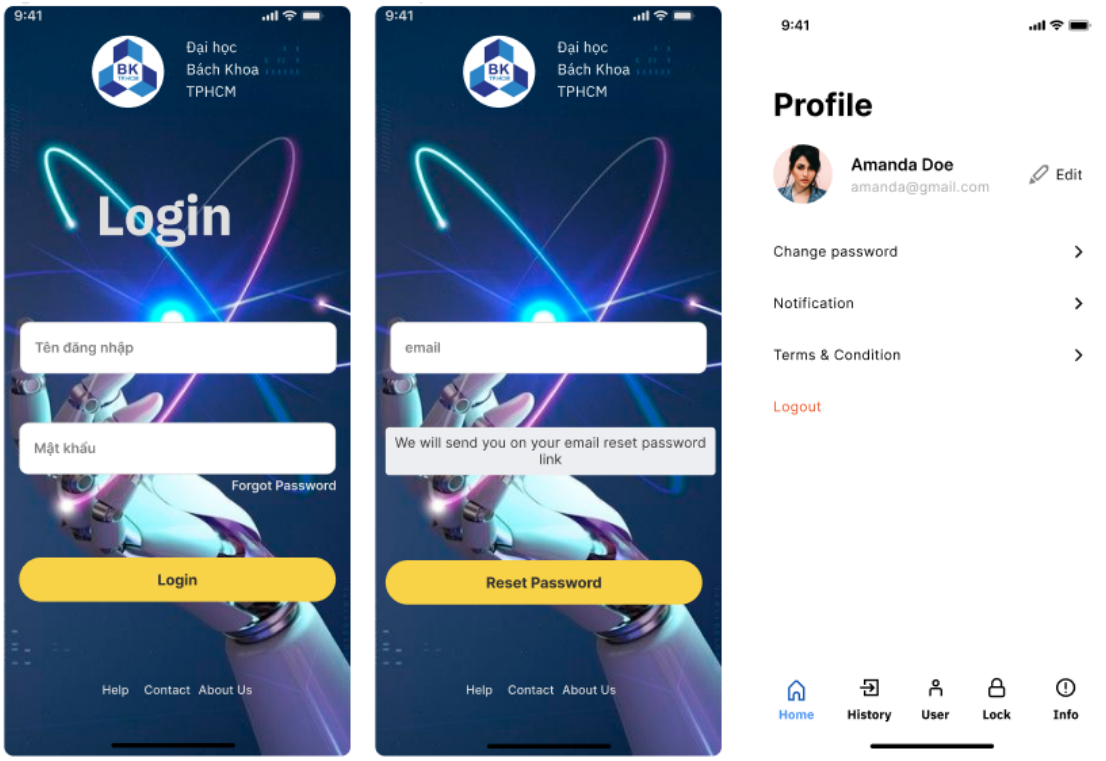
Link hiện thực: https://bom.so/a5ZyZb

9.3.1 Đăng nhập

Màn hình đầu tiên hiện ra khi người dùng sử dụng ứng dụng là màn hình đăng nhập. Màn hình đăng nhập bao gồm: các ô trống để nhập tên đăng nhập và mặt khẩu, nút bấm đăng nhập, nút quên mật khẩu và các liên kết trợ giúp.

Khi người dùng chọn vào nút quên mật khẩu, màn hình Reset mật khẩu sẽ hiện lên và yêu cầu người dùng nhập địa chỉ email để ứng dụng reset và gửi mật khẩu mới đến địa chỉ email của người dùng.

Khi đã đăng nhập thành công, người dùng sẽ được chuyển đến trang chủ của ứng dụng bao gồm thông tin người dùng và các tác vụ khác như: đổi mật khẩu, chỉnh sửa thông tin cá nhân, xem thông báo, đăng xuất, ...

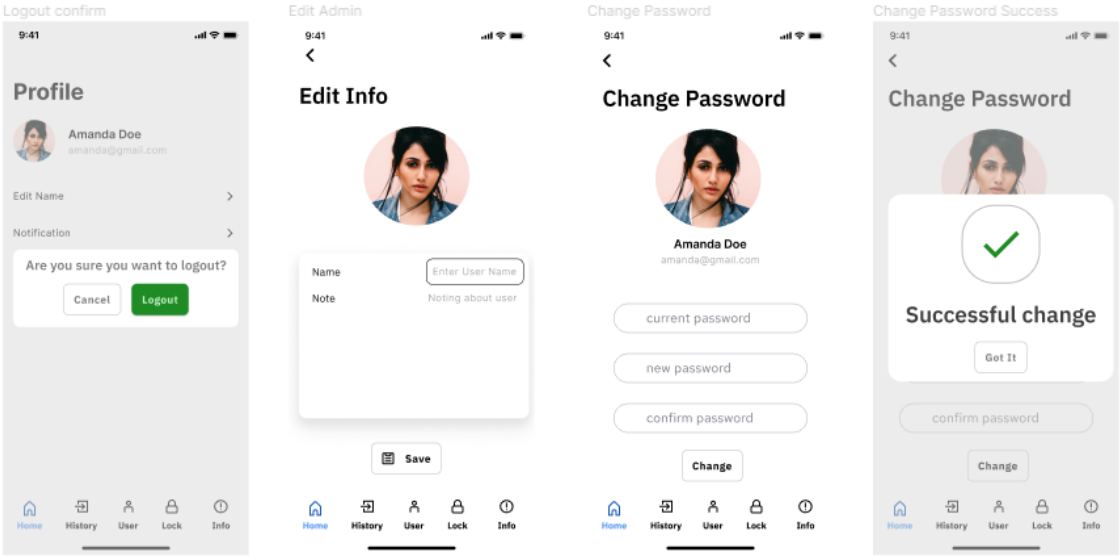


Hình 9: UI cho đăng nhập, reset mật khẩu và trang chủ

Khi người dùng chọn vào Logout, hệ thống sẽ hiển thị thông báo để xác nhận lại yêu cầu đăng xuất của người dùng.

Khi đã đăng nhập và vào trang chủ của ứng dụng, người dùng có thể tự thay đổi thông tin cá nhân của chính mình bằng cách chọn vào nút Edit trên góc phải. Lúc này ứng dụng sẽ chuyển đến trang chỉnh sửa thông tin, tại đây người dùng có thể chỉnh sửa hình ảnh, họ tên và ghi chú về mình.

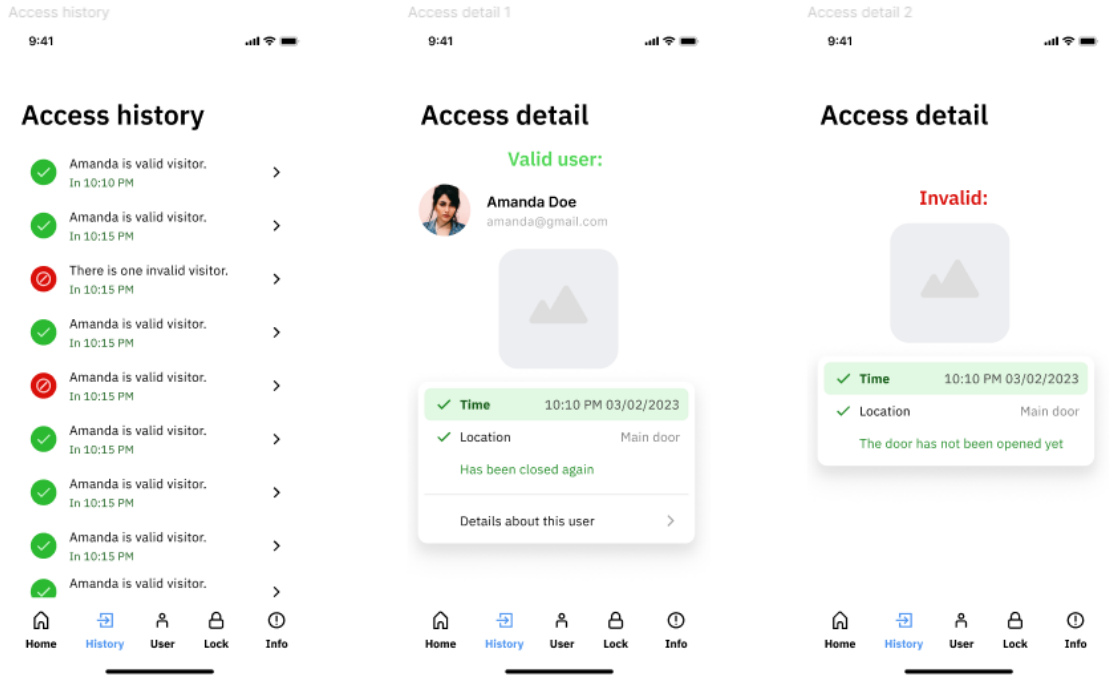
Một tác vụ khác có ở trang chủ đó là thay đổi mật khẩu, khi người dùng chọn thay đổi mật khẩu ứng dụng sẽ chuyển đến trang đổi mật khẩu và yêu cầu người dùng nhập lại mật khẩu cũ và chọn mật khẩu mới. Nếu người dùng đổi mật khẩu thành công, sẽ có thông báo đã thay đổi thành công lên trên cùng.



Hình 10: UI cho xác nhận đăng xuất, chỉnh sửa thông tin, đổi mật khẩu và thông báo đổi mật khẩu thành công

9.3.2 Lịch sử ra vào

Ở màn hình này sẽ cho ra một danh sách các lần xâm nhập của người dùng vào cửa, thể hiện được ảnh về lần xâm nhập này, thời gian, vị trí (ở cửa nào), thông tin về user và chi tiết lần xâm nhập ấy có hợp lệ hay không.

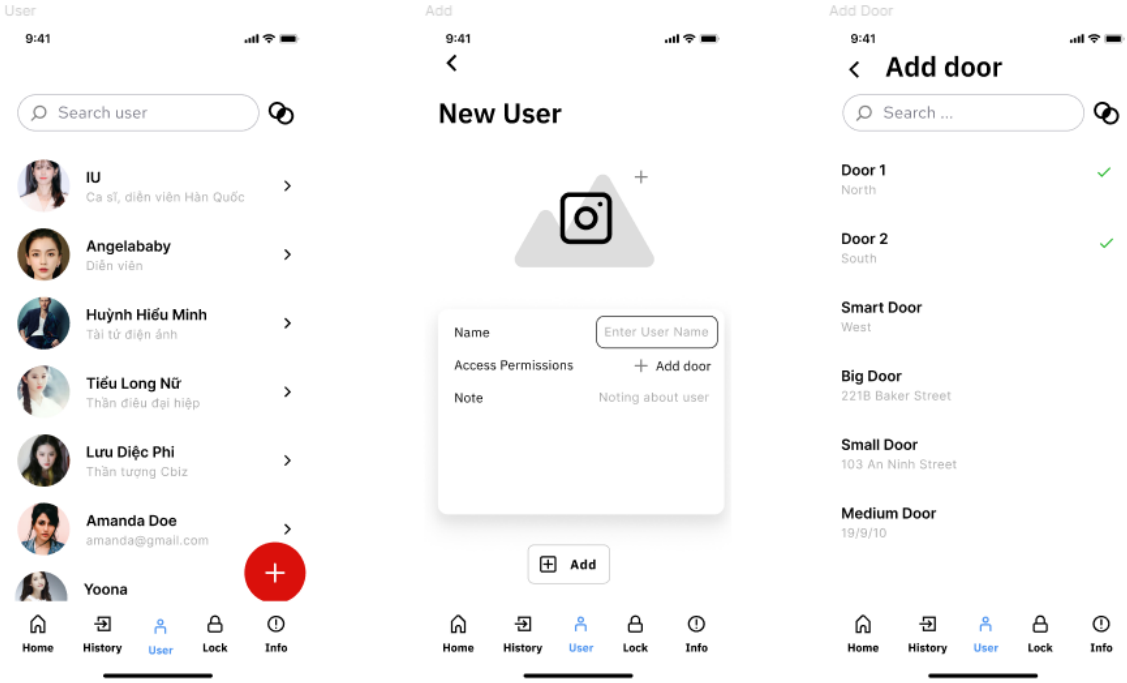


Hình 11: UI cho lịch sử ra vào

9.3.3 Quản lý người dùng

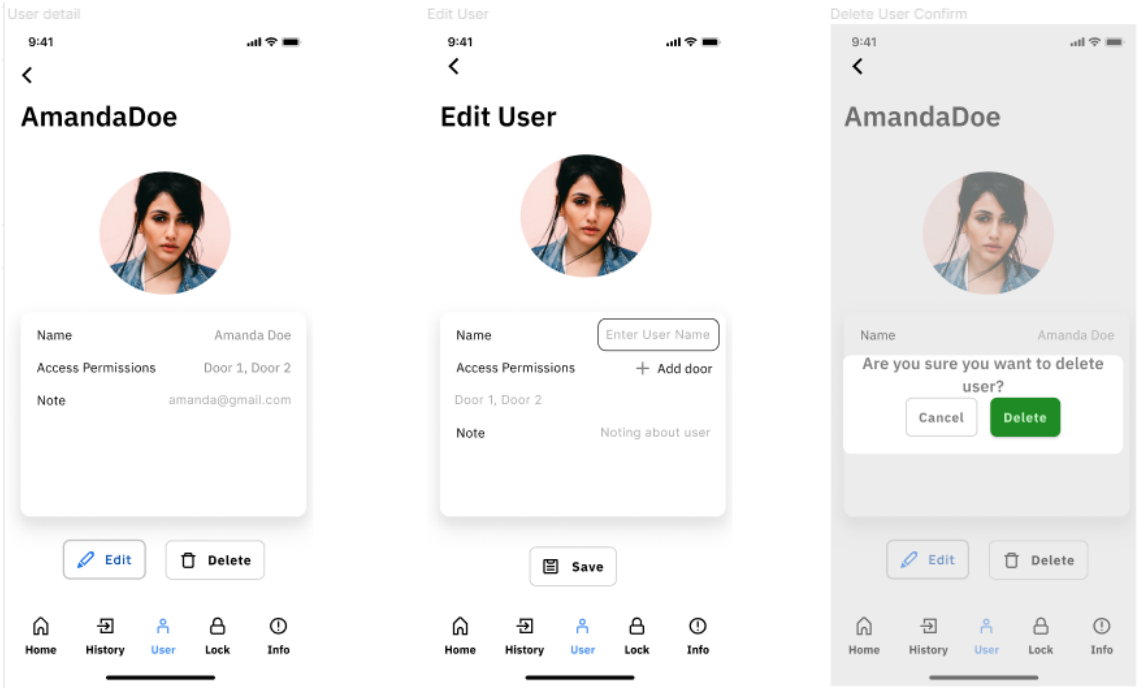
Màn hình User bao gồm danh sách về toàn bộ người dùng trong hệ thống. Các thông tin được hiển thị dưới dạng danh sách (list), bao gồm: Hình ảnh nhận diện, tên và ghi chú. Màn hình User cung cấp tính năng tìm kiếm và lọc để giúp tìm kiếm thông tin một cách nhanh chóng và thuận tiện, tính năng xem chi tiết user và thêm người dùng.

Khi admin chọn vào nút thêm người dùng, ứng dụng sẽ chuyển sang giao diện để người dùng nhập các thông tin cần thiết khi thêm người dùng bao gồm: Tên, quyền truy cập, thông tin nhận diện dạng ảnh và ghi chú.



Hình 12: UI cho quản lý và thêm người dùng

Chức năng xem chi tiết người dùng cung cấp thông tin chi tiết về người dùng bên cạnh đó có các tùy chọn như chỉnh sửa thông tin, xóa người dùng. Trang chỉnh sửa người dùng gồm các thông tin người dùng có thể chỉnh sửa như: hình ảnh, họ tên và quyền ra vào của người dùng. Khi admin nhấn vào nút xóa người dùng, ứng dụng sẽ hiện form để xác nhận lại yêu cầu của admin.

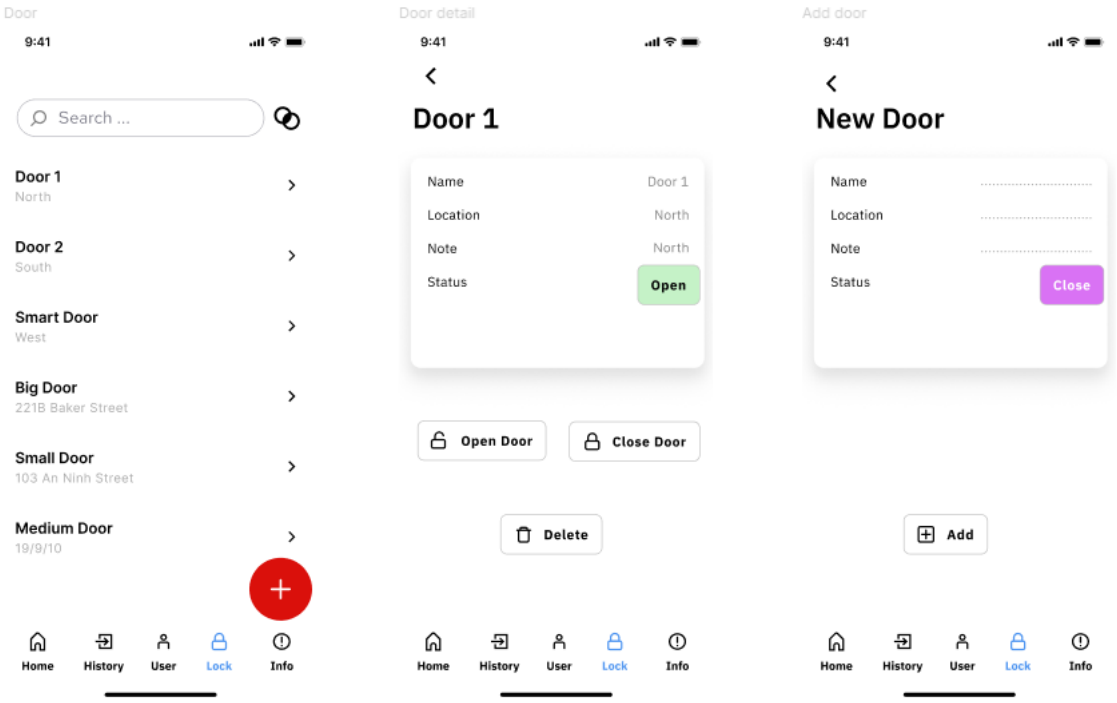


Hình 13: UI cho chỉnh sửa thông tin và xóa người dùng

9.3.4 Quản lý cửa

Màn hình Door bao gồm danh sách về toàn bộ cửa được điều khiển trong hệ thống. Các thông tin được hiển thị dưới dạng danh sách (list), bao gồm: Tên cửa và vị trí. Màn hình Door cung cấp tính năng tìm kiếm và lọc để giúp tìm kiếm thông tin một cách nhanh chóng và thuận tiện, tính năng xem chi tiết cửa và thêm cửa.

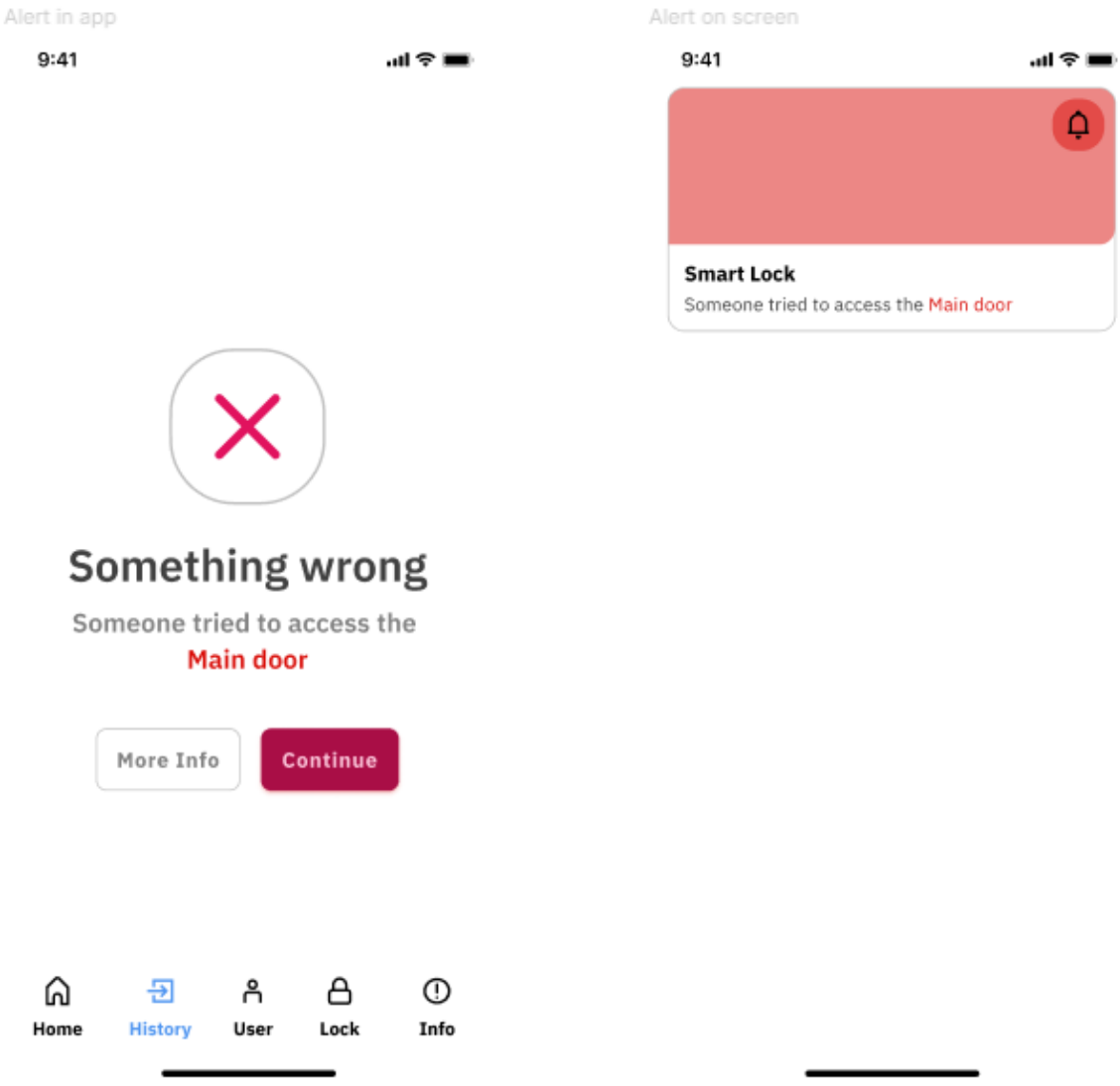
Xem chi tiết cửa cung cấp thông tin chi tiết về cửa có các tùy chọn như chỉnh sửa thông tin, mở cửa, đóng cửa, xóa cửa. Một số thông tin cần nhập khi thêm cửa bao gồm: Tên, vị trí, ghi chú.



Hình 14: UI cho quản lý cửa

9.3.5 Thông báo khi có người lạ

Ở đây trình bày 2 cách thông báo trong ứng dụng: một là khi người dùng đang sử dụng app *Alert in app* và thông báo đẩy *Alert in screen*



Hình 15: Thông báo khi có người lạ

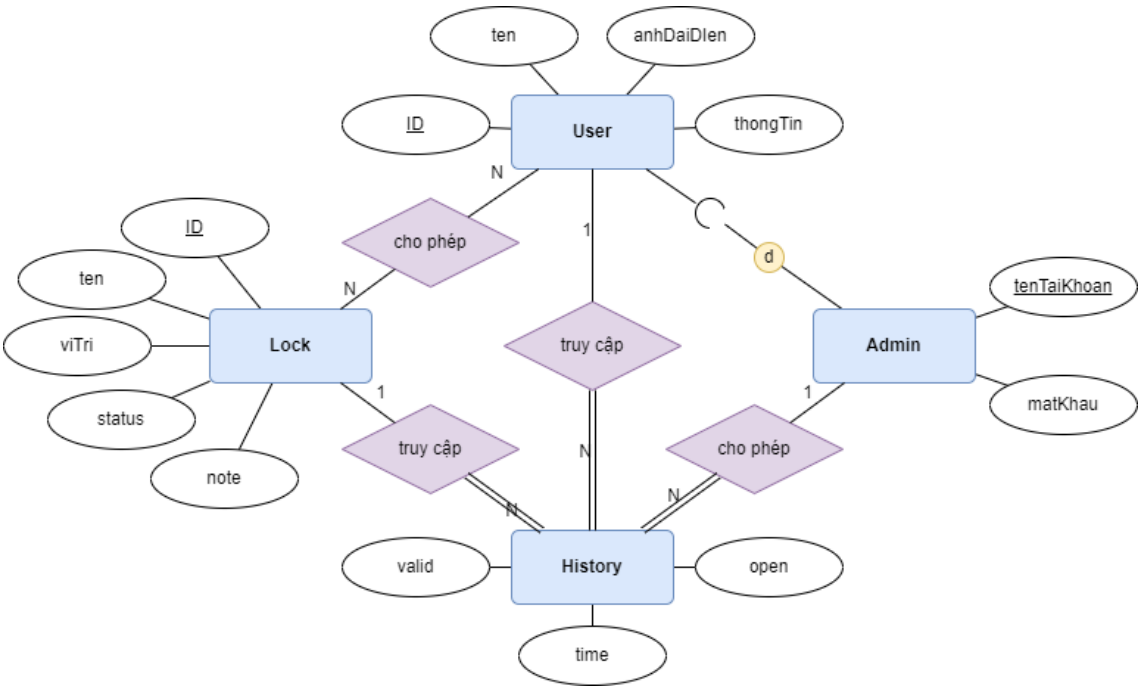
9.4 Thiết kế Cơ sở dữ liệu

9.4.1 ERD

ERD là viết tắt của Entity-Relationship Diagram, một loại sơ đồ được sử dụng để biểu diễn cấu trúc và quan hệ giữa các thực thể (entity) trong một hệ thống cơ sở dữ liệu. ERD được sử dụng để mô hình hóa các quan hệ giữa các thực thể và thuộc tính của chúng, giúp cho người thiết kế có thể hiểu rõ cấu trúc dữ liệu của hệ thống và thiết kế cơ sở dữ liệu một cách hiệu quả.

ERD (Entity-Relationship Diagram) là một công cụ được sử dụng trong quản lý cơ sở dữ liệu để biểu diễn cấu trúc của cơ sở dữ liệu dưới dạng mô hình hóa. Nó biểu diễn các thực thể và quan hệ giữa chúng thông qua các đường nối.

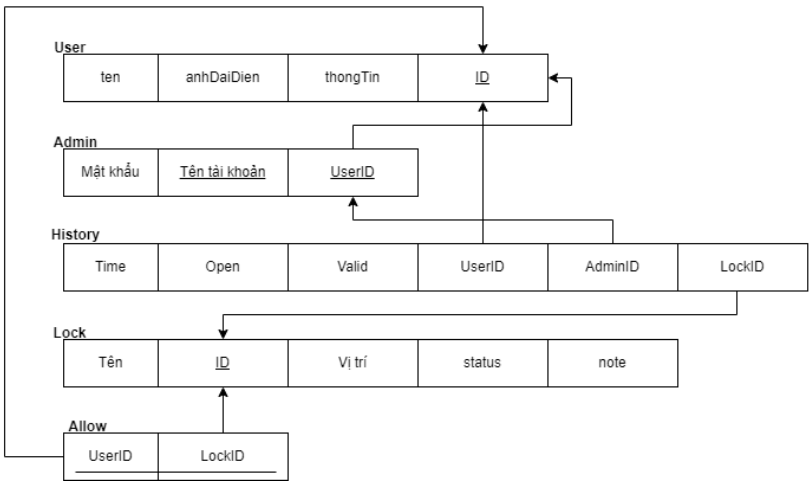
Sau đây là sơ đồ ERD của hệ thống:



Hình 16: Sơ đồ ERD

9.4.2 Mapping

Từ sơ đồ ERD ở trên, ánh xạ sang lược đồ cơ sở dữ liệu như sau:



Hình 17: Lược đồ cơ sở dữ liệu

a) **User**:
Khi 1 người dùng muốn sử dụng hệ thống, thông tin về tên người dùng, các thông tin cá nhân và hình ảnh khuôn mặt cần được lưu trữ trên hệ thống. Relation User được dùng để lưu lại thông tin của 1 người dùng, bao gồm các trường:

- **ID**: Mỗi người dùng khi tham gia vào hệ thống sẽ được lưu trữ với 1 ID riêng biệt.
- **ten**: Dùng để lưu họ tên người dùng.
- **thongTin**: Dùng để lưu lại các thông tin cần thiết về người dùng như chức vụ, vai trò.
- **anhDaiDien**: ảnh khuôn mặt của người dùng cần được lưu lại để hệ thống có thể đối chiếu và nhận diện.

b) **Admin**:
Admin là 1 loại người dùng đặc biệt có thể đăng nhập và hệ thống và có 1 số quyền nhất định để thêm, chỉnh sửa hay xóa người dùng cũng như các dữ liệu khác. Relation Adim gồm có các trường là:

- **taiKhoan**: dùng để lưu lại tên tài khoản người dùng dùng để đăng nhập vào hệ thống, tên tài khoản này phải là duy nhất giữa các admin.
- **matKhau**: mật khẩu để đăng nhập ứng với từng tên tài khoản.
- **userID**: do admin là một loại người dùng đặc biệt, vì vậy đối tượng admin là một đối tượng kế thừa của người dùng. Trường userID của admin là một khóa ngoại trỏ đến ID của 1 người dùng.

c) **History**:
Khi người dùng sử dụng hệ thống để ra vào các khu vực, lịch sử ra vào và nhận diện của người dùng sẽ được lưu lại trên hệ thống. Relation History dùng để lưu lại 1 lượt truy cập để nhận diện của người dùng đối với hệ thống, relation này bao gồm các trường:

- **time**: dùng để lưu lại thời gian đứng trước camera nhận diện của người dùng.

- **open**: dùng để lưu lại trạng thái thành công của người dùng, nếu trong lần truy cập đó cửa được mở và người dùng vào thành công khu vực, trường open sẽ lưu trữ giá trị True. Nếu người dùng bị từ chối vào khu vực, giá trị False sẽ được lưu lại.
- **valid**: dùng để lưu lại trạng thái nhận diện thành công của người dùng. Sau khi hệ thống nhận diện thành công người dùng, nếu người dùng được phép vào khu vực đang nhận diện, trường valid sẽ lưu giá trị True. Nếu người dùng không được phép vào khu vực cố gắng truy cập, trường sẽ có giá trị False. Khác với open, khi người dùng có giá trị valid là False vẫn có thể vào bên trong khu vực nếu như admin ử dụng chức năng mở cửa từ xa để cho phép người dùng vào khu vực.
- **userID**: là một khóa ngoại trỏ đến trường ID của user để lưu trữ ID của user đang muốn ra vào khu vực.
- **lockID**: là một khóa ngoại trỏ đến trường ID của lock để lưu trữ ID của lock đang được truy cập.
- **adminID**: là một khóa ngoại trỏ đến trường userID của Admin. Trong trường hợp người dùng không có quyền ra vào khu vực nhưng yêu cầu ra vào được chấp nhận bởi 1 admin từ xa, ID của admin đó sẽ được lưu vào trường adminID. Trong các trường hợp còn lại, giá trị của trường adminID sẽ là null.

d) **Lock**:
Relation Lock dùng để lưu các thông tin về các khóa cửa được áp dụng hệ thống nhận diện khuôn mặt, gồm có các trường:

- **ID**: mỗi khóa khi được thêm vào hệ thống sẽ được lưu trữ với 1 ID riêng biệt.
- **ten**: dùng để lưu lại tên khóa.
- **viTri**: là một trường có kiểu dữ liệu dạng String, dùng để lưu lại các miêu tả về vị trí của khóa. Cách đặt tên cho vị trí sẽ tùy thuộc vào từng cơ quan, tổ chức sử dụng hệ thống.
- **status**: dùng để lưu lại trạng thái đóng mở của khóa.
- **note**: dung để lưu lại các ghi chú về khóa.

e) **Allow**: Relation Allow dùng để lưu lại quyền ra vào của người dùng đối với các khóa. Một nngười dùng được xem là có quyền ra vào khu vực nếu trong relation Allow tồn tại một đối tượng chứa ID của người dùng và ID khóa của khu vực đó. Relation Allow gồm có các trường:

- **userID**: ID của người dùng được ra vào cửa.
- **lockID**: ID của khóa cửa.

9.4.3 Hiện thực cơ sở dữ liệu MongoDB

9.4.3.a Giới thiệu về MongoDB

MongoDB là một hệ thống quản lý cơ sở dữ liệu phi quan hệ (non-relational database management system) mã nguồn mở lần đầu ra đời bởi MongoDB Inc., tại thời điểm đó là thế hệ 10, vào tháng Mười năm 2007, nó là một phần của sản phẩm PaaS (Platform as a Service) tương tự như Windows Azure và Google App Engine. Sau đó nó đã được chuyển thành nguồn mở từ năm 2009.

MongoDB hoạt động trên mô hình dữ liệu tài liệu (document-oriented data model), trong đó dữ liệu được tổ chức thành các tài liệu JSON (JavaScript Object Notation). Các tài liệu này được

lưu trữ trong collections (tương đương với bảng trong cơ sở dữ liệu quan hệ), và các collections có thể được phân chia thành các shard để tăng tính mở rộng và hiệu suất.

Ưu điểm của MongoDB:

- Dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, linh hoạt trong việc lưu trữ dữ liệu, hiệu suất lớn và dễ dàng mở rộng lưu trữ.
- Dữ liệu được caching (ghi đệm) lên RAM, hạn chế truy cập vào ổ cứng nên tốc độ đọc và ghi cao.
- MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node mới vào cluster:
- Trường dữ liệu "_id" luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
- Khi có một truy vấn dữ liệu, bản ghi được cached lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.
- Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn thì thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.

Hạn chế

- Không ứng dụng được cho các mô hình giao dịch nào có yêu cầu độ chính xác cao do không có ràng buộc.
- Không có cơ chế transaction (giao dịch) để phục vụ các ứng dụng ngân hàng.
- Dữ liệu lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn.
- Mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao.

MongoDB được ưa chuộng trong các ứng dụng web và mobile như social networking, e-commerce, gaming và real-time analytics. Với thiết kế linh hoạt và hiệu suất tốt, MongoDB là sự lựa chọn hàng đầu cho các nhà phát triển khi cần lưu trữ dữ liệu cho các ứng dụng có khả năng mở rộng lớn và đáp ứng yêu cầu của người dùng.

Với những lý do trên, nhóm em quyết định sử dụng MongoDB để hiện thực hệ cơ sở dữ liệu cho ứng dụng.

9.4.3.b Khởi tạo Database

Cấu trúc dữ liệu của chương trình gồm 4 collection chính là:

- **user**:
  User là collection dùng để lưu thông tin và ảnh nhận diện khuôn mặt của người dùng.

Listing 1: User

```
{
  ten: {
    type: String,
    required: true,
  },
  thongTin: {
    type: String,
  },
  anhDaiDien: {
    type: String,
  },
}
```

- **admin**:
  Admin dùng để lưu lại tên tài khoản và mật khẩu của các user đặc biệt có thể đăng nhập vào hệ thống, vùng userID của admin chỉ đến key ID của collection User.

Listing 2: Admin

```
{
  userID: {
    type: Schema.Types.ObjectId,
    required: true,
    ref: 'users'
  },
  taiKhoan: {
    type: String,
    unique: true
  },
  matKhau: {
    type: String
  },
}
```

- **lock**:
  Lock dùng để lưu thông tin của các khóa có trên hệ thống.

Listing 3: Lock

```
{
  ten: {
    type: String,
    required: true
  },
  viTri: {
    type: String,
    required: true
  }
}
```

- **allow**:
  Collection Allow dùng để lưu các quyền truy cập của một user đối với một lock.

Listing 4: Allow

```
{
  userID: {
    type: Schema.Types.ObjectId,
    ref: 'user',
    required: true
  },
  lockID: {
    type: Schema.Types.ObjectId,
    ref: 'allow',
    required: true
  }
}
```

- **history**:
  Collection History dùng để lưu lại lịch sử truy cập của các user đối với các lock và trạng thái hợp lệ của lượt truy cập. Nếu truy cập là không hợp lệ nhưng user được admin mở cửa từ xa để cho vào khu vực, vùng adminID trên collection sẽ lưu lại userID của admin đã mở cửa cho user.

Listing 5: History

```
{
  lockID: {
    type: Schema.Types.ObjectId,
    ref: "locks",
    required: true,
  },
  userID: {
    type: Schema.Types.ObjectId,
    ref: "users",
    required: true,
  },
  adminID: {
    type: Schema.Types.ObjectId,
    ref: "admins",
  },
  time: {
    type: Date,
    default: Date.now,
  },
  open: {
    type: Boolean,
    required: true,
  },
  valid: {
    type: Boolean,
    required: true,
  },
}
```

9.5 Hiện thực phần Mobile App

9.5.1 React Native

React Native là một framework phát triển ứng dụng di động, cho phép lập trình viên sử dụng JavaScript để phát triển ứng dụng di động cho cả iOS và Android. React Native được phát triển bởi Facebook và có thể được coi là một phiên bản của React, một thư viện JavaScript phổ biến cho phát triển ứng dụng web.

Với React Native, lập trình viên có thể sử dụng các thành phần UI tương tự như các ứng dụng di động được xây dựng bằng các ngôn ngữ lập trình khác như Swift hoặc Java. Điều này giúp cho việc phát triển ứng dụng di động trở nên dễ dàng hơn và nhanh chóng hơn.

Một số tính năng của React Native bao gồm:

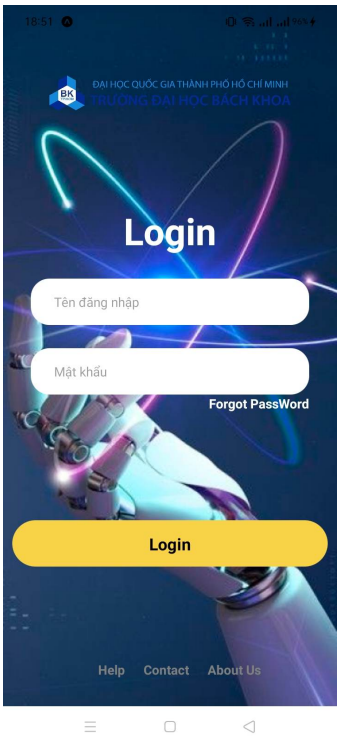
- Được xây dựng trên nền tảng React, một thư viện phổ biến cho phát triển ứng dụng web.
- Có thể sử dụng các thành phần UI được tùy chỉnh và có thể tái sử dụng.
- Hỗ trợ hot reloading, cho phép bạn xem trước các thay đổi trong giao diện người dùng mà không cần khởi động lại ứng dụng.
- Hỗ trợ các tính năng di động như định vị GPS, máy ảnh, micro, v.v.
- Hỗ trợ tính năng tương tác với server thông qua RESTful API hoặc WebSockets.

React Native là một lựa chọn tuyệt vời cho việc phát triển ứng dụng di động, đặc biệt là đối với những lập trình viên đã quen thuộc với JavaScript và React. Nó giúp tăng tốc quá trình phát triển ứng dụng di động và giảm chi phí phát triển đối với các ứng dụng di động cho cả iOS và Android. Chính vì lý do này nhóm em lựa chọn React Native làm framework để phát triển ứng dụng di động cho đồ án.

9.5.2 Kết quả hiện thực

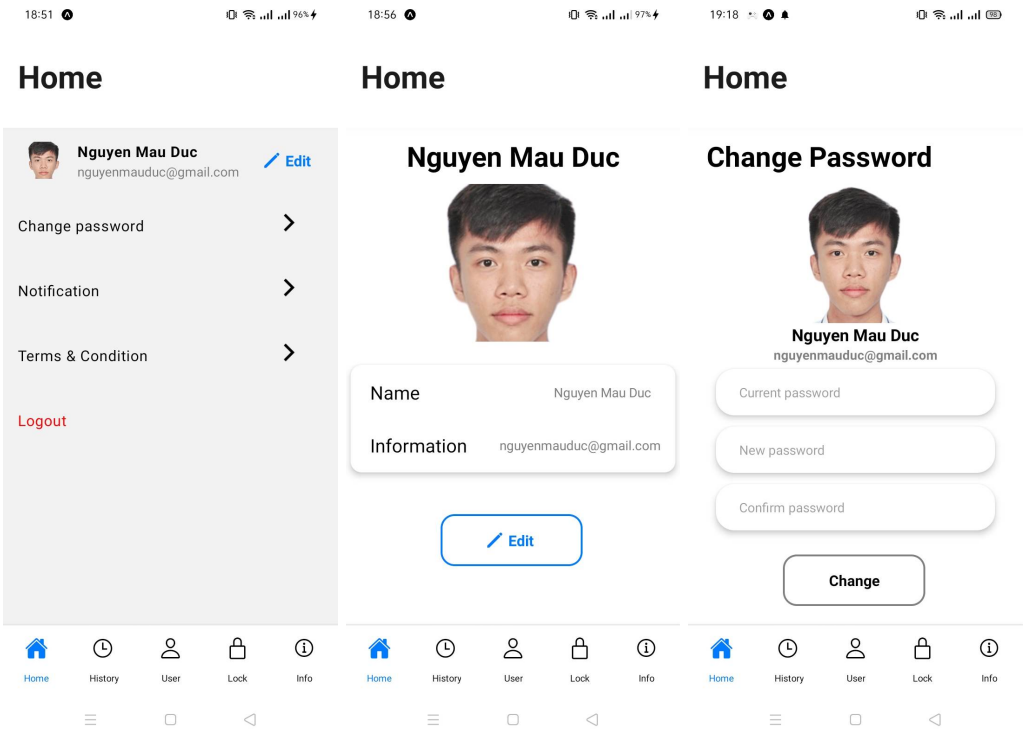
Nhóm thực hiện hiện thực UI theo mẫu đã thiết kế ở trên. Ứng dụng với các màn hình Home, History, User, Lock, Info có thể được truy cập sau khi người dùng thực hiện đăng nhập thành công trên màn hình Login.

9.5.2.a Login



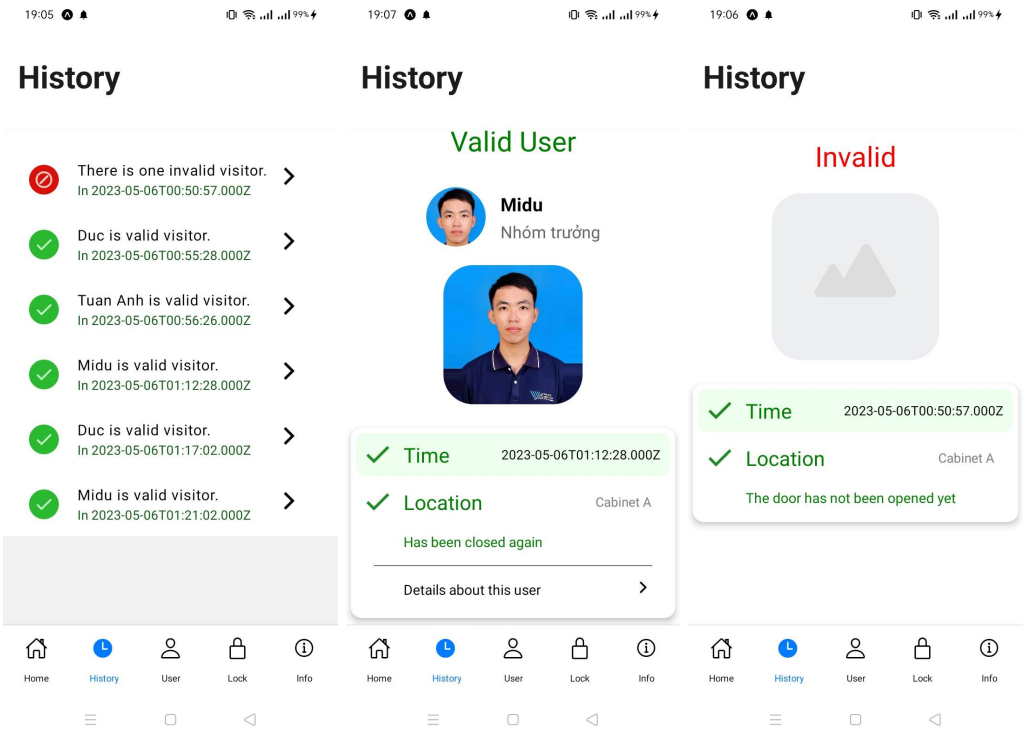
Hình 18: Giao diện Login

9.5.2.b Home



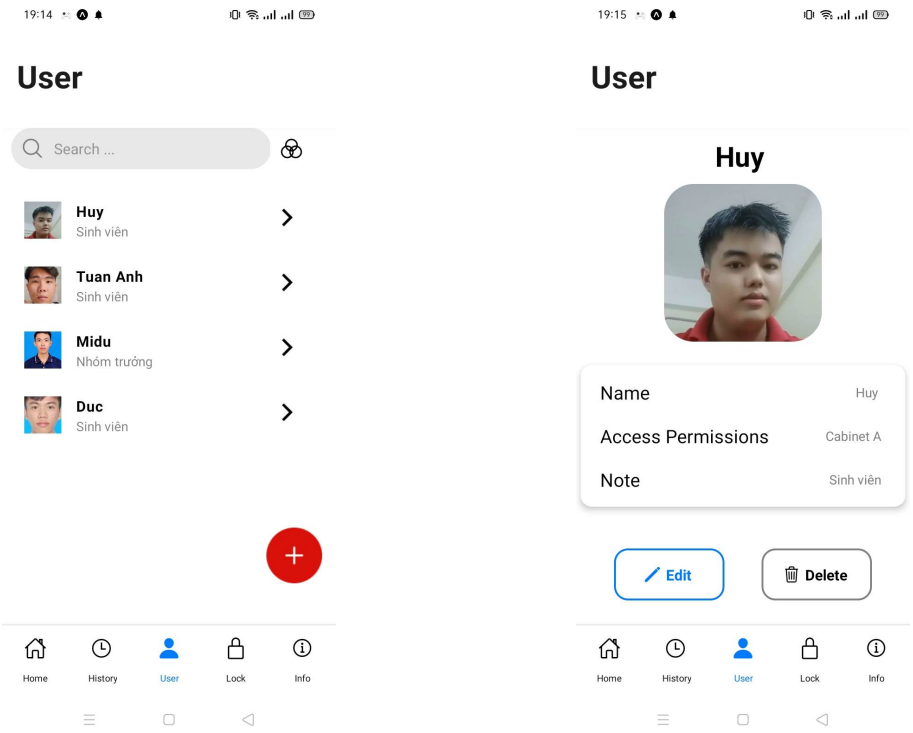
Hình 19: Giao diện Home

9.5.2.c History

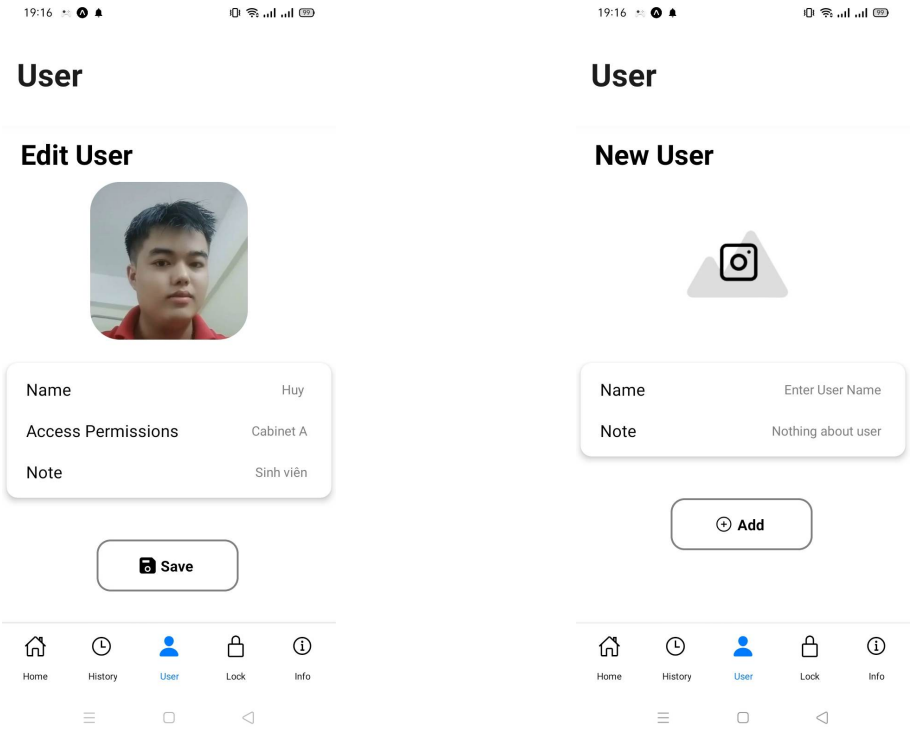


Hình 20: Giao diện History

9.5.2.d User

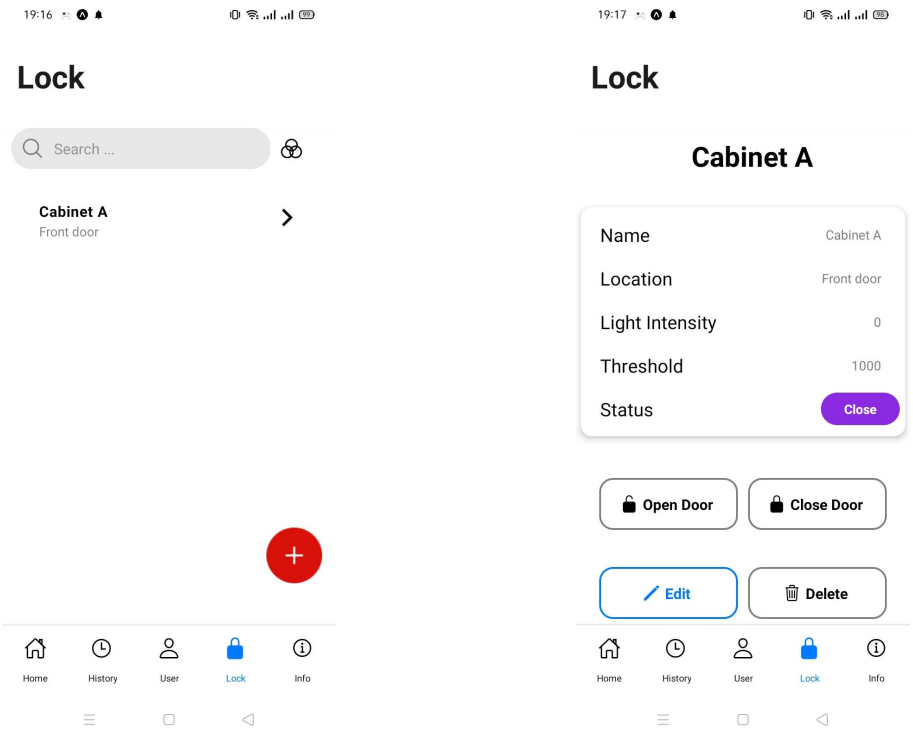


Hình 21: Giao diện danh sách User và chi tiết User

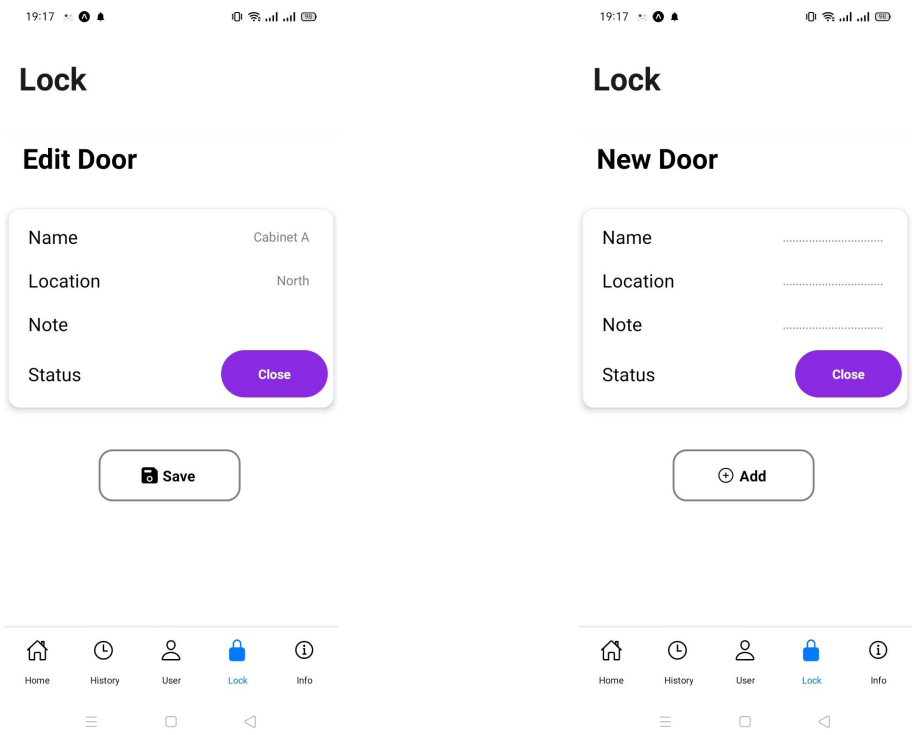


Hình 22: Giao diện chỉnh sửa User và thêm User mới

9.5.2.e Lock

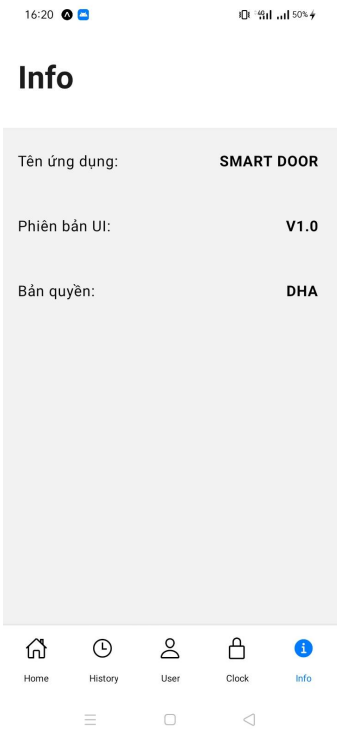


Hình 23: Giao diện danh sách Door và chi tiết Door



Hình 24: Giao diện chỉnh sửa Door và thêm Door mới

9.5.2.f Info



Hình 25: Giao diện Info

9.6 Hiện thực phần Web API cho cơ sở dữ liệu

9.6.1 Node.js - Express

Node.js là một nền tảng phát triển được xây dựng trên JavaScript runtime của Chrome, cho phép lập trình viên sử dụng JavaScript để phát triển các ứng dụng web nhanh chóng và hiệu quả. Node.js có thể được sử dụng để xây dựng các ứng dụng web backend, đặc biệt là các ứng dụng real-time và các ứng dụng đòi hỏi xử lý đa luồng.

Express là một framework web được xây dựng trên nền tảng Node.js, cung cấp các công cụ và tính năng để xây dựng các ứng dụng web backend nhanh chóng và dễ dàng. Express cho phép lập trình viên tạo các endpoint API và xử lý các yêu cầu HTTP từ phía client. Nó cũng cung cấp các tính năng như routing, middleware, và template engine để giúp lập trình viên phát triển ứng dụng web backend một cách dễ dàng và hiệu quả.

Khi sử dụng Node.js và Express để xây dựng backend cho ứng dụng web, lập trình viên có thể sử dụng các công nghệ và thư viện khác như MongoDB hay MySQL để quản lý cơ sở dữ liệu, các thư viện như Passport để xác thực người dùng, hoặc Socket.io để tạo các ứng dụng real-time. Điều này giúp cho việc phát triển các ứng dụng web backend trở nên dễ dàng hơn và nhanh chóng hơn.

Tóm lại, sử dụng Node.js và Express để xây dựng backend cho các ứng dụng React là một lựa chọn tuyệt vời cho các lập trình viên, đặc biệt là đối với những ai đã quen thuộc với JavaScript. Nó giúp tăng tốc quá trình phát triển ứng dụng web backend và giảm chi phí phát triển. Do vậy nhóm đã sử dụng Node.js cho phần backend của mình.

9.6.2 Kết quả hiện thực

Nhóm chúng tôi sẽ áp dụng mô hình MVC (Model-View-Controller) trong việc xây dựng phần Backend cho ứng dụng của mình. Trong mô hình MVC, mỗi đối tượng sẽ có ba thành phần chính là Controller, Model và View (Router) để đảm bảo tính phân tách và dễ bảo trì của ứng dụng.

- Controller: Đây là thành phần chịu trách nhiệm xử lý yêu cầu từ phía client và điều khiển việc xử lý của ứng dụng. Controller sẽ nhận các yêu cầu từ phía client, xử lý chúng và gửi kết quả trả về cho client. Controller cũng có thể gọi các phương thức của Model để thực hiện các thao tác truy vấn cơ sở dữ liệu.
- Model: Đây là thành phần đại diện cho dữ liệu và xử lý các thao tác truy vấn cơ sở dữ liệu.
- View (Router): Đây là thành phần đại diện cho giao diện người dùng. View sẽ nhận các yêu cầu từ phía client và điều hướng chúng đến các Controller tương ứng. View cũng có thể chứa các template để hiển thị dữ liệu trả về từ Controller.

Chi tiết hơn có thể xem phần hiện thực của nhóm ở: https://github.com/MauDucKG/DADN_TTNT_HK222_DHA/tree/main/backend

Còn trong phần này chúng em sẽ trình bày về collection **user**:

Phần View (Router):

```
    const webFramework = require("express");
const router = webFramework.Router();

const UserController = require("./user.controller");

router.get("/", UserController.getAllUser);
router.get("/:id", UserController.getUserById);
router.get("/:id/lock", UserController.getUserLocks);
router.post("/", UserController.newuser);

module.exports = router;
```

Phần Controller:

```
const userModel = require("./user.model");
const allowModel = require("../allow/allow.model");
const lockModel = require("../lock/lock.model");

class userController {
  getAllUser(request, respond) {
    userModel.find((error, users) => {
      if (error) {
        console.log(error);
```

```
    }

    respond.status(200).json({
      success: true,
      message: "Done!",
      users: users,
    });
  });
}

newuser = function (req, res) {
  const { ten, thongTin, anhDaiDien } = req.body;
  const user = new userModel({
    ten,
    thongTin,
    anhDaiDien,
  });
  user.save(function (error) {
    if (error) {
      res.status(500).send(error);
    } else {
      res.status(200).send("New user created!");
    }
  });
};

getUserById = async (req, res) => {
  const userId = req.params.id;
  try {
    const user = await userModel.findById(userId);
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }
    res.json(user);
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: "Server error" });
  }
};

getUserLocks = async (req, res) => {
  const userId = req.params.id;
  try {
    const result = await allowModel.find({ userID: userId });
    if (!result) {
      return res.status(404).json({ message: "Lock not found" });
    }
    const userLocks = [];

    for (const allow of result) {
      const lockResult = await lockModel.findById(allow.lockID);
      userLocks.push(lockResult);
    }
```

```
      res.status(200).json(userLocks);
    } catch (error) {
      console.error(error);
      res.status(500).json({ message: "Server error" });
    }
  };
}

module.exports = new userController();
```

Phần Model:

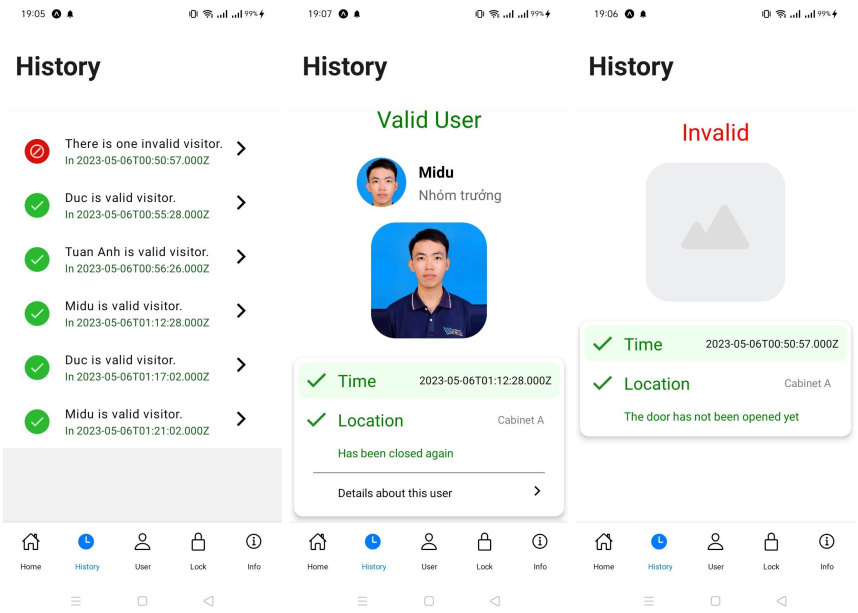
```
    const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const userSchema = new Schema(
  {
    ten: {
      type: String,
      required: true,
    },
    thongTin: {
      type: String,
    },
    anhDaiDien: {
      type: String,
    },
  },
  {
    collection: "users",
  }
);

const user = mongoose.model("users", userSchema);

module.exports = user;
```

9.7 Kết nối với cơ sở dữ liệu

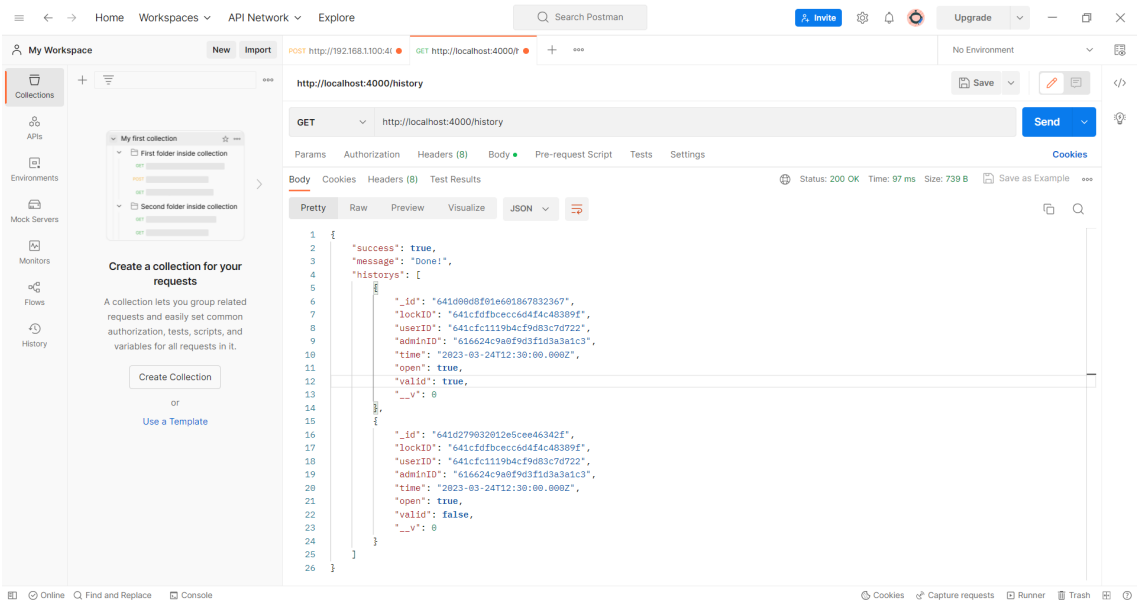
Để kết nối cơ sở dữ liệu giữa Backend (Node.js) và Frontend (React Native), chúng ta cần sử dụng một giao thức truyền thông như HTTP hoặc HTTPS để giao tiếp giữa hai phía.

Một cách đơn giản, sau khi tạo API từu phía Backend, sử dụng thư viện Axios để gửi các yêu cầu truy vấn đến Backend từ phía Frontend. Axios là một thư viện HTTP client đơn giản và dễ sử dụng, cho phép chúng ta gửi các yêu cầu GET, POST, PUT và DELETE đến Backend.

Kết quả có thể được minh họa trong phần màn hình của History và khi test bằng Postman như sau:



Hình 26: Giao diện History



Hình 27: Khóa cửa thông minh

9.8 Kết nối với thiết bị

9.8.1 Giới thiệu về MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức máy tính dựa trên giao thức TCP/IP được dùng để truyền tin nhắn giữa các thiết bị IoT (Internet of Things). MQTT sử dụng mô hình publish/subscribe để truyền thông, cho phép các thiết bị đăng ký theo dõi (subscribe) các chủ đề (topic) mà chúng quan tâm. Khi có thiết bị khác công bố (publish) tin nhắn trên một chủ đề, thông báo này sẽ được gửi đến tất cả các thiết bị đã đăng ký chủ đề đó.

MQTT sử dụng mô hình máy chủ - khách hàng. Các ứng dụng IoT hoạt động như các khách hàng MQTT, kết nối với máy chủ MQTT. Máy chủ MQTT là nơi tất cả dữ liệu được truyền và nhận lưu trữ trong các chủ đề. Các khách hàng có thể công bố dữ liệu cho chủ đề, hoặc đăng ký để nhận thông điệp được công bố trên chủ đề. MQTT hỗ trợ các phương thức bảo mật như mã hóa và xác thực để bảo vệ dữ liệu.

MQTT có lợi thế nhẹ, không phức tạp so với các giao thức khác. Nó đòi hỏi ít băng thông và thời gian phản hồi ngắn, phù hợp cho các thiết bị IoT có ít tài nguyên. Đây là lý do MQTT phổ biến trong các ứng dụng IoT đám mây. Nhóm chúng tôi sử dụng MQTT để thu thập dữ liệu từ các thiết bị IoT của Adafruit và đưa lên AWS IoT Core.

9.8.2 Kết nối của nhóm

Nhóm sử dụng cơ chế giao thức MQTT để subscribe dữ liệu từ Adafruit. MQTT là một giao thức truyền thông máy tính được thiết kế để gửi và nhận các tin nhắn đơn giản giữa các thiết bị IoT và các ứng dụng đám mây. Cơ chế này giúp giảm tải cho server bằng cách chỉ lấy dữ liệu khi có dữ liệu mới từ Adafruit.

Hiện tại, nhóm đang subscribe 3 topic là "detect-raw", "lock status" và "light status". Ở topic "detect-raw", nhóm đã cài đặt để khi có dữ liệu mới, dữ liệu đó sẽ được lưu vào collection với tên người truy cập. Tương tự, ở "lock status", dữ liệu được lưu sẽ là tên của admin mở cửa. Còn ở "light status", dữ liệu sẽ được lấy từ cảm biến ánh sáng và hiển thị lên giao diện của cửa đó.

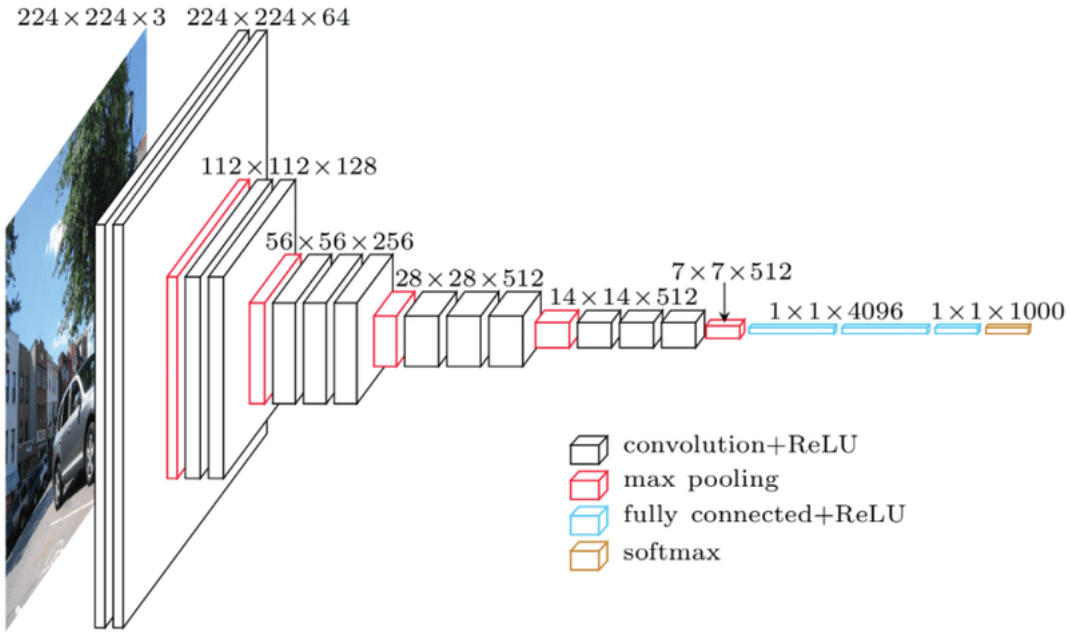
Nhóm đang sử dụng MQTT để lấy dữ liệu mới từ Adafruit IoT và xử lý, lưu trữ dữ liệu đó. Bằng cách này, server được giảm tải vì chỉ lấy dữ liệu mới khi có, tránh truy xuất liên tục. Khi có dữ liệu mới, nhóm sẽ lưu vào cơ sở dữ liệu với thông tin người dùng tương ứng để theo dõi. MQTT giúp truyền tin nhắn nhanh chóng, đơn giản giữa các thiết bị IoT và ứng dụng, phù hợp với mục đích của nhóm.

9.9 Hiện thực module AI

Model áp dụng cho module xử lí nhận diện khuôn mặt được train dựa trên kiến trúc mạng VGG16

9.9.1 Giới thiệu về mô hình VGG16

VGG16 là một mô hình mạng nơ-ron tích chập (CNN) do K. Simonyan và A. Zisserman đến từ Đại học Oxford đã đề xuất. Mô hình VGG16 có thể đạt được độ chính xác kiểm tra 92,7% trong ImageNet, một bộ dữ liệu chứa hơn 14 triệu hình ảnh đào tạo trên 1000 lớp đối tượng.



Hình 28: Kiến trúc mô hình VGG16

VGG16, như tên gọi của nó, là một mạng nơ-ron bao gồm 16 lớp. Mô hình mạng này tương đối lớn lớn với tổng cộng 138 triệu tham số. Dù có nhiều tham số như vậy, mô hình này có kiến trúc khá đơn giản. Ảnh đầu vào với kích thước 224x224 (quy ước của ImageNet) được đưa qua hệ thống các lớp tích chập và Max Pooling có kích thước giảm dần với số lượng các lớp lần lượt là 64, 128, 256, 512. Có 13 lớp tích chập với kích thước bộ lọc là 3x3 và sử dụng hàm kích hoạt ReLU. Sau mỗi lớp tích chập, có một lớp chuẩn hóa theo batch (batch normalization) để tăng tốc độ huấn luyện và giảm thiểu hiện tượng quá khớp (overfitting). Sau mỗi khối gồm nhiều lớp tích chập liên tiếp, có một lớp Max Pooling để giảm kích thước của đầu vào và trích xuất các đặc trưng quan trọng. Đặc trưng sau khi qua các lớp trên sẽ được đưa qua lớp Average Pooling và các lớp Fully Connected với hàm kích hoạt là softmax. Đầu ra của hàm softmax chính là xác suất ảnh thuộc về từng lớp ứng với các lớp trong bài toán phân loại.

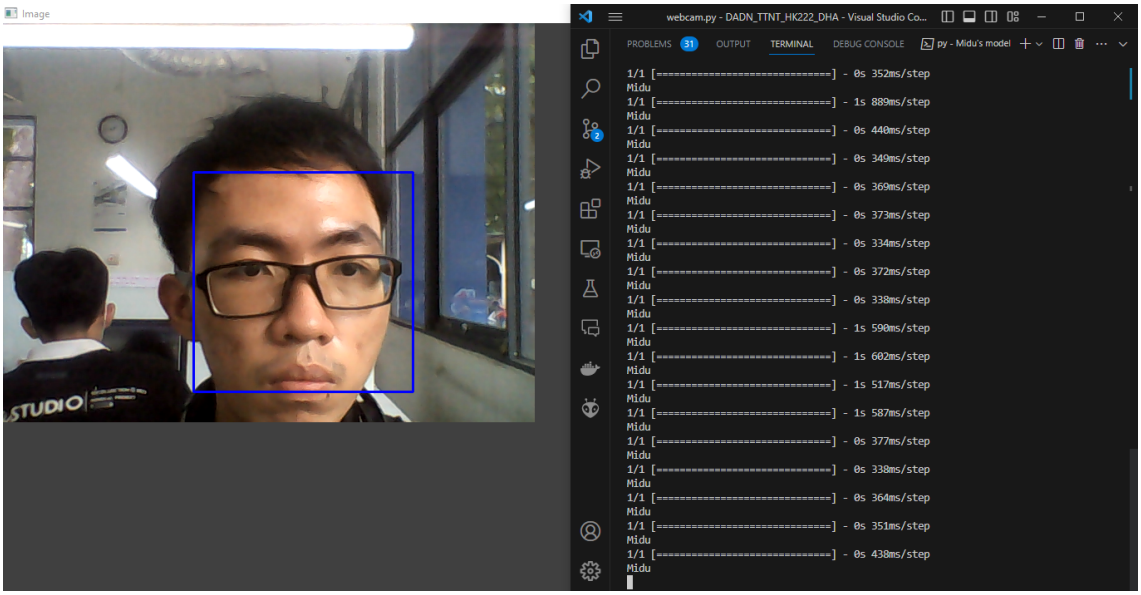
So với các mô hình tiền nhiệm ví dụ như AlexNet, VGG cho ra kết quả tốt hơn hẳn với tỉ lệ lỗi ít hơn. Tuy nhiên, hạn chế của mô hình này là thời gian train khá lâu và kích thước mô hình sẽ lớn nếu như bài toán phân loại phức tạp.

9.9.2 Quá trình hiện thực và kết quả

Do bài toán phân loại của nhóm không quá phức tạp, nhóm quyết định sử dụng kiến trúc của mô hình VGG16 cho bài toán nhận diện khuôn mặt. Nhóm sử dụng phương pháp huấn luyện Transfer Learning, dùng bộ hệ số VGGFace. Phương pháp này giúp giảm thời gian huấn luyện do không phải huấn luyện lại mô hình từ đầu nhưng vẫn sẽ giữ được độ chính xác ở mức tốt. Nhóm chỉ huấn luyện lại lớp Average Pooling và các lớp Dense để phù hợp với bài toán lần này.

Các ảnh đầu vào được lấy và phân loại thành từng thư mục ứng với các lớp thông qua việc gọi script *createData.py*. Các ảnh trên sẽ được tiền xử lí sao cho phù hợp với yêu cầu đầu vào của mô hình. Do tập dữ liệu có kích thước hạn chế nên nhóm có áp dụng các phép augmentation để làm giàu, giảm nguy cơ dẫn đến overfit. Sau khi chuẩn bị xong tập dữ liệu, việc huấn luyện mô hình sẽ được thực hiện thông qua việc gọi script *train.py*. Kết quả thu được của mô hình khá tốt với accuracy khoảng 97.6% và loss là 0.0877 ở epoch cuối cùng. Thực tế cho thấy, với ngưỡng chấp nhận kết quả là 80%, mô hình cho ra kết quả gần như tuyệt đối đối với các lớp có ảnh đầu vào chất lượng tốt (chứa cả đầy đủ khuôn mặt, không bị mờ, nhòe). Đối với các lớp có ảnh đầu vào chất lượng không tốt, độ chính xác của mô hình có bị giảm đi ít nhiều.

Mô hình sau khi được huấn luyện xong sẽ được sử dụng để nhận diện khuôn mặt thông qua việc gọi script *webcam.py*. Kết quả nhận diện khuôn mặt được gửi lên server Adafruit sử dụng giao thức MQTT để hỗ trợ việc điều khiển thiết bị.



Hình 29: Module AI trong khi chạy thực tế

10 Kết luận

Thông qua đồ án lần này, nhóm đã thực hiện được thành công **Hệ thống hỗ trợ mở khóa cửa thông minh** với đầy đủ các chức năng như đã đặc tả. Qua đó học hỏi thêm được nhiều kiến thức mới về tổng quan và cách xây dựng một hệ thống IoT, cách huấn luyện mô hình học sâu để giải quyết bài toán phân loại, cách hiện thực app mobile sử dụng React Native... Bên cạnh đó, chúng em còn rèn luyện được kĩ năng sử dụng LaTeX viết báo cáo, ứng dụng các kiến thức được học trên lớp vào bài tập lớp, rèn luyện kỹ năng làm việc nhóm hiệu quả.

Trong quá trình thực hiện đề tài, mặc dù phải làm việc, trao đổi online khá nhiều, nhưng các thành viên trong nhóm đều cố gắng hết khả năng của mình. Sắp xếp thời gian hợp lý, đúng giờ, nghiêm túc trong các buổi họp, luôn tìm tòi, học hỏi kiến thức để đồ án của nhóm được hoàn thành một cách tốt nhất. Cũng do chỉ trao đổi nhiều thông qua online nên trong quá trình thực hiện đề tài khó tránh khỏi sai sót. Nhóm chúng em rất mong nhận được ý kiến đóng góp của các thầy để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn ở những dự án sắp tới.

Tài liệu

[1] Nhiều tác giả, **Phát triển Gateway IoT bằng Python**, The Dariu Foundation.

[2] OhStem - Khơi nguồn sáng tạo, **YoloFarm Nông nghiệp công nghệ Nông nghiệp dựa trên AI và IoT**