

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA



ĐỒ ÁN ĐA NGÀNH - CÔNG NGHỆ PHẦN MỀM

SMART LIGHT & SMART FAN CHO SMART HOME

GVHD : Trần Thị Quế Nguyệt
SV thực hiện : Ngô Vũ Anh Khoa – 2011403
Lê Công Cường – 2010973
Nguyễn Đình Thi – 2012085
Nguyễn Quang Vinh – 2010430
Lê Nguyễn Đức Huy – 2010285

Tp. Hồ Chí Minh, Tháng 12/2023

Mục lục

I	Phân công	3
II	Giới thiệu:	3
III	Danh sách các thiết bị:	5
1	Thiết bị Input	5
2	Thiết bị Output	6
IV	Yêu cầu của người dùng:	6
V	Yêu cầu chức năng:	7
1	Khía cạnh hệ thống	7
2	Khía cạnh người dùng	8
VI	Yêu cầu phi chức năng:	8
1	Khía cạnh hệ thống	8
2	Khía cạnh người dùng	8
VII	Use-case:	9
1	Use-case scenario:	9
1.1	Use-case 1: Hiển thị trạng thái của đèn & quạt trên ứng dụng .	9
1.2	Use-case 2: Sử dụng ứng dụng để điều khiển độ sáng của đèn & tốc độ quay của quạt	10
1.3	Use-case 3: Cảm biến hồng ngoại nhận diện con người	11
1.4	Use-case 4: Nhận diện sự hiện diện của con người thông qua cảm biến khoảng cách	12
1.5	Use-case 5: Tự động điều khiển quạt và đèn tùy thuộc vào môi trường xung quanh	13
1.6	Use-case 6: Hẹn giờ điều khiển đèn & quạt trên ứng dụng . . .	15
2	Use-case diagram:	17
VIII	Kiến trúc hệ thống:	18

1	Tổng quan hệ thống:	18
1.1	Front-end:	18
1.2	Back-end & Database:	19
1.3	Services:	19
1.4	Gateway:	20
2	Định nghĩa Cơ sở dữ liệu:	21
2.1	Lịch hẹn - Schedules:	21
2.2	Ngưỡng - Thresholds:	22
IX	Back-end Implementation	23
1	Model	23
1.1	Threshold - Ngưỡng:	23
1.2	Schedule - Lịch trình:	26
2	API endpoint	28
X	Front-end Implementation	28
XI	Design Pattern:	31
XII	Các màn hình thực tế:	33
1	Home Screen:	33
2	Schedule:	35
3	Statistic:	38
4	Threshold:	41
XIII	Phụ lục:	46
1	AdafruitIO Feed:	46
2	Thiết kế OhStem:	47
3	Github Repo:	47

I Phân công

Thành viên	Nhiệm vụ	Đóng góp
Lê Nguyễn Đức Huy	Lắp mạch Microbit, Set up adafruitIO, lập trình gateway, vẽ usecase diagram tổng hợp	100%
Ngô Vũ Anh Khoa	Code front-end phần Home Screen, Schedule và Edit Schedule; Vẽ figma; viết usecase scenario	100%
Lê Công Cường	Code back-end, front-end phần Statistics, Threshold và Edit Threshold; Viết service; deploy CSDL; xây dựng kiến trúc hệ thống	100%
Nguyễn Đình Thi	Vẽ figma, viết giới thiệu, viết usecase scenario	60%
Nguyễn Quang Vinh	Vẽ figma, viết yêu cầu chức năng/phi chức năng, viết usecase scenario	50%

II Giới thiệu:

Trong thời đại công nghệ 4.0, IoT (Internet of Things) là một khái niệm ngày càng phổ biến và quan trọng. IoT là sự kết nối của các thiết bị thông minh với nhau và với internet, cho phép chúng giao tiếp, thu thập và trao đổi dữ liệu. IoT có thể ứng dụng vào nhiều lĩnh vực khác nhau, trong đó có lĩnh vực nhà thông minh (smart home).

Nhà thông minh là một hệ thống gồm nhiều thiết bị điện tử, điện gia dụng, an

ninh, chiếu sáng, điều hòa không khí và các thiết bị khác được kết nối với nhau qua mạng không dây hoặc có dây, cho phép người dùng điều khiển và quản lý chúng từ xa qua smartphone, tablet, laptop hoặc các thiết bị khác. Nhà thông minh mang lại nhiều lợi ích cho người dùng, như tiết kiệm năng lượng, tăng hiệu quả sử dụng, nâng cao an toàn và an ninh, tạo ra môi trường sống thoải mái và tiện nghi.

Trong số các thiết bị nhà thông minh, smart light (đèn thông minh) và smart fan (quạt thông minh) là hai thiết bị phổ biến và cần thiết. Smart light là đèn có khả năng điều chỉnh độ sáng, màu sắc, chế độ nhấp nháy và thời gian bật tắt theo ý muốn của người dùng hoặc theo các kịch bản được lập trình sẵn. Smart fan là quạt có khả năng điều chỉnh tốc độ, hướng gió, chế độ xoay và thời gian bật tắt theo ý muốn của người dùng hoặc theo các kịch bản được lập trình sẵn. Cả hai thiết bị này đều có thể kết nối với internet và được điều khiển từ xa qua smartphone hoặc các thiết bị khác.

Mục tiêu của đề tài này là thiết kế và xây dựng một hệ thống smart light và smart fan cho smart home, sử dụng công nghệ IoT. Hệ thống này sẽ bao gồm các thành phần sau:

- Các thiết bị smart light và smart fan, được trang bị các cảm biến và vi điều khiển để nhận và gửi dữ liệu qua mạng wifi.
- Một ứng dụng di động, được cài đặt trên smartphone của người dùng, để điều khiển và quản lý các thiết bị smart light và smart fan, cũng như hiển thị các thông tin về trạng thái, nhiệt độ, độ ẩm, ánh sáng và tiêu thụ năng lượng của chúng.
- Một máy chủ đám mây, được kết nối với internet, để lưu trữ và xử lý các dữ liệu từ các thiết bị smart light và smart fan

III Danh sách các thiết bị:

1 Thiết bị Input

1. *Cảm biến hồng ngoại*

- **Đặc tính:** Phát hiện vật cản hiệu quả.
- **Ứng dụng:** Dùng để phát hiện sự hiện diện của con người ở trong nhà.

2. *Camera kết nối cảm biến hồng ngoại*

- **Đặc tính:**
- **Ứng dụng:**

3. *Cảm biến khoảng cách*

- **Đặc tính:** Xác định khoảng cách từ điểm đang xét đến vật thể
- **Ứng dụng:** Kích hoạt các hành động được lập trình trước khi xác định có con người xuất hiện trong tầm hoạt động

4. *Cảm biến nhiệt độ*

- **Đặc tính:** Xác định nhiệt độ môi trường xung quanh
- **Ứng dụng:** Kích hoạt các hành động đèn & quạt được lập trình trước khi xác định nhiệt độ đạt một ngưỡng nào đó

5. *Cảm biến độ ẩm*

- **Đặc tính:** Xác định độ ẩm môi trường xung quanh
- **Ứng dụng:** Kích hoạt các hành động đèn & quạt được lập trình trước khi xác định có con người xuất hiện trong tầm hoạt động

6. *Yolo:Bit mainboard*

- **Đặc tính:** Đây là vi điều khiển chính.

- **Ứng dụng:** Kết nối với và nhận tín hiệu từ tất cả các thiết bị cảm biến. Đồng thời đảm nhiệm việc kết nối giữa server với các thiết bị quản lý (smartphone)

2 Thiết bị Output

1. Quạt mini

- **Đặc tính:** Thiết bị điện tử phổ biến trong đời sống, khi được cung cấp điện sẽ làm xoay trục động cơ, từ đó tạo nên nhiều ứng dụng khác nhau.
- **Ứng dụng:** Mô phỏng quạt trong nhà mà được kết nối với ứng dụng Smart Home

2. Đèn LED

- **Đặc tính:** Thiết bị điện tử phổ biến trong đời sống, khi được cung cấp điện sẽ tạo ra ánh sáng.
- **Ứng dụng:** Dùng để mô phỏng đèn trong nhà mà được kết nối với ứng dụng Smart Home

IV Yêu cầu của người dùng:

Trong Smart House, đối với người dùng cuối, đèn thông minh và quạt thông minh phải vận hành đúng theo kỳ vọng của người dùng cho một thiết bị thông minh tiêu chuẩn, bao gồm các tiêu chí:

1. Điều khiển từ xa: Người dùng có thể điều khiển quạt và đèn thông qua ứng dụng điện thoại hoặc remote từ xa.
2. Điều chỉnh tốc độ quay của quạt: Người dùng có thể điều chỉnh tốc độ quay của quạt theo các mức độ khác nhau.

3. Điều chỉnh cường độ ánh sáng: Người dùng có thể điều chỉnh độ sáng của đèn để tạo không gian phù hợp và đáp ứng đủ nhu cầu đối với môi trường xung quanh.
4. Điều chỉnh theo lịch trình: Người dùng có thể lập lịch trình để tự động bật/tắt quạt và đèn hoặc điều chỉnh mức độ của 2 thiết bị theo thời gian cụ thể.
5. Thông báo trạng thái: Hệ thống có thể thông báo cho người dùng về trạng thái hoạt động của quạt và đèn: trạng thái bật/tắt, tốc độ hiện tại, mức độ đèn sáng,...
6. Tích hợp cảm biến: Hệ thống có thể tích hợp cảm biến để tự động kích hoạt quạt và đèn khi phát hiện sự hiện diện hoặc khi môi trường trở nên thiếu sáng.
7. Cho phép kích hoạt theo điều kiện được cài đặt trước: Khi có một số yếu tố thỏa mãn theo các yếu tố được cài đặt trước, hệ thống có thể kích hoạt quạt và đèn theo các ngưỡng đã được cài đặt.

V Yêu cầu chức năng:

1 Khía cạnh hệ thống

- Tự động điều chỉnh ánh sáng theo độ sáng.
- Duy trì điều kiện ánh sáng lý tưởng trong nhà theo các cách khác nhau như thủ công, tự động hoặc định kì theo lịch trình.
- Báo cáo các vấn đề cần can thiệp thủ công cho người dùng như khi gặp trục trặc máy móc, ảnh hưởng xấu của thời tiết,...
- Định kì nhắc nhở người dùng kiểm tra kĩ thuật cho các thiết bị cảm biến.
- Các cảm biến sẽ phát hiện xem có người ở trong nhà hay không và báo về cho người dùng

2 Khía cạnh người dùng

- Cho phép người dùng điều chỉnh thủ công độ sáng, tốc độ quạt khi cần.
- Thông báo cho người dùng khi ánh sáng không đạt điều kiện lý tưởng.
- Người dùng có thể lên lịch điều chỉnh độ sáng như khi nào cần sử dụng hệ thống để điều chỉnh độ sáng lý tưởng.

VI Yêu cầu phi chức năng:

1 Khía cạnh hệ thống

- Hệ thống hoạt động liên tục 24/7.
- Tốc độ phản hồi nhanh ($<200\text{ms}$).
- Hệ thống chỉ cho phép duy nhất 1 người (admin) thực hiện thao tác thiết lập, chỉnh sửa thông tin thiết bị.
- Hệ thống cho phép tối đa 5 người (không tính admin) được quyền truy xuất thông tin của hệ thống.

2 Khía cạnh người dùng

- Giao diện ứng dụng thiết kế đơn giản, trực quan, dễ tiếp cận.
- Người dùng biết và sử dụng thuần thục các chức năng của ứng dụng sau 10 phút training.
- Mọi công việc thiết lập, chỉnh sửa phải được tối giản hóa (thực hiện dưới 4 thao tác).

VII Use-case:

1 Use-case scenario:

1.1 Use-case 1: Hiển thị trạng thái của đèn & quạt trên ứng dụng

Use case ID	1
Use case	Hiển thị trạng thái của đèn & quạt trên ứng dụng
Actor	Người dùng
Description	Người dùng xem được trạng thái hiện tại như độ sáng đèn và tốc độ quay của quạt trong ứng dụng
Trigger	Người dùng mở ứng dụng
Preconditions	Người dùng đang ở màn hình chính điện thoại (không phải màn hình chính ứng dụng)
Postconditions	Người dùng vào được trang mặc định/trang chủ/trang home của ứng dụng
Main Flow	1. Người dùng truy cập vào ứng dụng 2. Hệ thống điều hướng người dùng vào trang chủ của ứng dụng và hiện trạng thái của đèn và quạt trên ứng dụng
Exception Flow	Ở bước 2, nếu người chưa đăng nhập, hệ thống gửi thông báo và yêu cầu người dùng đăng nhập lại
Alternative Flow	2.1. Người dùng chọn hình thức đăng nhập bằng gmail 2.2. Hệ thống gửi mã xác nhận về tài khoản của người dùng 2.3. Người dùng nhập mã xác nhận vào giao diện và ấn nút xác nhận 2.4. Hệ thống xác nhận thành công và cho phép người dùng sử dụng ứng dụng

1.2 Use-case 2: Sử dụng ứng dụng để điều khiển độ sáng của đèn & tốc độ quay của quạt

Sử dụng ứng dụng để điều khiển độ sáng của đèn

UJsecase ID	2.1
Use-case	Sử dụng ứng dụng để điều khiển độ sáng của đèn
Diễn viên	Người dùng
Mô tả	Ngoài việc sử dụng cảm biến ánh sáng để tự động cập nhật, hệ thống của chúng tôi cho phép người dùng thay đổi mức độ sáng của đèn thông qua giao diện người dùng
Kích hoạt	Không
Điều kiện trước	Đèn đã được bật và người dùng phải tắt tùy chọn tự động trong cài đặt để thay đổi độ sáng một cách thủ công
Điều kiện sau	Độ sáng của đèn thay đổi theo mức độ của ứng dụng
Luồng chính	1. Người dùng phải tắt chế độ tự động nhận tín hiệu từ cảm biến 2. Hệ thống cung cấp hai tùy chọn cho người dùng: một thanh trượt có 7 mức từ Độ sáng Ấm nhất đến Độ sáng Lạnh nhất cho người dùng hoặc một nút cho chế độ đèn đêm. 3 Người dùng có thể chọn chế độ đèn đêm giảm ánh sáng xanh hoặc thanh trượt để chọn mức độ sáng phù hợp 4 Màu sắc và độ sáng của đèn thay đổi tương ứng với yêu cầu của người dùng.

Luồng ngoại lệ	Ở bước 1, nếu người dùng không tắt chế độ tự động, người dùng không thể thay đổi độ sáng một cách thủ công
Luồng thay thế	Không

Sử dụng ứng dụng để điều khiển tốc độ quay của quạt

Usecase ID	2.2
Use-case	Sử dụng ứng dụng để điều khiển tốc độ quay của quạt
Diễn viên	Người dùng
Mô tả	Cho phép người dùng thay đổi mức độ quay của quạt thông qua giao diện người dùng
Kích hoạt	Không
Điều kiện trước	Không
Điều kiện sau	Mức độ quay của quạt thay đổi theo mức độ của ứng dụng
Luồng chính	1. Hệ thống cung cấp một tùy chọn cho người dùng: một thanh trượt có 5 mức từ Quay Chậm nhất đến Quay Nhanh nhất cho người dùng. 3 Người dùng có thể chọn mức độ quay phù hợp nhất. 4 Mức độ quay của quạt thay đổi tương ứng với yêu cầu của người dùng.
Luồng ngoại lệ	Không
Luồng thay thế	Không

1.3 Use-case 3: Cảm biến hồng ngoại nhận diện con người

Usecase ID	3
-------------------	----------

Usecase	Nhận diện sự hiện diện của con người qua cảm biến hồng ngoại (camera kết nối cảm biến hồng ngoại)
Actor	Cảm biến hồng ngoại (camera kết nối cảm biến hồng ngoại)
Description	Phát hiện sự hiện diện của con người ở trong nhà
Trigger	None
Precondition	1. Người dùng đã thiết lập trước các chức năng sẽ thực hiện khi phát hiện thành công sự hiện diện (bật/tắt đèn, bật/tắt quạt) 2. Cảm biến hồng ngoại đã được kết nối với hệ thống và đang hoạt động
Postconditions	None
Main Flow	1. Cảm biến hồng ngoại ở trạng thái standby cho tới khi phát hiện sự hiện diện con người 2. Cảm biến hồng ngoại phát hiện ra hiện diện con người trong nhà 3. Cảm biến gửi tín hiệu tới hệ thống 4. Hệ thống thực hiện các chức năng liên quan (bật/tắt đèn, bật/tắt quạt,...)
Exception Flow	Exception xảy ra tại bước 3 - Nếu qua trình gửi tín hiệu về hệ thống có vấn đề, flow sẽ không thể tiếp tục
Alternative Flow	None

1.4 Use-case 4: Nhận diện sự hiện diện của con người thông qua cảm biến khoảng cách

Usecase ID	4
------------	---

Use case name	Nhận diện sự hiện diện của con người thông qua cảm biến khoảng cách
Actor	Cảm biến khoảng cách
Description	Phát hiện sự hiện diện của con người trong một khoảng cách nhất định
Trigger	Khi có con người xuất hiện trong phạm vi của cảm biến trong một khoảng cách nhất định
Precondition	Cảm biến đã được kết nối với hệ thống và đang hoạt động
Postconditions	Hệ thống đèn, quạt được khởi động
Main Flow	1. Cảm biến phát hiện ra sự hiện diện của con người trong một khoảng cách được cài đặt trước đó 2. Cảm biến gửi tín hiệu tới hệ thống đèn, quạt 3. Hệ thống đèn, quạt tự động khởi động
Exception Flow	Ở bước 2, Nếu qua trình gửi tín hiệu về hệ thống có vấn đề, flow sẽ không thể tiếp tục
Alternative Flow	Không

1.5 Use-case 5: Tự động điều khiển quạt và đèn tùy thuộc vào môi trường xung quanh

Use case ID	5.1
Use case	Tự động điều khiển quạt và đèn tùy thuộc vào môi trường (trong trường hợp có người)
Actor	Quạt, đèn, camera hồng ngoại cảm biến
Description	Hệ thống tự động khởi động quạt, đèn khi phát hiện có người ở nhà
Trigger	Không

Preconditions	Cảm biến hồng ngoại ở trạng thái standby cho tới khi phát hiện sự hiện diện con người
Postconditions	Hệ thống đèn, quạt được khởi động
Main Flow	<ol style="list-style-type: none"> 1. Cảm biến hồng ngoại phát hiện ra hiện diện con người trong nhà 2. Cảm biến gửi tín hiệu tới hệ thống đèn, quạt 3. Hệ thống quạt, đèn tự động khởi động
Exception Flow	Ở bước 2, Nếu qua trình gửi tín hiệu về hệ thống có vấn đề, flow sẽ không thể tiếp tục
Alternative Flow	Không

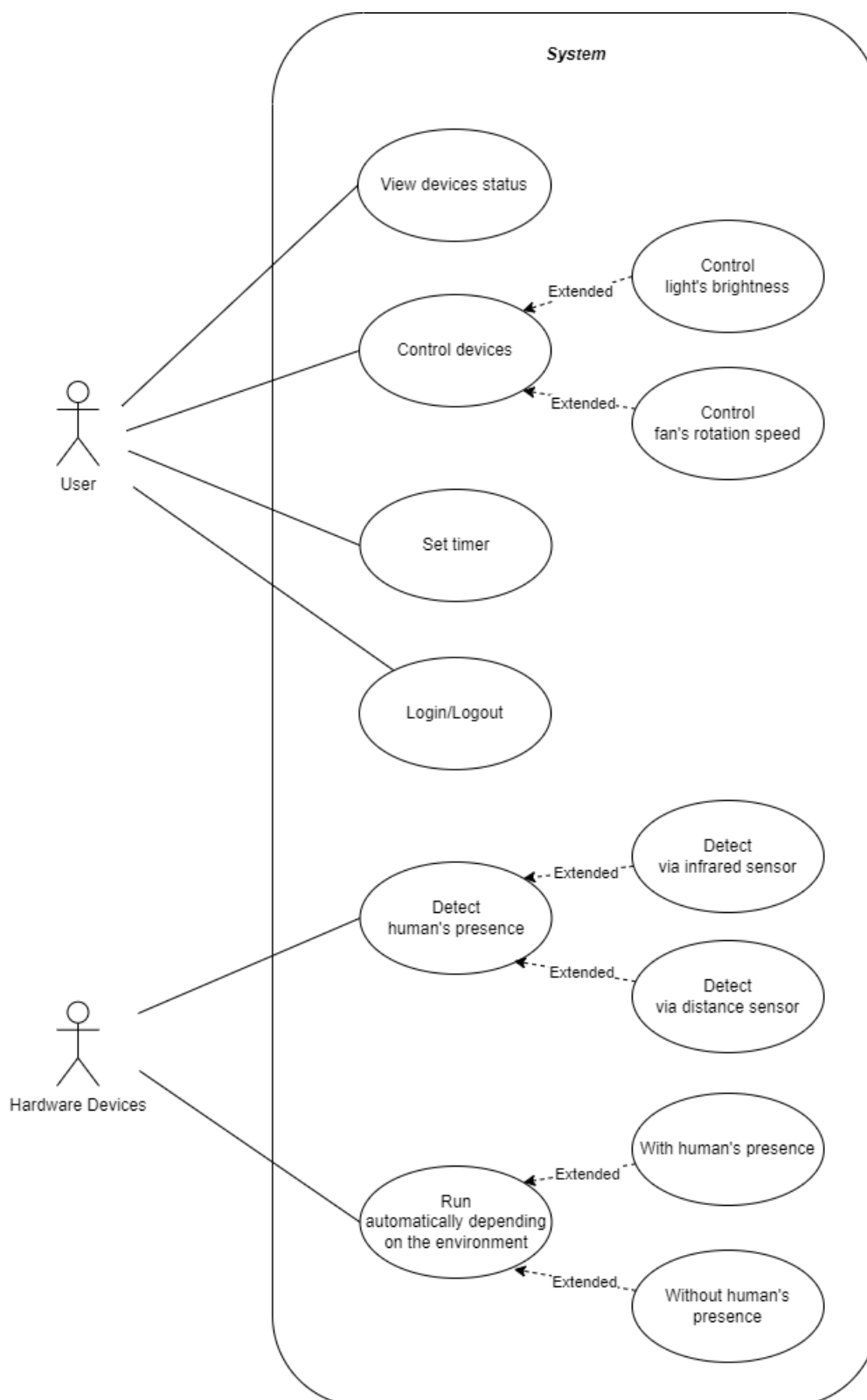
Use case ID	5.2
Use case	Tự động điều khiển quạt và đèn tùy thuộc vào môi trường (trong trường hợp không có người)
Actor	Quạt, đèn, camera hồng ngoại cảm biến
Description	Hệ thống tự động tắt các quạt, đèn khi phát hiện có người ở nhà
Trigger	Không
Preconditions	Cảm biến hồng ngoại ở trạng thái standby cho tới khi không phát hiện sự hiện diện con người
Postconditions	Hệ thống đèn, quạt được tắt
Main Flow	<ol style="list-style-type: none"> 1. Cảm biến hồng ngoại phát hiện ra không hiện diện con người trong nhà 2. Cảm biến gửi tín hiệu tới hệ thống đèn, quạt 3. Hệ thống quạt, đèn tự động tắt
Exception Flow	Ở bước 2, Nếu qua trình gửi tín hiệu về hệ thống có vấn đề, flow sẽ không thể tiếp tục
Alternative Flow	Không

1.6 Use-case 6: Hẹn giờ điều khiển đèn & quạt trên ứng dụng

Use case ID	6
Use case	Hẹn giờ điều khiển đèn & quạt trên ứng dụng
Actor	Người dùng
Description	Người dùng có thể hẹn giờ để đặt trước các trạng thái như độ sáng đèn và tốc độ quay của quạt trong ứng dụng
Trigger	Người dùng bấm vào mục "Hẹn giờ"
Preconditions	Người dùng đang ở màn hình chính màn hình chính của ứng dụng
Postconditions	Người dùng đặt thành công một bộ hẹn giờ
Main Flow	<ol style="list-style-type: none">1. Người dùng truy cập vào mục "Hẹn giờ"2. Hệ thống hiển thị các bộ hẹn giờ đã được đặt trên ứng dụng3. Người dùng bấm vào nút "Thêm"4. Hệ thống hiển thị các thông số mặc định để tạo một bộ hẹn giờ mới5. Người dùng nhập khung giờ để kích hoạt bộ hẹn giờ, bao gồm nhưng không nhất thiết phải nhập tất cả: giờ, phút, giây, thứ, ngày, tháng, các ngày trong tuần (thứ 2, thứ 3)...6. Người dùng thay đổi độ sáng của đèn và tốc độ quay của quạt7. Người dùng chọn "Lưu"8. Hệ thống trở lại mục "Hẹn giờ" và hiển thị các bộ hẹn giờ với bộ hẹn giờ mới được xuất hiện

Exception Flow	Ở bước 5, nếu người dùng nhập khung giờ trùng với bộ hẹn giờ bất kỳ đã lưu, thì ở bước 7, khi người dùng bấm "Lưu" sẽ hiện thông báo không thể đặt bộ hẹn giờ do xung đột với bộ hẹn giờ khác và trở lại với bước 5.
Alternative Flow	

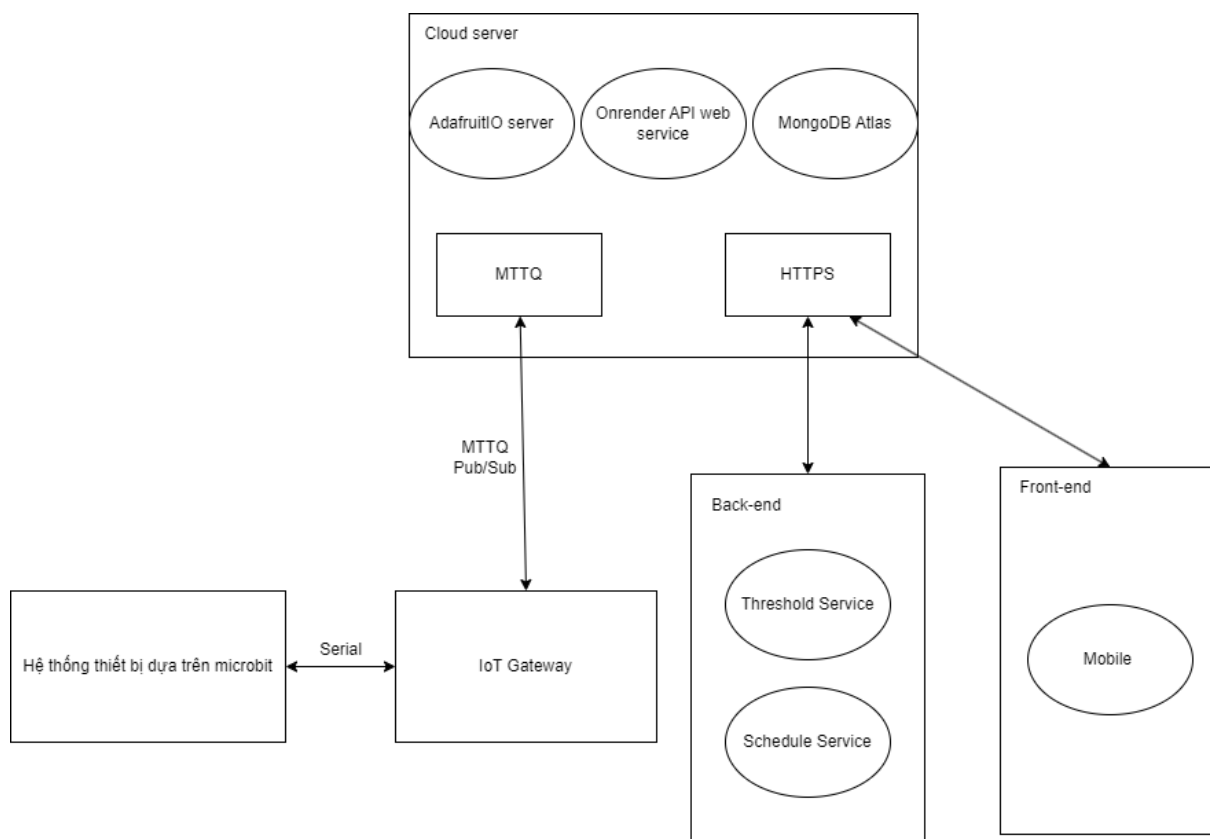
2 Use-case diagram:



Hình 1: Use case diagram

VIII Kiến trúc hệ thống:

1 Tổng quan hệ thống:



Hình 2: Sơ đồ kiến trúc tổng thể hệ thống

1.1 Front-end:

Ứng dụng smart house sẽ được xây dựng bằng React Native, một thư viện JavaScript cho phép bạn tạo ra các ứng dụng di động native cho Android, iOS và nhiều hơn nữa bằng cách sử dụng React. React Native kết hợp những phần tốt nhất của phát triển native (native development) với React, một thư viện JavaScript phổ biến trong việc xây dựng giao diện người dùng.

1.2 Back-end & Database:

Ứng dụng này được viết bằng ExpressJS, một framework ứng dụng web back-end cho Node.js. ExpressJS bắt đầu bằng việc tạo API đơn giản trong khi thêm và phát triển các thành phần và tính năng mạnh mẽ hơn tùy thuộc vào trường hợp sử dụng của ứng dụng.

Ứng dụng này sử dụng CSDL MongoDB, một cơ sở dữ liệu NoSQL dựa trên tài liệu (record) phổ biến, dễ sử dụng, cung cấp lưu trữ hiệu quả và linh hoạt cho nhiều loại tập dữ liệu khác nhau. MongoDB cho phép hỗ trợ các tình huống sử dụng giao dịch (transaction), tìm kiếm, phân tích trong khi sử dụng giao diện truy vấn chung.

1.3 Services:

1. **Threshold Service:** Service này có tác dụng chạy để nhận các ngưỡng dữ liệu được người dùng cấu hình trước. Nó sẽ liên tục kiểm tra xem các điều kiện có thỏa mãn ngưỡng này để có thể kích hoạt các hành vi được cài đặt sẵn đối với đèn và quạt hay không.

Các tiêu chí để đánh giá ngưỡng có thỏa mãn hay không bao gồm:

- Ngưỡng đó có được người dùng bật hay không
- Nhiệt độ cao hơn nhiệt độ ngưỡng
- Độ ẩm cao hơn độ ẩm ngưỡng
- Khoảng cách (đo được từ cảm biến khoảng cách) nhỏ hơn khoảng cách ngưỡng
- Sự hiện diện của con người (đo được từ cảm biến hồng ngoại) trùng với ngưỡng hiện diện

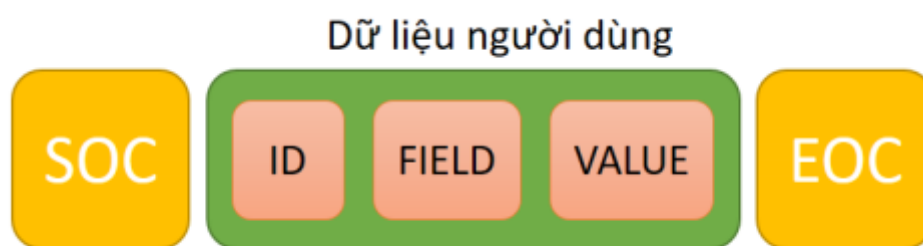
Khi ngưỡng được thỏa mãn, hệ thống sẽ gửi tín hiệu lên AdafruitIO nhằm gửi dữ liệu được về đèn với quạt có trong ngưỡng đó, từ đó AdafruitIO sẽ gửi tín hiệu về mạch thiết bị và gửi tín hiệu về cho đèn với quạt trên mạch đó.

2. **Schedule Service:** Service này có tác dụng chạy để nhận các lịch hẹn được lên lịch trước. Nó sẽ liên tục kiểm tra xem lịch hẹn được kích hoạt đã tới thời điểm khởi chạy chưa nhằm kích hoạt các hành vi được cài đặt sẵn đối với đèn và quạt.

Khi lịch được báo hiệu và kích hoạt, hệ thống sẽ gửi tín hiệu lên AdafruitIO nhằm gửi dữ liệu được về đèn với quạt có trong ngưỡng đó, từ đó AdafruitIO sẽ gửi tín hiệu về mạch thiết bị và gửi tín hiệu về cho đèn với quạt trên mạch đó.

1.4 Gateway:

Nhóm có xây dựng Gateway IoT dựa trên ngôn ngữ Python, kết nối Yolo:Bit và Dashboard của Adafruit. Khi Yolo:Bit nhận được dữ liệu từ sensor ví dụ như là nhiệt độ hay độ ẩm, Yolo:Bit sẽ tiến hành gửi giá trị đo được lên gateway thông qua cửa sổ Serial. Cụ thể hơn, Yolo:Bit sẽ đóng gói giá trị sensor thành một dataframe có cấu trúc "ID:FIELD:VALUE#", với là SOF và # là EOF, rồi in ra cửa sổ Serial. Gateway sẽ theo dõi cửa sổ Serial, và khi có dòng dữ liệu mới được Yolo:Bit gửi vào, Gateway sẽ phân tách dataframe ra và gửi giá trị VALUE lên feed tương ứng với trường FIELD.



Hình 3: *Format Dataframe*

2 Định nghĩa Cơ sở dữ liệu:

2.1 Lịch hẹn - Schedules:

Trường (Field)	Kiểu dữ liệu	Mô tả
_id	Mongodb Id	ID duy nhất dùng để đánh chỉ mục trên CSDL MongoDB
hour	Number	Giờ
minute	String	Phút
date	Date	Ngày tháng năm
repeat	Array[String]	các ngày lặp lại lịch hẹn
fanDevice	String	Thiết bị quạt dùng để điều khiển, tuân theo feed trên AdafruitIO
lightDevice	String	Thiết bị đèn dùng để điều khiển, tuân theo feed trên AdafruitIO
fanSpeed	Number	Trạng thái của thiết bị quạt khi lịch hẹn tới thời điểm kích hoạt
lightStatus	Bool	Trạng thái của thiết bị đèn khi lịch hẹn tới thời điểm kích hoạt
notification	Bool	Lịch hẹn có đang được bật hay không

Bảng 10: Định nghĩa cho lịch hẹn - Schedules

2.2 Ngưỡng - Thresholds:

Trường (Field)	Kiểu dữ liệu	Mô tả
_id	Mongodb Id	ID duy nhất dùng để đánh chỉ mục trên CSDL MongoDB
fanDevice	String	Thiết bị quạt được dùng để điều khiển, tuân theo feed trên AdafruitIO
lightDevice	String	Thiết bị đèn được dùng để điều khiển, tuân theo feed trên AdafruitIO
tempSensor	String	Cảm biến nhiệt độ được dùng để đo, tuân theo feed trên AdafruitIO
humidSensor	String	Cảm biến độ ẩm được dùng để đo, tuân theo feed trên AdafruitIO
distanceSensor	String	Cảm biến khoảng cách được dùng để đo, tuân theo feed trên AdafruitIO
humid	Decimal128	Ngưỡng độ ẩm được cấu hình
temp	Decimal128	Ngưỡng nhiệt độ được cấu hình
distance	Number	Ngưỡng khoảng cách được cấu hình
pirSensor	String	Cảm biến hồng ngoại được dùng để đo, tuân theo feed trên AdafruitIO
presented	Bool	Ngưỡng có/không có sự hiện diện của con người được cấu hình
lightStatusWhenReached	Bool	Trạng thái của đèn khi ngưỡng thoả mãn
fanSpeedWhenReached	Number	Trạng thái của quạt khi ngưỡng thoả mãn
lightStatusOriginal	Bool	Trạng thái của đèn khi ngưỡng không thoả mãn
fanSpeedOriginal	Number	Trạng thái của quạt khi ngưỡng không thoả mãn
currentState	Bool	Trạng thái ngưỡng có thoả mãn hay không
active	Bool	Ngưỡng có đang được bật hay không

Bảng 11: Định nghĩa cho Ngưỡng - Thresholds

IX Back-end Implementation

1 Model

1.1 Threshold - Ngưỡng:

```
const thresholdSchema = new mongoose.Schema({
  fanDevice: {
    type: String,
    default: "",
  },
  lightDevice: {
    type: String,
    default: "",
  },
  tempSensor: {
    type: String,
    default: "",
  },
  humidSensor: {
    type: String,
    default: "",
  },
  distanceSensor: {
    type: String,
    default: "",
  },
  pirSensor: {
    type: String,
    default: "",
  },
  fanSpeedOriginal: {
    type: Number,
    required: false,
    enum: [0, 20, 40, 60, 80, 100],
  },
});
```



```
    },
    lightStatusOriginal: {
      type: Boolean,
      required: false,
    },
    fanSpeedWhenReached: {
      type: Number,
      required: false,
      enum: [0, 20, 40, 60, 80, 100],
    },
    lightStatusWhenReached: {
      type: Boolean,
      required: false,
    },
    humid: {
      type: "decimal128",
      required: true,
      default: 0.0,
    },
    temp: {
      type: "decimal128",
      required: true,
      default: 0.0,
    },
    distance: {
      type: Number,
      required: true,
      default: 0,
    },
    presented: {
      type: Boolean,
      default: false,
    },
    active: {
      type: Boolean,
```

```
        required: true,
        default: true,
    },
    currentState: {
        type: Boolean,
        required: true,
        default: false,
    },
});
```

```
1  {
2      "_id": "6579c5225962c1b45060ecd0",
3      "fanDevice": "fan-speed",
4      "lightDevice": "light-switch",
5      "tempSensor": "temp",
6      "humidSensor": "humid",
7      "distanceSensor": "distance",
8      "humid": {
9          "$numberDecimal": "60"
10     },
11     "temp": {
12         "$numberDecimal": "38"
13     },
14     "distance": 29,
15     "__v": 0,
16     "active": true,
17     "pirSensor": "pir",
18     "presented": false,
19     "lightStatusWhenReached": true,
20     "fanSpeedWhenReached": 100,
21     "lightStatusOriginal": false,
22     "fanSpeedOriginal": 40,
23     "currentState": false
24 }
```

Listing 1: Ví dụ trả về của 1 bảng record threshold

1.2 Schedule - Lịch trình:

```
const scheduleSchema = new mongoose.Schema({
  hour: {
    type: Number,
    required: true,
  },
  minute: {
    type: Number,
    required: true,
  },
  date: {
    type: Date,
    required: true,
  },
  repeat: {
    type: [String],
    required: true,
    enum: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
  },
  fanDevice: {
    type: String,
    default: "",
  },
  lightDevice: {
    type: String,
    default: "",
  },
  fanSpeed: {
    type: Number,
    required: true,
    enum: [0, 20, 40, 60, 80, 100],
    default: 0,
  },
  lightStatus: {
```

```
        type: Boolean,  
        required: true,  
        default: false,  
    },  
    notification: {  
        type: Boolean,  
        required: true,  
    },  
},  
});
```

```
1  [  
2      {  
3          "_id": "6579c1eea11efb927b9c74b0",  
4          "hour": 13,  
5          "minute": 8,  
6          "date": "2023-12-16T00:00:00.000Z",  
7          "fanDevice": "fan-speed",  
8          "lightDevice": "light-switch",  
9          "fanSpeed": 20,  
10         "lightStatus": true,  
11         "notification": true,  
12         "__v": 0  
13     },  
14     {  
15         "_id": "6579c203a11efb927b9c74b3",  
16         "hour": 13,  
17         "minute": 5,  
18         "date": "2023-12-16T00:00:00.000Z",  
19         "fanDevice": "fan-speed",  
20         "lightDevice": "light-switch",  
21         "fanSpeed": 60,  
22         "lightStatus": false,  
23         "notification": true,  
24         "__v": 0  
25     }  
26 ]
```

Listing 2: Ví dụ trả về của 1 bảng record schedule

2 API endpoint

1. Các API tương tác với bảng Schedule

Method	Endpoint
GET	https://smart-house-api.onrender.com/schedules
GET	https://smart-house-api.onrender.com/schedules/:id
POST	https://smart-house-api.onrender.com/schedules
PUT	https://smart-house-api.onrender.com/schedules/:id
DELETE	https://smart-house-api.onrender.com/schedules/:id

Bảng 12: *API Endpoints cho Schedules*

2. Các API tương tác với bảng Threshold

Method	Endpoint
GET	https://smart-house-api.onrender.com/thresholds
GET	https://smart-house-api.onrender.com/thresholds/:id
POST	https://smart-house-api.onrender.com/thresholds
PUT	https://smart-house-api.onrender.com/thresholds/:id
DELETE	https://smart-house-api.onrender.com/thresholds/:id

Bảng 13: *API Endpoints for Thresholds*

X Front-end Implementation

Các API do AdafruitIO được gọi để hiển thị dữ liệu lên màn hình:

1. Đối với hiển thị thông tin từ cảm biến & thiết bị

Các API này được gọi đồng thời với bước nhảy (interval) 1.5 giây, tức là mỗi 1.5 giây sẽ được gọi một lần để luôn đảm bảo dữ liệu hiển thị được cập nhật sát với thực tế môi trường nhất.

Method	Endpoint
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.fan-speed/data/last
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.light-switch/data/last
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.temp/data/last
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.humid/data/last

Bảng 14: *API Endpoints để lấy dữ liệu gần nhất*

2. Đối với hiển thị các dữ liệu gần đây của cảm biến & thiết bị (Vẽ biểu đồ đường thống kê)

Các API này được gọi đồng thời với bước nhảy (interval) 5 giây.

Method	Endpoint
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.temp/data?limit=20
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.distance/data?limit=20
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.humid/data?limit=20
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.fan-speed/data?limit=20
GET	https://io.adafruit.com/api/v2/dadnhk231nhom9/feeds/group-9.light-switch/data?limit=20

Bảng 15: *API Endpoints để lấy 20 điểm dữ liệu gần nhất (recent data points)*

Đối với các API gọi từ AdafruitIO, Nếu gọi API quá nhiều trong một khoảng thời gian ngắn thì sau một thời gian sẽ bị lỗi HTTP 429 do vượt giới hạn gọi API trong một khoảng thời gian (API rate limiting). Để tránh giới hạn gọi API thì khi người dùng đang ở màn hình nào, chỉ có các API được phục vụ để hiển thị cho màn hình đó sẽ hoạt động, các API ở các màn hình khác sẽ không được gọi cho

đến khi người dùng chuyển qua màn hình đó. Và tất nhiên khi người dùng chuyển qua màn hình khác thì các API được hiện thực cho màn hình này sẽ được khởi động để bắt đầu gọi, và các API của màn hình trước sẽ ngưng hoạt động.

Đối với các API để tương tác với CSDL, mặc dù nền tảng để triển khai lên đám mây (Onrender) cũng có giới hạn cho việc gọi API cũng như giới hạn băng thông (bandwidth) để gửi dữ liệu qua lại giữa CSDL và hệ thống, tuy nhiên giới hạn lại rộng hơn khá nhiều nên việc kiểm soát gọi API không khắt khe như các API bên AdafruitIO

3. Đối với các thao tác CRUD khi đặt lịch hẹn

- **GET:** Lấy và hiển thị thông tin các lịch hẹn
- **POST:** Tạo một lịch hẹn mới
- **PUT/id:** Cập nhật một lịch hẹn cụ thể
- **DELETE/id:** Xoá một lịch hẹn cụ thể
- **GET/id:** Lấy và hiển thị một lịch hẹn cụ thể, thường được dùng để hiển thị thông tin trong màn hình sửa đổi lịch hẹn khi ta muốn sửa một lịch hẹn nào đó

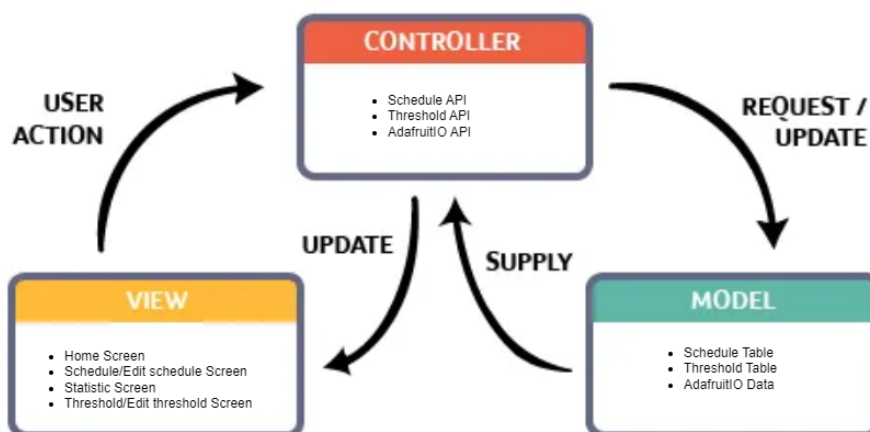
4. Đối với các thao tác CRUD khi cài đặt ngưỡng (threshold)

- **GET:** Lấy và hiển thị thông tin các ngưỡng
- **POST:** Tạo một ngưỡng mới
- **PUT/id:** Cập nhật một ngưỡng cụ thể
- **DELETE/id:** Xoá một ngưỡng cụ thể
- **GET/id:** Lấy và hiển thị một ngưỡng cụ thể, thường được dùng để hiển thị thông tin trong màn hình sửa đổi ngưỡng khi ta muốn sửa một ngưỡng nào đó

XI Design Pattern:

Hệ thống sử dụng mô hình thiết kế MVC. MVC là viết tắt của cụm từ "Model-View-Controller". Đây là mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành 3 phần được kết nối với nhau và mỗi thành phần đều có 1 nhiệm vụ riêng của nó và độc lập với các thành phần khác. Tên gọi 3 thành phần là:

- Model (dữ liệu): Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng và là cầu nối giữa 2 thành phần bên dưới là View và Controller. Một model là dữ liệu được sử dụng bởi chương trình. Đây có thể là cơ sở dữ liệu, hoặc file XML bình thường hay một đối tượng đơn giản.
- View (giao diện): Đây là phần giao diện (theme) dành cho người sử dụng. View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.
- Controller (bộ điều khiển): Là bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View. Một controller bao gồm cả Model lẫn View. Nó nhận input đầu vào và thực hiện các update tương ứng.

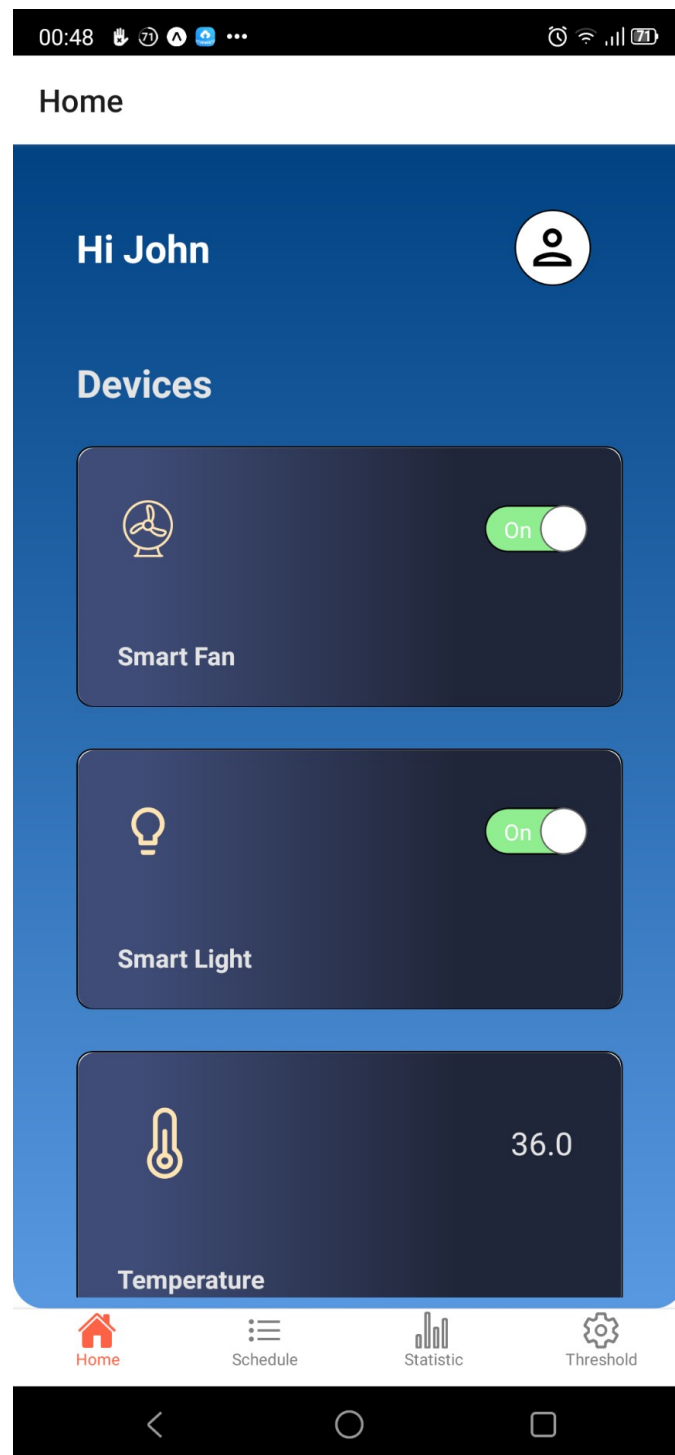


Hình 4: Mô hình thiết kế MVC cụ thể cho hệ thống này

Trong hệ thống này. Dù CSDL có thể được thiết kế để scale các thiết bị & cảm biến để có thể được hiển thị cũng như làm đầu vào để lưu vào các bảng dữ liệu khác như Schedule & Threshold, giao diện hiện tại lại không hỗ trợ việc quản lý thiết bị & cảm biến. Mặc dù vậy hệ thống này vẫn tuân thủ đầy đủ các nguyên tắc thiết kế của MVC với khả năng thêm/sửa/xoá lịch hẹn & ngưỡng.

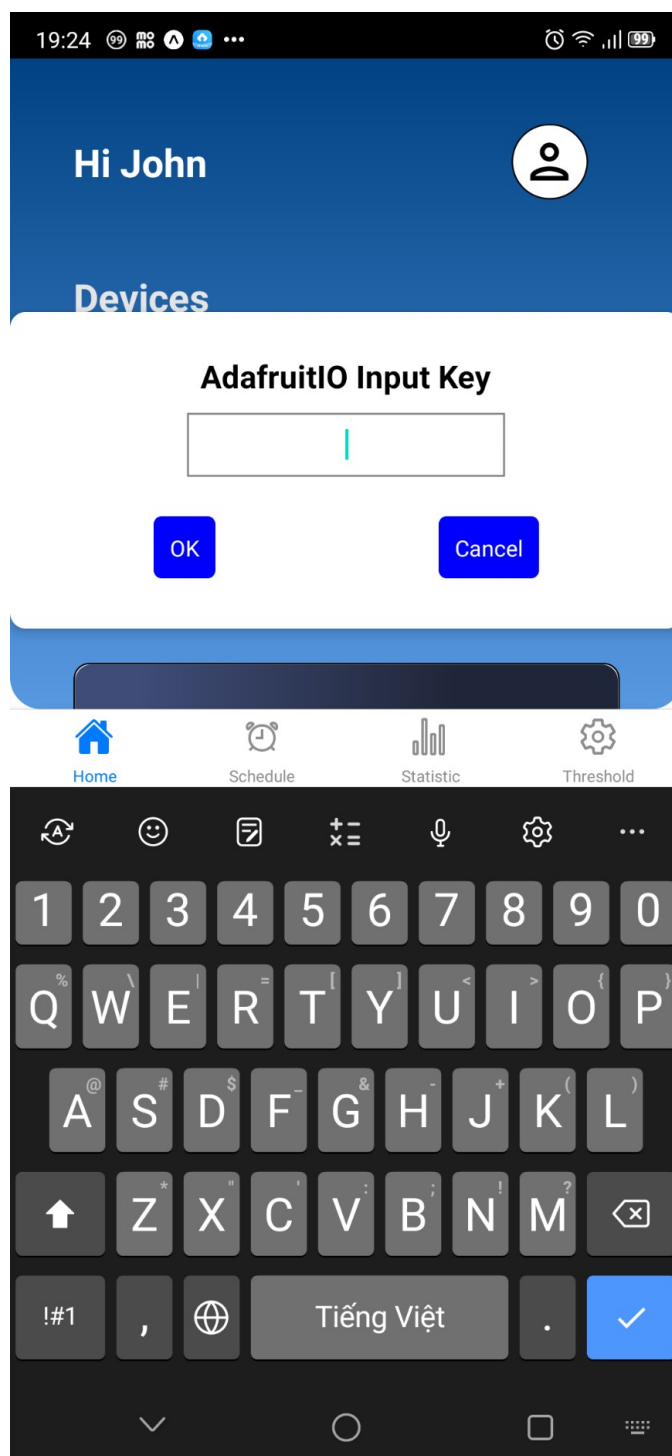
XII Các màn hình thực tế:

1 Home Screen:



Hình 5: Màn hình chính

Ở góc trên bên phải màn hình khi bấm vào ta có thể nhập AdafruitIO Key trên 1 modal. Vì AdafruitIO Key tự động thay đổi sau một khoảng thời gian nên việc nhập key này là cần thiết để hệ thống có thể tiếp tục tương tác được với AdafruitIO



Hình 6: Màn hình hiển thị Modal nhập AdafruitIO Key

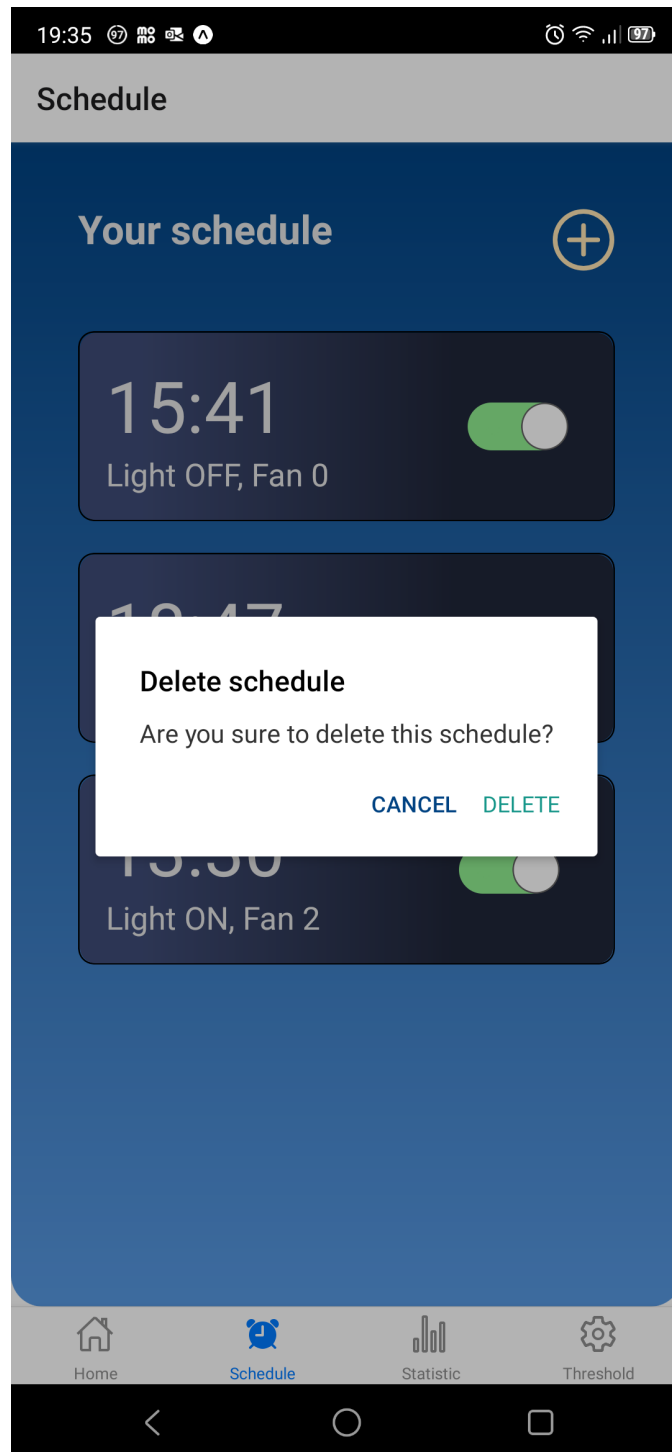
2 Schedule:



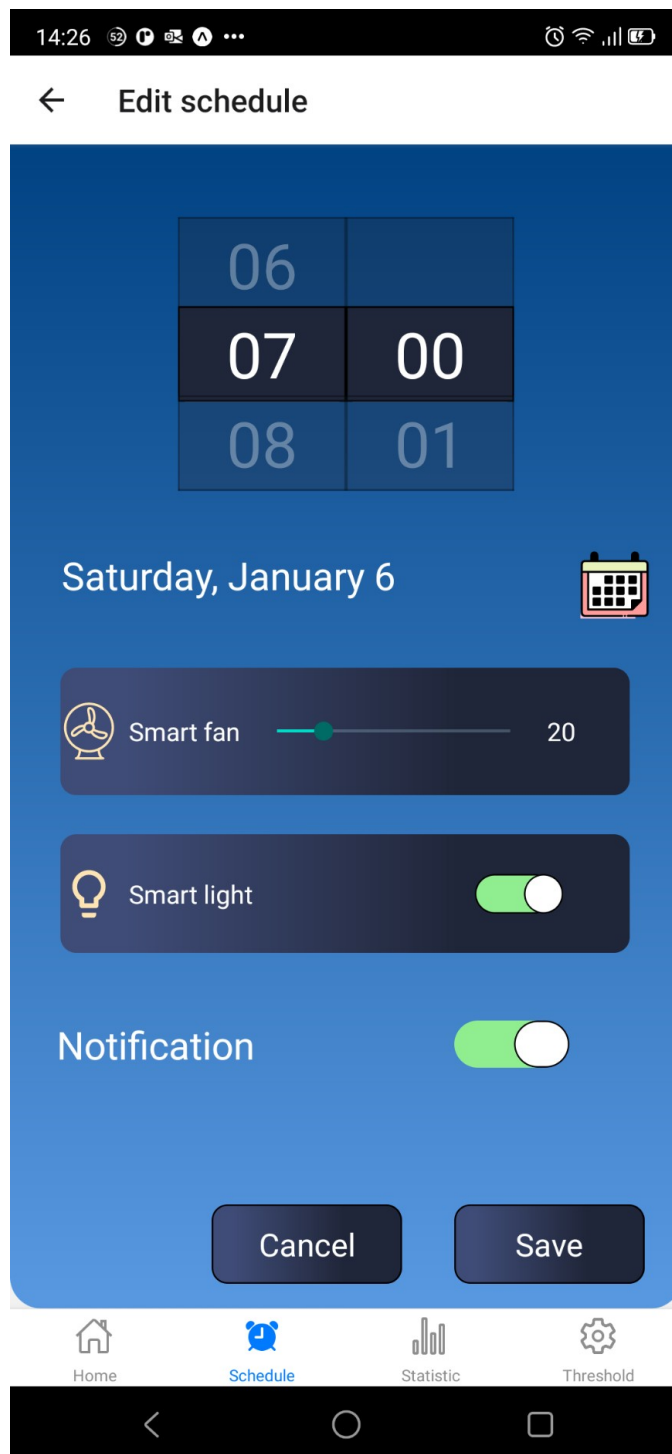
Hình 7: Màn hình hiển thị danh sách lịch hẹn

Có 3 hành động mà ta có thể thực hiện ở màn hình Schedule này:

- Thêm lịch mới: Bấm vào nút trên bên phải
- Sửa lịch: Bấm vào lịch muốn sửa
- Xoá lịch: Bấm và giữ vào lịch muốn xoá, sẽ hiện một dialog xác nhận xoá lịch.
Bấm "Confirm" và lịch được xoá



Hình 8: Màn hình xác nhận xoá lịch hẹn

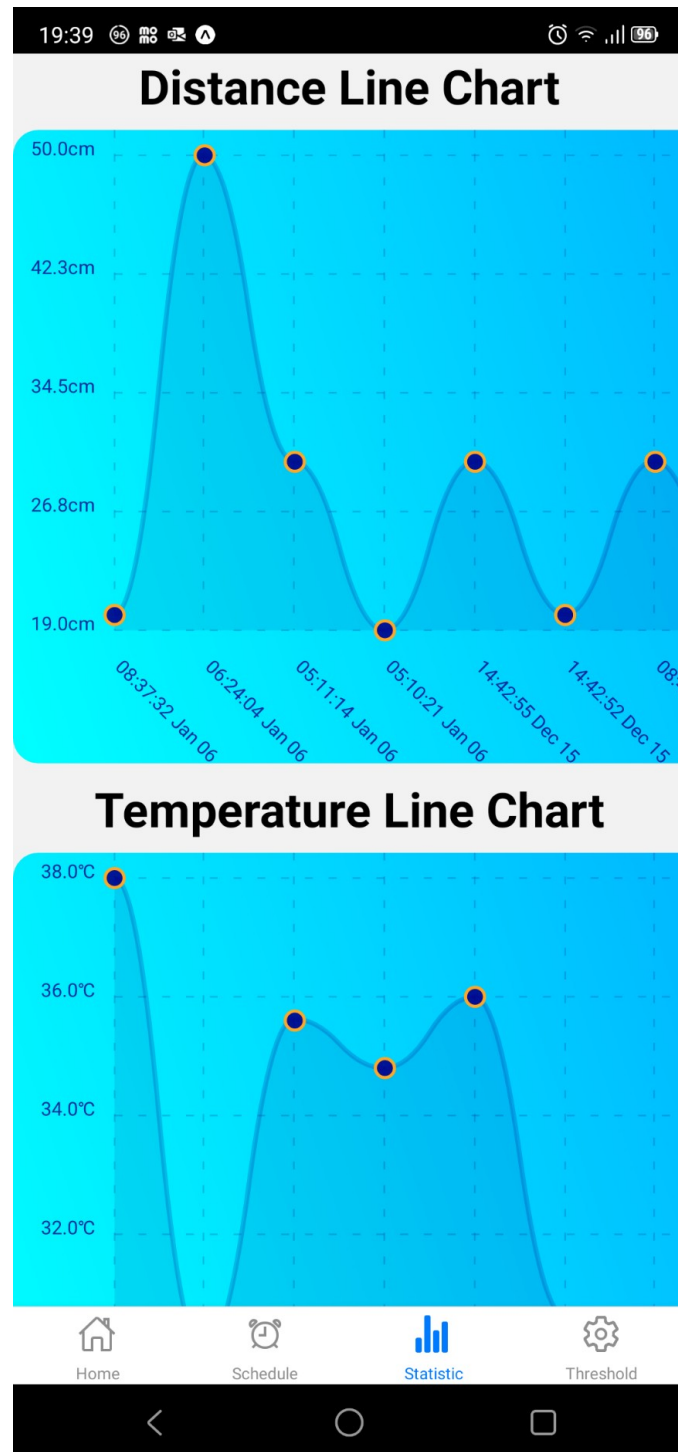


Hình 9: Màn hình chỉnh sửa/Tạo mới lịch hẹn

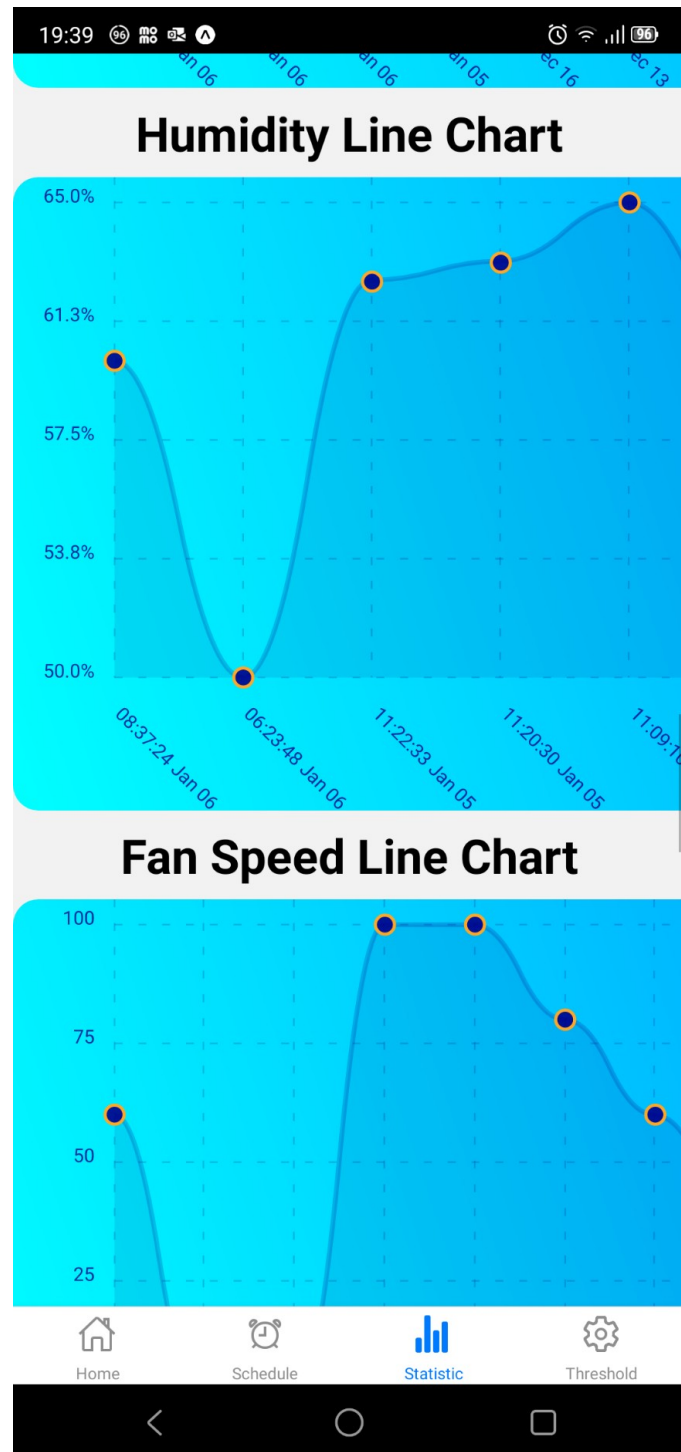
3 Statistic:

Màn hình hiển thị 5 biểu đồ thể hiện 20 điểm dữ liệu gần nhất của các thiết bị và cảm biến bao gồm:

- Đèn
- Quạt
- Nhiệt độ
- Độ ẩm
- Khoảng cách



Hình 10: Màn hình thống kê



Hình 11: Màn hình thống kê

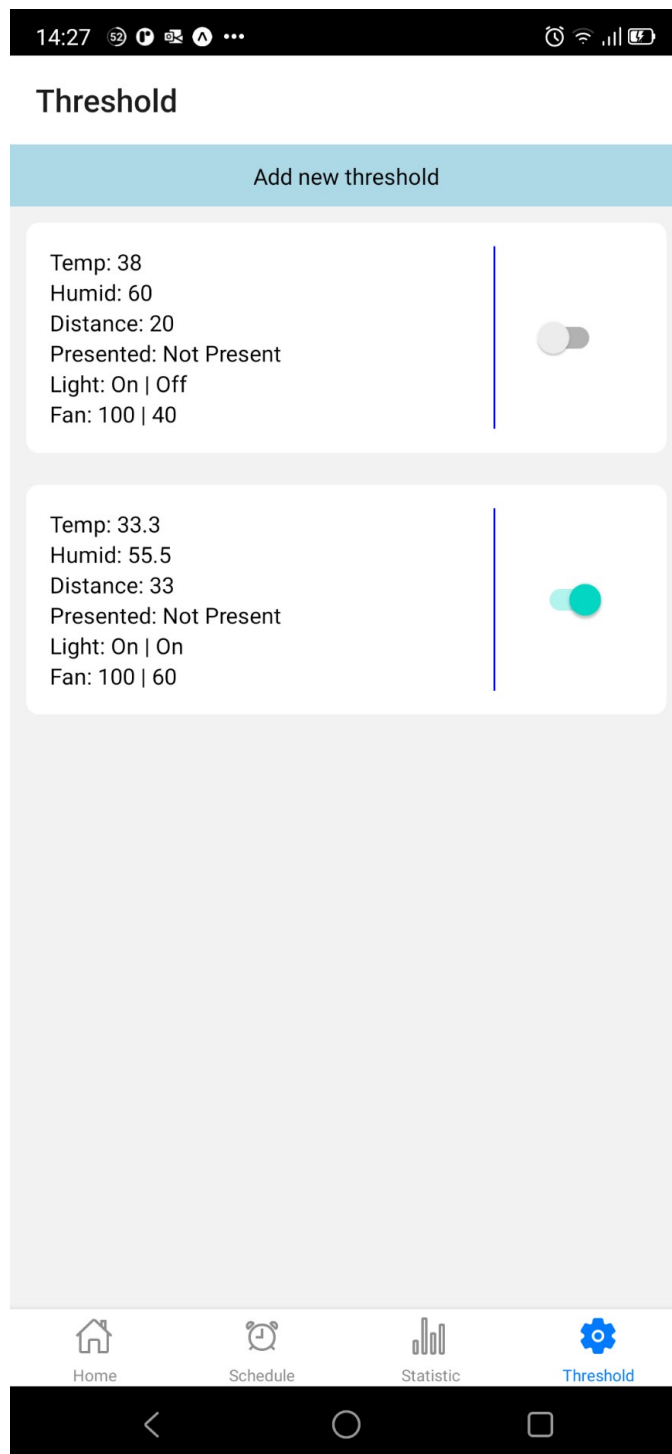
4 Threshold:

Một ngưỡng khi được bật có thể được kích hoạt nếu:

- Nhiệt độ (temp) cao hơn ngưỡng

- Độ ẩm (humid) cao hơn ngưỡng
- Khoảng cách (distance) thấp hơn ngưỡng
- Trùng với sự xuất hiện (presented)

Khi đó ngưỡng sẽ kích hoạt và điều khiển quạt với đèn theo cài đặt có sẵn. Nếu ngưỡng đang được thoả nhưng sau đó 1 hoặc nhiều điều kiện trên không thoả nữa thì khi đó quạt với đèn sẽ được cài đặt theo mức không thoả "Original"

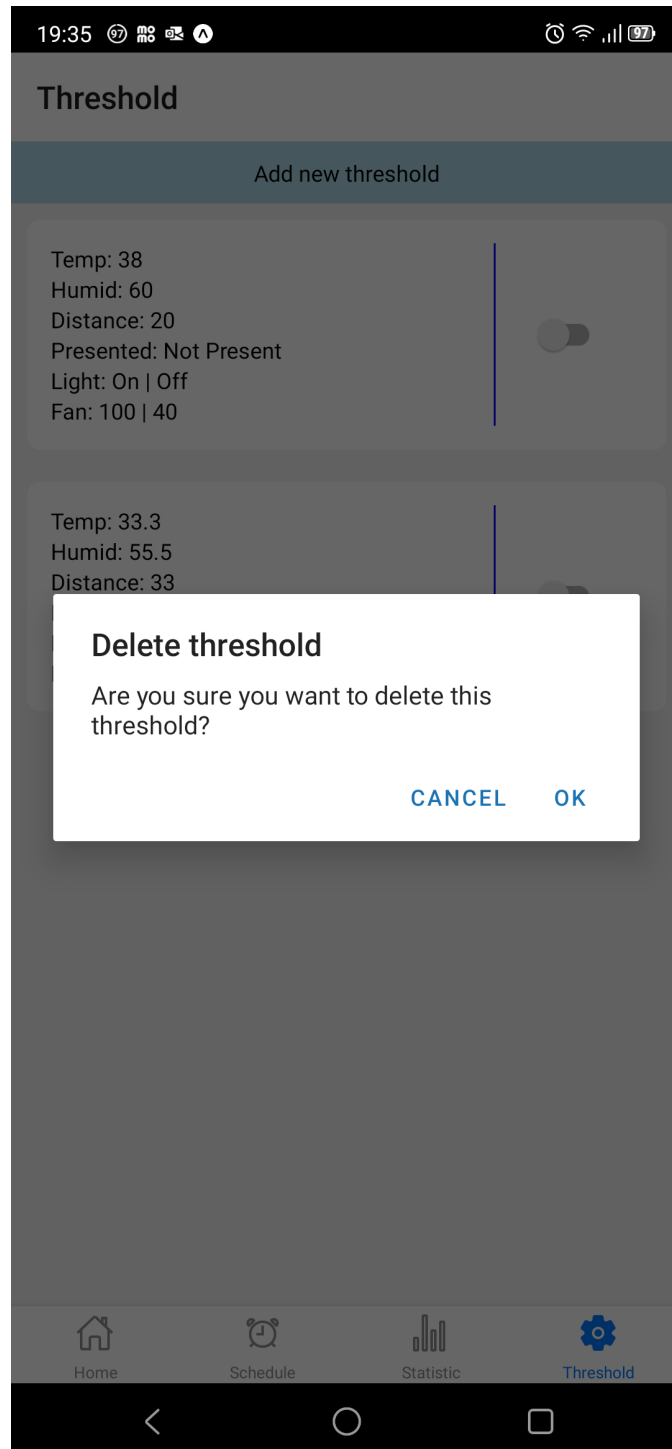


Hình 12: Màn hình hiển thị danh sách các ngưỡng





Có 3 hành động mà ta có thể thực hiện ở màn hình Threshold này:

- Thêm ngưỡng mới: Bấm vào nút trên cùng
- Sửa ngưỡng: Bấm vào ngưỡng muốn sửa

- Xoá ngưỡng: Bấm và giữ vào ngưỡng muốn xoá, sẽ hiện một dialog xác nhận xoá ngưỡng. Bấm "Confirm" và ngưỡng được xoá



Hình 13: Màn hình xác nhận xoá ngưỡng

14:26    

← Edit Threshold

Temp:

Humid:

Distance:

Presented: ☐





Light Status When Triggered: ☐

Light Status Original: ☐

Fan Status When Triggered:

Fan Status Original:

[Save](#)

 Home  Schedule  Statistic  Threshold

Hình 14: Màn hình chỉnh sửa/Tạo mới ngưỡng

XIII Phụ lục:

1 AdafruitIO Feed:

Group 9		
Feed Name	Key	Last value
<input type="checkbox"/> Distance	group-9.distance	20
<input type="checkbox"/> Fan Speed	group-9.fan-speed	60
<input type="checkbox"/> Humid	group-9.humid	60
<input type="checkbox"/> Light Switch	group-9.light-switch	0
<input type="checkbox"/> PIR	group-9.pir	CO NGUOI
<input type="checkbox"/> Temp	group-9.temp	38

Hình 15: Các Feed trên AdafruitIO

2 Thiết kế OhStem:



Hình 16: Thiết kế mạch trên OhStem

Link thiết kế OhStem: <https://app.ohstem.vn/#!/share/yolobit/2ZW4D1aiWenERtkHAIIR5JVOPik>

3 Github Repo:

Link Github: <https://github.com/Leino2604/smart-house>