

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



ĐỒ ÁN ĐA NGÀNH - CÔNG NGHỆ PHẦN MỀM

Dự án Smart Workplace

GVHD: Nguyễn Thiên Ân
Nhóm: 3
Sinh viên: Hồ Nguyễn Phi Hùng - 2211327.
Hồ Tuấn Linh - 2211848.
Hoàng Hữu Hà - 2113271.
Khương Gia Túc - 2014993.

TP.Hồ Chí Minh, Tháng 10-2024



Mục lục

1	Danh sách thành viên	3
2	Phiên bản	3
3	Giới thiệu về chủ đề	4
4	Yêu cầu	4
4.1	Funcional	4
4.1.1	Khía cạnh Internet vạn vật	4
4.1.2	Khía cạnh ứng dụng	5
4.2	Non-functional	5
4.2.1	Khía cạnh Internet vạn vật	5
4.2.2	Khía cạnh ứng dụng di động	5
5	Thiết bị	5
5.1	Thiết bị đầu vào	5
5.2	Thiết bị đầu ra	6
6	Use case chi tiết	6
6.1	Đăng nhập/ đăng ký	6
6.2	Thêm phòng làm việc	9
6.3	Hiển thị danh sách phòng đã thêm và trạng thái	11
6.4	Xem lịch sử sử dụng phòng	13
6.5	Thông báo định kỳ	16
6.6	Xem chi tiết các thông số trong phòng đang dùng	19
6.7	Điều khiển các thiết bị trong phòng đang dùng	21
7	Use case diagram	23
8	Mockup	23
8.1	Màn hình giới thiệu	24
8.2	Màn hình đăng nhập và đăng ký	25
8.3	Màn hình trang chủ hiển thị danh sách phòng	27
8.4	Màn hình trang xem thông số	28
8.5	Màn hình Trang điều khiển	29
9	Class diagram	30
10	Deployment View	31
11	Database Design	32
11.1	Entity-Relationship Diagram (ER diagram)	32
11.2	Relation Schema	33



12 Design Pattern	33
12.1 MVC Pattern (Model - View - Controller)	33
12.2 Observer Pattern	34
12.3 Singleton Pattern	35
12.4 Adapter Pattern	36
13 Thành phẩm	37



1 Danh sách thành viên

No.	Họ và tên	MSSV	Phần trăm đóng góp
1	Hồ Nguyễn Phi Hùng	2211327	100%
2	Hồ Tuấn Linh	2211848	100%
3	Hoàng Hữu Hà	2113271	100%
4	Khương Gia Túc	2014993	100%

2 Phiên bản

Phiên bản	Nội dung
v1	Thu thập yêu cầu (Giới thiệu đề tài và phạm vi dự án; Phân tích tổng quát, yêu cầu; Vẽ usecase diagram)
v2	Mô hình hệ thống (Vẽ các activity diagram, sequence diagram cho từng chức năng chính. Vẽ class diagram cho hệ thống) & Thiết kế kiến trúc (Mô tả database bằng ER diagram và lược đồ. Nêu ra các kiến trúc/công cụ/framework dùng cho backend và frontend)
v3	Sửa lại phần Thiết kế kiến trúc (thêm tổng quan kiến trúc)
v4	Thêm phần thành phẩm
v3	Làm phần các Design Pattern

3 Giới thiệu về chủ đề

Khu vực học tập và làm việc là một không gian quan trọng, nơi mà ý tưởng sáng tạo nảy nở và kiến thức được hình thành. Khu vực này là các phòng học, văn phòng, thư viện, phòng họp,... đối với học sinh và nhân viên văn phòng thì khu vực học tập và làm việc là nơi đòi hỏi môi trường thoải mái, điều kiện ánh sáng, nhiệt độ, độ ẩm lý tưởng để giúp con người thư giãn, thuận tiện và đầy cảm hứng giúp tăng cường khả năng tập trung, sự sáng tạo và năng suất. Đối với nhà trường và các doanh nghiệp thì khu vực học tập và làm việc cần được quản lý người dùng và các thiết bị để đảm bảo tính an toàn và bảo mật.

Với dự án khu vực học tập/làm việc thông minh được sử dụng các cảm biến để đo lường ánh sáng, nhiệt độ, độ ẩm,... và điều chỉnh đèn, điều hòa, phun sương... một cách tự động hoặc thủ công qua ứng dụng có thể quan sát và điều khiển các thiết bị để tạo một trường lý tưởng. Ngoài ra, mô hình AI nhận diện xem ai sử dụng khu vực này để cấp quyền truy cập thiết bị cho người dùng.



Hình 1: Smart Workplace

4 Yêu cầu

4.1 Funcional

4.1.1 Khía cạnh Internet vạn vật

- Tự động điều chỉnh ánh sáng, nhiệt độ, và độ ẩm dựa trên dữ liệu cảm biến và điều kiện môi trường hiện tại.
- Cho phép người dùng chọn chế độ điều chỉnh môi trường: tự động hoặc thủ công.



- Báo cáo các sự cố cần can thiệp thủ công khi không thể duy trì các điều kiện lý tưởng (ví dụ: nhiệt độ quá cao hoặc quá thấp).
- Cung cấp báo cáo trực tiếp về các thông số môi trường thông qua các chế độ hiển thị (biểu đồ, nhật ký...).

4.1.2 Khía cạnh ứng dụng

- Người dùng có thể tạo tài khoản và đăng nhập vào hệ thống để quản lý phòng học/làm việc.
- Cho phép người dùng thêm phòng làm việc, xóa phòng và quản lý các phòng làm việc.
- Cho phép người dùng điều khiển thủ công các thiết bị như đèn, điều hòa, và quạt.
- Cung cấp các thông số môi trường hiện tại và lịch sử.

4.2 Non-functional

4.2.1 Khía cạnh Internet vạn vật

- Hệ thống hoạt động liên tục 24/7 với thời gian ngừng hoạt động tối thiểu.
- Giao tiếp qua máy chủ MQTT để đảm bảo điều khiển thiết bị và thu thập dữ liệu.
- Thời gian trễ của các thiết bị không quá 5 giây.
- Ghi lại dữ liệu liên tục trong ít nhất 10 ngày để theo dõi và chẩn đoán.
- Cơ sở dữ liệu hỗ trợ lưu trữ ít nhất 500MB.
- Giao diện điều khiển trên thiết bị đơn giản và dễ sử dụng qua màn hình LCD hoặc nút bấm.

4.2.2 Khía cạnh ứng dụng di động

- Dễ sử dụng cho mọi đối tượng, đặc biệt là sinh viên và người đi làm.
- Giao diện dễ đọc trong mọi điều kiện ánh sáng.
- Hỗ trợ sử dụng trên nhiều loại thiết bị.
- Ứng dụng phải kết nối máy chủ dưới 2 giây và các thao tác phải phản hồi dưới 500ms.
- Hỗ trợ lưu trữ dữ liệu cục bộ lên đến 100MB và trên máy chủ.

5 Thiết bị

5.1 Thiết bị đầu vào

- Cảm biến DHT20 (Đo nhiệt độ, độ ẩm)

Mô tả: Đây là cảm biến đo nhiệt độ và độ ẩm với độ chính xác cao, tiêu thụ điện năng thấp và giao tiếp thông qua giao thức I²C.



Cách sử dụng: Kết nối cảm biến với vi điều khiển qua giao tiếp I²C. Bạn có thể đọc giá trị nhiệt độ và độ ẩm định kỳ để điều khiển các thiết bị khác như quạt hoặc máy phun sương dựa trên các giá trị này.

- Cảm biến ánh sáng

Mô tả: Đo cường độ ánh sáng môi trường xung quanh.

Cách sử dụng: Thường kết nối với vi điều khiển qua chân analog hoặc digital. Dữ liệu từ cảm biến có thể dùng để bật/tắt đèn LED tùy theo mức độ ánh sáng, hoặc điều chỉnh độ sáng đèn cho phù hợp với ánh sáng ban ngày.

- Camera + mic

Mô tả: Dùng để giám sát, thu nhận dữ liệu hình ảnh và âm thanh.

Cách sử dụng: Camera và mic có thể kết hợp với hệ thống gateway để giám sát từ xa, phát trực tiếp video hoặc ghi nhận âm thanh cho các mục đích xử lý khác.

5.2 Thiết bị đầu ra

- Màn hình LCD

Mô tả: Hiển thị các thông tin như nhiệt độ, độ ẩm, cường độ ánh sáng, hoặc trạng thái của hệ thống.

Cách sử dụng: Màn hình LCD thường sử dụng giao tiếp I²C hoặc SPI. Bạn có thể hiển thị dữ liệu từ các cảm biến trên màn hình để theo dõi.

- Quạt mini

Mô tả: Sử dụng để làm mát hoặc lưu thông không khí.

Cách sử dụng: Quạt được điều khiển bằng cách sử dụng transistor hoặc relay. Bạn có thể bật/tắt quạt dựa trên ngưỡng nhiệt độ từ cảm biến DHT20.

- Máy bơm (máy phun sương)

Mô tả: Dùng để phun sương, giúp tăng độ ẩm trong không khí.

Cách sử dụng: Tương tự như quạt, máy bơm có thể được điều khiển bằng relay hoặc transistor. Bạn có thể kích hoạt nó dựa trên dữ liệu độ ẩm từ DHT20, nếu độ ẩm thấp thì bật máy phun sương.

- Đèn Led

Mô tả: Đèn LED có thể dùng để chiếu sáng hoặc cung cấp tín hiệu thị giác.

Cách sử dụng: Bạn có thể điều khiển đèn LED bằng cách sử dụng chân GPIO để bật/tắt hoặc điều chỉnh độ sáng thông qua PWM (Điều chế độ rộng xung). Đèn có thể tự động bật khi trời tối hoặc theo điều kiện của cảm biến ánh sáng.

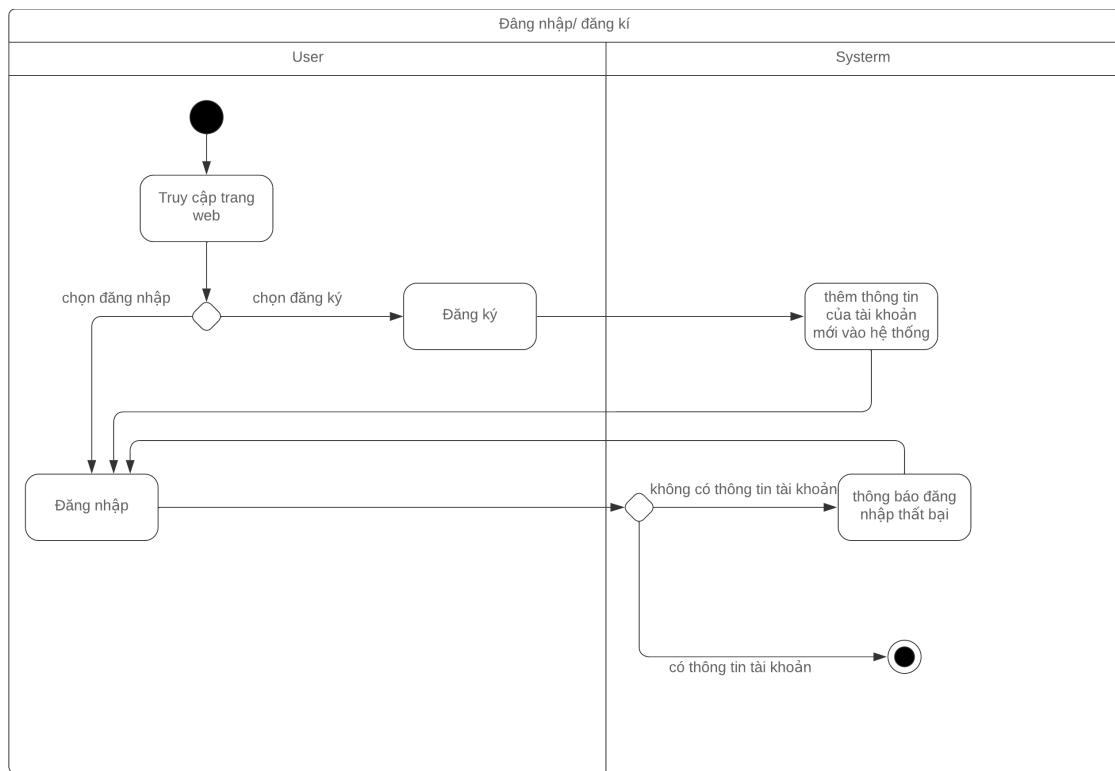
6 Use case chi tiết

6.1 Đăng nhập/ đăng kí

Use case name	Đăng nhập/ đăng kí
---------------	--------------------



Actor	User
Description	Người dùng đăng nhập hoặc đăng ký để sử dụng hệ thống
Precondition	Thiết bị người dùng kết nối internet khi thực hiện đăng nhập
Post condition	Người dùng đăng nhập thành công
Normal flow	<ol style="list-style-type: none">1. Người dùng truy cập trang web hệ thống2. Người dùng chọn đăng nhập3. Người dùng nhập thông tin tài khoản4. Hệ thống xác nhận thông tin thành công và di đến trang chủ
Alternative flow	Ở bước 2, nếu người dùng chọn đăng ký từ: <ul style="list-style-type: none">• (2.1) người dùng nhập thông tin cần thiết.• (2.2) hệ thống ghi nhận thông tin người dùng và quay lại luồng chính ở bước 3.
Exception flow	Ở bước 4, hệ thống xác nhận thông tin thất bại. <ul style="list-style-type: none">• (4.1) Hệ thống quay lại bước 2, cho phép người dùng nhập lại thông tin.

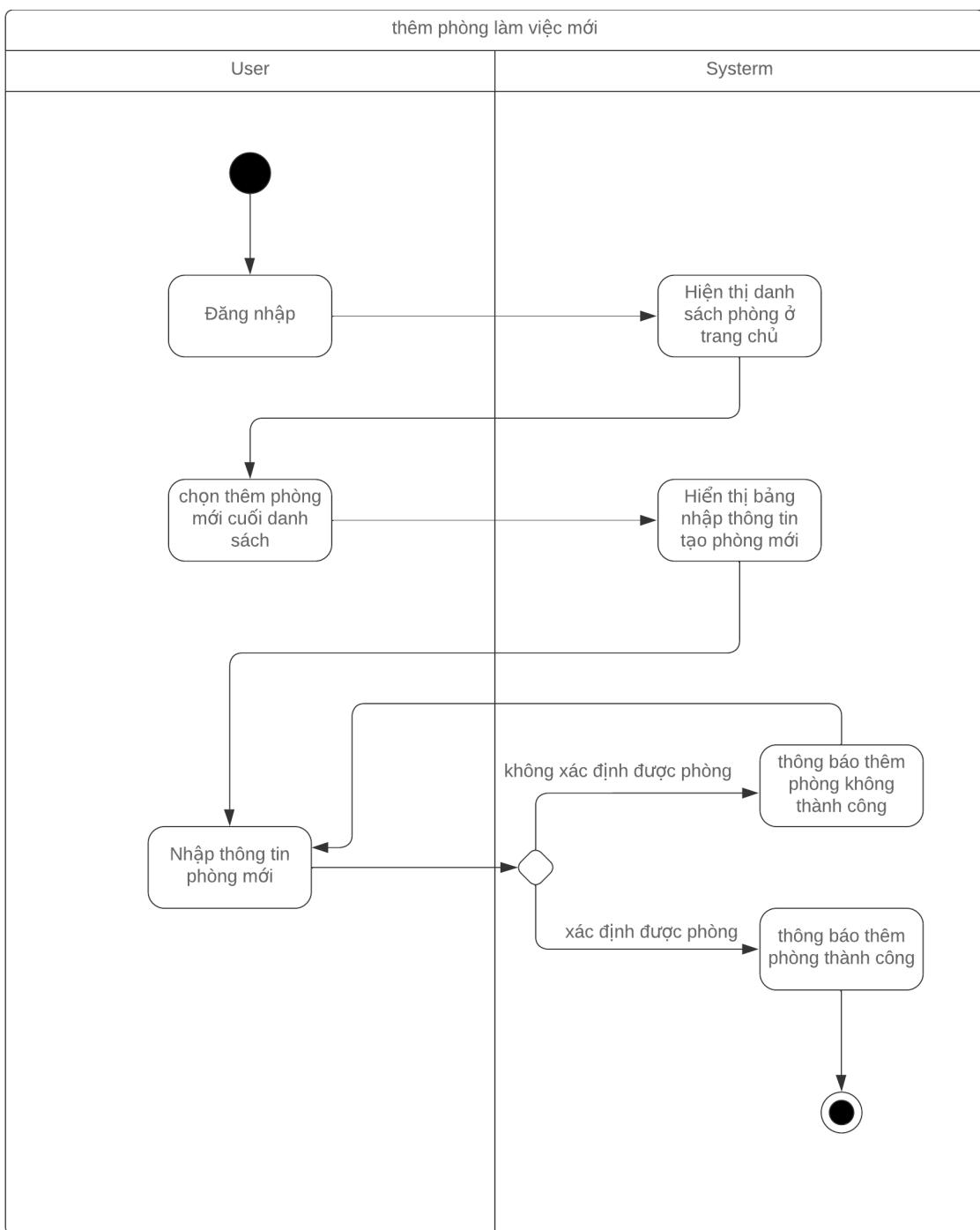


Hình 2: Activity Diagram Đăng nhập/Dáng kí



6.2 Thêm phòng làm việc

Use case name	Thêm phòng làm việc
Actor	User
Description	Hệ thống cho phép người dùng thêm các phòng làm việc/học tập mới của bản thân
Precondition	Người dùng đã đăng nhập vào hệ thống.
Post condition	Phòng mới được thêm hiển thị ở trang chủ.
Normal flow	<ol style="list-style-type: none">1. Hệ thống hiển thị danh sách các phòng mà người dùng đã từng sử dụng.2. Người dùng chọn thêm một phòng mới từ cuối danh sách.3. Hệ thống hiển thị thông tin cần thiết để thêm 1 phòng mới.4. Người dùng nhập thông tin vào hệ thống và nhấn “thêm”.5. Hệ thống xác nhận thông tin thành công và quay lại màn hình chính
Alternative flow	
Exception flow	Ở bước 5, hệ thống các nhận thông tin không thành công. <ul style="list-style-type: none">• (5.1) Hệ thống hiển thị thông báo "Không thêm thành công cho phòng này".• (5.2) Hệ thống quay lại bước 4, cho phép người dùng nhập lại thông tin.

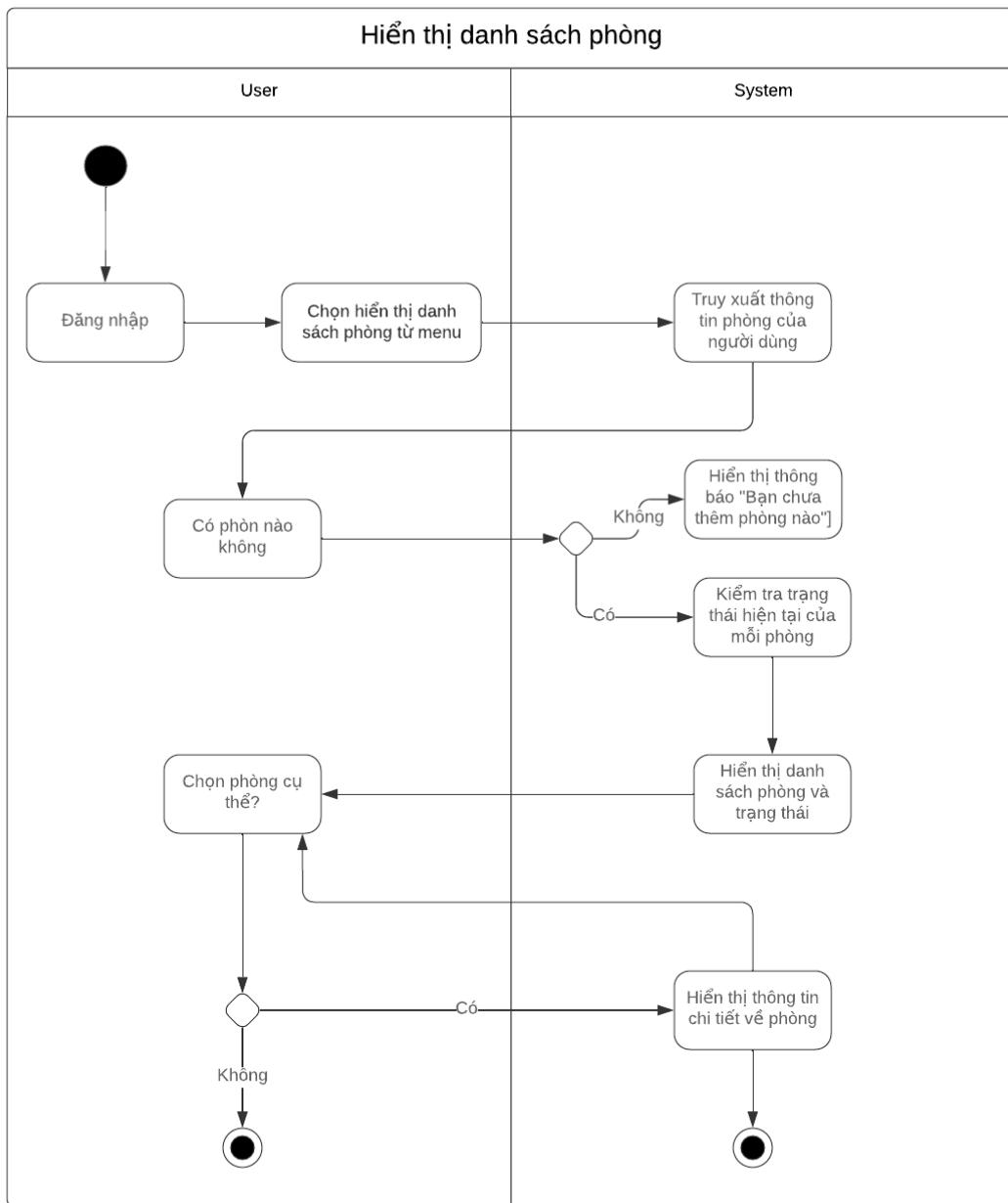


Hình 3: Activity Diagram Thêm phòng làm việc



6.3 Hiển thị danh sách phòng đã thêm và trạng thái

Use case name	Hiển thị danh sách phòng đã thêm và trạng thái
Actor	User
Description	Hệ thống hiển thị danh sách các phòng mà người dùng đã thêm cùng với trạng thái hiện tại của chúng
Precondition	Người dùng đã đăng nhập vào hệ thống.
Post condition	Danh sách phòng và trạng thái được hiển thị cho người dùng
Normal flow	<ol style="list-style-type: none">1. Người dùng chọn option "Xem danh sách phòng" từ menu chính2. Hệ thống truy xuất thông tin về các phòng của người dùng3. Hệ thống kiểm tra trạng thái hiện tại của mỗi phòng4. Hệ thống hiển thị danh sách phòng kèm theo thông tin:<ul style="list-style-type: none">• Tên phòng• Trạng thái (có người/không có người)• Các thông số môi trường hiện tại (nhiệt độ, độ ẩm, ánh sáng)5. Người dùng xem danh sách
Alternative flow	<p>5a Người dùng chọn một phòng cụ thể:</p> <ol style="list-style-type: none">1. Hệ thống hiển thị thông tin chi tiết về phòng đó2. Hiển thị chi tiết phòng đó
Exception flow	<p>2a Không có phòng nào trong danh sách:</p> <ol style="list-style-type: none">1. Hệ thống hiển thị thông báo "Bạn chưa thêm phòng nào"2. Kết thúc <p>3a Lỗi khi kiểm tra trạng thái phòng</p> <ol style="list-style-type: none">1. Hệ thống hiển thị trạng thái là "Không xác định" cho phòng đó



Hình 4: Activity Hiển thị danh sách phòng

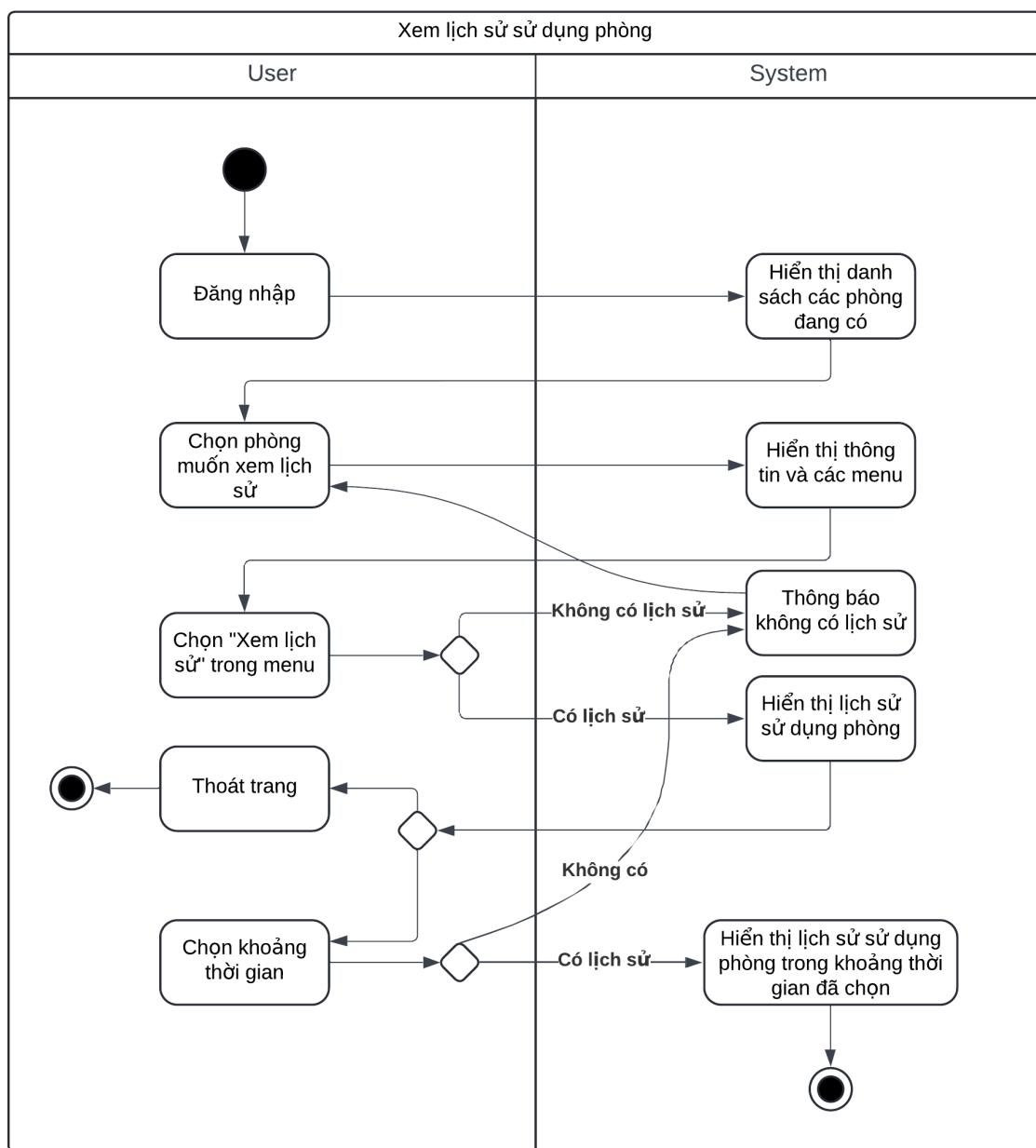


6.4 Xem lịch sử sử dụng phòng

Use case name	Xem lịch sử sử dụng phòng
Actor	User
Description	Hệ thống cho phép người dùng xem lại lịch sử sử dụng các phòng làm việc/học tập của bản thân, bao gồm thời gian sử dụng, các thông số môi trường, và các thao tác đã thực hiện.
Precondition	<ul style="list-style-type: none">Người dùng đã đăng nhập vào hệ thống.Người dùng đã từng sử dụng ít nhất một phòng.
Post condition	Lịch sử sử dụng phòng được hiển thị cho người dùng
Normal flow	<ol style="list-style-type: none">Hệ thống hiển thị danh sách các phòng mà người dùng đã từng sử dụng.Người dùng chọn một phòng cụ thể từ danh sách.Người dùng chọn chức năng "Xem lịch sử" từ menu.Hệ thống truy xuất và hiển thị lịch sử sử dụng của phòng đã chọn, bao gồm:<ul style="list-style-type: none">Ngày và thời gian bắt đầu sử dụngThời gian kết thúc sử dụngTổng thời gian sử dụngNgười dùng nhấn "Hoàn tất" để quay lại màn hình chính.
Alternative flow	Ở bước 4, nếu người dùng chọn "Chọn khoảng thời gian" từ thanh tùy chọn: <ul style="list-style-type: none">(4.1) Hệ thống hiển thị giao diện chọn ngày bắt đầu và kết thúc.(4.2) Người dùng chọn khoảng thời gian mong muốn và nhấn "Xác nhận".(4.3) Hệ thống cập nhật hiển thị lịch sử theo bộ lọc đã chọn.



Exception flow	<ul style="list-style-type: none">● Exception 1: Ở bước 4, không có dữ liệu lịch sử cho phòng đã chọn. (4.1) Hệ thống hiển thị thông báo "Không có dữ liệu lịch sử cho phòng này". (4.2) Hệ thống quay lại bước 2, cho phép người dùng chọn phòng khác.● Exception 2: Ở bước 7, khoảng thời gian người dùng chọn không có trong cơ sở dữ liệu. (7.1) Hệ thống hiển thị cửa sổ lỗi và đề xuất người dùng chọn khoảng thời gian phù hợp. (7.2) Quay lại bước 6 để người dùng chọn lại.
-----------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



Hình 5: Activity Diagram Xem lịch sử sử dụng phòng

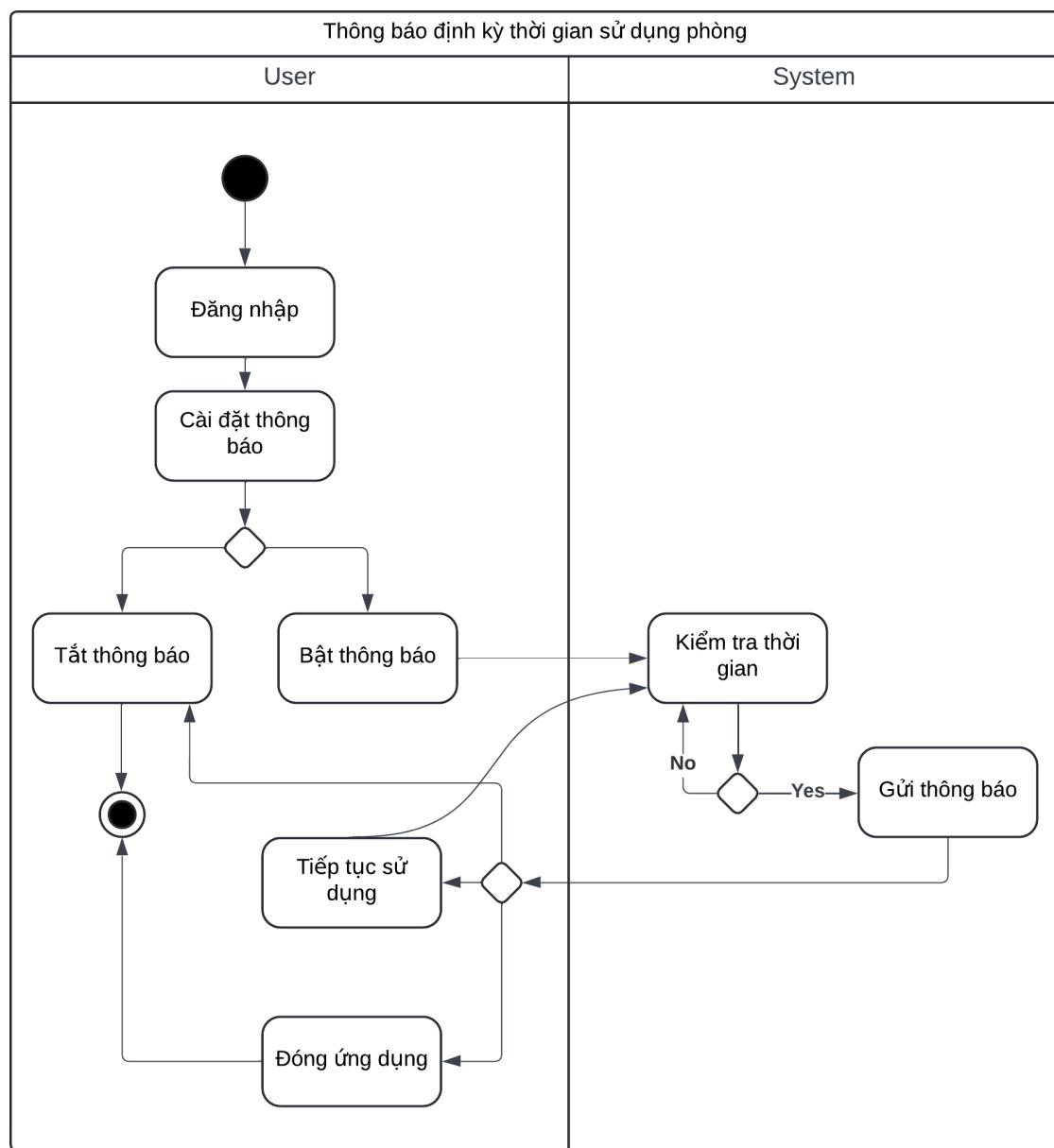


6.5 Thông báo định kỳ

Use case name	Thông báo định kỳ thời gian sử dụng phòng
Actor	User
Description	Hệ thống gửi thông báo định kỳ cho người dùng về thời gian họ đã sử dụng phòng, giúp họ theo dõi và quản lý thời gian làm việc/học tập hiệu quả.
Precondition	<ul style="list-style-type: none">Người dùng đã đăng nhập vào hệ thống.Người dùng đang sử dụng một phòng trong hệ thống.Tính năng thông báo đã được kích hoạt trong cài đặt người dùng.
Post condition	<ul style="list-style-type: none">Thông báo được ghi lại trong lịch sử thông báo của người dùng.Thời gian sử dụng phòng được cập nhật trong hệ thống.
Normal flow	<ol style="list-style-type: none">Hệ thống bắt đầu đếm thời gian khi người dùng bắt đầu sử dụng phòng.Khi đạt đến mốc thời gian định kỳ (mặc định là 30 phút), hệ thống tạo một thông báo.Hệ thống gửi thông báo đến thiết bị của người dùng (ví dụ: điện thoại, máy tính) với nội dung:<ul style="list-style-type: none">Tên phòng đang sử dụngThời gian đã sử dụngThông số môi trường hiện tại (nhiệt độ, độ ẩm, ánh sáng)Người dùng nhận và xem thông báo.Hệ thống tiếp tục đếm thời gian và lặp lại từ bước 2 cho đến khi người dùng kết thúc phiên sử dụng phòng.



Alternative flow	<p>Ở bước 4, nếu người dùng chọn "Tùy chỉnh thông báo" từ thông báo:</p> <ul style="list-style-type: none">• (4.1) Hệ thống hiển thị giao diện cài đặt thông báo.• (4.2) Người dùng có thể điều chỉnh:<ul style="list-style-type: none">– Tần suất thông báo (ví dụ: 15 phút, 30 phút, 1 giờ)– Loại thông tin hiển thị trong thông báo• (4.3) Người dùng xác nhận thay đổi.• (4.4) Hệ thống lưu cài đặt mới và áp dụng cho các thông báo tiếp theo.• (4.5) Quay lại luồng chính ở bước 5.
Exception flow	<ul style="list-style-type: none">• Exception 1: Người dùng tắt tính năng thông báo trong quá trình sử dụng phòng.<ul style="list-style-type: none">– (1.1) Hệ thống ngừng gửi thông báo cho phiên sử dụng hiện tại.– (1.2) Hệ thống hiển thị xác nhận rằng tính năng thông báo đã bị tắt.• Exception 2: Có lỗi kết nối khi gửi thông báo.<ul style="list-style-type: none">– (2.1) Hệ thống ghi lại thông báo vào hàng đợi.– (2.2) Hệ thống thử gửi lại thông báo khi kết nối được khôi phục.

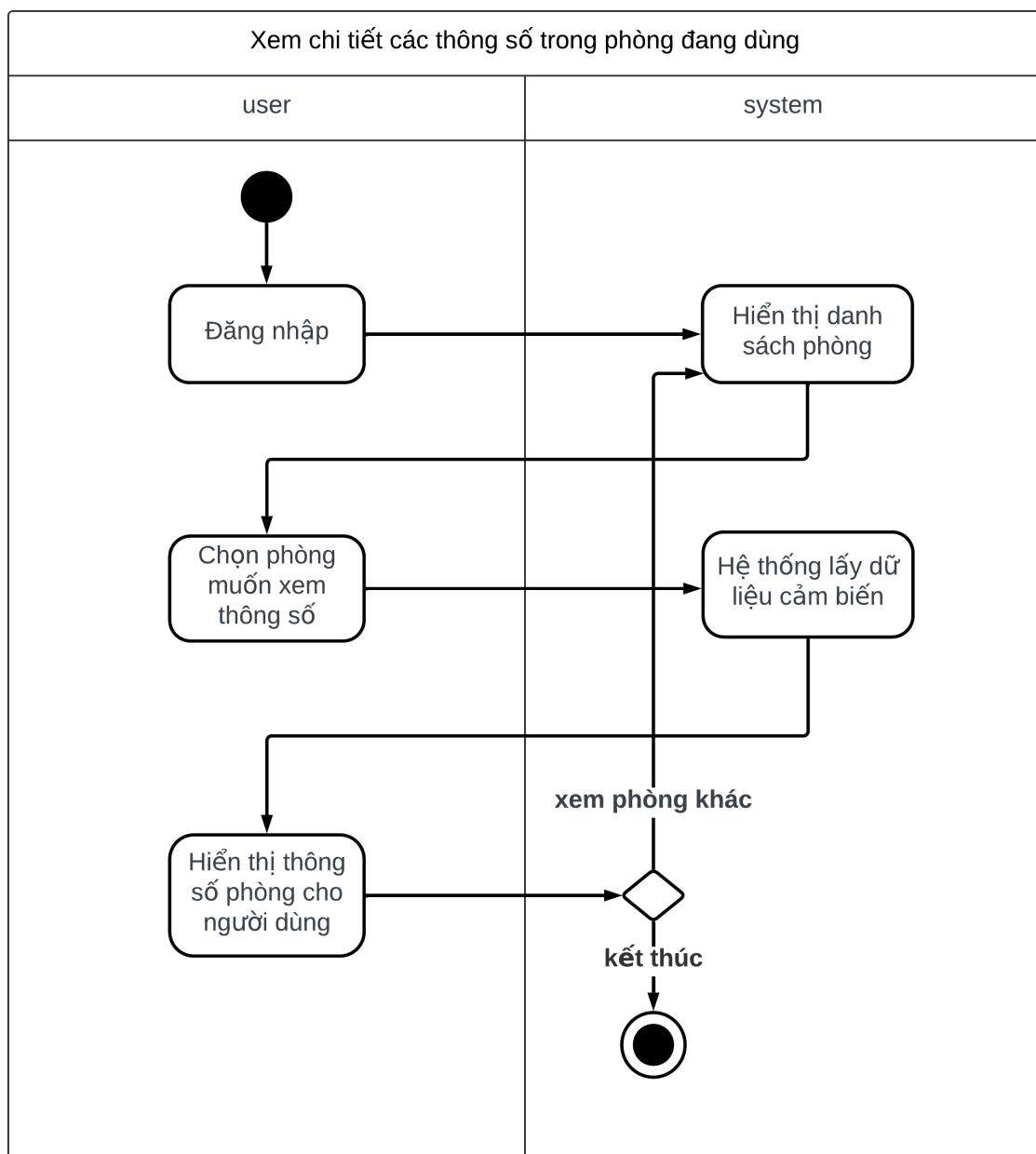


Hình 6: Activity Diagram Thông báo định kì



6.6 Xem chi tiết các thông số trong phòng đang dùng

Use case name	Xem chi tiết các thông số trong phòng đang dùng
Actor	User
Description	Người dùng yêu cầu xem dữ liệu nhiệt độ, độ ẩm và ánh sáng của căn phòng hiện tại. Hệ thống lấy dữ liệu từ các cảm biến và hiển thị nó.
Precondition	Người dùng đã đăng nhập và có quyền truy cập vào một phòng đã đăng ký.
Post condition	Dữ liệu môi trường được hiển thị cho người dùng.
Normal flow	<ol style="list-style-type: none">Người dùng chọn phòng từ danh sách phòng đang hoạt động.Hệ thống lấy dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng).Dữ liệu được hiển thị cho người dùng.
Alternative flow	
Exception flow	

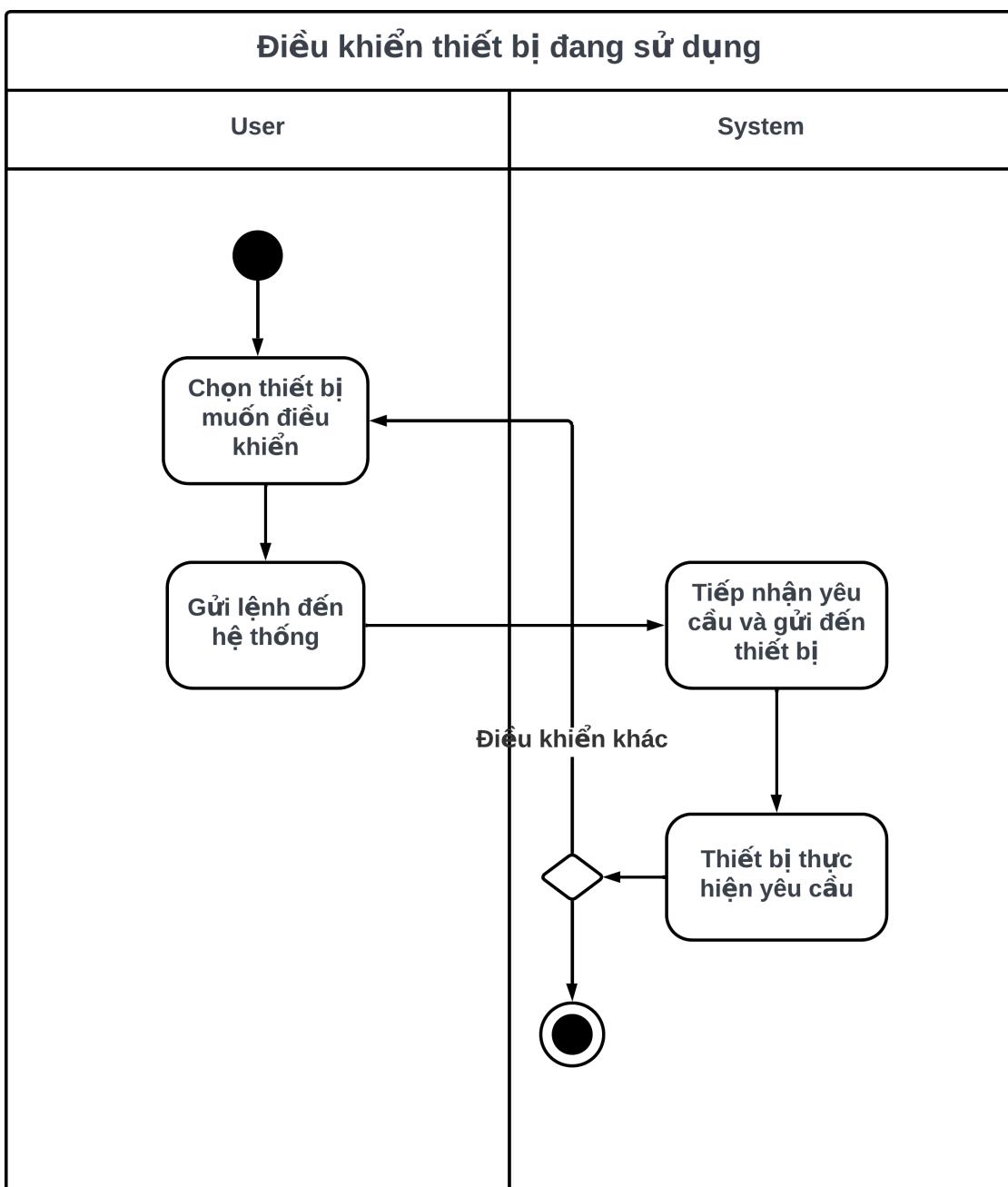


Hình 7: Activity Diagram Xem chi tiết phòng



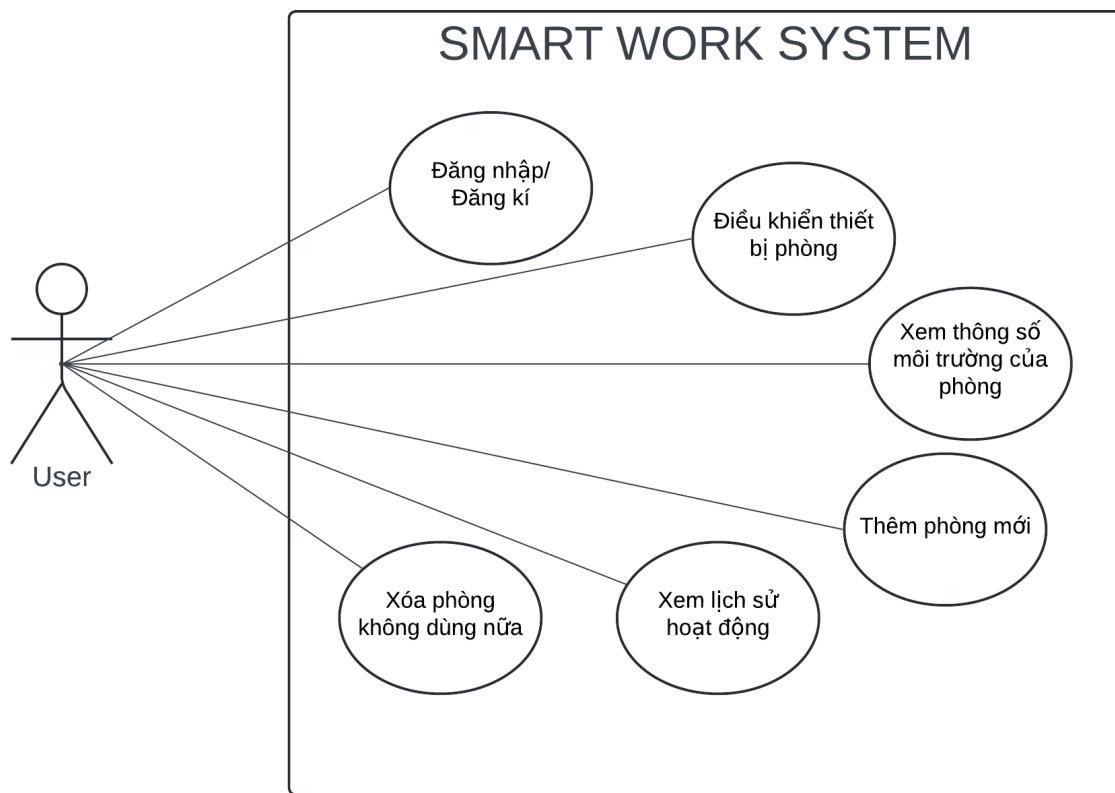
6.7 Điều khiển các thiết bị trong phòng đang dùng

Use case name	Điều khiển các thiết bị trong phòng đang dùng
Actor	User
Description	Người dùng có thể điều khiển các thiết bị như đèn, quạt hoặc máy phun sương trong phòng hiện tại bằng cách gửi lệnh qua hệ thống.
Precondition	Người dùng đã đăng nhập và có quyền truy cập vào một phòng đã đăng ký. Các thiết bị phải được kết nối và hoạt động.
Post condition	Các thiết bị trong phòng thực hiện được các hành động mà người dùng yêu cầu
Normal flow	<ol style="list-style-type: none">Người dùng chọn thiết bị (đèn, quạt, v.v.).Người dùng gửi lệnh (bật/tắt, điều chỉnh cài đặt).Hệ thống chuyển tiếp lệnh đến thiết bị.Thiết bị thực thi hành động theo yêu cầu.
Alternative flow	
Exception flow	Ở bước 3, hệ thống gửi yêu cầu thất bại. <ul style="list-style-type: none">(3.1) Hiển thị thông báo lỗi.(3.2) Quay lại bước 1.



Hình 8: Activity Diagram Điều khiển thiết bị

7 Use case diagram



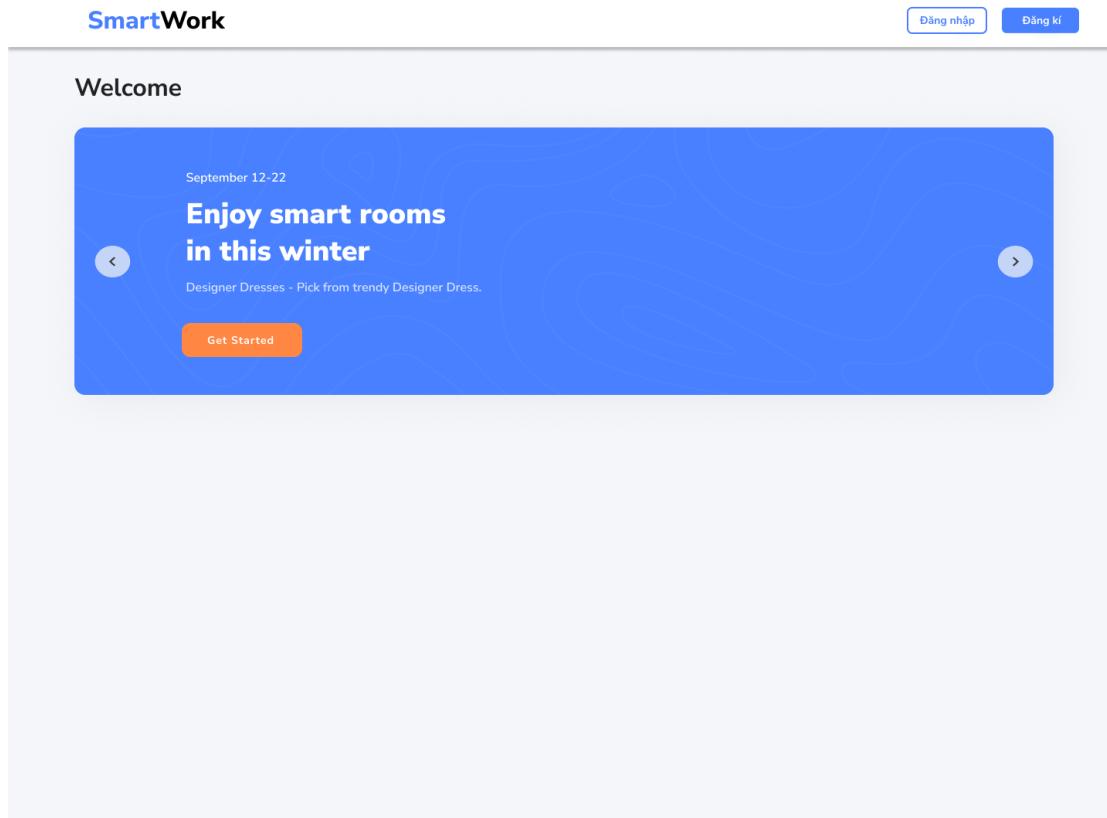
Hình 9: Usecase Diagram System

8 Mockup

Lưu ý: Một số mockups không còn phản ánh thiết kế hiện tại của nhóm và chỉ lưu trữ cho mục đích lưu trữ thiết kế và so sánh.



8.1 Màn hình giới thiệu

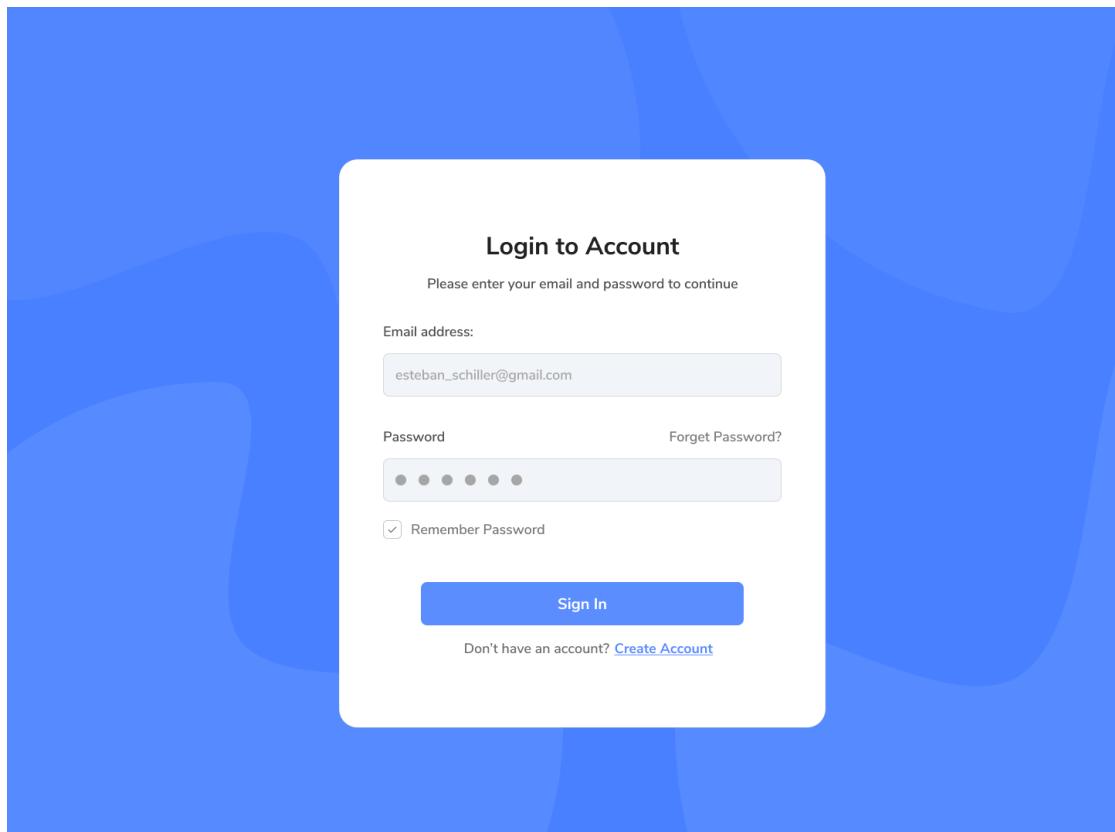


Hình 10: Màn hình giới thiệu

- Đây là trang đầu tiên cho web khi mới bắt đầu truy cập vào. Nó chứa các nút để thực hiện đăng nhập, đăng ký, và màn hình trang chủ này cũng sẽ cung cấp giới thiệu cơ bản, tin tức liên quan đến hệ thống,...

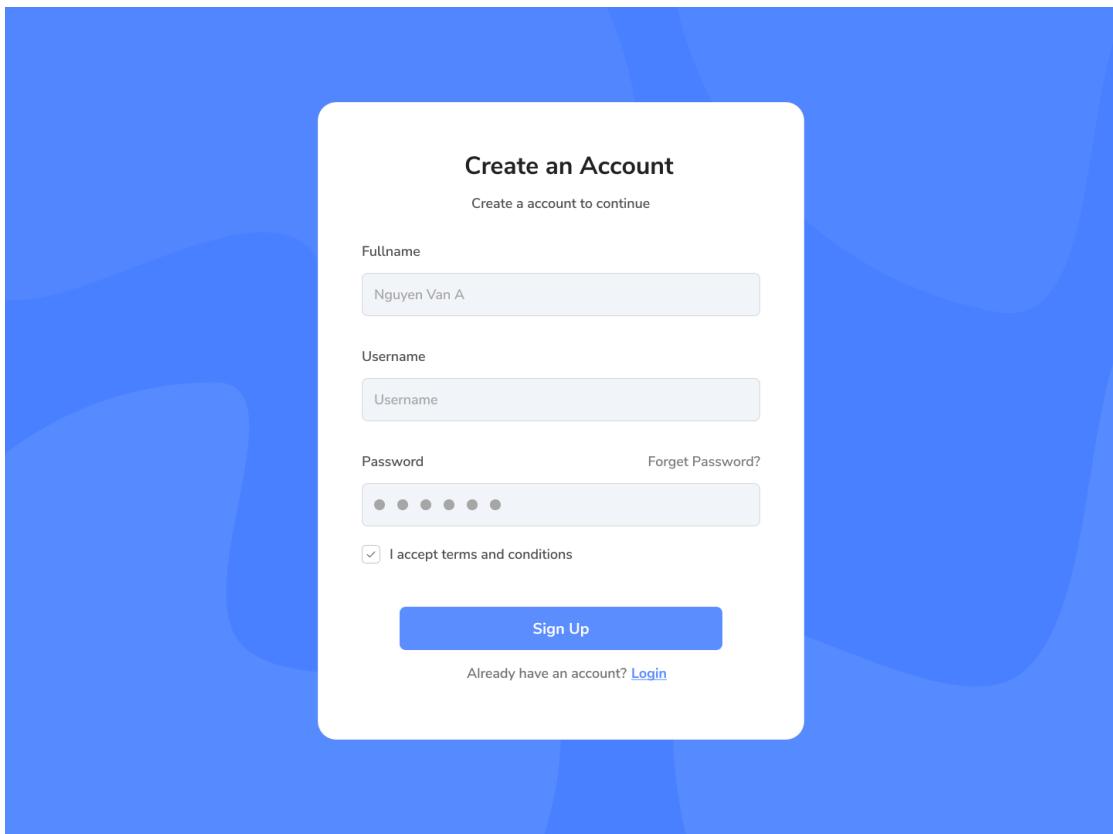


8.2 Màn hình đăng nhập và đăng ký



Hình 11: Màn hình đăng nhập.

- Yêu cầu người dùng nhập tên đăng nhập hoặc email và mật khẩu đã đăng ký trước đó.
- Màn hình đăng nhập xác thực thông tin của người dùng đã có tài khoản, sau đó chuyển họ đến trang chủ hiển thị danh sách phòng.
- Nếu thông tin đăng nhập không chính xác hoặc tài khoản không tồn tại, màn hình sẽ hiện thông báo lỗi với nội dung rõ ràng để người dùng hiểu và sửa lại.
- Sau khi đăng nhập thành công, hệ thống tạo và gửi một JWT cho người dùng để xác thực trong các phiên tiếp theo. JWT giúp bảo mật truy cập và giảm thiểu việc lưu trữ trạng thái trên server.



Hình 12: Màn hình đăng ký

- Yêu cầu người dùng nhập tên các thông tin sơ bộ cần bản để tạo tài khoản như tên người dùng, tên tài khoản hoặc địa chỉ email để đăng ký và mật khẩu xác nhận.
- Cho phép người dùng mới tạo tài khoản, sau đó họ sẽ được đưa vào trang đăng nhập để thực hiện đăng nhập.



8.3 Màn hình trang chủ hiển thị danh sách phòng

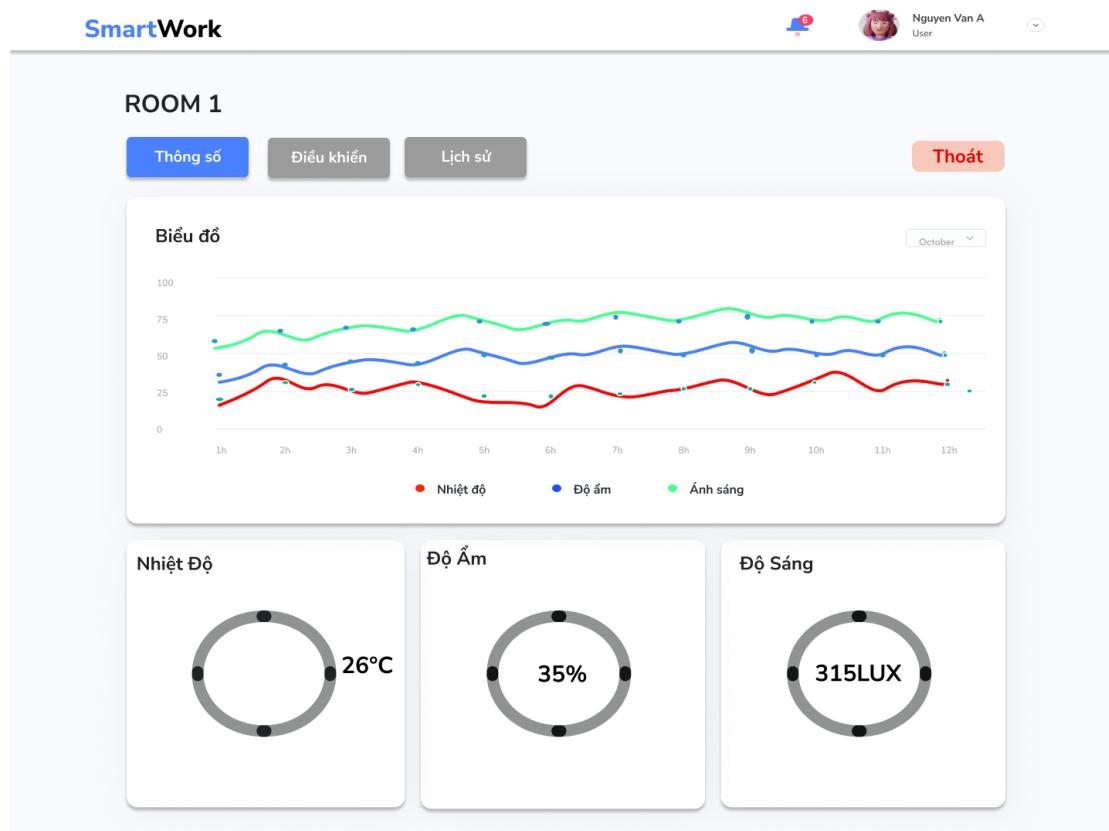
ID	Tên phòng	Adafruit Name	Adafruit Key	Trạng thái	Xem/Điều khiển
00001	Phòng đọc sách	us*****e1	*****_...*****	Trống	Xem ⏺
00002	Văn phòng 1	us*****e2	*****_...*****	Đang sử dụng	Xem ⏺
00003	Văn phòng 2	us*****e1	*****_...*****	Trống	Xem ⏺
00004	Phòng nghỉ 1	me*****e1	*****_...*****	Trống	Xem ⏺
00005	Phòng học	us*****e1	*****_...*****	Trống	Xem ⏺
00006	Phòng khách	cus*****c1	*****_...*****	Có người	Xem ⏺

Hình 13: Màn hình trang chủ hiển thị danh sách phòng

- Đây là màn hình trang chủ sau khi đăng nhập thành công nó hiển thị danh sách phòng đang có để người dùng dễ kiểm soát hơn.
- Danh sách phòng cung cấp thông tin về phòng như ID mã được cấp định dạng riêng cho từng phòng, tên phòng, tên tài khoản và mật khẩu kết nối với adafruit lưu trữ và quản lý dữ liệu từ các thiết bị IoT trong căn phòng đó. Danh sách phòng còn thể hiện trạng thái của phòng và cs nút điều hướng đến trang chi tiết



8.4 Màn hình trang xem thông số

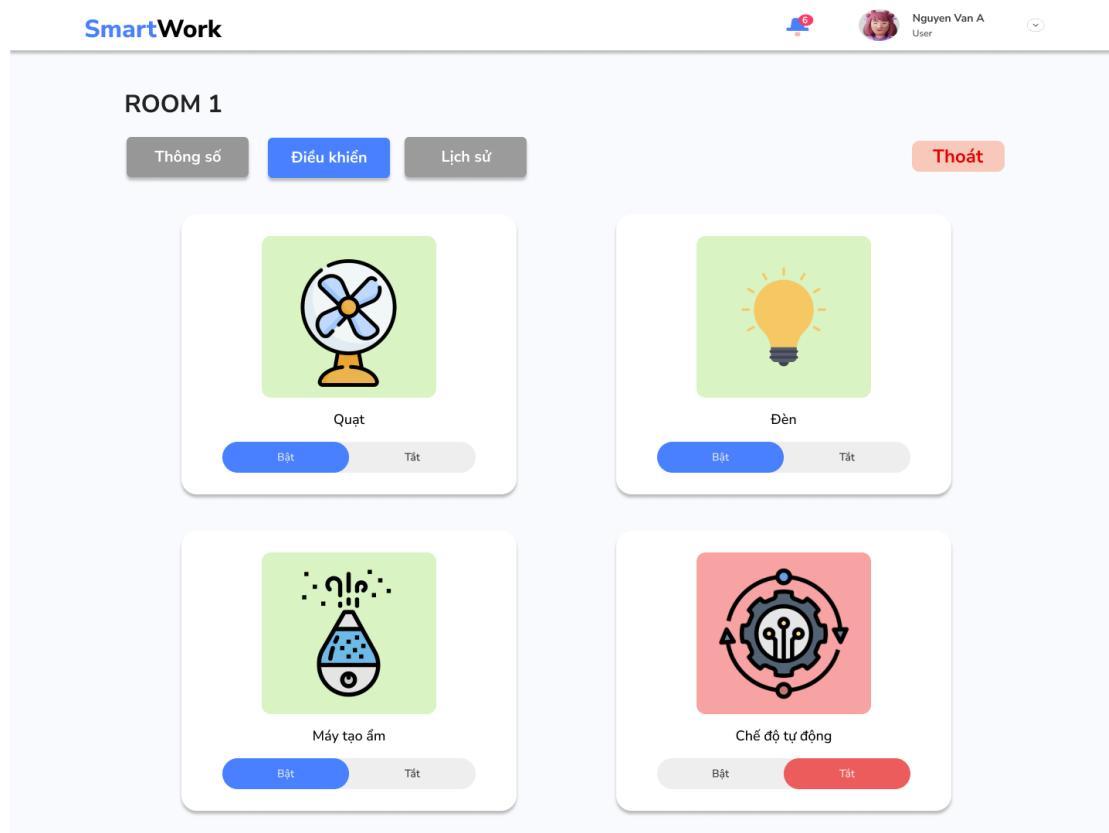


Hình 14: Màn hình trang xem thông số

- Trang có thể được truy cập khi ấn xem ở trang chủ hiển thị danh sách phòng, web sẽ được điều hướng qua trang xem thông số của phòng cụ thể, giúp theo dõi các thông số của phòng một cách tiện lợi.
- Trang hiển thị hai phần chính là biểu đồ các thông số nhiệt độ, độ ẩm, độ sáng của phòng theo thời gian, và các thông số của nó tại thời điểm hiện tại.
- Ngoài ra còn có các nút điều hướng đến các chức năng khác như trang điều khiển các thiết bị IoTs, trang lịch sử và nút thoát để quay lại trang danh sách



8.5 Màn hình Trang điều khiển

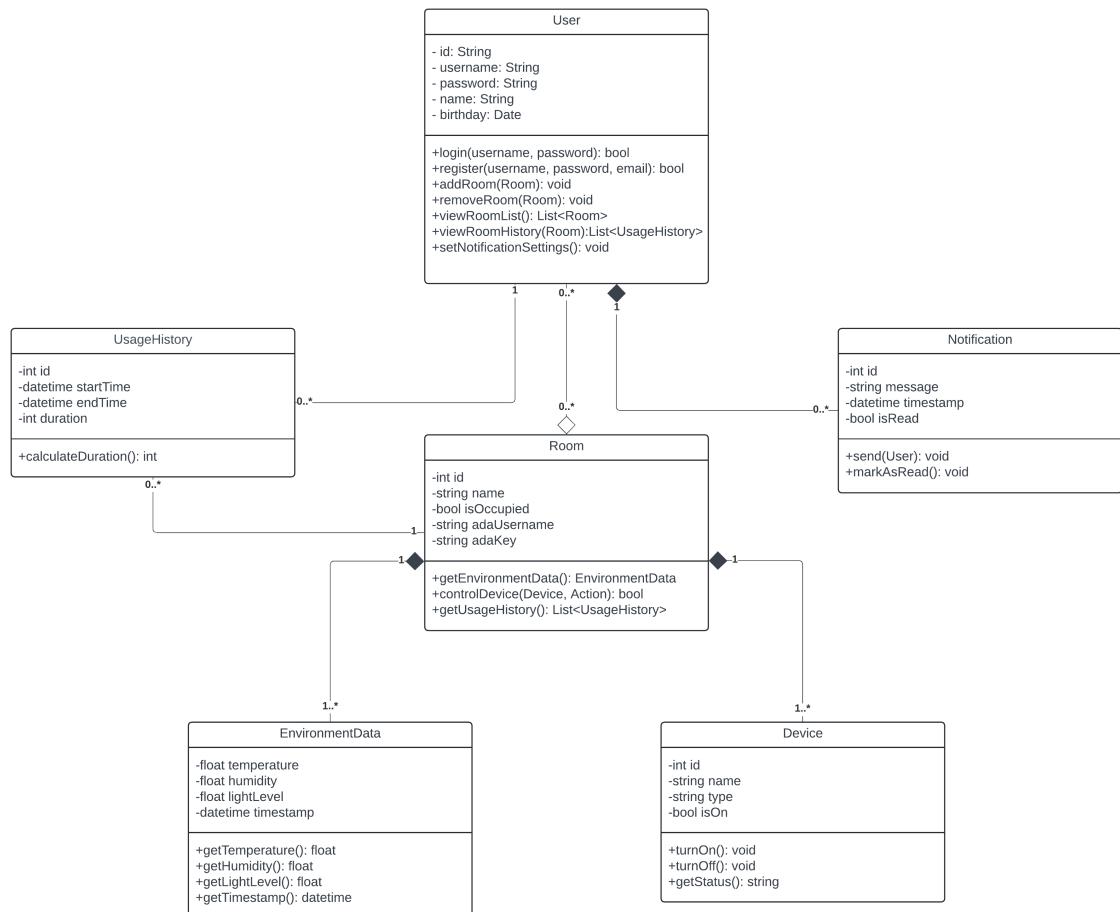


Hình 15: Trang điều khiển

- Trang điều khiển cho phép điều khiển các thiết bị IoT từ xa với các hình ảnh thể hiện trực quan đại diện cho các thiết bị để người dùng dễ dàng thao tác
- Ngoài ra còn có nút để bật chế độ tự động điều khiển các thiết bị để tạo môi trường thích hợp nhất với phòng nhất có thể để, Và nếu khi người dùng muốn tự kiểm soát có thể tự tắt chế độ này



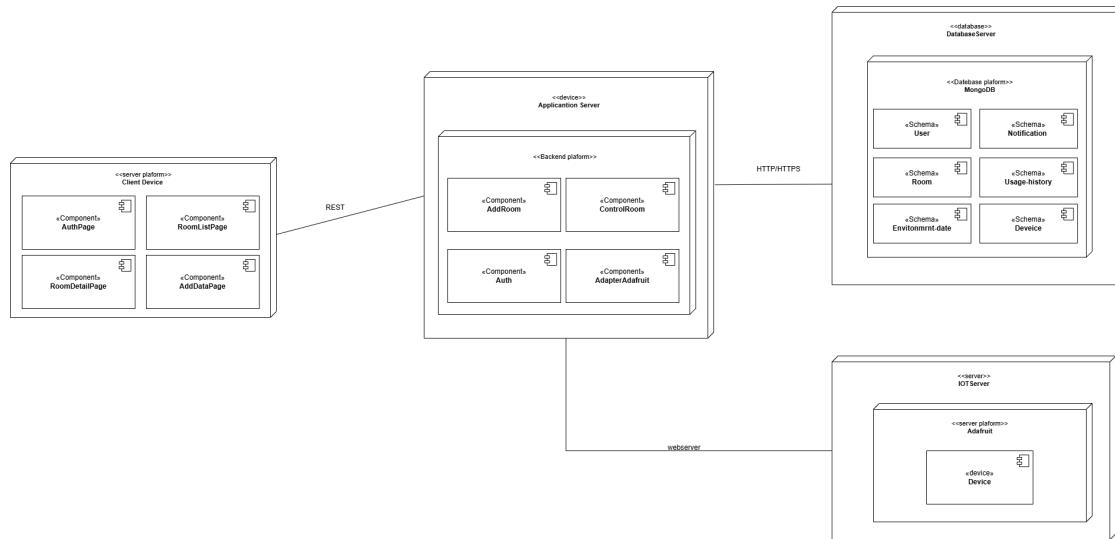
9 Class diagram



Hình 16: Class Diagram



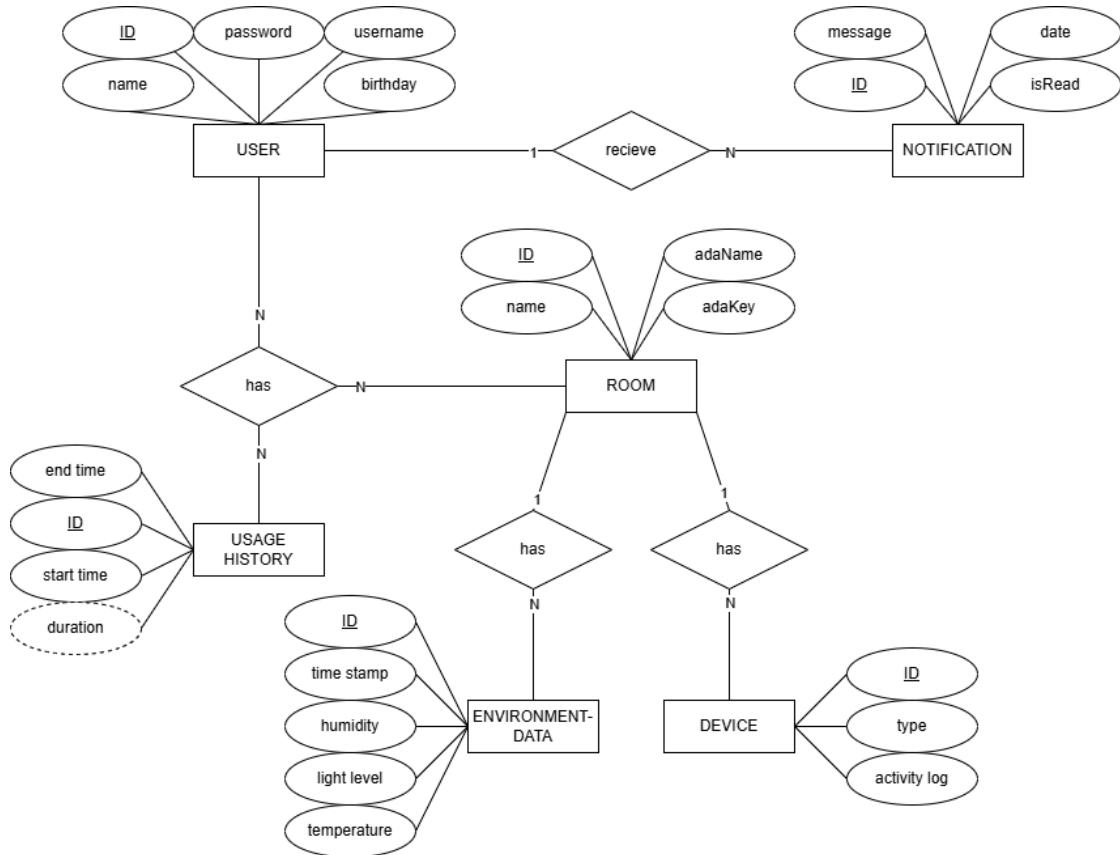
10 Deployment View



Hình 17: Deployment diagram

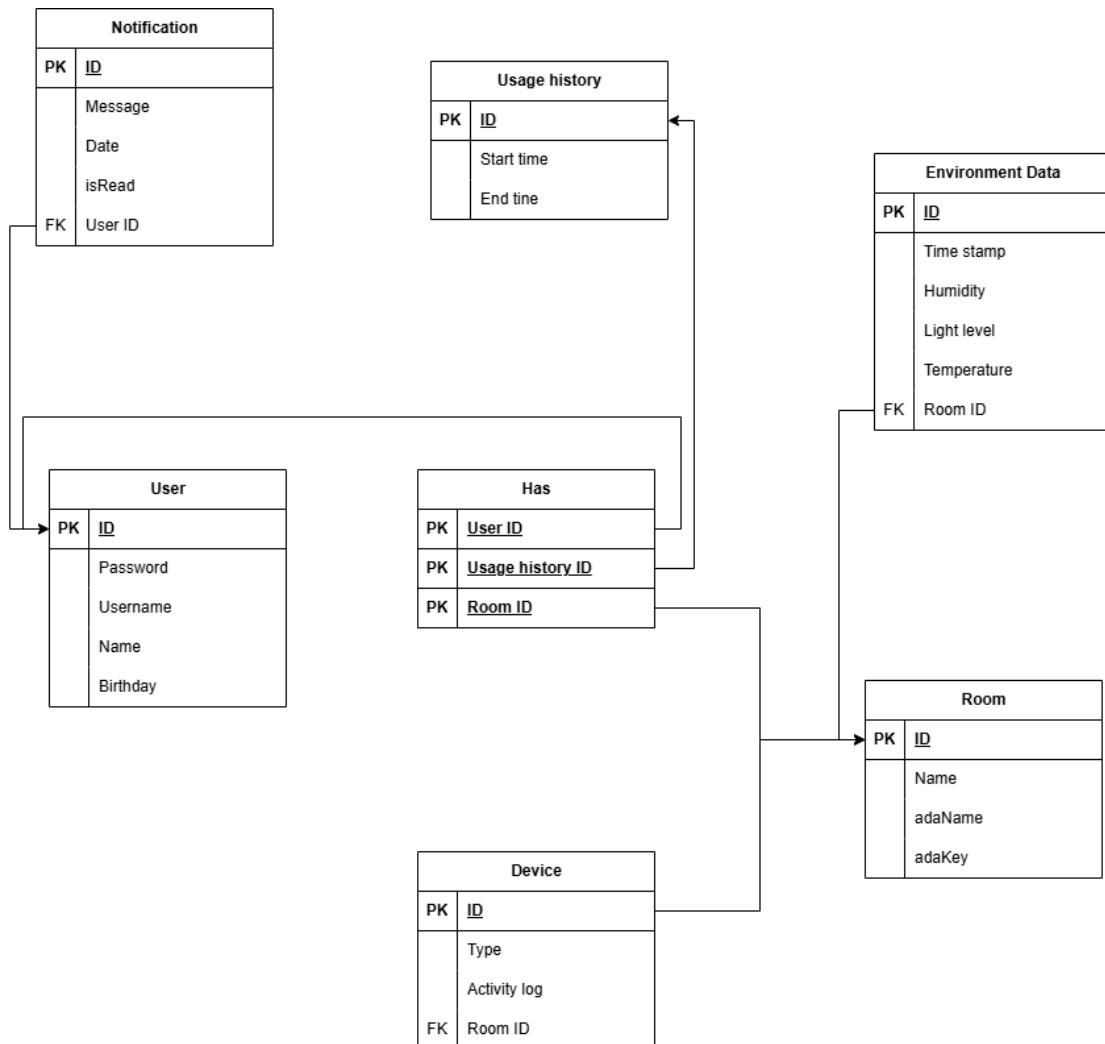
11 Database Design

11.1 Entity-Relationship Diagram (ER diagram)



Hình 18: ER Diagram

11.2 Relation Schema



Hình 19: Relation Schema

12 Design Pattern

12.1 MVC Pattern (Model - View - Controller)

Mô hình **Model-View-Controller** (MVC) là một mẫu thiết kế trong kỹ thuật phần mềm, được sử dụng để tách một ứng dụng thành ba thành phần chính: **Model**, **View**, và **Controller**. Việc phân chia này giúp tổ chức mã nguồn tốt hơn, làm cho ứng dụng dễ quản lý, mở rộng và dễ thích nghi với các thay đổi.

Dưới đây là chi tiết của từng thành phần

- **Model**: Chứa logic nghiệp vụ và quản lý dữ liệu của ứng dụng. Model chịu trách nhiệm



xử lý và lưu trữ dữ liệu, cung cấp các phương thức truy xuất dữ liệu. Trong dự án, các file như **Room.js**, **User.js**, và **Device.js** trong thư mục **src/models/** ở backend sẽ định nghĩa cấu trúc dữ liệu và các phương thức tương tác với cơ sở dữ liệu..

- **Controller:** Điều hướng các yêu cầu (requests) từ người dùng, xử lý logic nghiệp vụ, và tương tác với cả Model lẫn View. Trong dự án, các file như **roomController.js**, **authController.js** trong thư mục **src/controllers/** đảm nhận việc điều phối giữa Model và View.
- **View:** Là tầng chịu trách nhiệm giao tiếp với người dùng, hiển thị dữ liệu hoặc trả về kết quả cho client. Trong dự án, các API endpoints trong **src/routes/** cung cấp một giao diện để frontend hoặc client có thể tương tác với hệ thống qua các yêu cầu HTTP.

12.2 Observer Pattern

Observer Pattern là mẫu thiết kế cho phép một đối tượng tự động thông báo các thay đổi của mình cho các đối tượng khác đang theo dõi nó. Khi trạng thái của đối tượng được theo dõi thay đổi, các đối tượng đang đăng ký (observers) sẽ nhận được thông báo và thực hiện các hành động cần thiết.

Ứng dụng trong dự án:

- File **sensorDataObserver.js** trong **src/observers/** triển khai một lớp **SensorDataObserver** (các đối tượng theo dõi). Lớp này bao gồm các phương thức **subscribe**, và **notify** để quản lý và thông báo các thay đổi dữ liệu cảm biến.

```
class SensorDataObserver {  
    constructor() {  
        this.observers = [];  
    }  
  
    subscribe(observer) {  
        this.observers.push(observer);  
    }  
  
    unsubscribe(observer) {  
        this.observers = this.observers.filter(obs => obs !== observer);  
    }  
  
    notify(data) {  
        this.observers.forEach(observer => observer.update(data));  
    }  
}
```

- **DeviceStateObserver** được triển khai để theo dõi và thông báo các thay đổi trạng thái của thiết bị. Lớp **DeviceStateObserver** bao gồm các phương thức **subscribe**, **unsubscribe**, và **notify** giúp quản lý các observers và thông báo dữ liệu mới khi có sự thay đổi.

```
class DeviceStateObserver {  
    constructor() {
```



```
        this.observers = [];
    }

    subscribe(observer) {
        this.observers.push(observer);
    }

    unsubscribe(observer) {
        this.observers = this.observers.filter(obs => obs !== observer);
    }

    notify(data) {
        this.observers.forEach(observer => observer.update(data));
    }
}
```

Tác dụng:

- Giảm sự phụ thuộc giữa các component bằng cách cho phép chúng đăng ký hoặc hủy đăng ký khi cần thiết.
- Dễ dàng thêm hoặc xóa observers mà không ảnh hưởng đến code hiện có.
- Hữu ích trong các ứng dụng IoT khi cần cập nhật realtime dữ liệu cảm biến.

12.3 Singleton Pattern

Singleton Pattern đảm bảo rằng chỉ có một **instance** (thể hiện) duy nhất của một class được tạo ra và được chia sẻ trong suốt ứng dụng. Điều này giúp quản lý tài nguyên tập trung, đặc biệt hữu ích khi làm việc với các tài nguyên quan trọng như kết nối cơ sở dữ liệu.

Ứng dụng trong dự án:

- Trong **src/config/database.js**, một instance duy nhất của lớp **DatabaseConnection** được tạo và dùng làm kết nối với cơ sở dữ liệu.

```
// src/config/database.js
const instance = new DatabaseConnection();
Object.freeze(instance);
```

- Trong **src/services/adafruitHandler.js**, một instance duy nhất của lớp **AdafruitHandler** được tạo để xử lý kết nối với Adafruit IO.

```
// src/services/adafruitHandler.js
const adafruitHandlerInstance = new AdafruitHandler();
module.exports = adafruitHandlerInstance;
```

Tác dụng:



- Đảm bảo chỉ có một kết nối duy nhất với database hoặc MQTT để tránh lãng phí tài nguyên.
- Dễ dàng quản lý và duy trì tài nguyên hệ thống một cách tập trung, tiết kiệm tài nguyên.

12.4 Adapter Pattern

Adapter Pattern là mẫu thiết kế giúp chuyển đổi interface của một class thành **interface** mà client mong muốn. Nó cho phép các lớp với **interface** không tương thích có thể làm việc cùng nhau.

Ứng dụng trong dự án:

- File **adafruitAdapter.js** trong **src/adapters/** triển khai lớp **AdafruitAdapter** để kết nối với dịch vụ **MQTT** của **Adafruit IO**. Adapter này đóng gói toàn bộ logic kết nối, giúp các phần khác của ứng dụng có thể giao tiếp với **Adafruit IO** một cách dễ dàng mà không cần biết về chi tiết kết nối.

```
class AdafruitAdapter {  
    constructor(adaName, adaKey) {  
        this.client = mqtt.connect('mqtts://io.adafruit.com', {  
            username: adaName,  
            password: adaKey,  
        });  
    }  
  
    connect() {  
        return new Promise((resolve, reject) => {  
            this.client.on('connect', () => {  
                console.log('ket noi voi Adafruit IO thanh cong');  
                resolve();  
            });  
            this.client.on('error', (error) => {  
                console.error('Loi ket noi Adafruit IO:', error);  
                reject(error);  
            });  
        });  
    }  
  
    publish(feed, value) {  
        return new Promise((resolve, reject) => {  
            this.client.publish(`${this.client.options.username}/feeds/${feed}`,  
            value.toString(), (err) => {  
                if (err) {  
                    reject(err);  
                } else {  
                    resolve();  
                }  
            });  
        });  
    }  
  
    subscribe(feed, callback) {  
        const topic = `${this.client.options.username}/feeds/${feed}`;  
    }  
}
```



```
this.client.subscribe(topic);
this.client.on('message', (receivedTopic, message) => {
    if (receivedTopic === topic) {
        callback(message.toString());
    }
});

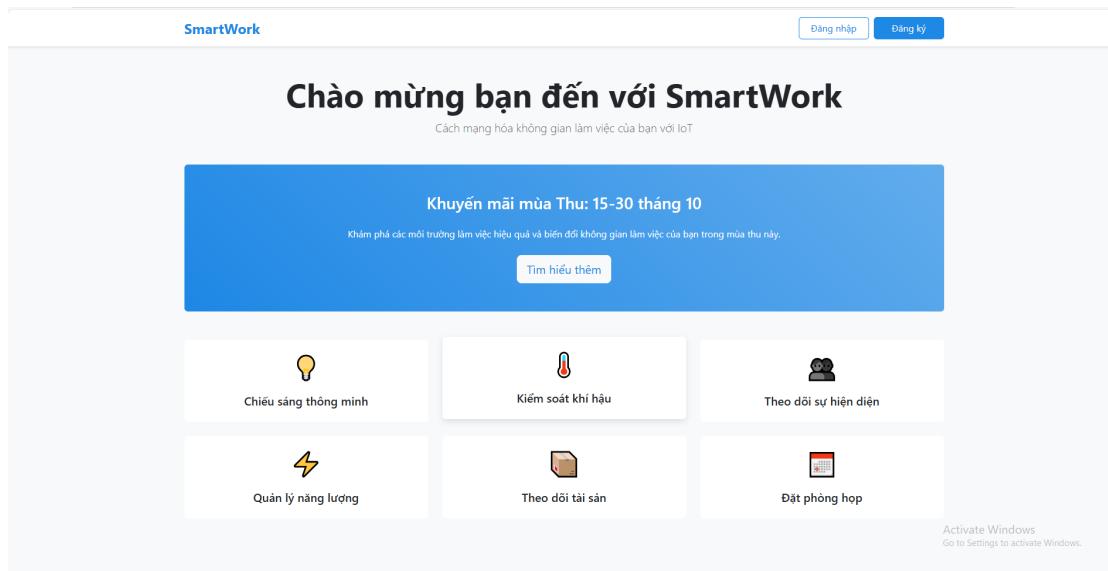
disconnect() {
    return new Promise((resolve) => {
        this.client.end(false, () => {
            console.log('Ngat ket noi khoi Adafruit IO');
            resolve();
        });
    });
}
```

Tác dụng:

- Cung cấp interface thống nhất cho việc giao tiếp với Adafruit IO, giúp dễ dàng thay đổi hoặc nâng cấp mà không ảnh hưởng đến mã nguồn đang sử dụng.
- Làm cho mã nguồn dễ dàng test vì adapter có thể được mock.
- Nếu cần thay đổi sang một dịch vụ IoT khác, có thể triển khai một adapter mới mà không cần thay đổi phần code còn lại.

13 Thành phẩm

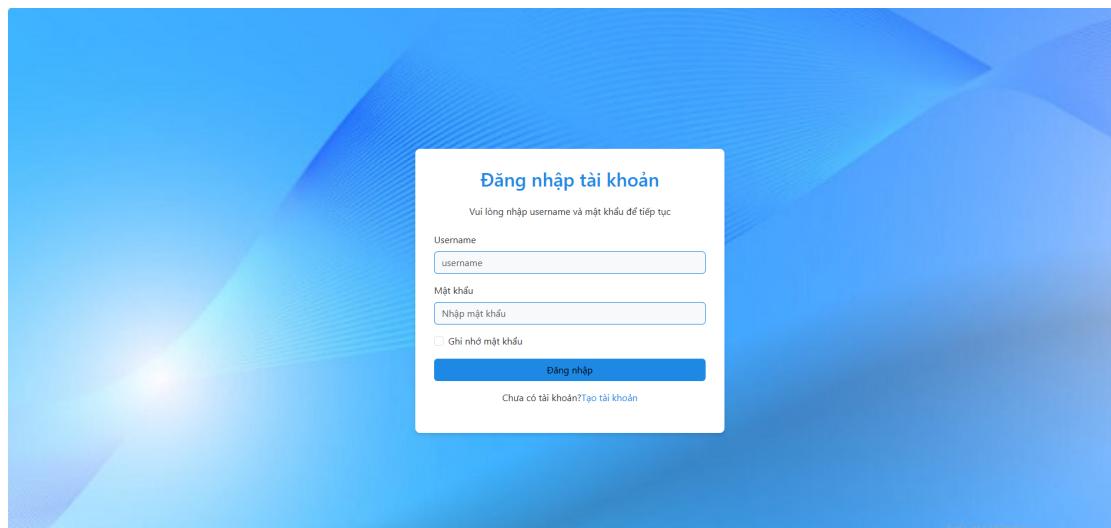
Nhóm đã triển khai ứng dụng web của mình với **Frontend** là giao diện người dùng cho hệ thống quản lý không gian làm việc IoT, được xây dựng bằng **React**; **Backend** là một hệ thống quản lý không gian làm việc IoT được xây dựng bằng **Node.js**, **Express** và **MongoDB**; **Gateway** là một ứng dụng gateway cho môi trường làm việc thông minh, được xây dựng bằng **Python**. Nó cung cấp các tính năng kiểm soát thiết bị, điều chỉnh tự động, điều khiển bằng giọng nói và nhận diện khuôn mặt để xác thực người dùng. Dưới đây là ảnh chụp màn hình của sản phẩm, minh họa quy trình làm việc điển hình:



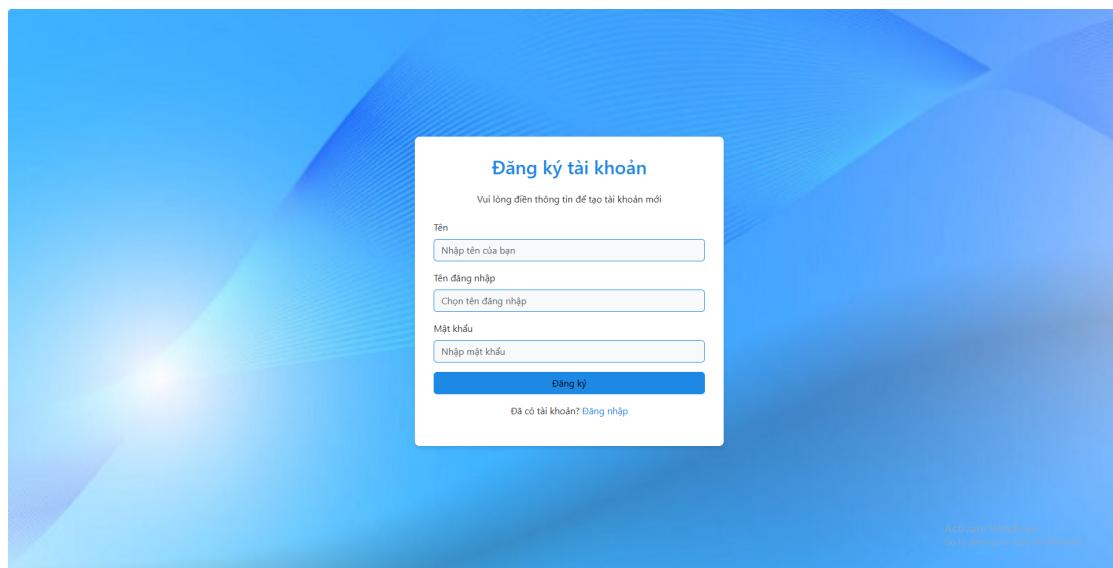
The screenshot shows the SmartWork landing page. At the top, there is a navigation bar with the SmartWork logo, a search bar, and two buttons: "Đăng nhập" (Login) and "Đăng ký" (Register). Below the header, a large banner says "Chào mừng bạn đến với SmartWork" (Welcome to SmartWork) and "Cách mạng hóa không gian làm việc của bạn với IoT" (Transform your workspace with IoT). A blue callout box highlights a "Khuyến mãi mùa Thu: 15-30 tháng 10" (Autumn promotion: October 15-30) offer. Below the banner are six cards representing different features: "Chiếu sáng thông minh" (Smart lighting), "Kiểm soát khí hậu" (Climate control), "Theo dõi sự hiện diện" (Presence tracking), "Quản lý năng lượng" (Energy management), "Theo dõi tài sản" (Asset tracking), and "Đặt phòng họp" (Meeting room booking). At the bottom right, there is a watermark: "Activate Windows Go to Settings to activate Windows."

Hình 20: Màn hình giới thiệu

- Đây là trang đầu tiên cho web khi mới bắt đầu truy cập vào. Nó có sự thay đổi nhỏ so với phiên bản mockup. Với lời chào mừng rõ ràng hơn và một slogan nhỏ,
- Thêm các khung với những icon mô tả chức năng của Smartwork, dễ người dùng hiểu rõ hơn về ứng dụng.
- Và vẫn còn những nút điều hướng cũ như Đăng nhập và đăng kí



Hình 21: Màn hình đăng nhập

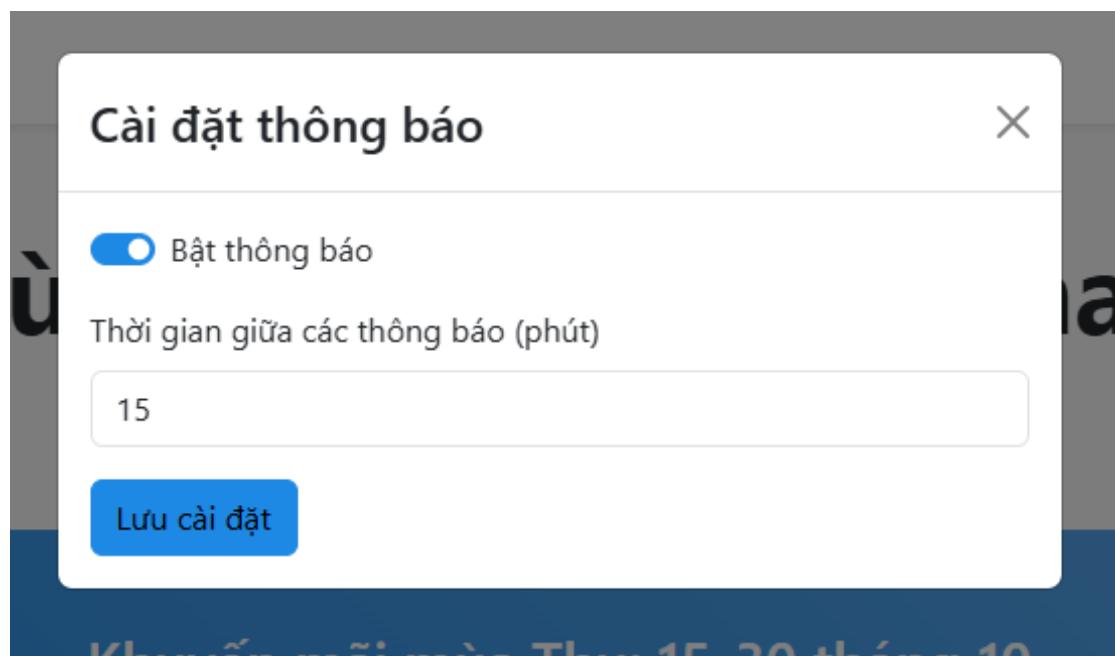


Hình 22: Màn hình đăng ký

- Trang đăng nhập và đăng ký không có nhiều thay đổi so với mockup chỉ thay hình nền và ngôn ngữ cho phù hợp với người dùng.

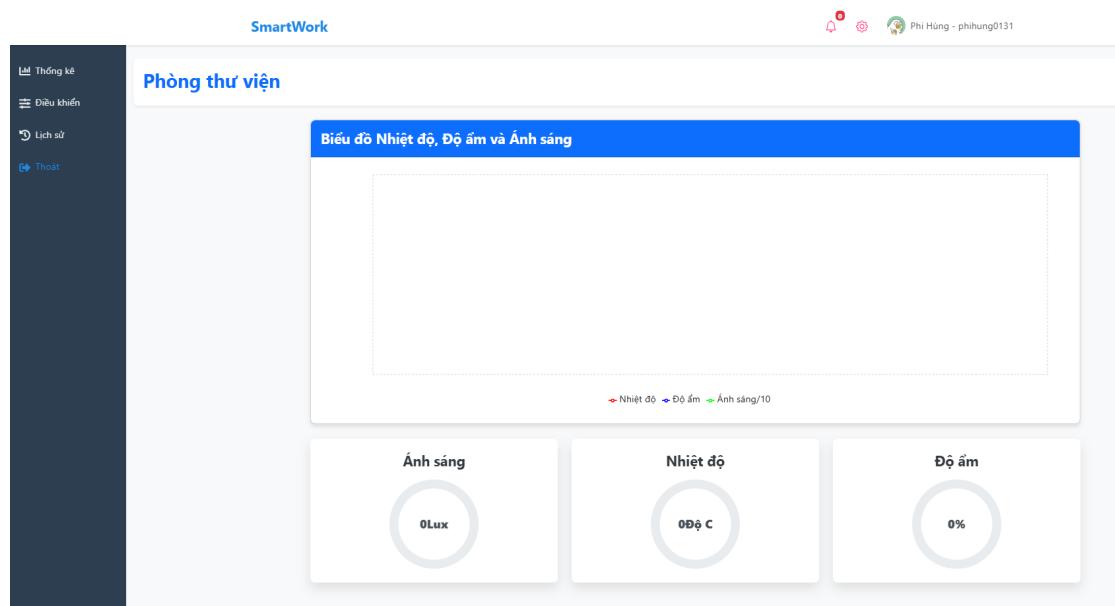
Hình 23: Màn hình danh sách phòng

- Trang danh sách phòng được hiển thị ngay sau khi đăng nhập thành công.
- Trang danh sách phòng cũng không có nhiều thay đổi so với mockup.
- Trang hiển thị các thông tin cơ bản của phòng như id, tên phòng, adafruit name và key nhưng đã che dấu và trạng thái của phòng.



Hình 24: module cài đặt thông báo

- Module này cho phép người dùng bật/tắt thông báo 1 cách chủ động với 1 khoản thời gian tùy chỉnh.



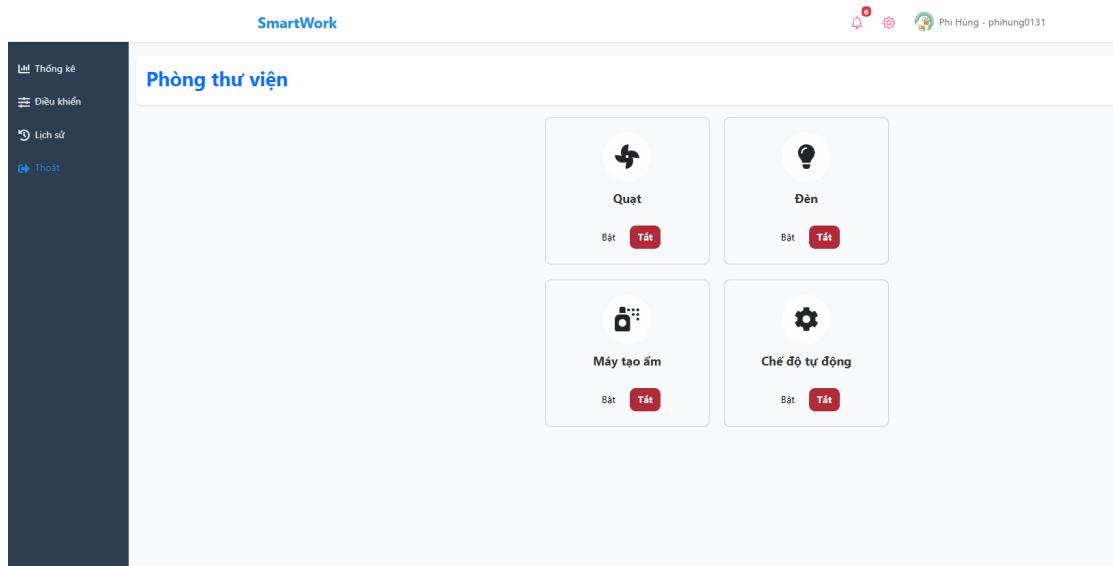
Hình 25: trang thông báo

- Khi ấn vào "xem" trong trang danh sách phòng tương ứng để mở trang thống kê của phòng



đó

- So với phiên bản mockup thì các trang hiển thị thông số của phòng cụ thể đã chuyển các nút thành thanh menu bên trái màn hình để người dùng dễ dàng hơn.
- Các dữ liệu được lấy từ adafruit hiện thị các thông số nhiệt độ, độ ẩm và ánh sáng của phòng, và trên là biểu đồ thay đổi của chúng theo thời gian.



Hình 26: trang điều khiển

- Không có sự thay đổi nhiều, chỉ đổi các icon biểu tượng cho thiết bị một cách trực quan hơn.
- So với phiên bản mockup thì các trang hiển thị thông số của phòng cụ thể đã chuyển các nút thành thanh menu bên trái màn hình để người dùng dễ dàng hơn.
- Trang điều khiển cho phép điều khiển các thiết bị IoT từ xa quan đại diện cho các thiết bị để người dùng dễ dàng thao tác và có chế độ tự động điều khiển các thiết bị cho phù hợp với mô trường phòng



The screenshot shows the SmartWork application interface. On the left is a dark sidebar with navigation links: 'Thống kê' (Statistics), 'Điều khiển' (Control), 'Lịch sử' (History), and 'Thoát' (Logout). The main content area has a header 'Phòng thư viện' (Library Room) and a sub-header 'Room Usage History'. It includes search fields for 'Username' (with placeholder 'Search by username'), 'Start Date' (with placeholder 'Select start date'), and 'End Date' (with placeholder 'Select end date'). There are also 'Refresh' and 'Reset Filters' buttons. Below these are two rows of buttons: 'Tìm kiếm' (Search) and 'Xóa' (Delete). The main table displays 'Room Usage History' with columns: 'Username', 'Status', 'Time', and 'Duration'. The data shows multiple entries for 'phihung0131' and 'none' at different times on 10/28/2024, with various status changes between 'IN' (green) and 'OUT' (red). The duration for most entries is listed as '-' or very short intervals like '0m 2s' or '0m 2s'. At the bottom right of the table are page navigation buttons: '<', '1', '2', '3', and '>'.

Username	Status	Time	Duration
phihung0131	IN	10/28/2024, 11:30:42 AM	-
none	OUT	10/28/2024, 11:30:29 AM	0m 2s
phihung0131	OUT	10/28/2024, 11:30:27 AM	2m 8s
phihung0131	IN	10/28/2024, 11:28:19 AM	-
none	OUT	10/28/2024, 11:28:12 AM	8m 16s
phihung0131	IN	10/28/2024, 11:19:56 AM	-
none	OUT	10/28/2024, 11:19:51 AM	1m 37s
phihung0131	IN	10/28/2024, 11:18:14 AM	-
none	OUT	10/28/2024, 11:18:14 AM	0m 2s
phihung0131	OUT	10/28/2024, 11:18:12 AM	-

Hình 27: trang điều khiển

- Được thêm mới so với phiên bản mockup
- Trang hiển thị lịch sử người dùng phòng nếu người chưa xác định danh tính trong dữ liệu thì sẽ hiển thị none, nếu người dùng ra khỏi phòng thì trạng thái hiển thị out và sẽ được tính toán thời gian trong phòng của người dùng
- Trang giúp kiểm soát những người đã vào phòng bằng camera để dễ dàng quản lý và tăng tính an toàn cho phòng.