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

THỰC TẬP ĐỒ ÁN MÔN HỌC ĐA NGÀNH

HƯỚNG CÔNG NGHỆ PHẦN MỀM (CO3109)

Báo cáo

Hệ thống nhà kho thông minh

GVHD: Mai Đức Trung
Nhóm: 4CE + 1CS

Họ và tên sinh viên	MSSV
Bùi Xuân Hùng	1913600
Nguyễn Thủy Ngọc	1911704
Đào Thị Tuyết Sương	1914979
Trương Phi Trường	1915749
Nông Trọng Thuyên	1915390

TP. Hồ Chí Minh, Tháng 5/2022

Mục lục

I Giới thiệu đề tài **3**

II Phân tích yêu cầu **5**

1 Yêu cầu chức năng 5

2 Yêu cầu phi chức năng 5

III Công nghệ sử dụng **6**

1 Giao thức MQTT 6

1.1 Publish, Subscribe 6

1.2 Kiến trúc mức cao của MQTT 6

1.3 Topic 7

1.4 MQTT Bridge 7

2 IoT Gateway 8

3 Adafruit 9

4 MVC Design Pattern 9

5 XAMPP 10

5.1 XAMPP 10

5.2 PHPMyAdmin 11

6 NodeJS và ExpressJS 13

6.1 NodeJS 13

7 ReactJS và Recharts 13

7.1 ReactJS 13

7.2 Recharts 15

IV Thiết bị sử dụng **16**

V Phân chia công việc **17**

VI Thiết kế hệ thống **18**

1 Usecase Diagram 18

1.1 Đăng kí 18

1.2 Đăng nhập 19

1.3 Đăng xuất 19

1.4 Xem dữ liệu cũ 19

1.5 Xem dữ liệu hiện tại 19

1.6 Điều khiển máy lạnh 19

1.7 Điều khiển cửa 20

1.8 Điều khiển đèn . . . 20
1.9 Quản lý thiết bị trong hệ thống . . . 20
1.10 Quản lý nhà kho trong hệ thống . . . 20
1.11 Phân quyền . . . 20
1.12 Cập nhật dữ liệu . . . 20
1.13 Kiểm tra độ ẩm, nhiệt độ . . . 20
1.14 Kiểm tra ánh sáng . . . 21
1.15 Kiểm tra nồng độ gas . . . 21
1.16 Đóng mở cửa . . . 21
1.17 Đóng mở máy lạnh . . . 21
1.18 Bật tắt đèn . . . 21
2 Activity Diagram . . . 22
3 Giao diện dự kiến . . . 22
3.1 Dashboard . . . 22
3.2 Control . . . 23
3.3 Manage . . . 23

VII Hiện thực hệ thống 24
1 Cấu trúc thư mục . . . 24
2 Xây dựng hệ cơ sở dữ liệu . . . 25
3 Xây dựng hệ thống backend . . . 25
4 Xây dựng hệ thống frontend . . . 26

VIII Báo cáo công việc 27

I Giới thiệu đề tài

Công nghệ là một phần thiết yếu trong cuộc sống ngày nay. Nhờ việc ứng dụng công nghệ vào đời sống sản xuất mà năng suất cũng như chất lượng và trải nghiệm người dùng ngày càng được cải thiện. Một trong những khía cạnh của công nghệ hiện hữu rất rõ trong cuộc sống ngày nay mà ta có thể kể đến là tự động hóa. Việc tự động hóa các dịch vụ trong đời sống giúp người dùng thuận tiện hơn trong việc sử dụng, cũng như thúc đẩy sự phát triển của nhiều dịch vụ mới. Rất nhiều mặt của đời sống đã được các công ty công nghệ ứng dụng việc tự động hóa, có thể kể đến như: dây chuyền sản xuất tự động, thiết bị cảm biến nhận biết giúp cảnh báo cháy nổ, thiết bị robot thông minh, linh kiện điện tử,...

Sự ra đời của Robot, xe tải không người lái, công nghệ cảm biến, IoT cùng những dòng kệ để hàng thông minh, hiện đại đã hội tụ đủ để khai sinh ra một nhà kho thông minh – smart warehouse. Với sự phát triển của Internet vạn vật, công nghệ IOT giúp nâng cao hiệu quả công việc, giảm chi phí lao động và tăng lợi nhuận của doanh nghiệp.

Hiện nay, ngày càng nhiều phần mềm được phát triển cho việc quản lý nhà kho. Các phần mềm này cung cấp giao diện dễ nhìn, dễ sử dụng và nâng cao hiệu quả trong việc theo dõi tình trạng nhà kho. Mỗi phần mềm thích hợp với một mô hình nhà kho khác nhau, được phát triển những chức năng khác nhau, do đó đều có những điểm mạnh riêng.

Trong đề tài này, nhóm chúng em sẽ mô phỏng mô hình nhà kho lạnh, xoay quanh các yếu tố cơ bản nhưng cũng không kém phần quan trọng ảnh hưởng đến việc lưu trữ các mặt hàng là ánh sáng, nhiệt độ, độ ẩm, khí gas và trạng thái đóng mở cửa của kho. Cụ thể, mô hình và phần mềm sẽ có một số đặc điểm và chức năng sau:

- Áp dụng cho nhà kho vừa và nhỏ, các yếu tố được áp dụng chung cho toàn bộ nhà kho.
- Đối với kho lạnh thì nhiệt độ, độ ẩm và ánh sáng là một vấn đề quan trọng.
- Với những kho đông lạnh cần nhiệt độ và độ ẩm ổn định, chúng ta cần theo dõi các thông số này nhờ các cảm biến. Trạng thái đóng mở cửa của kho cũng cần phải chú ý.
- Để tránh hiện tượng rò rỉ khí gas cho các thực phẩm đông lạnh, cảm biến khí gas là một điều cần thiết.
- Hướng đến khách hàng ở nhiều độ tuổi với trình độ sử dụng công nghệ khác

nhau. Hỗ trợ người dùng quản lý nhiều nhà kho.

Từ những đặc điểm và chức năng vừa xác định, có thể nhận thấy ta cần thiết kế một phần mềm đơn giản, dễ sử dụng, tập trung vào tốc độ và độ ổn định thay vì những chức năng phức tạp.

II Phân tích yêu cầu

1 Yêu cầu chức năng

- Hệ thống cho phép người dùng đăng nhập hoặc đăng kí một tài khoản mới và đăng xuất.
- Hệ thống cho phép người dùng thêm kho mới vào hệ thống.
- Hệ thống cho phép người dùng xem các dữ liệu hiện tại (gas, nhiệt độ, độ ẩm, cường độ ánh sáng, trạng thái đóng/mở cửa).
- Hệ thống cho phép người dùng xem các dữ liệu quá khứ (gas, nhiệt độ, độ ẩm, cường độ ánh sáng).
- Hệ thống cho phép người dùng bật/tắt máy lạnh.
- Hệ thống cho phép người dùng bật/tắt đèn.
- Hệ thống cho phép người dùng đóng/mở cửa.
- Hệ thống cảm biến báo hiệu nồng độ gas vượt ngưỡng trong không khí.
- Hệ thống cảm biến báo hiệu nhiệt độ không khí.
- Hệ thống cảm biến báo hiệu độ ẩm không khí.
- Hệ thống cảm biến báo hiệu cường độ ánh sáng.

2 Yêu cầu phi chức năng

- Hệ thống hoạt động toàn thời gian (24/7).
- Hệ thống dễ bảo trì và dễ sử dụng.
- Hệ thống cảm biến báo hiệu theo thời gian thực.

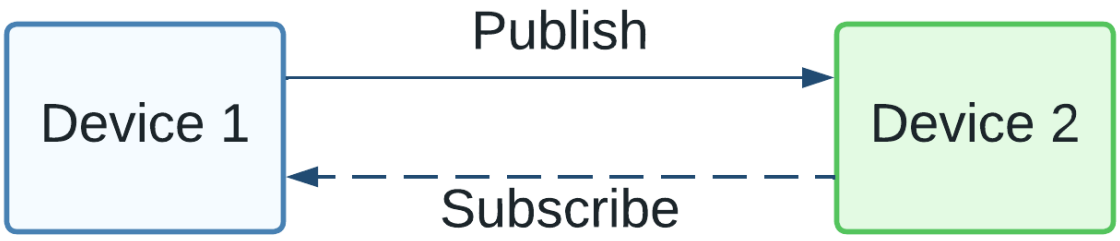
III Công nghệ sử dụng

1 Giao thức MQTT

Giao thức MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông điệp (message) theo mô hình publish/subscribe, sử dụng băng thông thấp, độ tin cậy cao và có khả năng hoạt động trong điều kiện đường truyền không ổn định.

1.1 Publish, Subscribe

Trong một hệ thống sử dụng giao thức MQTT để truyền nhận thông tin từ gateway lên server và ngược lại, nhiều node trạm (client) hay gateway kết nối tới một MQTT server (broker). Mỗi client sẽ đăng ký một hoặc vài kênh (topic, ví dụ như "/client1/channel1", "/client1/channel2". Quá trình đăng ký này gọi là **subscribe**, giống như chúng ta đăng ký nhận thông báo trên một kênh Youtube. Mỗi client sẽ nhận được dữ liệu khi có bất kì sự thay đổi thông tin nào trên kênh đã đăng ký tức là đã có trạm nào đó gửi dữ liệu và kênh đã đăng ký. Khi một client gửi dữ liệu tới kênh đó, quá trình này gọi là **publish**.



Hình 3.1: MQTT Publish - Subscribe
Nguồn: smartfactoryvn.com

1.2 Kiến trúc mức cao của MQTT

Kiến trúc mức cao (high-level) của MQTT gồm 2 phần chính là Broker (MQTT server) và Clients (devices/software components):

- Broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ client. Nhiệm vụ chính của broker là nhận message từ publisher, xếp các message theo hàng đợi rồi chuyển chúng tới một địa chỉ cụ thể.
- Client được chia thành 2 nhóm là publisher và subscriber. Client chỉ làm ít nhất một trong hai việc là xuất (publish) các message lên một topic cụ thể hoặc đăng

ký (subscribe) một topic nào đó để nhận message từ topic này.

Mỗi client sẽ đăng ký theo dõi các kênh thông tin (topic) hoặc gửi dữ liệu lên kênh thông tin đó. Quá trình đăng ký này gọi là "subscribe" và hành động một client gửi dữ liệu lên kênh thông tin được gọi là "publish". Mỗi khi kênh thông tin đó được cập nhật dữ liệu (dữ liệu này có thể đến từ các client khác) thì những client nào đã đăng ký theo dõi kênh này sẽ nhận được dữ liệu cập nhật đó.

1.3 Topic

Topic có thể được xem như một "đường truyền" logic giữa 2 điểm là publisher và subscriber. Về cơ bản, khi message được publish vào một topic thì tất cả những subscriber của topic đó sẽ nhận được message này.

Giao thức MQTT cho phép khai báo topic kiểu phân cấp.

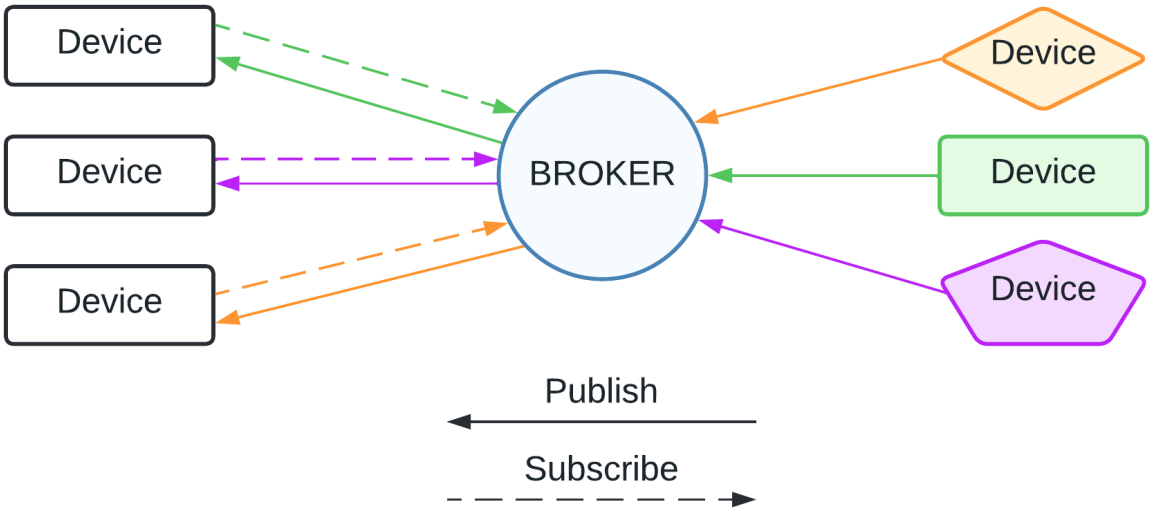
Giả sử chúng ta có một hệ thống cảm biến đo thông tin môi trường trong nhà kho của chúng ta. Như vậy, các topic phục vụ truyền tải thông tin môi trường cho nhà kho có thể được khai báo như sau:

- feeds/smart-warehouse.bbc-temp: topic thông tin nhiệt độ nhà kho.
- feeds/smart-warehouse.bbc-humi: topic thông tin độ ẩm nhà kho.
- feeds/smart-warehouse.bbc-light: topic thông tin cường độ ánh sáng nhà kho.

1.4 MQTT Bridge

MQTT Bridge là một tính năng của MQTT Broker cho phép các MQTT Broker có thể kết nối và trao đổi dữ liệu với nhau. Để sử dụng tính năng này, ta cần tối thiểu 2 Broker, trong đó, một Broker bất kỳ sẽ được cấu hình thành Bridge. Khi cấu hình MQTT bridge, ta cần lưu ý tới các thông số sau:

- **address**: địa chỉ của Broker cần kết nối.
- **bridge-protocol-version**: phiên bản của giao thức MQTT đang sử dụng chung cho 2 broker.
- **topic**: phần này định nghĩa 3 thông số: tên topic được trao đổi giữa 2 broker, chiều trao đổi (1 chiều hay 2 chiều) và topic mapping giữa 2 broker.



Hình 3.2: MQTT Bridge

2 IoT Gateway

IoT Gateway là một máy tính nhúng làm cầu kết nối giữa các cảm biến (sensor), tác nhân (actor) tới mạng diện rộng Internet hoặc mạng nội bộ Intranet.

Một số công nghệ truyền thông được sử dụng bởi các cảm biến: UART, Bluetooth, WiFi, BLE, Zigbee, Z-Wave, 6LoWPAN, NFC, WiFi Direct, GSM, LTE, LoRa, NB-IoT, LTE-M,..

Một số cảm biến có thể tạo ra hàng chục ngàn điểm dữ liệu mỗi giây. IoT gateway cung cấp một nơi để xử lý trước dữ liệu cục bộ ở vùng biên trước khi nó được gửi lên đám mây. Khi dữ liệu được tổng hợp, thu gọn và phân tích một cách khôn ngoan ở vùng biên, nó sẽ giảm thiểu khối lượng dữ liệu cần chuyển tiếp lên đám mây, điều này có thể tác động lớn đến thời gian hồi đáp và chi phí đường truyền mạng.

Một lợi ích khác của cổng IoT là nó có thể cung cấp cơ chế bảo mật bổ sung cho mạng IoT và dữ liệu mà nó vận chuyển. Vì gateway quản lý thông tin di chuyển theo cả hai chiều, nó có thể bảo vệ dữ liệu khi di chuyển lên đám mây khỏi các rò rỉ, hạn chế các thiết bị IoT bị xâm phạm bởi nguồn tấn công bên ngoài với các tính năng như phát hiện giả mạo, mã hóa, tạo số ngẫu nhiên bằng phần cứng và công cụ mã hóa.

Một IoT Gateway đa năng có thể thực hiện rất nhiều nhiệm vụ, ví dụ như:

- Tạo điều kiện giao tiếp với các thiết bị cũ hoặc không có kết nối internet.
- Bộ nhớ đệm dữ liệu, caching và media streaming.

– Xử lý trước dữ liệu (data pre-processing), làm sạch, lọc và tối ưu hóa.

– Tổng hợp dữ liệu thô.

– Liên lạc Deive-to-Device / Machine-to-Machine.

– Tính năng kết nối mạng và lưu trữ dữ liệu trực tiếp.

– Trực quan hóa dữ liệu và phân tích dữ liệu cơ bản thông qua các ứng dụng IoT Gateway.

– Tính năng lịch sử dữ liệu ngắn hạn.

– Bảo mật: quản lý truy cập người dùng và các tính năng bảo mật mạng.

– Quản lý cấu hình thiết bị.

– Chẩn đoán hệ thống

3 Adafruit

Adafruit là một nền tảng phần cứng và thị trường có một cộng đồng lớn và có thể trở thành nơi tốt nhất cho những người mới trong lĩnh vực điện tử và máy tính nhúng. Nền tảng này vừa cung cấp phần cứng và phụ kiện của riêng mình, vừa bán bo mạch của các nhà cung cấp khác như Raspberry Pi và Arduino.

Các bảng và phần mở rộng (cánh) của Adafruit Feather cực kỳ linh hoạt - phần mở rộng có thể hoạt động với bất kỳ bảng nào. Để kết nối các thiết bị với Internet và xử lý tất cả dữ liệu chúng tạo ra, Adafruit cung cấp dịch vụ đám mây Adafruit IO hoạt động cả với phần cứng Adafruit và Arduino.

Adafruit có thể không có nền tảng phần mềm IDE hoặc IoT của riêng mình, nhưng có một trong những cộng đồng mạnh nhất, hỗ trợ và cơ sở kiến thức để xây dựng dự án IoT và kết nối các đối tượng vật lý với Internet. Phần cứng của Adafruit cũng có lợi thế cạnh tranh. Bảng lông vũ cực kỳ nhẹ và đơn giản, vì vậy chúng sẽ trở thành bước khởi đầu tuyệt vời cho một thiết bị IoT nhỏ và không phức tạp như cảm biến đất hoặc thiết bị đeo theo dõi.

4 MVC Design Pattern

MVC là từ viết tắt của 'Model View Controller'. Nó đại diện cho các nhà phát triển kiến trúc áp dụng khi xây dựng các ứng dụng. Với kiến trúc MVC, chúng ta xem xét cấu trúc ứng dụng liên quan đến cách luồng dữ liệu của ứng dụng của chúng ta hoạt động như thế nào.

– **Model**: là nơi chứa những nghiệp vụ tương tác với dữ liệu hoặc hệ quản trị cơ sở dữ liệu (mysql, mssql...); nó sẽ bao gồm các class/function xử lý nhiều nghiệp vụ như kết nối database, truy vấn dữ liệu, thêm – xóa – sửa dữ liệu...

- **View**: là nơi chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh... nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.
- **Controller**: là nơi tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class/ function xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu thông tin cần thiết nhờ các nghiệp vụ lớp Model cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp View.

Sự tương tác giữa các thành phần trong mô hình MVC:

- Controller tương tác với qua lại với View.
- Controller tương tác qua lại với Model.
- Model và View không có sự tương tác với nhau mà nó tương tác với nhau thông qua Controller.

5 XAMPP

Nhóm lựa chọn xây dựng cơ sở dữ liệu sử dụng ngôn ngữ MySQL, đồng thời sử dụng Xampp để hỗ trợ Apache và dễ dàng quản lý databas với phpMyAdmin.

5.1 XAMPP

XAMPP hoạt động dựa trên sự tích hợp của 5 phần mềm chính là Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) và Perl (P), nên tên gọi XAMPP cũng là viết tắt từ chữ cái đầu của 5 phần mềm này:

- Chữ X đầu tiên là viết tắt của hệ điều hành mà nó hoạt động với: Linux, Windows và Mac OS X.
- Apache: Web Server mã nguồn mở Apache là máy chủ được sử dụng rộng rãi nhất trên toàn thế giới để phân phối nội dung Web. Ứng dụng được cung cấp dưới dạng phần mềm miễn phí bởi Apache Software Foundation.
- MySQL / MariaDB: Trong MySQL, XAMPP chứa một trong những hệ quản trị cơ sở dữ liệu quan hệ phổ biến nhất trên thế giới. Kết hợp với Web Server Apache và ngôn ngữ lập trình PHP, MySQL cung cấp khả năng lưu trữ dữ liệu cho các dịch vụ Web. Các phiên bản XAMPP hiện tại đã thay thế MySQL bằng MariaDB (một nhánh của dự án MySQL do cộng đồng phát triển, được thực hiện bởi các nhà phát triển ban đầu).
- PHP: Ngôn ngữ lập trình phía máy chủ PHP cho phép người dùng tạo các trang Web hoặc ứng dụng động. PHP có thể được cài đặt trên tất cả các nền tảng và hỗ trợ một số hệ thống cơ sở dữ liệu đa dạng.

– Perl: ngôn ngữ kịch bản Perl được sử dụng trong quản trị hệ thống, phát triển Web và lập trình mạng. Giống như PHP, Perl cũng cho phép người dùng lập trình các ứng dụng Web động.

Các ưu điểm của XAMPP bao gồm:

– Miễn phí, dễ cài đặt và sử dụng.

– Xampp hoạt động tốt trên cả Cross-platform, Linux, Window và MacOS.

– XAMPP có cấu hình đơn giản cũng như nhiều chức năng hữu ích cho người dùng. Tiêu biểu gồm: giả lập Server, giả lập Mail Server, hỗ trợ SSL trên Localhost.

– XAMPP tích hợp các thành phần với nhiều tính năng như Apache; PHP; MySql;... giúp hỗ trợ phát triển web toàn diện, không cần cài đặt thêm nhiều phần mềm.

– Mã nguồn mở: Không như Appserv, XAMPP có giao diện quản lý khá tiện lợi. Nhờ đó, người dùng có thể chủ động bật tắt hoặc khởi động lại các dịch vụ máy chủ bất kỳ lúc nào.

Tuy vậy, Xampp còn tồn tại một vài nhược điểm nhất định. Hãy cân nhắc chúng để đưa ra quyết định có nên sử dụng chương trình này không. Cụ thể:

– Không hỗ trợ Module

– Không được tích hợp Version MySQL, do đó, đôi khi sẽ mang đến sự bất tiện cho người dùng.

– Dung lượng khá nặng, khoảng 141MB cho file cài đặt.

– Lỗi Xampp thường gặp là Apache không start được, gây bất tiện vì người dùng thường phải đi sửa lỗi Xampp không start.

5.2 PHPMyAdmin

PhpMyAdmin là một công cụ mã nguồn mở miễn phí được viết bằng PHP nhằm giúp người dùng (các nhà quản trị cơ sở dữ liệu...) có thể quản lý cơ sở dữ liệu MySQL thông qua giao diện web thay vì sử dụng giao diện cửa sổ dòng lệnh (Command line interface). Sử dụng phpMyAdmin người dùng có thể thực hiện nhiều tác vụ khác nhau như khi sử dụng cửa sổ dòng lệnh. Bao gồm như việc tạo, cập nhật và xóa cơ sở dữ liệu, các bảng, phân quyền user...

Các tính năng chính:

– Giao diện web.

– Quản lý cơ sở dữ liệu MySQL.

– Nhập dữ liệu từ CSV và SQL.

– Xuất dữ liệu sang các định dạng khác nhau: CSV, SQL, XML, PDF (thông qua thư viện TCPDF), ISO/IEC 26300 – OpenDocument văn bản và bảng tính, Word, Excel, LaTeX và các định dạng khác.

– Quản lý nhiều máy chủ.

– Tạo PDF đồ họa của bố trí cơ sở dữ liệu.

– Tạo các truy vấn phức tạp bằng cách sử dụng Query-by-example (QBE).

– Tìm kiếm tổng quan trong cơ sở dữ liệu hoặc một tập hợp con của nó.

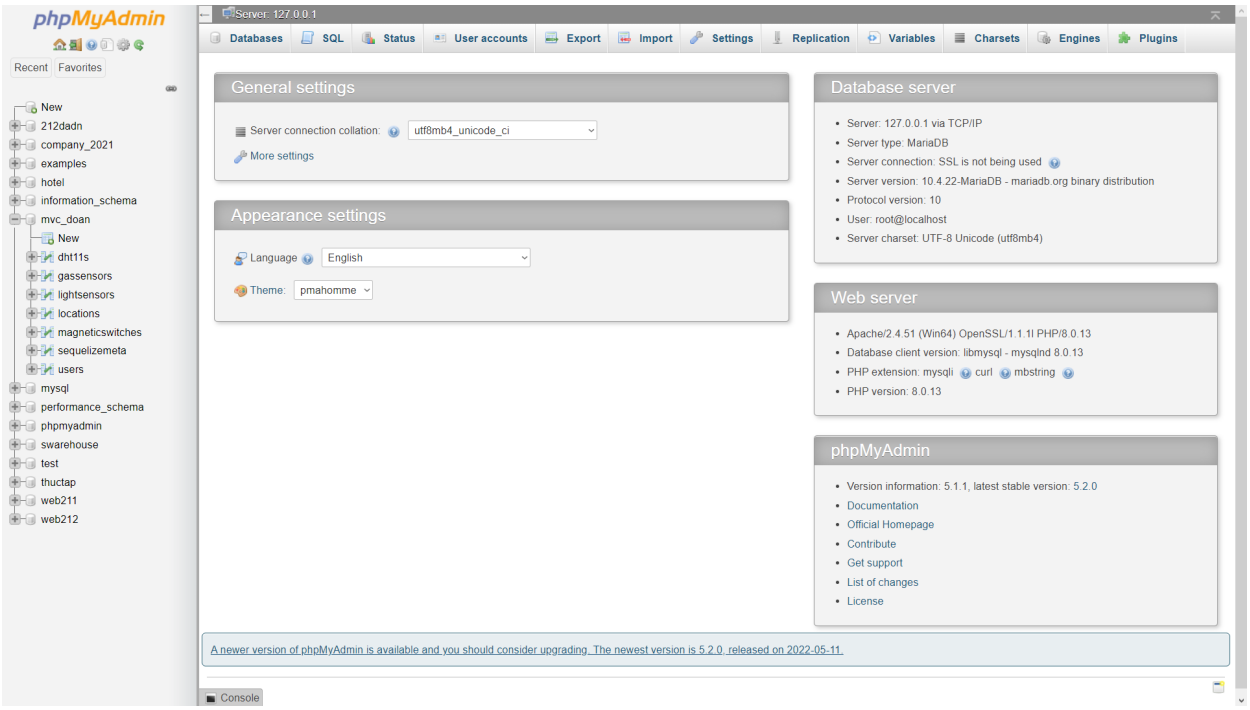
– Chuyển đổi dữ liệu được lưu trữ thành các định dạng bằng cách sử dụng một tập hợp các chức năng được xác định trước, như hiển thị dữ liệu BLOB như hình ảnh hoặc tải về liên kết.

– Giám sát các truy vấn (quy trình).

Ưu điểm:

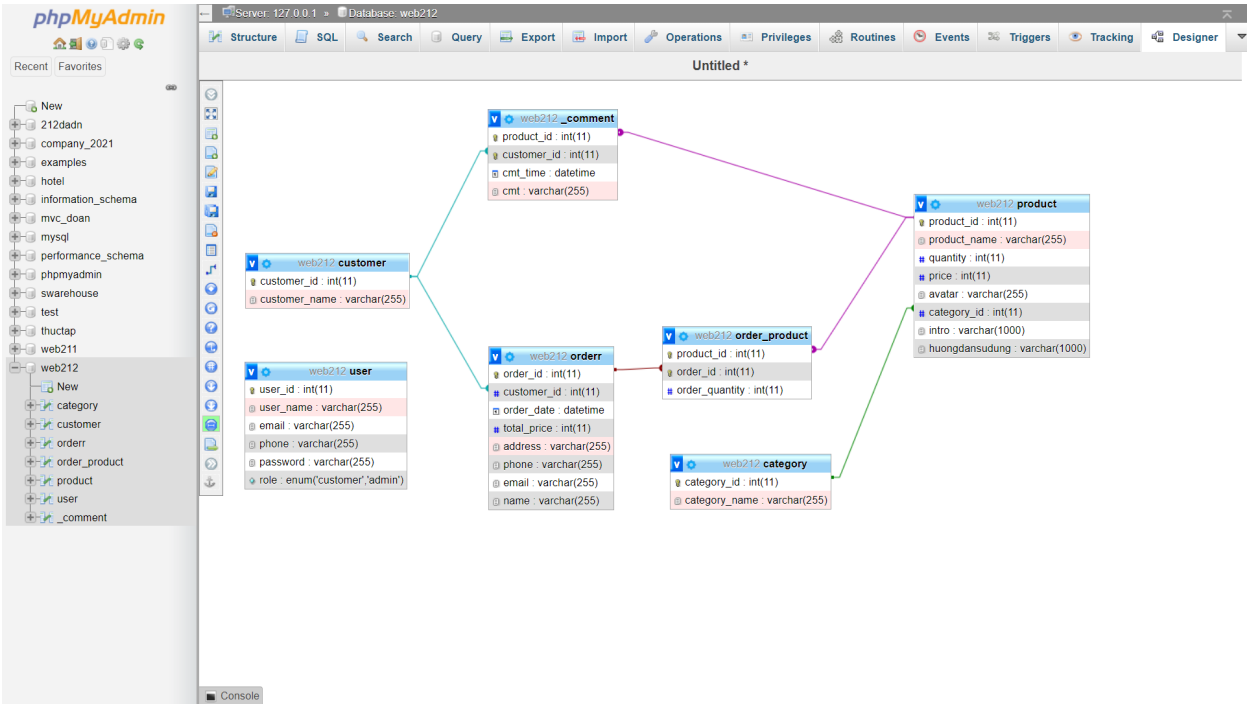
– PhpMyAdmin được tích hợp sẵn vào Xampp trong khi cài đặt.

– Giúp cho việc thực hiện các công việc như xem danh sách các database, cấu trúc table, chèn dữ liệu và thay đổi cấu trúc bảng 1 cách nhanh chóng và trực quan.



Hình 3.3: Trang chủ phpMyAdmin

– Design View: giúp người dùng xem thiết kế của database 1 các trực quan hơn.



Hình 3.4: Design View trong phpMyAdmin

6 NodeJS và ExpressJS

6.1 NodeJS

Nodejs là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

Nodejs được xây dựng và phát triển từ năm 2009, bảo trợ bởi công ty Joyent, trụ sở tại California, Hoa Kỳ. Dù sao thì chúng ta cũng nên biết qua một chút chút lịch sử của thứ mà chúng ta đang học một chút chứ nhỉ? =))

Phần Core bên dưới của Nodejs được viết hầu hết bằng C++ nên cho tốc độ xử lý và hiệu năng khá cao.

Nodejs tạo ra được các ứng dụng có tốc độ xử lý nhanh, realtime thời gian thực.

Nodejs áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án Startup nhanh nhất có thể.

7 ReactJS và Recharts

7.1 ReactJS

React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng

đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code javasscript trong code html thông qua các attribute như ng-model, ng-repeat...thì với react là một library cho phép nhúng code html trong code javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong JS.Tích hợp giữa javascript và HTML vào trong JSX làm cho các component dễ hiểu hơn

React là một thư viện UI phát triển tại Facebook để hỗ trợ việc xây dựng những thành phần (components) UI có tính tương tác cao, có trạng thái và có thể sử dụng lại được. React được sử dụng tại Facebook trong production, và www.instagram.com được viết hoàn toàn trên React.

Một trong những điểm hấp dẫn của React là thư viện này không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau. React so sánh sự thay đổi giữa các giá trị của lần render này với lần render trước và cập nhật ít thay đổi nhất trên DOM. Trước khi đến cài đặt và cấu hình, chúng ta sẽ đi đến một số khái niệm cơ bản:

- **Virtual DOM**: Công nghệ DOM ảo giúp tăng hiệu năng cho ứng dụng. Việc chỉ node gốc mới có trạng thái và khi nó thay đổi sẽ tái cấu trúc lại toàn bộ, đồng nghĩa với việc DOM tree cũng sẽ phải thay đổi một phần, điều này sẽ ảnh hưởng đến tốc độ xử lý. React JS sử dụng Virtual DOM (DOM ảo) để cải thiện vấn đề này.Virtual DOM là một object Javascript, mỗi object chứa đầy đủ thông tin cần thiết để tạo ra một DOM, khi dữ liệu thay đổi nó sẽ tính toán sự thay đổi giữa object và tree thật, điều này sẽ giúp tối ưu hoá việc re-render DOM tree thật.

  React sử dụng cơ chế one-way data binding – luồng dữ liệu 1 chiều. Dữ liệu được truyền từ parent đến child thông qua props. Luồng dữ liệu đơn giản giúp chúng ta dễ dàng kiểm soát cũng như sửa lỗi.

  Với các đặc điểm ở trên, React dùng để xây dựng các ứng dụng lớn mà dữ liệu của chúng thay đổi liên tục theo thời gian. Dữ liệu thay đổi thì hầu hết kèm theo sự thay đổi về giao diện. Ví dụ như Facebook: trên Newsfeed của bạn cùng lúc sẽ có các status khác nhau và mỗi status lại có số like, share, comment liên tục thay đổi. Khi đó React sẽ rất hữu ích để sử dụng.

- **JSX**: JSX là một dạng ngôn ngữ cho phép viết các mã HTML trong Javascript. **Faster**: Nhanh hơn. JSX thực hiện tối ưu hóa trong khi biên dịch sang mã Javacsript. Các mã này cho thời gian thực hiện nhanh hơn nhiều so với một mã tương đương viết trực tiếp bằng Javascript. Safer: an toàn hơn. Ngược với Javascript, JSX là kiểu statically-typed, nghĩa là nó được biên dịch trước khi

chạy, giống như Java, C++. Vì thế các lỗi sẽ được phát hiện ngay trong quá trình biên dịch. Ngoài ra, nó cũng cung cấp tính năng gỡ lỗi khi biên dịch rất tốt.

Easier: Dễ dàng hơn. JSX kế thừa dựa trên Javascript, vì vậy rất dễ dàng để cho các lập trình viên Javascripts có thể sử dụng.

- **Components**: React được xây dựng xung quanh các component, chứ không dùng template như các framework khác. Trong React, chúng ta xây dựng trang web sử dụng những thành phần (component) nhỏ. Chúng ta có thể tái sử dụng một component ở nhiều nơi, với các trạng thái hoặc các thuộc tính khác nhau, trong một component lại có thể chứa thành phần khác. Mỗi component trong React có một trạng thái riêng, có thể thay đổi, và React sẽ thực hiện cập nhật component dựa trên những thay đổi của trạng thái. Mọi thứ React đều là component. Chúng giúp bảo trì mã code khi làm việc với các dự án lớn. Một react component đơn giản chỉ cần một method render. Có rất nhiều methods khả dụng khác, nhưng render là method chủ đạo.
- **Props và State**:

  Props: giúp các component tương tác với nhau, component nhận input gọi là props, và trả thuộc tính mô tả những gì component con sẽ render. Prop là bất biến.

  State: thể hiện trạng thái của ứng dụng, khi state thay đổi thì component đồng thời render lại để cập nhật UI.

7.2 Recharts

Recharts là một thư viện được xây dựng dựa trên ReactJS và D3. D3.js là thư viện Javascript dùng để trực quan hoá dữ liệu, biến dữ liệu thành những đồ thị, bảng biểu xây dựng từ những native HTML5 elements (chứ không phải file ảnh), tăng tính tương tác với người dùng trong môi trường web.

Mục đích chính của Recharts là giúp các bạn vẽ biểu đồ trong thư viện ReactJS trở nên dễ dàng hơn bao giờ hết. Nguyên tắc chính của Recharts là:

- Dễ dàng triển khai nó trong Component của ReactJS.
- Hỗ trợ SVG và cực kỳ nhẹ.
- Các thành phần bên trong chart được trình bày rất "thuần khiết".

IV Thiết bị sử dụng

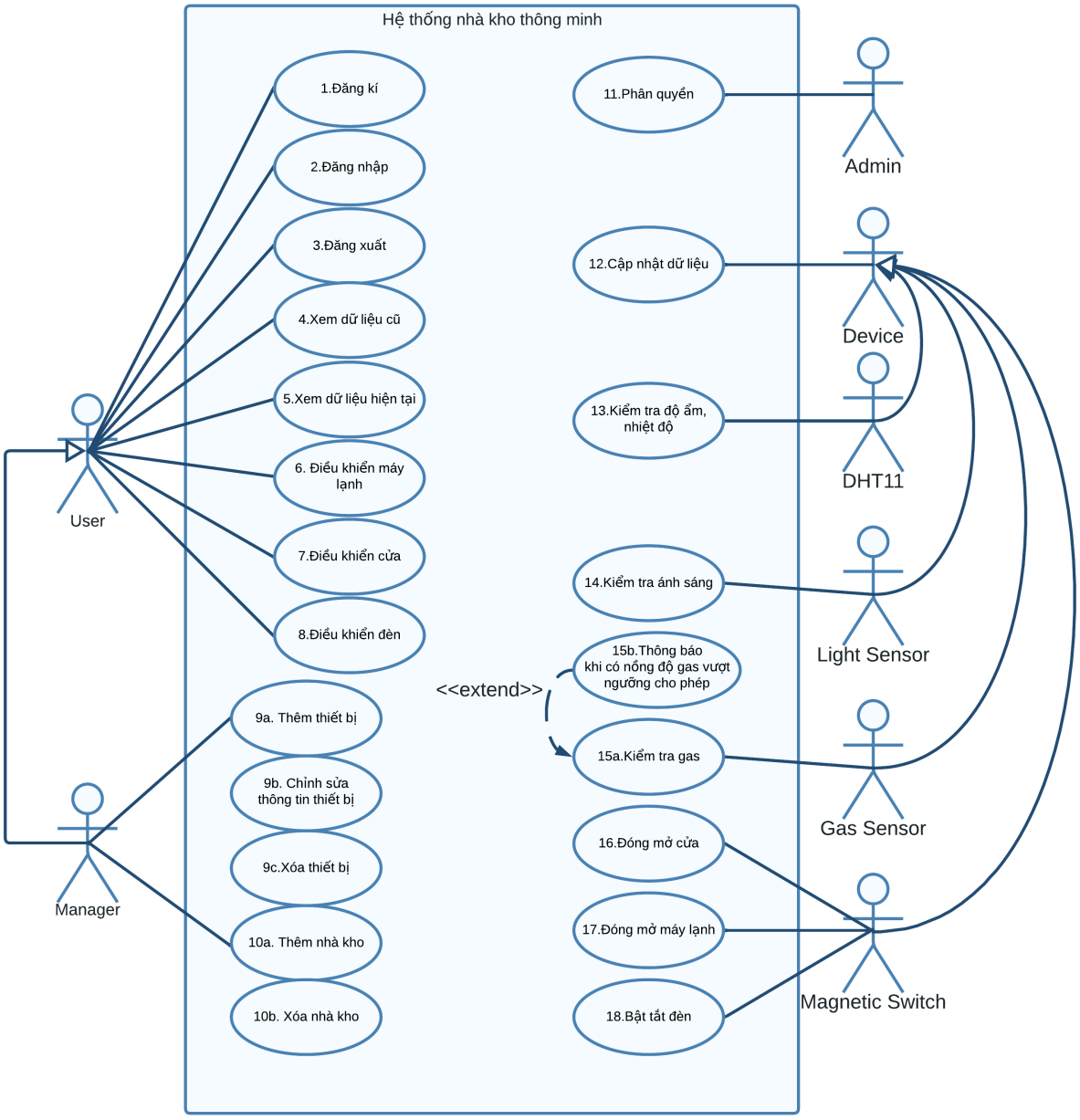
Input	Output
DHT11	LCD I2C
Light sensor	Mini pump
Magnetic switch	Relay circuit
Gas sensor	Buzzer
	A 2-color single LED

V Phân chia công việc

Họ và tên	**MSSV**	**Nội dung công việc**	**Hoàn thành**
Bùi Xuân Hùng	1913600	Triển khai backend (NodeJS theo mô hình MVC), thiết lập tương tác giữa database (MySQL) với backend - sử dụng Sequelize	100%
Nguyễn Thủy Ngọc	1911704	Thiết kế cơ sở dữ liệu, vẽ diagram	100%
Đào Thị Tuyết Sương	1914979	Thiết kế và xây dựng frontend với ReactJS	100%
Trương Phi Trường	1915749	Thiết lập phần cứng và dữ liệu adafruit	100%
Nông Trọng Thuyên	1915390	Thiết lập kết nối MQTT, xây dựng API và thiết lập tương tác giữa frontend và backend	100%

VI Thiết kế hệ thống

1 Usecase Diagram



Hình 6.1: Usecase Diagram

1.1 Đăng kí

Use-case name	Đăng kí
Actor	User
Description	User tạo tài khoản để đăng nhập vào ứng dụng
Preconditions	Tài khoản chưa được đăng ký trước đó
Postcondition	Một tài khoản mới được tạo và user có thể sử dụng nó để đăng nhập vào ứng dụng. User được chuyển đến giao diện chính.
Normal Flows	1. Ở giao diện đăng nhập, nhất vào dòng chữ "Not a member? Sign up now!". 2. Nhập đầy đủ thông tin. 3. Nhấn nút ""Sign Up"". 4. User được chuyển đến giao diện chính.
Exceptions	Không có (nếu trùng thông tin)
Alternative Flows	Không có

1.2 Đăng nhập

Use-case name	Đăng nhập
Actor	User, Manager
Description	User đăng nhập vào ứng dụng
Preconditions	User đã tạo tài khoản trước đó
Postcondition	User được chuyển đến giao diện chính
Normal Flows	1. User nhập username và password. 2. Nhấn nút "Log in" 3. Website chuyển tới giao diện tương ứng
Exceptions	Nếu user chưa có tài khoản hoặc sai mật khẩu thì hiển thị thông báo "blablo"
Alternative Flows	Không có

1.3 Đăng xuất

1.4 Xem dữ liệu cũ

1.5 Xem dữ liệu hiện tại

1.6 Điều khiển máy lạnh

1.7 Điều khiển cửa

1.8 Điều khiển đèn

1.9 Quản lý thiết bị trong hệ thống

a. Thêm thiết bị

b. Chỉnh sửa thông tin thiết bị

c. Xóa thiết bị

1.10 Quản lý nhà kho trong hệ thống

a. Thêm nhà kho

b. Xóa nhà kho

1.11 Phân quyền

1.12 Cập nhật dữ liệu

1.13 Kiểm tra độ ẩm, nhiệt độ

1.14 Kiểm tra ánh sáng

1.15 Kiểm tra nồng độ gas

a. Kiểm tra nồng độ gas

b. Thông báo khi nồng độ gas vượt ngưỡng cho phép

1.16 Đóng mở cửa

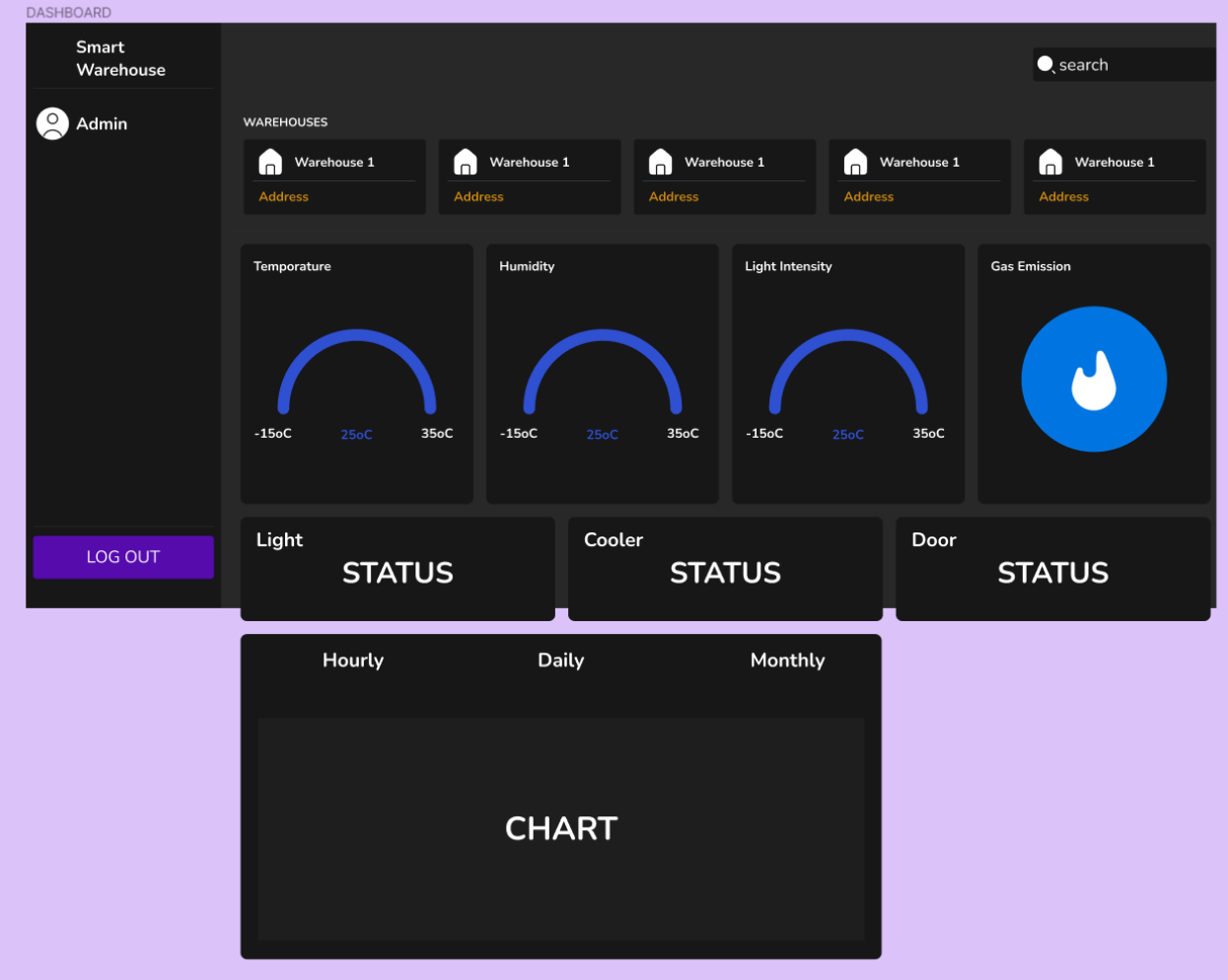
1.17 Đóng mở máy lạnh

1.18 Bật tắt đèn

2 Activity Diagram

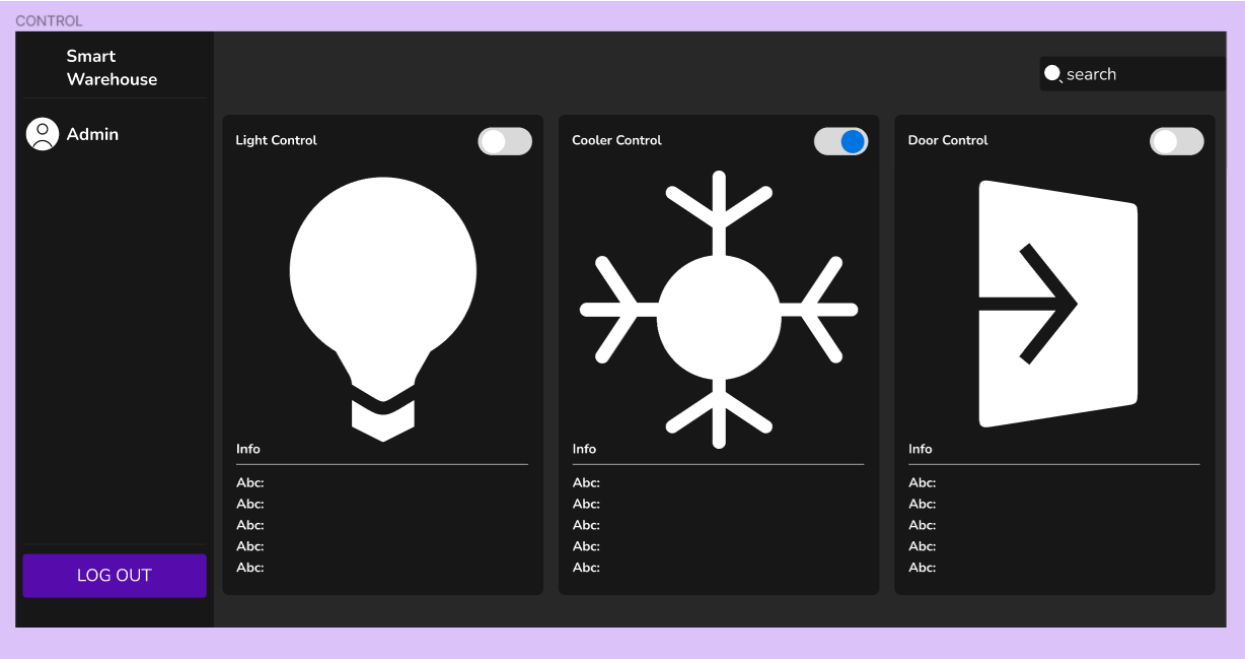
3 Giao diện dự kiến

3.1 Dashboard



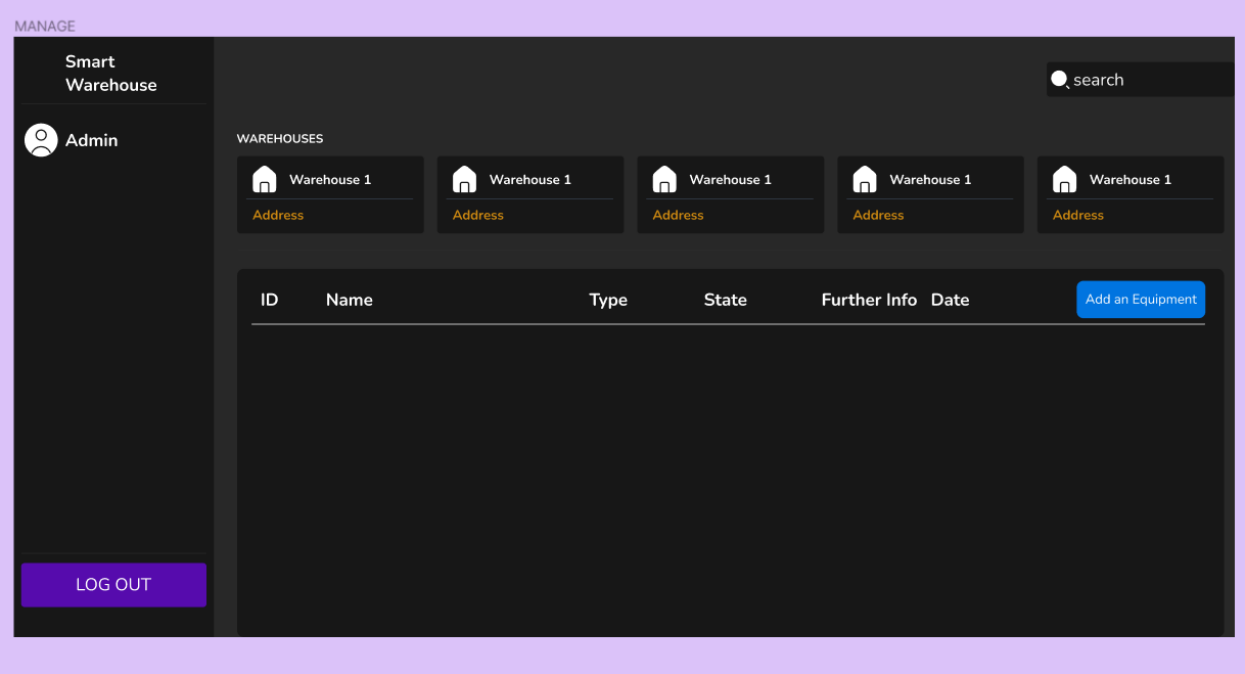
Hình 6.2: Giao diện hiển thị thông tin dữ liệu từ thiết bị

3.2 Control



Hình 6.3: Giao diện điều khiển thiết bị

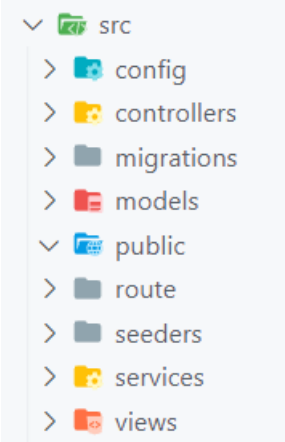
3.3 Manage



Hình 6.4: Giao diện quản lý thiết bị và nhà kho

VII Hiện thực hệ thống

1 Cấu trúc thư mục



Hình 7.1: Cấu trúc thư mục

Những phần bắt buộc phải có:

- **public**: chứa các thành phần public như file css, js, hình ảnh, video...
- **config**: thư mục này liên quan đến việc connect giữa các thành phần. Nó xử lí các vấn đề giữa các bên cần giao tiếp như giao tiếp giữa backend và mysql, backend và phần hardware (MQTT).
- **route**: đảm nhận nhiệm vụ điều hướng người dùng thông qua các url và theo chuẩn RestAPI tức là có 4 giao thức chính: GET, PUT, POST, DELETE.
- **controllers**: đóng vai trò như một middleware. Sau khi route nhận request từ người dùng, controller, nằm giữa model và view, sẽ lấy dữ liệu thông qua models và hiển thị đến view.
- **models**: khi liên kết database thành công, thì các bảng dữ liệu sẽ được xem như là một class với các thuộc tính là các trường dữ liệu và các hàm liên quan sẽ là method.
- **views**: những thành phần hiển thị bên phía người dùng - client side.

Dựa trên thiết kế của phần mềm thì sẽ có thêm các thư mục khác. Trong đồ án này, sau đây là những phần được thêm vào để góp phần hoàn thiện phần mềm:

- **migrations**: Migration là kỹ thuật trong việc tương tác với cơ sở dữ liệu, theo đó việc thay đổi về cấu trúc CSDL ở code sẽ được cập nhật lên CSDL đảm bảo dữ liệu đang tồn tại không bị mất.

– **seeders**: mô phỏng dữ liệu trong quá trình phát triển phần mềm.

– **services**: chứa các hàm bổ sung trong quá trình sử dụng dữ liệu, quy định các hàm public/subcribe giữa MQTT và NodeJS.

2 Xây dựng hệ cơ sở dữ liệu

3 Xây dựng hệ thống backend

Sau đây là các thư viện được sử dụng trong đồ án này:

```
"author": "HungBui",
"license": "ISC",
"dependencies": {
  "body-parser": "^1.19.0",
  "dotenv": "^8 .2.0",
  "ejs": "^3.1.5",
  "express": "^4.17.1",
  "mysql2": "^2.2.5",
  "sequalize": "^6.6.2"
},
"devDependencies": {
  "@babel/core": "^7.12.10",
  "@babel/node": "^7.12.10",
  "@babel/preset-env": "^7.12.11",
  "nodemon": "^2.0.7",
  "sequelize-cli": "^6.2.0"
}
```

Hình 7.2: Các thư viện được sử dụng

* **dependencies**: chứa các thư viện dùng cho NodeJS.
  – **body-parser**: Là phần mềm phân tích phần body của NodeJS, trả về một function hoạt động như một middleware.
  – **dotenv**: là một module zero-dependency, tải các biến môi trường từ tệp .env vào process.env. Dotenv lưu cấu hình trong môi trường tách biệt với code.
  – **ejs**: viết tắt của "Embedded JavaScript templating", đây là một thư viện, được sử dụng để phân tích các tập tin ejs, và tạo ra HTML trả về cho client (Trình duyệt).
  – **express**: Express.js (Hoặc đơn giản là Express) là một Web Application Framework cho NodeJS. Cung cấp bộ tính năng mạnh mẽ cho các ứng dụng web và mobile. Express hỗ trợ các phương thức HTTP và midleware tạo ra

môt API vô cùng mạnh mẽ và dễ sử dụng.

- **mysql2**: hỗ trợ NodeJS làm việc với MySQL.
- **sequelize**: là một ORM (Object Relational Mapping) dành cho Node.js và io.js. Nó hỗ trợ bạn truy cập một cách dễ dàng đến PostgreSQL, MySQL, MariaDB, SQLite và MSSQL cùng với các tính năng như là relations, transaction, replication ...

* **devDependencies**: chứa các thư viện dùng cho việc phát triển phần mềm.
  - **babel**: Babel là một bộ công cụ lập trình được sử dụng chủ yếu để chuyển đổi mã Javascript ECMAScript từ đời 2015 trở lên thành các mã Javascript đời thấp hơn để có thể tương thích ngược với các trình duyệt, môi trường hiện tại hoặc cũ hơn.
  - **@babel/core**: Gói core cơ bản của Babel, dùng để chạy bất kỳ các thiết lập / cấu hình của babel.
  - **@babel/node**: Là một CLI tương tự Node CLI, sử dụng để biên dịch tương thích với các cài đặt của Babel Preset và Babel Plugins.
  - **@babel/preset-env**: là một smart preset tự động sử dụng phiên bản Javascript mới nhất mà không cần khai báo cụ thể từng phiên bản một.
  - **nodemon**: là một tiện ích giao diện dòng lệnh (CLI) được phát triển bởi @rem để bao bọc ứng dụng Node của bạn, xem hệ thống file và tự động khởi động lại quá trình.
  - **sequalize-cli**: hỗ trợ liên kết NodeJS với database như Postgres, MySQL, MariaDB, SQLite và Microsoft SQL Server.

4 Xây dựng hệ thống frontend

VIII Báo cáo công việc