HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO CUỐI KỲ MÔN IoT VÀ ỨNG DỤNG

Đề tài: Hệ thống cho thú cưng ăn tự động

Nhóm 8

Giáo viên hướng dẫn: TS. Kim Ngọc Bách

Thành viên:

1. B22DCDT192 Nguyễn Bình Minh
2. B22DCVT345 Nguyễn Đức Minh
3. B22DCAT233 Đỗ Trung Quân
4. B22DCCN845 Trần Thanh Thúy

Hà Nội - 11/2025

MỤC LỤC

I. Giới thiệu đề tài

1. Mô tả đề tài

Ngày nay với sự phát triển vượt bậc của khoa học kỹ thuật, công nghệ IoT (Internet of Things) đã được ứng dụng ở rất nhiều lĩnh vực trong thực tế để phục vụ nhu cầu: chăm sóc sức khỏe, quản lý thời gian và nâng cao chất lượng cuộc sống cho con người.

Khi đời sống con người được cải thiện thì việc nuôi thú cưng trong gia đình đang trở nên phổ biến, đặc biệt là ở các thành phố lớn. Bên cạnh niềm vui mà thú cưng mang lại, một vấn đề khác cũng được quan tâm đó là: chăm sóc thú cưng khi chủ nhân bận rộn. Khi con người phải đi làm xa, công tác dài ngày hoặc đơn giản là quên mất giờ cho thú cưng ăn, việc này có thể ảnh hưởng nghiêm trọng đến sức khỏe và tâm lý của vật nuôi. Không những vậy, việc không kiểm soát được lượng thức ăn có thể dẫn đến tình trạng thú cưng ăn quá nhiều hoặc quá ít, gây béo phì hoặc suy dinh dưỡng. Vì vậy, vấn đề cho thú cưng ăn đúng giờ, đúng lượng và theo dõi được chế độ dinh dưỡng là một việc rất cần thiết với người thường xuyên nuôi thú cưng. Đặc biệt là các bạn sinh viên, người đi làm văn phòng thường xuyên phải đi làm cả ngày, không có thời gian chăm sóc thú cưng chu đáo. Xuất phát từ ý tưởng và tình hình thực tế nhóm chúng em thấy đây là một đề tài hay, có tính ứng dụng cao và có thể phát triển nên em đã chọn đề tài này làm đề tài chính trong bài tập lớn môn học.

Để khắc phục vấn đề chăm sóc thú cưng khi bận rộn, nhóm 8 chúng em đã quyết định chọn đề tài: hệ thống cho thú cưng ăn tự động thông minh. Thiết bị này giúp chúng ta dễ dàng cho thú cưng ăn từ xa thông qua ứng dụng điện thoại, đặt lịch cho ăn tự động theo giờ, và theo dõi được lượng thức ăn mà thú cưng đã tiêu thụ. Hệ thống có thể được sử dụng trong các hộ gia đình, cửa hàng thú cưng, và các cơ sở chăm sóc động vật, góp phần nâng cao chất lượng cuộc sống cho cả người và vật nuôi.

2. Mục tiêu hệ thống

*** Mục tiêu tổng quát:**

Xây dựng hệ thống cho ăn thú cưng thông minh có khả năng tự động và điều khiển từ xa thông qua nền tảng IoT. Hệ thống cho phép người dùng quản lý, giám sát và kiểm soát chế độ ăn của thú cưng một cách chủ động, thuận tiện và chính xác, kể cả khi không có mặt tại nhà.

*** Mục tiêu cụ thể:**

- Điều khiển từ xa: Cho phép người dùng kích hoạt quá trình cho ăn tức thì thông qua ứng dụng web (ReactJS) kết nối với hệ thống qua giao thức MQTT.

- Lập lịch tự động: Cho phép đặt lịch cho ăn định kỳ trong ngày; Node.js Backend lưu lịch và gửi lệnh điều khiển đến ESP32 đúng thời gian đã định.

- Theo dõi lượng thức ăn: Sử dụng Load Cell Sensor + HX711 để đo khối lượng thức ăn trước và sau khi phân phối, ghi nhận và hiển thị dữ liệu thống kê trên web dashboard.

- Điều khiển bằng giọng nói: Hỗ trợ người dùng ra lệnh bằng giọng nói thông qua Web Speech API, giúp tăng tính tiện dụng và hiện đại.

- Chức năng mở rộng: Hệ thống mở rộng tích hợp AI để tối ưu khẩu phần ăn dựa trên thói quen của từng vật nuôi.

3. Phạm vi triển khai

*** Số lượng và loại thiết bị:**

- 01 module ESP32: Đảm nhiệm việc điều khiển các cảm biến, cơ cấu xả thức ăn, kết nối Internet và giao tiếp dữ liệu với backend thông qua giao thức MQTT.
- 01 cảm biến Load Cell (5kg) + module HX711: Dùng để đo khối lượng thức ăn trong khay và theo dõi lượng thức ăn đã tiêu thụ.
- 01 động cơ Servo: Điều khiển nắp hoặc trục quay xả thức ăn ra khay.
- 01 hệ thống LED và nút nhấn (Button): Hiển thị trạng thái và cho phép thao tác thủ công khi cần.
- 01 bộ nguồn DC ổn định (hoặc pin sạc dự phòng): Cung cấp năng lượng cho toàn bộ hệ thống.

*** Môi trường hoạt động:**

- Vị trí đặt thiết bị: Trong nhà hoặc khu vực bán ngoài trời (nơi đặt khay ăn của thú cưng).
- Điều kiện hoạt động: Môi trường có độ ẩm và bụi nhẹ; cần đảm bảo khung bảo vệ thiết bị chống nước cơ bản cho các thành phần điện tử (đặc biệt là Load Cell và ESP32).
- Nguồn điện: Sử dụng adapter hoặc pin sạc.

*** Phạm vi kết nối và quản lý:**

- Kết nối mạng: Thiết bị ESP32 kết nối Wi-Fi nội bộ và truyền dữ liệu đến backend (Node.js) thông qua MQTT broker.
- Quản lý và giám sát: Người dùng thao tác qua web dashboard (ReactJS), có thể theo dõi dữ liệu và điều khiển cho ăn từ xa.
- Số lượng hệ thống triển khai: Mô hình thử nghiệm 1 bộ.

4. Tổng quan phương hướng

Đề tài "Hệ thống cho thú cưng ăn tự động" được nhóm triển khai theo hướng phát triển một mô hình IoT hoàn chỉnh, bao gồm cả phần cứng (thiết bị thực tế) và phần mềm (ứng dụng điều khiển và giám sát). Hệ thống hướng đến việc giải quyết bài toán thực tế: người nuôi thú cưng thường không có thời gian cho ăn đúng giờ hoặc kiểm soát lượng thức ăn phù hợp.

Phương hướng tổng thể của nhóm được chia thành 4 giai đoạn chính:

Giai đoạn 1 – Nghiên cứu và thiết kế hệ thống:

Nhóm tiến hành tìm hiểu các thành phần cơ bản của một hệ thống IoT theo mô hình ba lớp:

- *Lớp cảm nhận (Perception Layer):* Bao gồm Load Cell Sensor (đo khối lượng thức ăn), Servo Motor (xả thức ăn), LED và Button (hiển thị và thao tác).
- *Lớp mạng (Network Layer):* ESP32 làm thiết bị trung gian kết nối Wi-Fi, truyền dữ liệu qua giao thức MQTT đến máy chủ.

- *Lớp ứng dụng (Application Layer):* Ứng dụng web được xây dựng bằng ReactJS giúp người dùng điều khiển, theo dõi và quản lý thiết bị.

Giai đoạn 2 – Xây dựng và tích hợp phần cứng:
Trong hệ thống này, vi điều khiển ESP32 đóng vai trò là khối xử lý trung tâm. Nó đảm nhiệm đồng thời cả ba tác vụ: thu thập dữ liệu từ cảm biến (Load Cell), điều khiển cơ cấu chấp hành (Servo Motor) và thực hiện kết nối truyền thông IoT (WiFi/MQTT). Các thành phần linh kiện được kết nối trực tiếp với ESP32 và tích hợp thành một mô hình thực tế hoàn chỉnh.

Giai đoạn 3 – Phát triển phần mềm và giao tiếp IoT:
Hệ thống backend được xây dựng bằng NodeJS + Express, kết hợp cơ sở dữ liệu MongoDB để lưu trữ thông tin người dùng, lịch cho ăn và dữ liệu từ cảm biến. Giao tiếp giữa thiết bị và server sử dụng MQTT nhằm đảm bảo tốc độ, ổn định và khả năng truyền thời gian thực. Giao diện người dùng được phát triển bằng ReactJS, cung cấp các chức năng điều khiển thủ công, thiết lập lịch, theo dõi lịch sử cho ăn và cảnh báo.

Giai đoạn 4 – Mở rộng và tích hợp AI:
Hệ thống dự kiến phát triển thêm các tính năng AI và điều khiển bằng giọng nói thông qua Web service offline, giúp người dùng ra lệnh tự nhiên như “Cho mèo ăn” hoặc “Kiểm tra lượng thức ăn”. Ngoài ra, nhóm hướng đến việc ứng dụng các mô hình học máy đơn giản để phân tích thói quen ăn uống của thú cưng, từ đó đưa ra gợi ý khẩu phần hợp lý.

Định hướng phát triển:

- Hoàn thiện mô hình thử nghiệm, kiểm tra độ ổn định truyền dữ liệu giữa thiết bị và server.
- Tối ưu giao diện người dùng (UI/UX) trên web.
- Mở rộng hệ thống nhiều thiết bị, cho phép người dùng quản lý nhiều thú cưng cùng lúc.

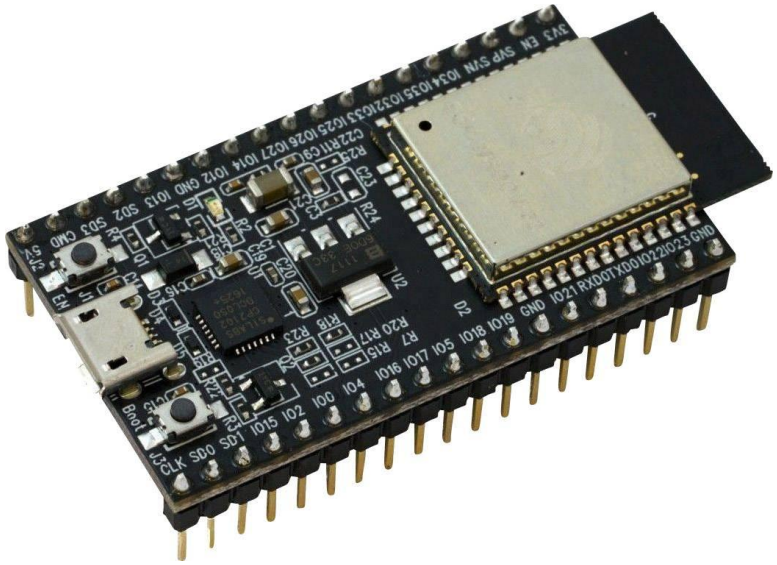
- Nghiên cứu bổ sung AI cho chuyển đổi giọng nói nhằm điều khiển hệ thống từ xa.

Với phương hướng trên, nhóm hướng tới việc xây dựng một hệ thống IoT thông minh, có tính ứng dụng cao, ổn định và có thể mở rộng, góp phần đưa công nghệ IoT vào đời sống hằng ngày theo hướng tiện ích và tự động hóa.

II. Các công nghệ, lý thuyết áp dụng cho dự án

A. Phần cứng

1. ESP32



- ESP32 là vi điều khiển tích hợp Wi-Fi và Bluetooth, được phát triển bởi Espressif Systems. Khác với Arduino Uno chỉ xử lý tín hiệu ngoại vi, ESP32 có khả năng kết nối mạng, giao tiếp thời gian thực và xử lý song song, nhờ bộ vi xử lý dual-core 32-bit Tensilica LX6, tốc độ lên tới 240MHz, và bộ nhớ RAM 520KB.

- Thành phần:

+ Wi-Fi Module (2.4GHz): Cho phép ESP32 kết nối Internet và giao tiếp với server MQTT.

+ Bluetooth BLE: Hỗ trợ giao tiếp không dây tầm ngắn (mở rộng kết nối với mobile).

+ GPIO (General Purpose I/O): Dùng để kết nối với cảm biến, servo hoặc Arduino Uno.

+ ADC/DAC: Đọc và xuất tín hiệu tương tự (analog).

+ UART, SPI, I2C: Giao tiếp nối tiếp với các thiết bị ngoại vi.

+ Flash Memory: Lưu trữ chương trình và dữ liệu cấu hình.

+ Nguồn 3.3V & Reset Button: Duy trì hoạt động ổn định của bo mạch.

- Trong hệ thống IoT cho thú cưng ăn tự động, ESP32 là cầu nối IoT (Internet Gateway) trong toàn bộ hệ thống, giúp đồng bộ dữ liệu và điều khiển giữa ứng dụng người dùng, backend, và phần cứng (Arduino Uno, Load Cell, Motor).

2. Load Cell Sensor

- Load Cell Sensor là cảm biến đo lực hoặc khối lượng, hoạt động dựa trên nguyên lý biến dạng điện trở (strain gauge): khi có lực tác động, cảm biến bị biến dạng rất nhỏ, dẫn đến thay đổi điện trở – sự thay đổi này được chuyển thành tín hiệu điện áp tỉ lệ với khối lượng vật thể.

- Thành phần:

+ Thanh cảm biến Load Cell (Strain Gauge): Thành phần cơ khí có nhiệm vụ phát hiện biến dạng vật lý do khối lượng tác động.

+ Module HX711: Bộ khuếch đại và chuyển đổi tín hiệu analog (biên độ rất nhỏ, tính bằng mV) từ Load Cell sang tín hiệu số (digital) chuẩn 24-bit để gửi trực tiếp về ESP32 xử lý.

+ Khung bàn cân (Mounting Frame): Giúp cố định cảm biến, đảm bảo lực truyền tải trọng tập trung chính xác vào thanh Load Cell để có kết quả đo ổn định.

- Trong hệ thống IoT cho thú cưng ăn tự động, Load Cell Sensor được sử dụng để đo trọng lượng thức ăn trong khay, giúp hệ thống theo dõi chính xác lượng thức ăn đã cấp và còn lại, phục vụ tính năng food tracking.

3. Servo

- Servo Motor là loại động cơ có thể điều khiển chính xác vị trí góc quay, hoạt động dựa trên nguyên lý phản hồi vòng kín và tín hiệu điều chế độ rộng xung (PWM): vi điều khiển gửi một chuỗi xung, độ rộng của xung này sẽ quy định góc quay cụ thể (thường từ 0° đến 180°), mạch điều khiển bên trong sẽ liên tục so sánh vị trí thực tế với vị trí mong muốn để tự động điều chỉnh.

- Thành phần:

+ Động cơ DC: Thành phần tạo ra chuyển động quay cơ bản với tốc độ cao.

+ Hệ thống bánh răng: Có nhiệm vụ giảm tốc độ vòng quay của động cơ DC để tăng mô-men xoắn, giúp servo có lực kéo đủ mạnh để đóng/ mở các cơ cấu cơ khí chịu tải.

+ Mạch điều khiển & Biến trở: Biến trở đóng vai trò cảm biến vị trí trục quay, gửi tín hiệu phản hồi về mạch điều khiển để đảm bảo động cơ quay đúng góc độ chính xác được yêu cầu bởi ESP32.

- Trong hệ thống IoT cho thú cưng ăn tự động, Servo Motor đóng vai trò là cơ cấu chấp hành điều khiển cửa xả thức ăn. ESP32 gửi lệnh điều khiển servo quay một góc xác định (ví dụ: mở 90°) để thức ăn rơi xuống khay, sau đó quay ngược lại để đóng kín, giúp hệ thống kiểm soát lượng thức ăn được cấp trong mỗi bữa.

Ngoài ba thành phần chính trên, hệ thống IoT còn sử dụng một số linh kiện hỗ trợ giúp hiển thị, điều khiển và đảm bảo hoạt động ổn định của mô hình:

+ LED: Hiển thị trạng thái hoạt động của hệ thống (đang cho ăn, chờ lệnh, lỗi kết nối…).

+ Button: Cho phép người dùng thao tác thủ công, ví dụ nhấn để cho ăn ngay tại chỗ.

+ Điện trở 2 x 220Ω và 1 x 100kΩ: Dùng để hạn dòng cho LED, nút nhấn và ổn định tín hiệu đọc từ cảm biến.

B. Phần mềm

1. NodeJS

- Node.js là một nền tảng chạy mã JavaScript phía máy chủ (server-side), được xây dựng trên Google Chrome V8 Engine. Nhờ đó, Node.js cho phép lập trình viên sử dụng cùng một ngôn ngữ (JavaScript) cho cả frontend và backend, giúp phát triển hệ thống nhanh và đồng nhất hơn.
- Express.js là một framework nhẹ của Node.js, hỗ trợ xây dựng API và dịch vụ web một cách đơn giản, linh hoạt và hiệu quả. Express cung cấp các chức năng như định

tuyến (routing), xử lý yêu cầu HTTP, và quản lý middleware để phát triển ứng dụng web RESTful.

- Trong hệ thống IoT cho thú cưng ăn tự động, Node.js (Express) đảm nhiệm vai trò là backend chính, giúp:

+ Nhận và xử lý dữ liệu từ thiết bị IoT (ESP32, cảm biến trọng lượng) gửi lên qua giao thức HTTP hoặc MQTT.

+ Quản lý và lưu trữ dữ liệu vào cơ sở dữ liệu trung tâm (ví dụ: MongoDB hoặc PostgreSQL).

+ Cung cấp API RESTful cho ứng dụng web truy xuất thông tin (như lượng thức ăn, lịch cho ăn, trạng thái thiết bị).

+ Thực hiện xử lý logic nghiệp vụ, như tính toán lượng thức ăn cần thiết, kiểm tra lịch cho ăn, và thông báo trạng thái cho người dùng.

- Nhờ sử dụng Node.js (Express), hệ thống đạt được tốc độ xử lý nhanh, khả năng mở rộng tốt, và dễ dàng tích hợp với các thiết bị IoT khác trong tương lai.

2. MongoDB

- MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL lưu trữ dữ liệu dưới dạng document (tài liệu JSON) thay vì bảng như trong SQL truyền thống. Cấu trúc này giúp MongoDB linh hoạt, dễ mở rộng và phù hợp cho các ứng dụng IoT, web và mobile có dữ liệu thay đổi liên tục.

- Trong hệ thống IoT cho thú cưng ăn tự động, MongoDB được sử dụng để lưu trữ và quản lý dữ liệu của toàn bộ hệ thống, bao gồm:

+ Thông tin người dùng và cấu hình thiết bị.

+ Lịch trình cho ăn tự động do người dùng cài đặt.

+ Lịch sử lượng thức ăn và hoạt động của thú cưng, được gửi từ cảm biến Load Cell.

- Nhờ đặc điểm linh hoạt và dễ mở rộng, MongoDB giúp hệ thống có thể xử lý dữ liệu thời gian thực từ nhiều thiết bị cùng lúc, đảm bảo tốc độ truy cập nhanh và tính ổn định cao cho backend Node.js.

3. MQTT

- MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông nhẹ được thiết kế đặc biệt cho các ứng dụng IoT. Nó hoạt động theo mô hình publish/subscribe,

cho phép các thiết bị gửi và nhận dữ liệu thông qua một máy chủ trung gian gọi là broker. MQTT có ưu điểm tiết kiệm băng thông, độ trễ thấp và ổn định cao, phù hợp cho hệ thống cần truyền dữ liệu liên tục giữa nhiều thiết bị.

- Trong hệ thống IoT cho thú cưng ăn tự động, MQTT đóng vai trò là kênh giao tiếp thời gian thực giữa thiết bị phần cứng (ESP32) và backend (Node.js):

+ Khi người dùng gửi lệnh cho ăn thủ công (manual feeding) từ ứng dụng, lệnh được publish lên topic MQTT, và ESP32 sẽ nhận (subscribe) để kích hoạt motor xả thức ăn.

+ Khi cảm biến Load Cell đo được lượng thức ăn, dữ liệu sẽ được gửi ngược lại qua MQTT để Node.js lưu vào MongoDB và hiển thị trên dashboard.

- MQTT giúp hệ thống đạt được phản hồi tức thời, hoạt động ổn định ngay cả khi mạng yếu — là yếu tố quan trọng trong các ứng dụng IoT thực tế.

4. ReactJS

- ReactJS là thư viện JavaScript mã nguồn mở do Facebook phát triển, dùng để xây dựng giao diện người dùng (UI) theo hướng component-based. React cho phép tạo các trang web động, mượt và tối ưu hiệu năng nhờ cơ chế Virtual DOM, giúp cập nhật dữ liệu nhanh mà không cần tải lại toàn bộ trang.

- Trong hệ thống IoT cho thú cưng ăn tự động, ReactJS được sử dụng để phát triển giao diện web dashboard, nơi người dùng có thể:

+ Điều khiển hệ thống cho ăn thủ công (manual feeding) chỉ bằng một nút bấm.

+ Thiết lập lịch cho ăn tự động và theo dõi trạng thái kết nối thiết bị.

+ Xem lịch sử lượng thức ăn mà thú cưng đã tiêu thụ thông qua biểu đồ trực quan.

- Frontend ReactJS kết nối với backend Node.js qua API RESTful và MQTT, giúp hiển thị thông tin theo thời gian thực và mang lại trải nghiệm thân thiện, hiện đại cho người dùng.

5. WebSpeech để chuyển đổi giọng nói trong hệ thống

Quy trình xử lý lệnh bằng giọng nói diễn ra qua các bước sau:

- Thu thập giọng nói (Frontend): Người dùng kích hoạt chức năng bằng cách nhấn nút ghi âm trên giao diện website (Frontend). Trình duyệt sẽ thu âm câu lệnh (ví dụ: “Cho mèo ăn” hoặc “Kiểm tra lượng thức ăn”).

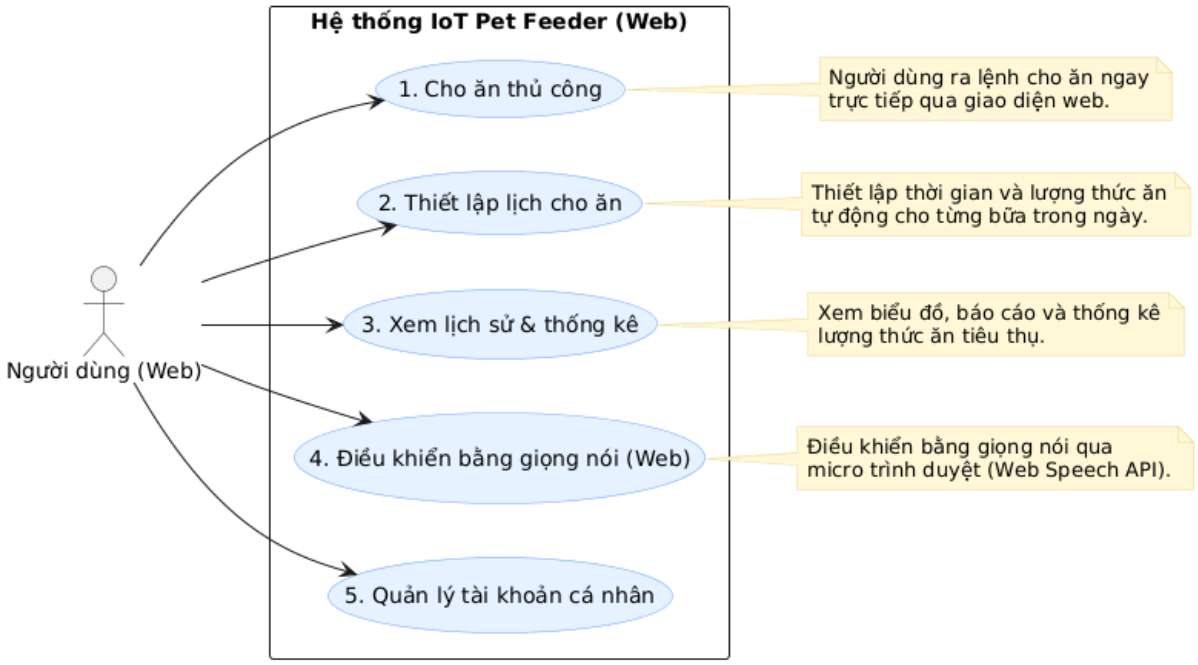
- Chuyển đổi Text (Frontend): Sau khi nhận diện, tín hiệu âm thanh được chuyển thành văn bản (text). Kết quả này sẽ được chuẩn hóa (ví dụ: chuyển sang chữ thường) và so khớp với danh sách từ khóa hợp lệ.
- Gửi lệnh lên Backend: Văn bản lệnh đã nhận diện (dạng String chứa câu lệnh) sẽ được Frontend gửi lên Backend Server thông qua API POST.

III. Các tính năng triển khai

1. Yêu cầu chức năng

- **Chức năng cho ăn thủ công:**
  - Người dùng có thể bấm nút "Feed Now" trên ứng dụng để cấp thức ăn ngay lập tức.
  - Lệnh điều khiển được gửi từ ứng dụng → máy chủ → thiết bị feeder qua giao thức MQTT.
  - Thiết bị thực hiện quay motor phân phối thức ăn và gửi phản hồi trạng thái (thành công/ thất bại).
- **Chức năng cho ăn theo lịch:**
  - Người dùng thiết lập thời gian, khối lượng thức ăn cho từng lần cho ăn.
  - Lịch được lưu trữ trên cơ sở dữ liệu và gửi xuống thiết bị.
  - Tới thời điểm định sẵn, thiết bị tự động thực hiện cho ăn và ghi lại kết quả.
- **Chức năng theo dõi lượng thức ăn:**
  - Cảm biến trọng lượng đo khối lượng thức ăn trước và sau khi phân phối.
  - Dữ liệu được gửi lên máy chủ, lưu vào cơ sở dữ liệu, và hiển thị dưới dạng biểu đồ thống kê.
  - Cho phép người dùng theo dõi lịch sử tiêu thụ thức ăn của vật nuôi theo ngày/tuần/tháng.
- **Chức năng điều khiển bằng giọng nói:**
  - Cho phép người dùng sử dụng các câu lệnh giọng nói như "Cho mèo ăn" hoặc "Kiểm tra lượng thức ăn".

Sơ đồ UC:



2. Yêu cầu phi chức năng

- **Hiệu năng:**
    - Thời gian phản hồi trung bình ≤ 5 giây cho lệnh điều khiển trực tiếp.
    - Dữ liệu cảm biến cập nhật định kỳ mỗi 10 – 15 giây.
- **Độ tin cậy:**
    - Thiết bị hoạt động ổn định khi được cấp nguồn.
    - Khi mất kết nối mạng, hệ thống vẫn tự thực hiện các lịch cho ăn đã lưu cục bộ.
    - Dữ liệu được lưu tạm trên thiết bị và đồng bộ lại sau khi có kết nối.
- **Bảo mật:**
    - Sử dụng kết nối MQTT qua kênh bảo mật TLS và giao thức HTTPS để bảo vệ dữ liệu truyền tải.
    - Thiết bị được định danh duy nhất bằng Device ID.
    - Tài khoản người dùng bảo vệ bằng mật khẩu mã hóa và cơ chế JWT.
- **Khả năng mở rộng:** Mã nguồn được tổ chức tách biệt (ESP32, Frontend, Backend, AI Service), dễ dàng bảo trì hoặc tích hợp thêm tính năng mới.
- **Tiết kiệm năng lượng:**
    - Thiết bị vào chế độ “sleep” khi không hoạt động để giảm tiêu thụ điện.

    - Chỉ kích hoạt cảm biến và motor khi có sự kiện thực tế.
- **Trải nghiệm người dùng:**
    - Giao diện ứng dụng trực quan, hiển thị biểu đồ, lịch sử cho ăn, thông báo lỗi rõ ràng.
    - Có thể cài đặt và cấu hình dễ dàng qua website.

3. Yêu cầu giao tiếp

- **Giao tiếp giữa thiết bị và máy chủ (IoT – Server):**
    - Sử dụng giao thức MQTT để truyền dữ liệu nhẹ, nhanh và ổn định.
    - Vi điều khiển ESP32 thiết lập kết nối đến MQTT Broker thông qua mạng Wi-Fi, đảm nhiệm vai trò truyền nhận dữ liệu hai chiều: nhận tín hiệu điều khiển hoạt động cho ăn và gửi phản hồi trạng thái hệ thống.
    - Dữ liệu truyền dạng JSON, gồm các trường: device_id, timestamp, type, value, status.
- **Giao tiếp giữa máy chủ và giao diện người dùng (Server – Frontend):**
    - Giao tiếp sử dụng RESTful API được xây dựng bằng NodeJS + ExpressJS.
    - Phía client ReactJS gửi các yêu cầu HTTP (GET, POST, PUT, DELETE) đến API để:
        - Đăng nhập, đăng ký, xác thực người dùng.
        - Quản lý thiết bị (thêm, xóa, cấu hình).
        - Cập nhật lịch cho ăn, truy xuất dữ liệu thống kê.
    - Dữ liệu phản hồi ở định dạng JSON để dễ xử lý phía frontend.
    - Tất cả các endpoint API đều yêu cầu xác thực JWT (JSON Web Token).
- **Kết nối cơ sở dữ liệu (Server – Database):**
    - Sử dụng MongoDB làm hệ quản trị cơ sở dữ liệu, kết nối thông qua thư viện Mongoose.
    - Dữ liệu được lưu trữ trong các collection:
        - users: thông tin người dùng.
        - schedules: lịch cho ăn định kỳ.
        - feed_logs: nhật ký cho ăn, lượng thức ăn, thời gian.
    - Các truy vấn và cập nhật thực hiện bất đồng bộ (async/await), đảm bảo hiệu suất.

- **Yêu cầu bảo mật trong giao tiếp:**
  - Toàn bộ luồng giao tiếp sử dụng HTTPS/ WSS (TLS) để mã hóa dữ liệu.
  - Mỗi thiết bị có device token riêng để xác thực với MQTT Broker.
  - Người dùng đăng nhập qua giao diện ReactJS, được cấp JWT cho các yêu cầu tiếp theo.

4. Yêu cầu hoạt động

- **Điều kiện mạng và điện:**
  - Kết nối mạng Wi-Fi 2.4 GHz, tốc độ tải lên tối thiểu 1 Mbps.
  - Thiết bị tự động tái kết nối khi mất mạng và lưu tạm dữ liệu để gửi lại sau.
  - Nguồn cấp chính: 5V DC, có thể qua USB Type-C hoặc adapter ngoài.
- **Điều kiện hoạt động phần mềm:**
  - Frontend: chạy trên trình duyệt hiện đại (Chrome, Edge, Firefox, Safari) với kết nối internet ổn định.
  - Backend: triển khai trên server NodeJS ≥ v18, hỗ trợ HTTPS và MQTT service.
  - Database: MongoDB ≥ v6.0, lưu trữ dữ liệu thời gian thực và lịch sử cho ăn.

5. Yêu cầu về thử nghiệm

- **Thử nghiệm phần mềm:**
  - Frontend (ReactJS):
    - Kiểm tra hiển thị biểu đồ, lịch cho ăn, trạng thái thiết bị.
    - Đảm bảo giao diện phản hồi nhanh, không treo khi mất mạng.
  - Backend (NodeJS + MongoDB):
    - Kiểm tra API REST hoạt động đúng (CRUD người dùng, lịch cho ăn, thống kê).
    - Kiểm tra hiệu năng khi có nhiều người dùng truy cập đồng thời.
    - Đảm bảo dữ liệu được lưu trữ chính xác và không trùng lặp.
- **Thử nghiệm giao tiếp IoT:**
  - Kiểm tra MQTT:
    - Đảm bảo thiết bị nhận lệnh “Feed Now” trong vòng ≤ 2 giây.

    - Thử mất mạng giữa chừng và xác minh khả năng gửi lại dữ liệu khi kết nối phục hồi.
- Kiểm tra an toàn truyền thông:
    - Thử kết nối sai token, token hết hạn, và xử lý xác thực.
    - Đảm bảo MQTT Broker từ chối thiết bị không hợp lệ.

IV. Phân tích thiết kế hệ thống

1. Các chức năng chính

- **Giám sát và trực quan hóa dữ liệu:**

Đây là giao diện trung tâm giúp người dùng nắm bắt tình trạng hoạt động của thiết bị một cách trực quan.

+ Giám sát thời gian thực: Hệ thống hiển thị liên tục các thông số nhận được từ cảm biến thông qua giao thức MQTT, bao gồm trọng lượng thức ăn hiện tại trong khay và thời gian của lần cho ăn gần nhất.

+ Biểu đồ thống kê: Tích hợp thư viện Recharts để vẽ biểu đồ, minh họa xu hướng tiêu thụ thức ăn của thú cưng theo từng ngày, hỗ trợ người dùng theo dõi sức khỏe vật nuôi.

- **Điều khiển chế độ cho ăn thủ công và sử dụng giọng nói:**

Cung cấp các phương thức tương tác linh hoạt để vận hành thiết bị theo nhu cầu tức thời.

+ Điều khiển cho ăn thủ công: Người dùng kích hoạt lệnh xả thức ăn ngay lập tức thông qua nút nhấn trên giao diện Web. Hệ thống chỉ xác nhận hoàn thành khi nhận được tín hiệu phản hồi từ thiết bị phần cứng.

+ Điều khiển bằng giọng nói: Hệ thống sử dụng Web Speech API (cụ thể là giao diện SpeechRecognition hoặc window.webkitSpeechRecognition) , một giải pháp nhận diện giọng nói nguyên bản được tích hợp sẵn trên các trình duyệt hiện đại (như Chrome/Edge). Quy trình nhận diện giọng nói được thực hiện trực tiếp phía Client (trên

trình duyệt) theo mô hình Online Recognition (Stream data trực tiếp qua Dịch vụ nhận diện giọng nói của Google) để chuyển đổi âm thanh thành văn bản (text), giúp giảm tải tài nguyên cho Server.

- **Điều khiển chế độ cho ăn theo lịch:**

Cho phép tự động hóa quy trình chăm sóc thông qua việc thiết lập các tác vụ định kỳ.

+ Thiết lập lịch trình: Người dùng có thể tạo mới các lịch cho ăn với đầy đủ các tham số: thời gian kích hoạt (giờ/ phút), định lượng thức ăn (gram) và chu kỳ lặp lại (các ngày trong tuần).

+ Quản lý danh sách lịch cho ăn: Hỗ trợ xem danh sách, chỉnh sửa thông số hoặc xóa các lịch trình cũ.

+ Trạng thái kích hoạt: Cung cấp chức năng bật/tắt (Toggle) nhanh cho từng lịch trình, giúp người dùng linh hoạt ngưng tạm thời một mốc thời gian mà không cần xóa dữ liệu.

- **Xác thực và bảo mật:**

Đảm bảo tính riêng tư dữ liệu và kiểm soát quyền truy cập hệ thống.

+ Cơ chế xác thực: Sử dụng phương pháp Token-based Authentication. Người dùng cần đăng ký và đăng nhập để nhận mã truy cập (Access Token).

+ Quản lý phiên làm việc: Hệ thống tự động kiểm tra tính hợp lệ của Token trong suốt quá trình sử dụng và tự động điều hướng (Redirect) về trang đăng nhập nếu phiên làm việc hết hạn.

- **Giao tiếp thời gian thực:**

+ Giao thức MQTT: Hệ thống sử dụng MQTT Broker (HiveMQ Cloud) làm trung gian truyền tải dữ liệu với độ trễ thấp.

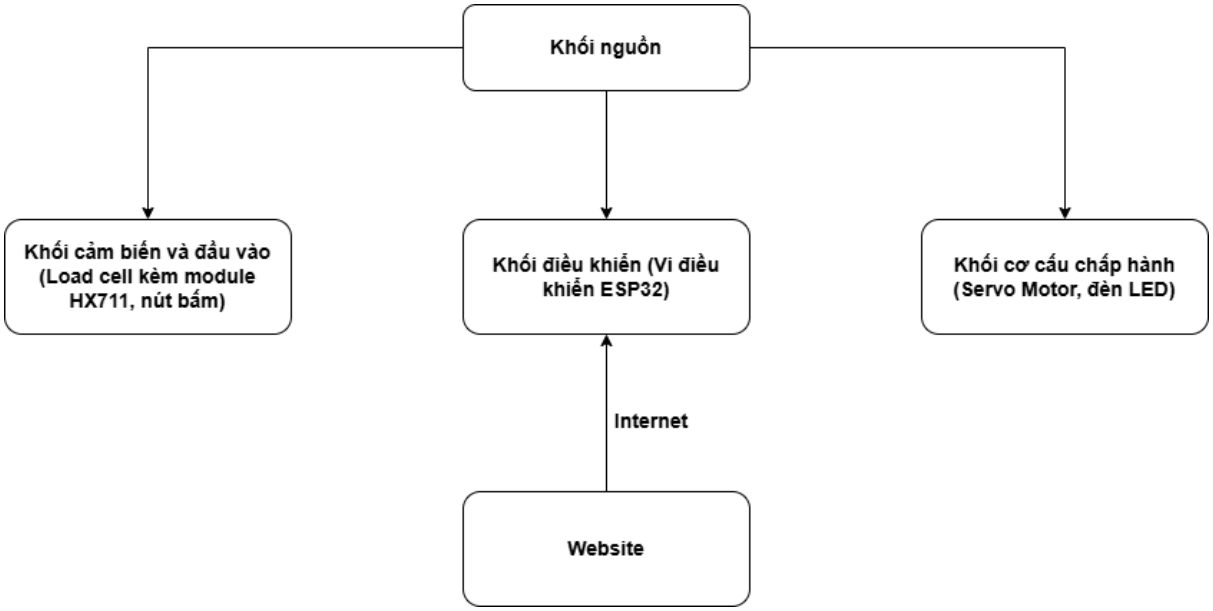
- Subscribe (Nhận): Ứng dụng Web lắng nghe các topic telemetry (dữ liệu cảm biến) và ack (xác nhận lệnh).

- Publish (Gửi): Ứng dụng Web gửi các lệnh điều khiển tới thiết bị.

+ Cơ chế Auto-reconnect: Tự động phát hiện sự cố mất kết nối mạng và thực hiện tái kết nối để đảm bảo tính liên tục của dữ liệu.

2. Phần cứng

A. Sơ đồ khối hệ thống



- Khối nguồn: Cung cấp điện áp DC ổn định (thường là 5V) để nuôi toàn bộ mạch hệ thống, bao gồm vi điều khiển, động cơ và các cảm biến.

- Khối điều khiển: Sử dụng vi điều khiển ESP32. Đây là "bộ não" của hệ thống, có nhiệm vụ:

+ Thu thập dữ liệu trọng lượng từ khối cảm biến.

+ Kết nối Wi-Fi và giao tiếp với Server qua giao thức MQTT để nhận lệnh điều khiển hoặc gửi báo cáo.

+ Xử lý logic và xuất tín hiệu điều khiển (PWM/ Digital) tới khối chấp hành.

- Khối cảm biến & Đầu vào:

+ Cảm biến Load Cell (kèm module HX711): Đo lường trọng lượng thức ăn thực tế trong khay và chuyển đổi thành tín hiệu số gửi về vi xử lý.

+ Nút nhấn (Button): Tiếp nhận thao tác vật lý nhấn để cho ăn ngay lập tức.

- Khối cơ cấu chấp hành:

+ Servo Motor: Nhận tín hiệu điều khiển (PWM) từ vi xử lý để thực hiện quay góc mở/đóng cửa xả thức ăn.

+ Đèn LED: Nhận tín hiệu logic để hiển thị trạng thái hoạt động của hệ thống khi kích hoạt cho ăn thủ công.

B. Thiết kế phần cứng

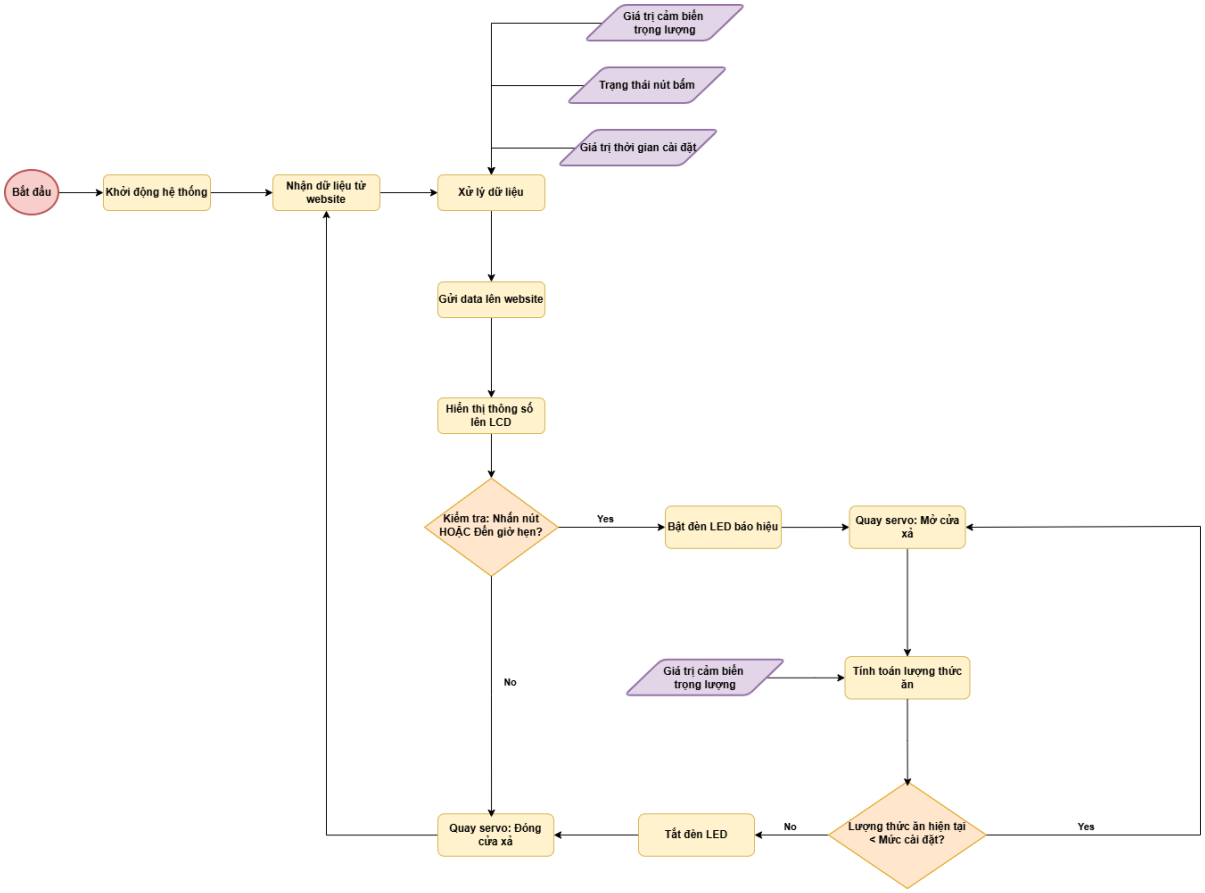
*** Sơ đồ kết nối chân linh kiện trong hệ thống:**

ESP32 Pin	Servo Motor	HX711 Module	I2C Interface (LCD)	Button	LED
GND	GND	GND	GND	PIN 1	CATHODE
VIN (5V)	VCC		VCC		
3.3V		VCC		PIN 3	
D13	PWM				
D21			SDA		
D22			SCL		
RX2		DT			
D4		SCK			
D27				PIN 2	
D26					ANODE

HX711 Module	Load Cell Sensor
E+	Red
E-	Black
A-	White
A+	Green

C. Thiết kế phần mềm

*** Lưu đồ thuật toán:**



*** Lập trình cho ESP32:**

Cài đặt môi trường lập trình cho ESP32 Dev Module trên Arduino IDE

- Bước 1: Khởi động phần mềm Arduino IDE để bắt đầu thiết lập.
- Bước 2: Trên thanh menu, chọn File → Preferences. Tại ô Additional Boards Manager URLs, dán đường dẫn sau vào (gói cài đặt lõi cho dòng ESP32): https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json. Sau đó nhấn OK để lưu lại.
- Bước 3: Vào Tools → Board → Boards Manager. Tại thanh tìm kiếm, nhập từ khóa "esp32". Tìm gói có tên "esp32 by Espressif Systems" và nhấn nút Install để tiến hành cài đặt.
- Bước 4: Kết nối board ESP32 vào máy tính.
  - Vào Tools → Board → esp32 → Chọn ESP32 Dev Module. Sử dụng cấu hình board ESP32 Dev Module giúp đảm bảo tính tương thích rộng rãi nhất với các biến thể phần cứng khác nhau của dòng ESP32-WROOM-32.
  - Vào Tools → Port và chọn cổng COM tương ứng (ví dụ: COM3, COM4...).

- Bước 5: Trong trường hợp máy tính không nhận cổng COM, hãy tải và cài đặt Driver cho chip giao tiếp USB-to-UART CP2102 (thường dùng trên Dev Module) tại liên kết: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads

3. Frontend

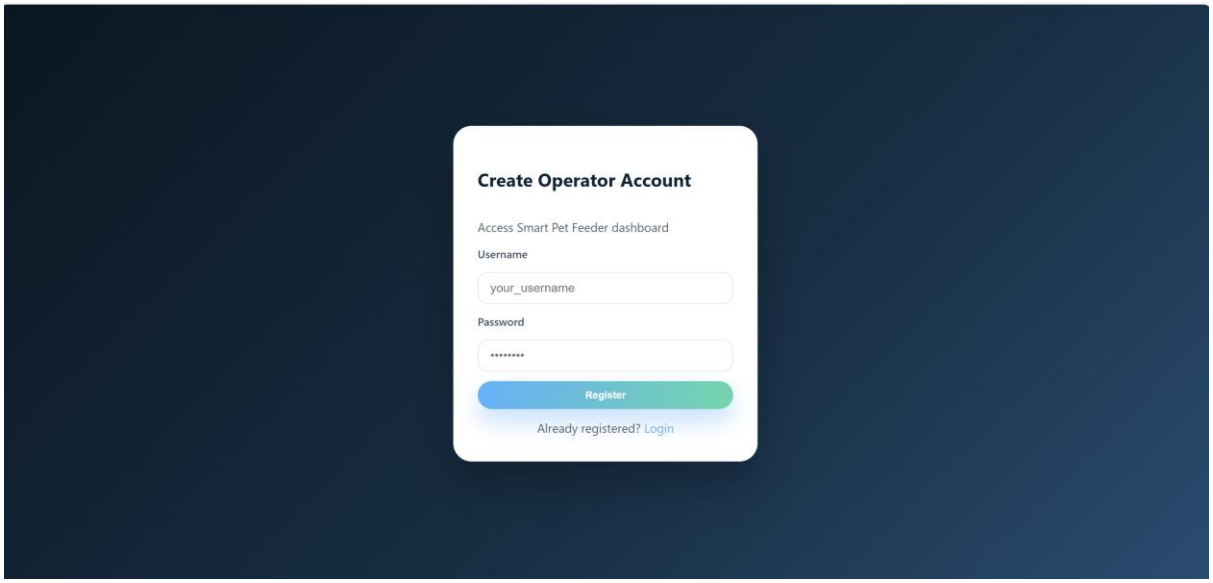
A. Cấu trúc mã nguồn

```
frontend/
├── src/
│   ├── components/          # Các thành phần UI tái sử dụng (Reusable Components)
│   │   ├── AppLayout.jsx    # Bố cục chính (Layout) bao bao quanh ứng dụng
│   │   ├── Sidebar.jsx      # Thanh điều hướng bên trái
│   │   ├── TopBar.jsx       # Thanh tiêu đề phía trên
│   │   ├── StatCard.jsx     # Card hiển thị số liệu thống kê
│   │   └── StatusBadge.jsx  # Nhãn hiển thị trạng thái (Online/Offline)
│   ├── pages/               # Các trang màn hình chính (Views)
│   │   ├── Dashboard.jsx    # Trang tổng quan giám sát hệ thống
│   │   ├── Login.jsx        # Trang đăng nhập
│   │   ├── Register.jsx     # Trang đăng ký tài khoản
│   │   ├── ManualFeed.jsx   # Trang điều khiển cho ăn thủ công
│   │   └── Schedule.jsx     # Trang cài đặt lịch trình
│   ├── services/            # Các service xử lý logic nghiệp vụ & kết nối
│   │   ├── api.js           # Client xử lý HTTP Request (RESTful API)
│   │   └── mqtt.js          # Client xử lý kết nối MQTT (Real-time)
│   ├── hooks/               # Các Custom Hooks (Logic tái sử dụng)
│   ├── App.jsx              # Component gốc, cấu hình Routing
│   ├── main.jsx             # Điểm khởi chạy ứng dụng (Entry point)
│   └── styles.css           # Định dạng CSS toàn cục
├── public/                  # Tài nguyên tĩnh (Images, Icons, Favicon)
├── package.json             # Khai báo thư viện phụ thuộc (Dependencies)
└── vite.config.js           # Cấu hình công cụ build Vite
```

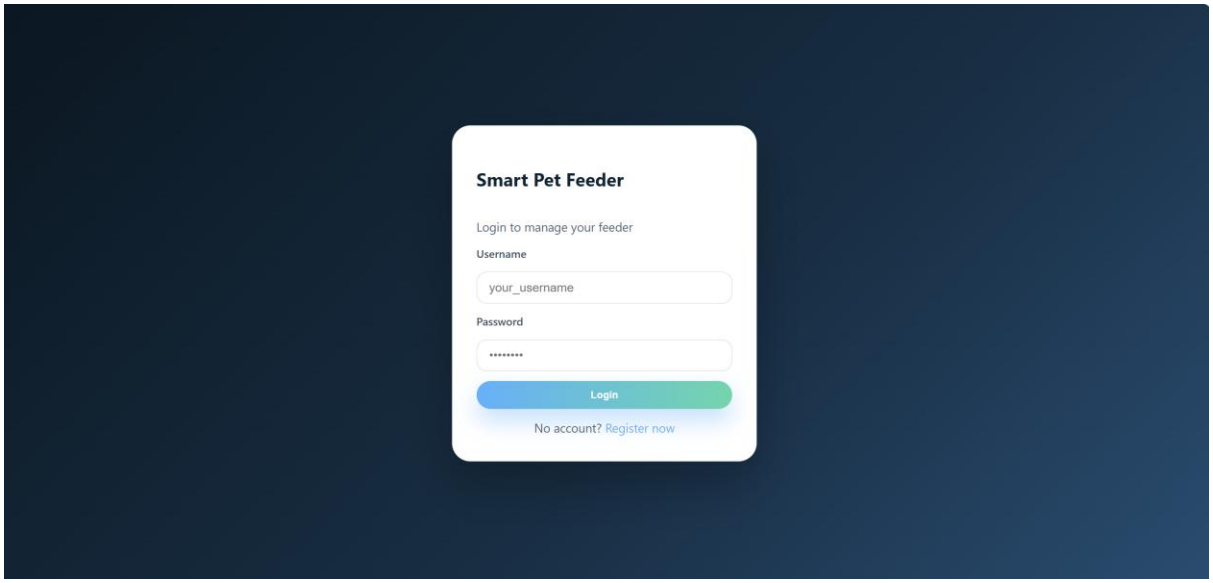
Hệ thống Frontend được thiết kế theo mô hình Component-Based Architecture, phân tách thành 4 lớp logic chính:

- Lớp Giao diện (Presentation Layer): Bao gồm các thành phần React (pages, components) chịu trách nhiệm hiển thị dữ liệu trực quan và xử lý tương tác người dùng.
- Lớp Dịch vụ (Service Layer): Đóng gói logic giao tiếp với Backend (qua REST API) và thiết bị IoT (qua MQTT) thông qua các module độc lập (api.js, mqtt.js), tách biệt hoàn toàn với logic hiển thị.
- Lớp Quản lý Trạng thái (State Management): Sử dụng cơ chế React Hooks (useState, useEffect) để kiểm soát luồng dữ liệu và đồng bộ trạng thái ứng dụng theo thời gian thực.
- Lớp Định tuyến (Routing Layer): Sử dụng React Router để điều hướng linh hoạt giữa các màn hình theo mô hình Single Page Application (SPA), giúp trải nghiệm người dùng mượt mà không cần tải lại trang.

B. Giao diện ứng dụng

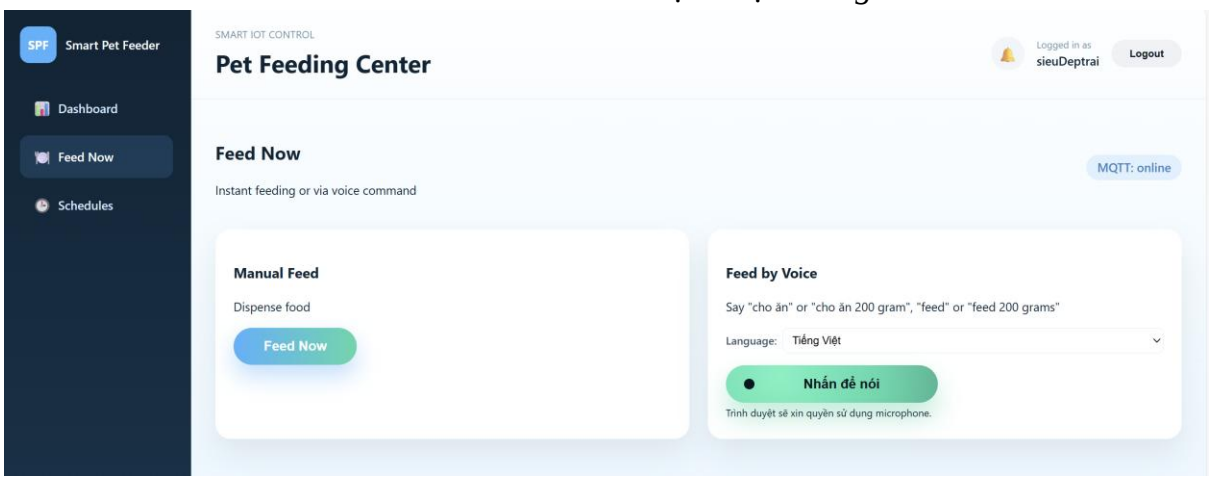


Hình 4.1. Màn hình đăng ký người dùng

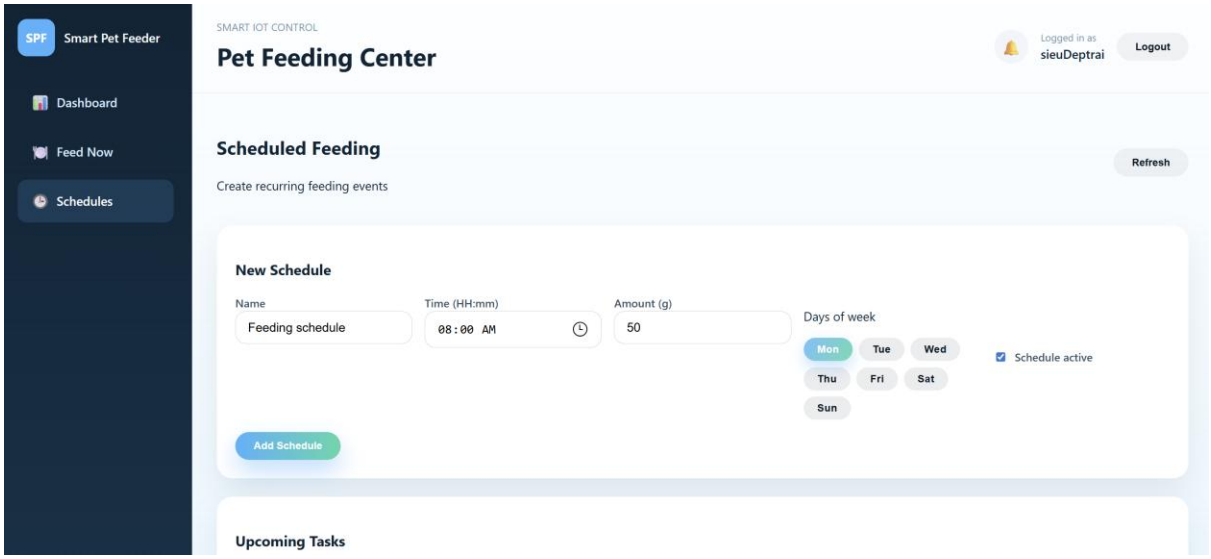


Hình 4.2. Màn hình đăng nhập vào hệ thống

Hình 4.3. Màn hình hiển thị số liệu thống kê



Hình 4.4. Màn hình điều khiển chế độ cho ăn thủ công và sử dụng giọng nói



Hình 4.5. Màn hình điều khiển chế độ cho ăn theo lịch

4. Backend

A. Cấu trúc mã nguồn

```
backend/
├── models/              # Định nghĩa cấu trúc dữ liệu (Data Schemas)
│   ├── User.js          # Schema thông tin người dùng
│   ├── Schedule.js      # Schema lưu trữ lịch trình cho ăn
│   └── FeedLog.js       # Schema nhật ký hoạt động cho ăn
│
├── services/            # Lớp xử lý logic nghiệp vụ cốt lõi (Business Logic)
│   ├── mqttService.js   # Quản lý kết nối và gửi/nhận tin nhắn MQTT
│   ├── feedService.js   # Logic xử lý quy trình cho ăn (tính toán, kiểm tra)
│   ├── scheduleService.js  # Logic CRUD cho lịch cho ăn
│   ├── schedulerService.js # Quản lý tác vụ định kỳ để kích hoạt cho ăn
│   └── voiceService.js  # Xử lý phân tích lệnh giọng nói
│
├── controllers/         # Lớp điều phối yêu cầu (Request Handlers)
│   ├── authController.js   # Xử lý đăng ký, đăng nhập
│   ├── scheduleController.js # Tiếp nhận request quản lý lịch trình
│   ├── voiceController.js  # Tiếp nhận file âm thanh từ Client
```

```
│   └── feedController.js    # Tiếp nhận lệnh cho ăn thủ công
│
├── routes/                 # Định tuyến API (API Endpoints)
│   ├── auth.js             # Routes: /api/auth
│   ├── schedule.js         # Routes: /api/schedule
│   └── feed.js             # Routes: /api/feed
│
├── middleware/             # Các hàm trung gian xử lý request
│   └── auth.js             # Middleware xác thực JWT Token bảo vệ routes
│
├── config/                 # Cấu hình hệ thống
│   └── database.js         # Thiết lập kết nối MongoDB
│
├── server.js               # Khởi tạo Server, kết nối DB & MQTT
├── package.json            # Khai báo thư viện (Express, Mongoose, MQTT...)
└── .env                    # Biến môi trường (DB URL, MQTT Credentials)
```

B. Một số cấu hình quan trọng

* MongoDB:

- Khởi tạo tài khoản và tạo Cluster**:** Truy cập trang web MongoDB Atlas tại *https://www.mongodb.com/atlas* và tiến hành đăng ký tài khoản. Sau khi đăng nhập, lựa chọn tạo mới một Shared Cluster.

→ Hệ thống sẽ tự động triển khai một cụm MongoDB trên nền tảng cloud.

- Tạo cơ sở dữ liệu và các collection**:** truy cập mục Database → Collections để khởi tạo cơ sở dữ liệu cho dự án. Ba collection chính được tạo tương ứng với các thành phần dữ liệu của hệ thống: users, schedules, feedlogs.

- Cấu hình quyền truy cập từ backend: Tại mục Security → Database Access, tiến hành tạo tài khoản người dùng phục vụ kết nối từ backend.

+ Chọn Add New Database User

+ Thiết lập thông tin:

- Username : root
- Password: 12345

- Cấu hình quyền truy cập IP để backend có thể kết nối đến MongoDB Atlas: truy cập Security → Network Access → Add IP Address → Chọn Allow access from anywhere (0.0.0.0/0)

- Lấy Connection String để sử dụng trong backend: Chọn Cluster → Connect → Chọn "Drivers" → Sao chép connection string và lưu vào file .env của folder backend: mongodb+srv://root:12345@cluster1.k28cwf8.mongodb.net/IOT_PetFeederDB?retryWrites=true&w=majority&appName=Cluster-1

* MQTT:

- Truy cập trang chủ: https://www.hivemq.com/mqtt-cloud-broker/ rồi tạo tài khoản và đăng nhập.

- Tạo một MQTT Cluster mới:

- Chọn loại Free Cluster.
- Chọn server region gần Việt Nam (ví dụ: Singapore).

→ Sau khi tạo thành công, hệ thống sẽ cung cấp:

- Broker URL dạng mqtts://<cluster_id>.s1.eu.hivemq.cloud:8883
- Port 8883 (kết nối TLS/SSL)
- Username / Password để client kết nối.

- Kích hoạt tính năng:
  - TLS/SSL encryption
  - Client authentication bằng username – password
- Lưu lại các thông tin cấu hình để sử dụng trong file môi trường .env.

C. Danh sách API

*** POST /auth/register**

Tạo tài khoản mới với username và password.

- Parameter:
  - username — string (min 3 characters)
  - password — string (min 6 characters)
- Response:

```
{
  "user": {
    "id": String,
    "username": String,
    "lastOnline": Date
  }
}
```

*** POST api/auth/login**

Đăng nhập bằng xác thực thông tin tài khoản, trả về jwt token

- Parameter:
  - username — string
  - password — string
- Response:

```
{
  "token": String,
  "user": {
    "id": String,
    "username": String,
    "lastOnline": Date
  }
}
```

*** POST /api/feed/manual**

Tạo lệnh cho ăn thủ công với lượng thức ăn mặc định 200 gram.

- Parameter: Bearer token authorization – jwt token

- Response:

```
{
  "message": String,
  "feedLog": {
    "_id": String,
    "user": String,
    "feedType": "manual",
    "amount": Number,
    "targetAmount": Number,
    "status": String,
    "startTime": Date,
    "endTime": Date,
    "createdAt": Date,
    "updatedAt": Date
  }
}
```

* **POST api/feed/voice**

Tạo lệnh cho ăn với lượng thức ăn tùy chỉnh bằng giọng nói

- Parameters: Text – String (voice command sentence)
- Response:

```
{
  "message": String,
  "parsedAmount": Number,
  "feedLog": {
    "_id": String,
    "user": String,
    "feedType": "voice",
    "amount": Number,
    "targetAmount": Number,
    "status": String,
    "voiceCommand": String,
```

"startTime": Date,
"endTime": Date,
"createdAt": Date,
"updatedAt": Date
}
}

*** POST /api/feed/stats/weekly**

Trả về thông số lượng thức ăn xả và số lần cho ăn mỗi ngày trong 7 ngày qua

- Parameters: Bearer token authorization – jwt token
- Response:

{
"days": Number,
"data": [
{
"date": String,
"totalAmount": Number,
"feedCount": Number
}
]
}

* **POST /api/schedules**

Tạo lịch hẹn cho ăn mới

- Parameters:
  - Bearer token authorization – jwt token
  - name — string
  - time — string (HH:MM)
  - daysOfWeek — array of number
  - amount — number
- Response:

{
"schedule": {

```
    "_id": String,
    "user": String,
    "name": String,
    "time": String,
    "daysOfWeek": [Number],
    "amount": Number,
    "isActive": Boolean,
    "createdAt": Date,
    "updatedAt": Date
  }
}
```

*** GET /api/schedules**

Lấy danh sách tất cả lịch hẹn cho ăn

- Parameters: Bearer token authorization – jwt token
- Response:

```
{
  "schedules": [
    {
      "_id": String,
      "name": String,
      "time": String,
      "daysOfWeek": [Number],
      "amount": Number,
      "isActive": Boolean
    }
  ]
}
```

*** PUT /api/schedules/:id**

Cập nhật thông tin của 1 lịch hẹn cho ăn cụ thể

- Parameters:
  - Bearer token authorization – jwt token

- name — string
- time — string
- daysOfWeek — array
- amount — number

- Response:

```
{
  "schedule": {
    "_id": String,
    "user": String,
    "name": String,
    "time": String,
    "daysOfWeek": [Number],
    "amount": Number,
    "isActive": Boolean,
    "createdAt": Date,
    "updatedAt": Date
  }
}
```

*** DELETE /api/schedules/:id**

Xóa 1 lịch hẹn cho ăn cụ thể

- Parameters: Bearer token authorization – jwt token
- Response:

```
{
  "message": String
}
```

* **PATCH /api/schedules/:id/toggle**

Kích hoạt hoặc vô hiệu hóa 1 lịch hẹn cho ăn cụ thể

- Parameters: Bearer token authorization – jwt token
- Response:

```
{
  "schedule": {
```

```
    "_id": String,
    "user": String,
    "name": String,
    "time": String,
    "daysOfWeek": [Number],
    "amount": Number,
    "isActive": Boolean,
    "createdAt": Date,
    "updatedAt": Date
  }
}
```


5. Chức năng speech – to – text

A. Đặc điểm mô hình sử dụng

- Cơ sở công nghệ: Hệ thống sử dụng Web Speech API (cụ thể là SpeechRecognition interface). Đây là giải pháp nhận diện giọng nói nguyên bản được tích hợp sẵn trên các trình duyệt hiện đại, thực hiện xử lý trực tiếp phía Client giúp giảm tải tài nguyên cho Server.
- Đặc tính kỹ thuật: Hoạt động theo mô hình thời gian thực với kiến trúc hướng sự kiện cho phép chuyển đổi lời nói thành văn bản với độ trễ thấp và không yêu cầu cài đặt thêm bất kỳ thư viện phụ thuộc nào.
- Thiết lập cấu hình: Mô hình được tinh chỉnh tối ưu cho tác vụ ra lệnh ngắn
    - Ngôn ngữ: Thiết lập en-US để đảm bảo độ chính xác cao nhất cho các từ khóa kỹ thuật.
    - Chế độ hoạt động: continuous: false (tự động ngắt lắng nghe sau khi nhận diện xong một câu lệnh) và maxAlternatives: 1 (chỉ lấy kết quả có độ tin cậy cao nhất) để đơn giản hóa quá trình xử lý logic.
    - Phản hồi: interimResults: true cho phép hiển thị kết quả tạm thời ngay khi người dùng đang nói.

B. Cách thức hoạt động

Quy trình xử lý giọng nói trong hệ thống được vận hành qua 3 giai đoạn chính:

- Khởi tạo và Cấu hình: Hệ thống kiểm tra tính tương thích của trình duyệt và khởi tạo đối tượng SpeechRecognition với các tham số tối ưu cho ra lệnh ngắn: ngôn ngữ en-US (tiếng Anh Mỹ), chế độ continuous: false (tự động ngắt sau khi nhận lệnh) và interimResults: true (cho phép phản hồi thị giác tức thì).
- Vòng đời xử lý sự kiện: Hệ thống hoạt động dựa trên mô hình hướng sự kiện bất đồng bộ
    - onstart: Kích hoạt trạng thái "Listening" trên giao diện để người dùng biết hệ thống đang thu âm.
    - onresult: Chuyển đổi tín hiệu âm thanh thành văn bản, thực hiện chuẩn hóa chuỗi ký tự về dạng lowercase và so khớp với danh sách từ khóa hợp lệ (ví dụ: "feed now", "cho ăn")
    - onerror/ onend: Bắt các ngoại lệ (mất kết nối, lỗi microphone) và tự động đặt lại trạng thái sẵn sàng của giao diện khi phiên làm việc kết thúc.
- Tích hợp và Đồng bộ hệ thống: Ngay khi xác định lệnh hợp lệ, Frontend gọi API gửi tín hiệu điều khiển xuống Backend, đồng thời đăng ký lắng nghe (Subscribe) kênh phản hồi từ MQTT để xác nhận trạng thái thực thi thực tế từ thiết bị phần cứng.

V. Kết quả đạt được

- Xây dựng thành công mô hình phần cứng hoàn chỉnh gồm: ESP32, Load Cell + HX711, Servo và hệ thống đèn LED – nút nhấn, hoạt động ổn định.
- Triển khai thành công kết nối IoT qua giao thức MQTT, đảm bảo thiết bị nhận lệnh nhanh (gần như tức thì) và gửi dữ liệu cảm biến về server chính xác.
- Hoàn thiện backend NodeJS kết nối với MongoDB, hỗ trợ đầy đủ các chức năng: quản lý người dùng, thiết bị, lịch cho ăn và lưu trữ nhật ký hoạt động.
- Phát triển giao diện web bằng ReactJS, cho phép người dùng điều khiển cho ăn thủ công, thiết lập lịch tự động, xem biểu đồ thống kê và theo dõi trạng thái thiết bị theo thời gian thực.

- Load Cell đo chính xác lượng thức ăn trước và sau khi phân phối, giúp hệ thống ghi lại lịch sử tiêu thụ để người dùng theo dõi.
- Tích hợp thành công tính năng điều khiển bằng giọng nói thông qua Web Speech Recognition API, hỗ trợ các lệnh cơ bản như cho ăn hoặc kiểm tra lượng thức ăn.
- Hệ thống vận hành ổn định trong các bài kiểm thử: xử lý lệnh nhanh, hiển thị dữ liệu đúng, hoạt động khi mất mạng tạm thời và đồng bộ lại khi có kết nối.
- Đáp ứng đầy đủ yêu cầu chức năng, phi chức năng và giao tiếp đã đặt ra trong đề tài, chứng minh tính khả thi và khả năng ứng dụng thực tế của mô hình.

VI. Kết luận và hướng phát triển

1. Ưu điểm

- Hệ thống hoạt động ổn định, đáp ứng đầy đủ các chức năng cho ăn thủ công, cho ăn theo lịch và theo dõi lượng thức ăn theo thời gian thực.
- Giao tiếp MQTT giúp thiết bị phản hồi nhanh, đảm bảo tính thời gian thực trong điều khiển IoT.
- Giao diện web ReactJS thân thiện, hiển thị biểu đồ và trạng thái thiết bị rõ ràng.
- Load Cell đo trọng lượng chính xác, hỗ trợ lưu lịch sử tiêu thụ thức ăn.
- Backend NodeJS – MongoDB xử lý dữ liệu hiệu quả, hỗ trợ quản lý người dùng, thiết bị và lịch cho ăn.
- Tích hợp thành công điều khiển bằng giọng nói qua Web Speech API, tăng tính tiện lợi.
- Hệ thống có tính khả thi cao, dễ triển khai trong các hộ gia đình hoặc cửa hàng thú cưng.

2. Hạn chế

- Mô hình phần cứng mới dừng lại ở dạng thử nghiệm, chưa có vỏ bảo vệ chống nước – chống bụi cho môi trường thực tế.
- Cơ cấu xả thức ăn còn đơn giản, dễ bị kẹt với những loại thức ăn kích thước lớn hoặc dạng viên không đồng đều.

- Tính năng AI mới chỉ dừng ở mức điều khiển bằng giọng nói, chưa có thuật toán phân tích thói quen ăn uống.

3. Hướng phát triển

- Hoàn thiện phần cứng: thiết kế vỏ hộp chống nước, tối ưu cơ cấu xả thức ăn, bổ sung cảm biến chống kẹt motor.
- Phát triển thêm ứng dụng mobile (Android/ iOS) để tiện sử dụng hơn
- Mở rộng chức năng sử dụng AI: phân tích thói quen ăn uống, dự đoán nhu cầu dinh dưỡng và cảnh báo bất thường.
- Hỗ trợ quản lý nhiều thiết bị và nhiều vật nuôi trên một tài khoản.
- Tăng cường bảo mật: mã hóa MQTT nâng cao, xác thực hai lớp, phát hiện truy cập bất thường.
- Tối ưu trải nghiệm người dùng: thêm biểu đồ chi tiết, lịch sử đầy đủ và thông báo đa kênh (Zalo/ Telegram/ Email).
- Cải thiện khả năng hoạt động ngoại tuyến, thêm pin dự phòng để thiết bị duy trì hoạt động khi mất điện dài hạn.

TÀI LIỆU THAM KHẢO

1. Tài liệu tiếng Việt

[1] https://dlib.ptit.edu.vn/handle/HVCNBCVT/3422

[2] https://kcokhi.duytan.edu.vn/tai-lieu-tham-khao-s4h/thiet-ke-he-thong-cham-soc-thu-cung-tu-dong-axd

2. Tài liệu tiếng Anh

[1] https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[2] https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

[3] https://www.hivemq.com/mqtt-essentials/

[4] https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

[5] https://react.dev/reference/react