NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Vĩnh Long, ngày ..... tháng …… năm ……

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Vĩnh Long, ngày ..... tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành tốt bài báo cáo này, trước hết tôi xin cảm ơn đến quý thầy cô giáo trong khoa Công nghệ thông tin, Trường Kỹ thuật và Công nghệ. Tôi xin cảm ơn đến ban lãnh đạo nhà trường đã tạo mọi điều kiện tốt nhất để tôi có cơ hội được trình bày và thể hiện ý tưởng của mình, từ đó áp dụng kiến thức, kỹ năng của mình để đưa ý tưởng ấy thành sản phẩm thực tế có thể áp dụng trong đời sống.

Đặc biệt, tôi xin gửi lời cảm ơn chân thành đến thầy Đoàn Phước Miền - Giảng viên trực tiếp hướng dẫn, thầy đã tận tình giúp đỡ tôi trong quá trình học tập cũng như trong việc nâng cao kiến thức chuyên môn, phát huy tính tự học, tự tìm hiểu, từ đó hình thành một hướng đi rõ ràng cho tôi trong thiết kế và lập trình một website, giúp tôi dễ dàng tiếp cận và ngày càng tiến sâu với chuyên ngành Công nghệ thông tin một cách nhanh chóng hơn. Đồng thời trau dồi cho tôi kỹ năng học tập năng động và sáng tạo, rèn luyện tư duy cũng như khả năng lập trình. Giúp cho tôi trau dồi đươc nhiều bài học mới, được tìm hiểu nhiều ngôn ngữ lập trình. Giảng viên hướng dẫn đã luôn theo sát từng giai đoạn và có những lời chỉ dẫn kịp thời để tôi có thể hoàn thành một website hoàn chỉnh và đúng với yêu cầu.

Do khả năng cũng như kiến thức chuyên môn của bản thân tôi còn hạn chế, nên trong quá trình làm bài báo cáo tôi không tránh khỏi những sai sót, tôi rất mong nhận được những ý kiến đóng góp của quý thầy, cô để tôi có thể khắc phục những thiếu sót và có thể hoàn thiện bài báo cáo hơn trong tương lai.

Cuối lời, tôi cũng xin kính chúc quý thầy, cô và ban lãnh đạo nhà trường có nhiều sức khỏe và ngày càng thành công hơn trong công việc.

Tôi xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Vấn đề nghiên cứu của đề tài xuất phát từ thực trạng ngành chăn nuôi lươn hiện nay dù mang lại giá trị kinh tế cao nhưng quy trình quản lý vẫn chủ yếu dựa trên phương pháp thủ công, thiếu sự hỗ trợ của các hệ thống số hóa, dẫn đến khó khăn trong việc tối ưu hóa năng suất và kiểm soát rủi ro dịch bệnh.

Để khắc phục hạn chế này, hướng tiếp cận của đề tài là ứng dụng công nghệ thông tin để chuyển đổi số toàn diện quy trình chăn nuôi, xây dựng một nền tảng quản lý tập trung và trực quan.

Cách giải quyết vấn đề được thực hiện thông qua việc phát triển website "Quản lý chăn nuôi lươn" sử dụng các công nghệ hiện đại (React.js, Node.js, MongoDB). Hệ thống cung cấp các giải pháp quản lý chặt chẽ từ cơ sở hạ tầng (bể nuôi), quy trình vận hành (nhật ký ăn, chỉ số môi trường nước pH/Oxy/Nhiệt độ, theo dõi sức khỏe) cho đến quản trị dòng tiền (chi phí, doanh thu).

Kết quả đạt được là một ứng dụng web hoàn chỉnh, cho phép người nuôi giám sát chặt chẽ tình hình trang trại, tự động hóa việc tổng hợp báo cáo tài chính và biểu đồ tăng trưởng, từ đó hỗ trợ ra quyết định chính xác để nâng cao hiệu quả kinh tế cho mô hình nuôi.

MỞ ĐẦU

1. Lí do chọn đề tài:

Trong bối cảnh nông nghiệp hiện đại, mô hình nuôi lươn không bùn đang trở thành điểm sáng kinh tế nhờ hiệu quả cao và nhu cầu thị trường lớn. Tuy nhiên, qua khảo sát thực tế tại các hộ gia đình và trang trại quy mô vừa và nhỏ, công tác quản lý vẫn còn tồn tại nhiều bất cập do phụ thuộc hoàn toàn vào phương pháp thủ công. Việc ghi chép bằng sổ sách rời rạc khiến người nuôi gặp khó khăn trong việc truy xuất lịch sử nguồn gốc, kiểm soát tỉ lệ hao hụt, và đặc biệt là thiếu sự minh bạch trong hạch toán dòng tiền. Xuất phát từ nhu cầu cấp thiết đó, đề tài **"Xây dựng website quản lý trại chăn nuôi lươn"** được thực hiện nhằm cung cấp một giải pháp chuyển đổi số toàn diện, giúp chuẩn hóa quy trình nuôi, quản lý chặt chẽ chi phí và tối ưu hóa lợi nhuận cho người nông dân.

2. Mục đích:

Website mong muốn thay thế phương thức ghi chép thủ công bằng hệ thống cơ sở dữ liệu tập trung, đảm bảo an toàn và dễ dàng tra cứu. Cung cấp công cụ theo dõi liên tục các chỉ số môi trường nước (pH, nhiệt độ, oxy) và sức khỏe vật nuôi để kịp thời phát hiện và xử lý rủi ro. Xây dựng thuật toán tự động tổng hợp chi phí đầu vào (con giống, thức ăn, thuốc, điện nước) và doanh thu đầu ra, từ đó tính toán chính xác lãi/lỗ cho từng bể nuôi theo thời gian thực.

3. Đối tượng nghiên cứu:

Quy trình kỹ thuật nuôi lươn (giống lươn, môi trường, bể nuôi, chăm sóc sức khỏe và chăm sóc cho ăn hằng ngày), các hoạt động ảnh hưởng đến việc thu và chi trong quá trình nuôi lươn. Các công nghệ hỗ trợ (React.js, Node.js, Mongo DB, JSON Web Token (JWT), Chart.js).

4. Phạm vi nghiên cứu:

Website được thiết kế phù hợp cho các mô hình nuôi lươn có quy mô vừa và nhỏ bao gồm quản lý bể nuôi, môi trường nuôi, quản lý nhật ký thu, quản lý nhật ký chi, quản lý sức khỏe lươn.

CHƯƠNG 1: TỔNG QUAN

1.1 Bối cảnh của đề tài:

Trong bối cảnh nông nghiệp 4.0 đang phát triển mạnh mẽ, việc ứng dụng công nghệ thông tin vào sản xuất nông nghiệp không còn là xu hướng mà đã trở thành yêu cầu cấp thiết để nâng cao năng suất và chất lượng. Tại Việt Nam, mô hình nuôi lươn không bùn (nuôi trong bể xi măng hoặc bể bạt) đang nổi lên như một hướng đi kinh tế hiệu quả, mang lại giá trị thương phẩm cao và thị trường tiêu thụ ổn định.

Tuy nhiên, thực trạng quản lý tại các hộ gia đình và trang trại quy mô vừa và nhỏ hiện nay vẫn còn tồn tại nhiều bất cập. Đa số quy trình quản lý từ nhập con giống, theo dõi lịch cho ăn, kiểm soát môi trường nước đến hạch toán chi phí đều được thực hiện thủ công, dựa trên kinh nghiệm cá nhân hoặc ghi chép sổ sách rời rạc.

Phương thức quản lý truyền thống này dẫn đến các vấn đề nghiêm trọng như:

- **Thiếu minh bạch tài chính:** Khó xác định chính xác chi phí đầu vào (thức ăn, thuốc, điện nước) so với doanh thu thực tế, dẫn đến việc hạch toán lãi/lỗ chỉ mang tính ước lượng.
- **Rủi ro vận hành cao:** Không có dữ liệu lịch sử về môi trường và dịch bệnh để phân tích nguyên nhân khi xảy ra sự cố (lươn chết, chậm lớn,…).
- **Khó khăn trong truy xuất:** Không quản lý được tồn kho vật tư và nguồn gốc con giống một cách hệ thống.

Xuất phát từ nhu cầu thực tiễn đó, đề tài **"Xây dựng website quản lý trại chăn nuôi lươn "** được lựa chọn nghiên cứu và thực hiện nhằm cung cấp một giải pháp chuyển đổi số toàn diện cho người nuôi lươn.

1.2 Mục tiêu cốt lõi

Đề tài tập trung giải quyết bài toán **số hóa quy trình chăn nuôi**, chuyển đổi từ việc ghi chép thủ công sang quản lý dựa trên dữ liệu. Hệ thống hướng tới việc giúp người nông dân:

- **Quản lý tập trung:** Theo dõi toàn bộ vòng đời của lươn từ lúc nhập giống đến khi xuất bán.

- **Kiểm soát chi phí:** Tự động hóa việc tính toán dòng tiền, theo dõi sát sao chi phí thức ăn, thuốc và vận hành để tối ưu hóa lợi nhuận.
- **Giám sát chặt chẽ:** Ghi nhận và cảnh báo các chỉ số môi trường, sức khỏe vật nuôi để giảm thiểu rủi ro hao hụt.

1.3 Hướng tiếp cận công nghệ

Để giải quyết các vấn đề trên, hệ thống được xây dựng trên nền tảng Web Application hiện đại, đảm bảo tính sẵn sàng, bảo mật và dễ dàng mở rộng:

- **Kiến trúc hệ thống:** Sử dụng mô hình **MERN Stack** (MongoDB, Express.js, React.js, Node.js). Đây là bộ công nghệ mạnh mẽ, linh hoạt, cho phép xử lý dữ liệu lớn và phản hồi thời gian thực.
- **Cơ sở dữ liệu:** Sử dụng **MongoDB** (NoSQL) để thiết kế các mô hình dữ liệu linh hoạt, phù hợp với đặc thù quản lý chăn nuôi có nhiều biến động (như nhật ký ăn, nhật ký sức khỏe, biến động kho).
- **Giao diện người dùng:** Xây dựng giao diện thân thiện, trực quan với **React.js** và các biểu đồ thống kê (Chart.js), giúp người dùng dễ dàng nắm bắt tình hình trang trại thông qua các Dashboard tổng quan.

Kết quả của đề tài không chỉ dừng lại ở một sản phẩm phần mềm, mà còn đề xuất một **quy trình quản lý chuẩn hóa** cho nghề nuôi lươn. Hệ thống giúp người nuôi chuyển đổi tư duy từ "làm theo kinh nghiệm" sang "quản lý bằng công nghệ", từ đó giảm thiểu rủi ro, tiết kiệm chi phí và nâng cao hiệu quả kinh tế bền vững.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Tổng quan về công nghệ sử dụng

2.1.1 Giới thiệu về ReactJS

2.1.1.1 Khái niệm

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, ra mắt vào năm 2013 với mục đích để xây dựng giao diện người dùng. Nó được sử dụng rộng rãi để xây dựng các trang web SPA (Single Page Application) và các ứng dụng trên nền tảng di động. Nó rất dễ sử dụng và cho phép người dùng có thể tạo các component UI có thể tái sử dụng [2].

Một điểm nổi bật của ReactJS là Virtual DOM, giúp tăng hiệu suất hiển thị. Khi dữ liệu thay đổi, React chỉ cập nhật những phần của giao diện cần thiết thay vì tải lại toàn bộ trang, từ đó cải thiện tốc độ phản hồi và trải nghiệm người dùng.

Hình 2.1: ReactJS là thư viện JavaScript chuyên cho xây dựng giao diện

2.1.1.2 Các tính năng nổi bật của ReactJS

ReactJS có nhiều tính năng hữu ích cho việc phát triển ứng dụng web, bao gồm:

- **Components**: ReactJS cho phép phát triển ứng dụng web theo mô hình component. Các component là các phần tử UI độc lập có thể được tái sử dụng trong nhiều phần khác nhau của ứng dụng.
- **Virtual DOM**: ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM là một bản sao của DOM được lưu trữ trong bộ nhớ và được cập nhật một cách nhanh chóng khi có thay đổi, giúp tăng tốc độ và hiệu suất của ứng dụng.

- **JSX**: JSX là một ngôn ngữ lập trình phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI. JSX kết hợp HTML và JavaScript, giúp cho việc viết mã dễ hiểu và dễ bảo trì hơn.
- **State và Props**: ReactJS cho phép quản lý trạng thái của các thành phần UI thông qua State và Props. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.
- **Hỗ trợ đa nền tảng**: ReactJS không chỉ được sử dụng để phát triển ứng dụng web, mà còn được sử dụng để phát triển ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.

2.1.1.3 Cách thức hoạt động của ReactJS

ReactJS có thể cho phép truyền mã HTML với JavaScript với những lợi ích:

- DOM là một cấu trúc cây biểu diễn tài liệu HTML, và JavaScript có thể tương tác với DOM để thay đổi nội dung và cấu trúc của trang web.
- ReactJS sử dụng một thuật toán diff hiệu quả để tìm ra những phần tử cần thay đổi và chỉ cập nhật những phần đó trên DOM thực tế, tránh việc cập nhật lại toàn bộ DOM.
- Khi có nhu cầu đọc và ghi vào DOM, JSX sẽ sử dụng DOM ảo của nó. Sau đó DOM ảo sẽ cố gắng tìm cách hiệu quả để cập nhật DOM của trình duyệt.

Trong ReactJS, bạn tạo ra các phần tử React bằng cách sử dụng các hàm JSX như , <button>,… Các phần tử này không phải là DOM thực, mà là các đối tượng đơn giản được tạo ra dễ dàng.

ReactJS sử dụng một DOM ảo (Virtual DOM) để tối ưu quá trình cập nhật DOM thực. Khi bạn cập nhật một phần tử React, ReactJS sẽ so sánh phần tử mới với phần tử cũ trong DOM ảo, sau đó chỉ cập nhật những phần cần thiết trong DOM thực, nhờ tốc độ xử lý nhanh của JavaScript.

Ngoài ra, mặc dù được tạo ra để sử dụng trong trình duyệt nhưng thiết kế của ReactJS sẽ khiến chúng có lợi khi sử dụng trên máy chủ Node.JS.

2.1.1.4 So sánh ReactJS và các thư viện Javascript khác

Khi so sánh ReactJS với các framework hoặc thư viện JavaScript phổ biến khác như Angular hoặc Vue.js thì tính linh hoạt sẽ là điểm nổi bật của framework ReactJS. Không giống như Angular hay Vue.js, chúng chỉ ra những cách cụ thể để giải quyết vấn đề thì ReactJS không có quan điểm nhất định. Điều này cho phép các lập trình viên có nhiều quyền tự do hơn trong việc lựa chọn cách xây dựng cấu trúc ứng dụng của họ.

Khi nói về giao tiếp giữa máy khách (client-side) và máy chủ (server-side), cả Vue.js và ReactJS thường sử dụng Axios trong khi Angular cung cấp module client-side HTTP. Một điểm khác biệt khác về kích thước khi xem xét các đối tác nhẹ như Preact, một giải pháp thay thế nhỏ hơn cung cấp chức năng tương tự như ReactJS.

Tóm lại, ReactJS nổi bật hơn so với các framework và thư viện JavaScript khác nhờ vào tính linh hoạt nó mang lại. Giúp các lập trình viên xây dựng ứng dụng một trang (SPA) một cách dễ dàng hơn.

2.1.1.5 Ưu điểm và nhược điểm của ReactJS

- **Ưu điểm:**

- Hiệu suất cao nhờ Virtual DOM: ReactJS sử dụng Virtual DOM để theo dõi mọi thay đổi trong giao diện. Khi state hoặc props thay đổi, React so sánh Virtual DOM và DOM thực và chỉ cập nhật những phần khác biệt, giúp tăng hiệu suất và giảm thời gian render.

- Quản lý trạng thái hiệu quả: ReactJS có state và props để quản lý dữ liệu nội bộ và dữ liệu truyền giữa các component. Luồng dữ liệu một chiều giúp kiểm soát trạng thái dễ dàng và giảm xung đột dữ liệu trong ứng dụng.

- Dễ dàng tích hợp với các thư viện và công nghệ khác: ReactJS có thể kết hợp với các thư viện như Tailwind CSS, Redux, React Router, hoặc các thư viện API khác, giúp xây dựng các ứng dụng web phức tạp một cách linh hoạt.

- Hỗ trợ phát triển đa nền tảng: React Native kế thừa kiến thức từ ReactJS, cho phép phát triển ứng dụng di động cho cả iOS và Android bằng cùng một mã nguồn.

- Cộng đồng lớn và tài nguyên phong phú: ReactJS có cộng đồng lớn, nhiều thư viện, plugin và tài liệu hướng dẫn chi tiết, thuận tiện cho việc học tập và phát triển ứng dụng.

- **Nhược điểm:**

- Chỉ là thư viện, không phải framework toàn diện: ReactJS chỉ tập trung vào UI, không cung cấp sẵn giải pháp cho routing, quản lý state toàn cục hay tương tác với backend, vì vậy cần kết hợp với các thư viện bổ sung.

- Đường cong học tập cao: ReactJS sử dụng JSX, Virtual DOM, component-based architecture và Hooks, có thể khó tiếp cận với người mới học.

- Cập nhật liên tục: Các phiên bản mới và thay đổi trong ReactJS và các thư viện liên quan có thể gây khó khăn khi bảo trì ứng dụng.

- Phụ thuộc vào JavaScript: Ứng dụng ReactJS yêu cầu trình duyệt hỗ trợ JavaScript. Nếu JavaScript bị tắt hoặc trình duyệt quá cũ, ứng dụng có thể không hiển thị đúng.

2.1.2 Giới thiệu về Tailwind CSS

2.1.2.1 Khái niệm

Tailwind CSS là một framework CSS utility-first, tương tự như Bootstrap, cung cấp một tập hợp các lớp CSS được tích hợp sẵn mà chúng ta có thể sử dụng trong ứng dụng của mình. Tuy nhiên, điểm đặc biệt của Tailwind CSS là nó cung cấp một số lượng lớn các lớp CSS với các thuộc tính, quy tắc CSS khác nhau. Thay vì sử dụng các lớp CSS được định nghĩa sẵn cho các thành phần cụ thể, Tailwind CSS tập trung vào việc cung cấp các lớp CSS cơ bản mà chúng ta có thể kết hợp để xây dựng giao diện.

Hình 2.2. Tailwindcss

Ví dụ, giả sử bạn muốn tạo một nút có chiều cao cố định, đệm ngang, màu nền đen, các cạnh bo tròn và phông chữ màu trắng, in đậm. Sau đây là HTML bạn sẽ sử dụng:

<button class="h-10 px-6 font-semibold rounded-md bg-black text-white" type="submit">Buy now</button>

HTML chứa 6 lớp tiện ích. Hãy phân tích từng lớp bên dưới:

- **h-10**: Lớp này đặt nút ở chiều cao cố định là 10 đơn vị.
- **px-6**: Lớp này đặt đệm ngang của nút ở 6 đơn vị.
- **font-semibold**: Lớp này đặt độ đậm của phông chữ của nút thành semibold.
- **rounded-md**: Lớp này đặt bán kính đường viền của nút sao cho các góc được bo tròn.
- **bg-black**: Lớp này đặt màu nền của nút thành màu đen.
- **text-white**: Lớp này đặt màu văn bản của nút thành màu trắng.

2.1.2.2 Các đặc điểm nổi bật của Tailwind CSS

- **Khả năng Tùy Biến Cao (Highly Customizable)**

- Tailwind được thiết kế để dễ dàng tùy chỉnh thông qua một tệp cấu hình JavaScript (tailwind.config.js).

- Thiết kế hệ thống (Design System): Bạn có thể định nghĩa toàn bộ hệ thống thiết kế của mình, bao gồm bảng màu, phông chữ, khoảng cách, breakpoints,...

- Mở rộng: Dễ dàng thêm các lớp tiện ích mới hoặc mở rộng các lớp hiện có.

- Ví dụ cấu hình: Thay đổi thang màu mặc định hoặc thêm một font chữ mới.

- **Thiết Kế Đáp Ứng (Responsive Design) Dễ Dàng**

- Tailwind sử dụng các tiền tố để áp dụng các lớp tiện ích cho các kích thước màn hình cụ thể (Breakpoints).

- Các tiền tố mặc định:
  - sm: màn hình nhỏ (Small)
  - md: màn hình trung bình (Medium)
  - lg: màn hình lớn (Large)
  - xl: màn hình cực lớn (Extra-Large)
  - 2xl: màn hình siêu lớn

- Cách sử dụng:

** ... **

  - Mặc định (mọi màn hình): chiếm 100% chiều rộng (w-full).

- Từ màn hình trung bình trở lên (md:): chiếm 50% chiều rộng (w-1/2).
- Từ màn hình lớn trở lên (lg:): chiếm 33.3% chiều rộng (w-1/3).

2.1.2.3 Cấu trúc hệ thống và cách Tailwind CSS tổ chức giao diện

- **Hệ thống phân chia theo thang đo (Design Scale)**

- Tailwind thiết kế giao diện dựa trên các thang đo được chuẩn hóa. Ví dụ:
  - khoảng cách (spacing)
  - cỡ chữ (typography scale)
  - độ bo góc
  - độ đậm nhạt màu sắc
  - độ rộng – độ cao theo tỷ lệ

- Nhờ việc mọi giá trị đều nằm trong quy chuẩn chung, giao diện của sản phẩm trở nên đồng bộ, tránh tình trạng mỗi thành phần sử dụng một kích thước khác nhau.

- **Hệ thống giao diện responsive**

Tailwind áp dụng phương pháp thiết kế tăng tiến (progressive enhancement), hay còn gọi là *mobile-first*. Giao diện được thiết kế phù hợp cho thiết bị nhỏ trước, sau đó mở rộng dần cho thiết bị lớn. Hệ thống breakpoint của Tailwind rõ ràng, dễ nhớ và có thể tùy chỉnh theo nhu cầu dự án. Điều này giúp quá trình xây dựng giao diện responsive trở nên dễ dàng hơn nhiều.

- **Hỗ trợ các trạng thái và chế độ hiển thị phức tạp**

- Không chỉ hỗ trợ các trạng thái như rê chuột, focus hay active, Tailwind còn hỗ trợ các trạng thái nâng cao như:
  - chế độ nền tối (Dark Mode)
  - trạng thái nhóm (group modifier)
  - trạng thái phụ thuộc vào phần tử cha
  - các hiệu ứng chuyển động cơ bản

2.1.2.4 Ưu nhược điểm của Tailwind CSS

- **Ưu điểm**

- Tốc độ phát triển nhanh: Do có thể viết style trực tiếp trong HTML, Tailwind giúp rút ngắn đáng kể thời gian thiết kế giao diện. Bên cạnh đó, việc sử dụng hệ thống lớp tiện ích chuẩn hoá giúp hạn chế tình trạng phải điều chỉnh nhiều lần.

- Không áp đặt thiết kế: Không giống các framework như Bootstrap vốn có phong cách mặc định tương đối rõ rệt, Tailwind hoàn toàn không áp đặt bất kỳ giao diện nào. Điều này giúp lập trình viên tự do thiết kế giao diện phù hợp với cá tính thương hiệu hoặc yêu cầu của sản phẩm mà không cần tốn công ghi đè style có sẵn.

- Giảm độ phức tạp của CSS thủ công: Tailwind giúp giảm thiểu sự phụ thuộc vào file CSS riêng. Việc không phải viết CSS mới cho từng thành phần giúp tránh tình trạng phân mảnh style, đặc biệt trong những dự án lớn với nhiều lập trình viên cùng tham gia.

- Tối ưu hiệu năng: Nhờ cơ chế chỉ giữ lại các lớp được sử dụng thực tế, Tailwind giúp kích thước CSS trong môi trường sản phẩm (production) nhỏ hơn nhiều so với các framework truyền thống. Điều này giúp cải thiện tốc độ tải trang, ảnh hưởng tích cực đến trải nghiệm người dùng và cả điểm chất lượng SEO [3].

- **Nhược điểm**

- Mã HTML có thể trở nên dài và khó đọc: Do mọi kiểu dáng đều được đưa trực tiếp vào HTML, các thẻ có thể chứa nhiều lớp tiện ích, khiến mã nguồn trở nên dài. Tuy nhiên, điều này có thể được khắc phục bằng việc tổ chức code hợp lý hoặc tách các khối giao diện thành component.

- Đòi hỏi thời gian làm quen: Người mới phải nhớ hệ thống class khá nhiều (ví dụ: bg-blue-500, px-4, text-lg). Ban đầu có thể cảm thấy phức tạp so với CSS thuần.

- Khó quản lý khi thiếu quy ước chung: Nếu team không có guideline, code dễ bị trùng lặp class và khó duy trì về lâu dài.

- Không phù hợp cho dự án nhỏ, đơn giản: Với website tĩnh, ít trang, việc tích hợp Tailwind có thể "thừa tính năng" so với CSS thuần [3].

2.1.3 Giới thiệu về NodeJS

2.1.3.1 Khái niệm

Node.js là một môi trường chạy JavaScript phía máy chủ (server-side runtime environment), được xây dựng trên JavaScript V8 Engine của Google – đây cũng chính là động cơ thực thi JavaScript của trình duyệt Chrome. Khác với JavaScript truyền thống vốn chỉ chạy trên trình duyệt, Node.js cho phép lập trình viên sử dụng JavaScript để xây dựng các ứng dụng phía máy chủ như API, dịch vụ web, hệ thống real-time và các ứng dụng mạng [4].

Hình 2.3: Node.js

- Nguồn mở (Open-source): Mã nguồn của Node.js được công bố công khai, điều này có nghĩa là bất kỳ ai cũng có thể truy cập, sử dụng, và đóng góp vào mã nguồn. Node.js được duy trì bởi cộng đồng lập trình viên trên toàn thế giới, và hướng dẫn đóng góp của Node.js hướng dẫn bạn cách để bạn có thể góp phần phát triển nó.

- Đa nền tảng (Cross-platform): Node.js không phụ thuộc vào bất kỳ hệ điều hành nào cụ thể nào, nghĩa là nó có thể chạy trên Linux, macOS hoặc Windows. Điều này làm cho Node.js trở thành một lựa chọn linh hoạt cho các nhà phát triển muốn xây dựng các ứng dụng có thể hoạt động trên nhiều nền tảng khác nhau mà không cần thay đổi mã nguồn.

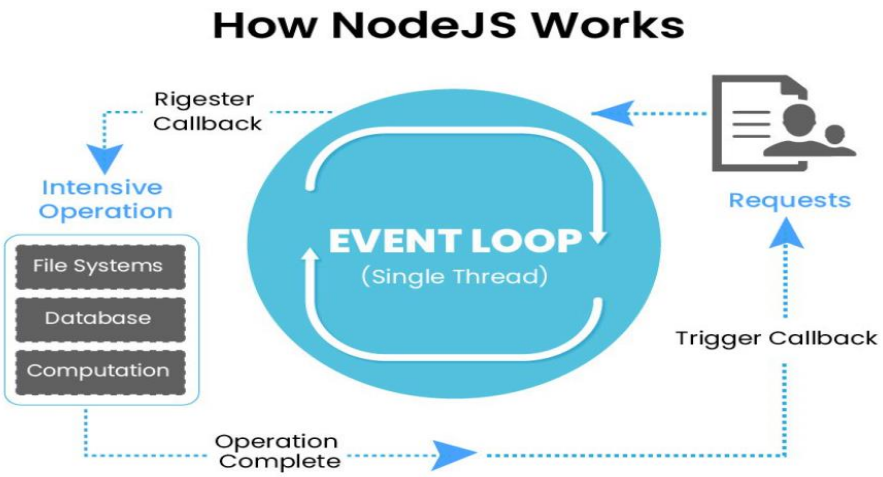
- Môi trường thực thi JavaScript (JavaScript runtime environment): Để mã JavaScript có thể được thực thi, nó cần một môi trường chạy phù hợp. Trong khi trình duyệt như Chrome và Firefox cung cấp một môi trường thực thi cho JavaScript, Node.js mở rộng khả năng này ra ngoài trình duyệt. Node.js cho phép chạy JavaScript trên máy chủ, hoặc trong bất kỳ môi trường máy tính nào khác, không chỉ trong trình duyệt.

- Dựa trên V8 JavaScript Engine: Node.js được xây dựng dựa trên V8, động cơ JavaScript được phát triển bởi Google cho trình duyệt Chrome. Điều này giúp Node.js có khả năng thực thi JavaScript nhanh và hiệu quả, đồng thời hỗ trợ các tính năng mới nhất của ngôn ngữ JavaScript [1].

2.1.3.2 Cách hoạt động của NodeJS

Node.js hoạt động dựa trên một số nguyên tắc cơ bản giúp nó hiệu quả trong việc xử lý các ứng dụng có nhiều hoạt động nhập/xuất (I/O) mà không bị chặn, đồng

thời giảm đáng kể sự phức tạp trong quản lý các luồng thực thi. Dưới đây là một số thành phần chính giải thích cách thức hoạt động của Node.js [1]:



Hình 2.4 How Node.js works.

- Kiến trúc Non-blocking I/O và Event-Driven:

Node.js sử dụng một mô hình non-blocking I/O (input/output) và event-driven, nghĩa là các hoạt động như đọc file, truy vấn cơ sở dữ liệu, hoặc giao tiếp mạng được thực hiện mà không chặn tiến trình chính. Điều này cho phép xử lý nhiều yêu cầu cùng lúc mà không cần tạo nhiều luồng (thread), giúp giảm bớt chi phí liên quan đến quản lý luồng và tối ưu hóa hiệu suất.

- V8 JavaScript Engine

Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, đây là một động cơ rất nhanh cho phép biên dịch mã JavaScript thành mã máy để thực thi trực tiếp trên phần cứng, làm tăng hiệu suất thực thi.

- Single-Threaded

Mặc dù Node.js hoạt động trên một luồng duy nhất cho logic ứng dụng của người dùng, nó vẫn sử dụng nhiều luồng ở tầng thấp hơn thông qua thư viện libuv để xử lý các hoạt động I/O. Tuy nhiên, những chi tiết này được ẩn giấu khỏi người dùng, giúp việc lập trình đơn giản hơn mà vẫn đảm bảo hiệu suất.

- Event Loop

Trái tim của Node.js là "event loop". Đây là vòng lặp sự kiện mà ở đó Node.js tiếp tục lắng nghe sự kiện và thực hiện các hàm gọi lại khi một sự kiện được kích hoạt. Vòng lặp sự kiện cho phép Node.js xử lý hàng nghìn kết nối đồng thời mà không cần phải tạo ra chi phí quản lý luồng.

- Trigger Callback

Khi thao tác I/O hoàn tất, hệ điều hành thông báo cho Node.js, và Node.js sau đó thực thi hàm callback tương ứng để xử lý kết quả hoặc tiếp tục xử lý logic.

- NPM (Node Package Manager)

NPM là hệ thống quản lý gói cho Node.js, cho phép các nhà phát triển dễ dàng chia sẻ và sử dụng mã nguồn từ nhau. NPM là một trong những kho lưu trữ mã nguồn mở lớn nhất thế giới và chứa hàng ngàn module có thể được tích hợp vào ứng dụng của bạn.

- Require

Require làm 3 thứ:

- Tải module đi kèm với Node.js như hệ thống file và HTTP từ Node.js API.
- Tải thư viện thứ 3 như Express và Mongoose mà bạn cài đặt từ npm.
- Giúp bạn require file của bạn và mo-đun hoá project.

Require là 1 chức năng, và nó nhận tham số path tinh chỉnh và trả về *module.export* [1].

2.1.3.3 Ưu nhược điểm của NodeJS

- **Ưu điểm**

- Có tốc độ xử lý nhanh nhờ cơ chế xử lý bất đồng bộ (non-blocking). Bạn có thể dễ dàng xử lý hàng ngàn kết nối trong khoảng thời gian ngắn nhất.

- Giúp dễ dàng mở rộng khi có nhu cầu phát triển website.

- Nhận và xử lý nhiều kết nối chỉ với một single-thread. Nhờ đó, hệ thống xử lý sẽ sử dụng ít lượng RAM nhất và giúp quá trình xử Nodejs lý nhanh hơn rất nhiều.

- Có khả năng xử lý nhiều Request/s cùng một lúc trong thời gian ngắn nhất.

- Có khả năng xử lý hàng ngàn Process cho hiệu suất đạt mức tối ưu nhất.

-Phù hợp để xây dựng những ứng dụng thời gian thực như các ứng dụng chat, mạng xã hội …

- **Nhược điểm**

- Nodejs gây hao tốn tài nguyên và thời gian. Nodejs được viết bằng C++ và JavaScript nên khi xử lý cần phải trải qua một quá trình biên dịch. Nếu bạn cần xử lý những ứng dụng tốn tài nguyên CPU thì không nên sử dụng Nodejs.

- Nodejs so với các ngôn ngữ khác như PHP, Ruby và Python sẽ không có sự chênh lệch quá nhiều. Nodejs có thể sẽ phù hợp với việc phát triển ứng dụng mới. Tuy nhiên khi xây dựng và triển khai dự án quan trọng thì Nodejs không phải là sự lựa chọn hoàn hảo nhất.

2.1.4 Giới thiệu về MongoDB

2.1.4.1 Khái niệm

MongoDB là một chương trình cơ sở dữ liệu mã nguồn mở được thiết kế theo kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào. Chính do cấu trúc linh hoạt này nên MongoDB có thể được dùng để lưu trữ các dữ liệu có cấu trúc phức tạp và đa dạng và không cố định (hay còn gọi là Big Data).

Hình 2.5 Mongo DB

2.1.4.2 Các tính năng nổi bật của MongoDB

Mongodb được sử dụng rất nhiều và được đánh giá vô cùng cao nhờ sở hữu nhiều đặc điểm nổi trội là:

- Mongodb chính là một database hướng tài liệu, nên khi đó mọi dữ liệu sẽ được lưu trữ trong document theo kiểu JSON thay vì lưu theo dạng bảng như CSDL quan hệ nên việc truy cập vấn sẽ nhanh chóng hơn.
- Với các CSDL quan hệ thì chúng ta sẽ có khái niệm bảng, khi đó các cơ sở dữ liệu quan hệ sẽ sử dụng các bảng để có thể lưu trữ dữ liệu, còn với Mongodb thì bạn cần phải sử dụng khái niệm collection thay cho bảng.

- Mongodb chính là một hệ quản trị cơ sở dữ liệu mà trong đó mã nguồn mở là CSDL thường thuộc NoSql và được hàng triệu người sử dụng.
- So với RDBMS thì trong Mongodb collection thường sẽ ứng với table, còn document sẽ tương ứng với row. MongoDB sẽ sử dụng các document để thay cho row trong RDBMS.
- Với các collection có trong Mongodb thường sẽ được cấu trúc rất linh hoạt nên nó cho phép các dữ liệu được lưu trữ mà không cần phải tuân theo bất kỳ một cấu trúc nhất định nào.
- Các thông tin có liên quan đều sẽ được lưu trữ cùng với nhau để người dùng có thể truy cập truy vấn nhanh hơn thông qua các ngôn ngữ truy vấn Mongodb.

2.1.4.3 Ưu nhược điểm của MongoDB

- **Ưu điểm**

- Bởi vì Mongodb sử dụng các dữ liệu dưới dạng Document JSON nên mỗi một collection đều có kích cỡ và document khác nhau. Nhưng chúng lại rất linh hoạt khi thực hiện lưu trữ bởi vậy nếu bạn muốn thứ gì thì chỉ cần insert thoải mái.

- Các dữ liệu có trong Mongodb thường không ràng buộc lẫn nhau, chúng không có join như trong RDBMS, nên khi bạn insert, xóa hoặc update thì sẽ không phải bỏ ra quá nhiều thời gian để kiểm tra chúng có thỏa mãn các ràng buộc như trong RDBMS hay không.

- Mongodb dễ mở rộng được, và trong Mongodb luôn có khái niệm cluster chính là cụm các node sẽ có chứa các dữ liệu giao tiếp với nhau. Nên chỉ cần bạn muốn mở rộng hệ thống thì chỉ việc thêm một node mới vào cluster.

- Các trường hợp dữ liệu "_id" sẽ luôn được đánh tự động index, nên tốc độ truy vấn thông tin sẽ luôn đạt hiệu suất cao nhất.

- Nếu như có một truy vấn dữ liệu, thì bản ghi sẽ được cached lên bộ nhớ Ram. Khi đó, việc phục vụ lượt truy vấn sau sẽ diễn ra nhanh hơn mà bạn không cần phải đọc từ ổ cứng.

- Tốc độ truy vấn của Mongodb luôn nhanh hơn so với các hệ quản trị cơ sở dữ liệu quan hệ. Nhờ có một lượng đủ dữ liệu nên việc thử nghiệm cho thấy tốc độ insert của Mongodb sẽ nhanh gấp 100 lần so với MySQL.

- **Nhược điểm**

- Mongodb không sở hữu các tính chất ràng buộc như trong RDBMS nên khi bạn thao tác với Mongodb cần phải cẩn thận hết sức.

- Khi thực hiện insert/update/remove bản ghi thì MongoDB sẽ chưa thể cập nhật ngay vào ổ cứng. Chỉ sau 60 giây thì Mongodb mới có thể ghi được toàn bộ dữ liệu được thay đổi từ RAM xuống phần ổ cứng. Điều này chính là nhược điểm bởi nó có thể mang lại nguy cơ mất dữ liệu khi các tình huống xấu như mất điện xảy ra.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả bài toán

Hệ thống website quản lý chăn nuôi lươn được xây dựng nhằm mục tiêu cung cấp một nền tảng quản lý chăn nuôi lươn theo xu hướng số hóa giúp người chăn nuôi có thể quản lý toàn bộ quá trình chăn nuôi lươn một cách thuận tiện hơn. Bài toán đặt ra cho hệ thống là thiết kế và hiện thực hóa một quy trình chăn nuôi, cho phép người dùng thao tác hiệu quả hơn với cơ sở dữ liệu trong chăn nuôi lươn, bao gồm việc tìm kiếm thông tin các danh mục, quản lý các danh mục,…Các yêu cầu cơ bản của bài toán có thể được mô tả như sau:

- **Quản lý bể nuôi:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách các bể nuôi.
- **Quản lý giống lươn:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách các giống lươn.
- **Quản lý kho thức ăn:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách thức ăn trong kho.
- **Quản lý kho thuốc:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách thuốc trong kho.
- **Quản lý nhật ký cho ăn:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách nhật ký cho ăn.
- **Quản lý môi trường nước:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách các môi trường nước.
- **Quản lý Sức khỏe & Dịch bệnh:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách các thông tin sức khỏe lươn hiện có.
- **Quản lý Xuất bán:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách xuất bán hiện có trong hệ thống.
- **Quản lý Chi phí vận hành:** Hệ thống cho phép người dùng thêm, xóa, sửa, xem danh sách các chi phí vận hành (điện, nước, vận chuyển,…) trong quá trình chăn nuôi.

- **Báo cáo Tài chính:** Cho phép người chăn nuôi theo dõi đầu ra / đầu vào của dòng tiền, phân loại chi phí theo bể nuôi, chi phí vận hành.

3.2 Yêu cầu hệ thống:

3.2.1 Yêu cầu về chức năng:

- Đăng ký, đăng nhập: Cho phép tạo tài khoản mới và đăng nhập để lưu trữ thông tin cá nhân.
- Quản lý các thông tin của lươn: Cho phép thêm, xóa, sửa, xem danh sách các thông tin trong quá trình chăn nuôi lươn (Giống lươn, bể nuôi, kho thức ăn, kho thuốc, nhật ký cho ăn, môi trường nước, sức khỏe & dịch bệnh, xuất bán, chi phí vận hành).
- Báo cáo tài chính: Cho phép người dùng xem thông tin đầu ra/đầu vào của dòng tiền, phân loại chi phí ra theo bể nuôi, chi phí vận hành khác.

3.2.2 Yêu cầu phi chức năng:

- Hiệu suất: Tốc độ tải trang nhanh, phản hồi thao tác người dùng mượt mà dưới 2 giây thông qua cơ chế SPA của ReactJS.
- Tính khả dụng: Hệ thống hoạt động ổn định 24/7, giao diện trực quan, dễ sử dụng ngay cả với người dùng không am hiểu công nghệ.
- Bảo mật cao: JWT hiệu quả trong việc xác thực và truyền tải dữ liệu an toàn giữa các bên, nhờ vào khả năng ký mã đảm bảo tính toàn vẹn và xác thực của thông tin.

3.3. Thiết kế cơ sở dữ liệu:

3.3.1. Danh sách các bảng dữ liệu:

Bảng 3.1: Quản lý thông tin người dùng:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
name	String		Tên người dùng
_id	ObjectId	PK	Mã người dùng

email	String		Email người dùng
password	String		Mật khẩu người dùng

Bảng 3.2: Quản lý thông tin bể nuôi:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
name	String		Tên bể nuôi
_id	ObjectId	PK	Mã bể nuôi
type	String		Loại lươn
Size	Number		Dung tích bể nuôi
location	String		Vị trí bể
status	String		Trạng thái bể (Trống/Đang nuôi)
currentBatchId	ObjectId	FK	Mã giống lươn
currentQuantity	Number		Số lượng lươn hiện tại
startQuantity	Number		Số lượng lươn ban đầu
startDate	Date		Ngày bắt đầu nuôi
createAt	Date		Ngày giờ tạo ra bể
updateAt	Date		Ngày giờ bị thay đổi gần nhất

Bảng 3.3: Quản lý chi phí vận hành:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
name	String		Tên chi phí vận hành
_id	ObjectId	PK	Mã chi phí vận hành

type	String		Loại khoản chi
amount	Number		Số tiền chi
date	Date		Ngày chi
payer	String		Người chi tiền
relatedTankid	ObjectId	FK	Mã bể nuôi
note	String		Ghi chú

Bảng 3.4: Quản lý giống lươn:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
name	String		Tên giống lươn
_id	ObjectID	PK	Mã giống lươn
quantity	Number		Số lượng con giống
sizeGrade	Number		Kích cỡ mẫu. Số con/1 kg
pricePerUnit	Number		Đơn giá của giống lươn
totalCost	Number		Tổng chi phí nhập giống
source	String		Tên nguồn nhập
tankID	ObjectId	FK	Mã bể nuôi
importDate	Date		Ngày nhập
notes	String		Ghi chú
createdAt	Date		Ngày giờ tạo phiếu nhập
update	Date		Ngày giờ lần chỉnh sửa gần nhất

Bảng 3.5: Quản lý môi trường:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
_id	ObjectId	PK	Mã môi trường
tankId	ObjectId	FK	Mã bể nuôi
pH	Number		Độ pH của nước
temperature	Number		Nhiệt độ nước (°C)
oxygen	Number		Hàm lượng Oxy (mg/L)
turbidity	Number		Độ đục / trong của nước
recordedAt	Date		Thời điểm ghi nhận tình trạng
createdAt	Date		Ngày giờ tạo bản ghi
updatedAt	Date		Ngày giờ chỉnh sửa lần gần nhất

Bảng 3.6: Quản lý Nhật ký cho ăn:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
_id	ObjectId	PK	Mã nhật ký cho ăn
tankId	ObjectId	FK	Mã bể nuôi
foodId	ObjectId	FK	Mã thức ăn
quantity	Number		Số lượng thức ăn cho ăn
estimatedCost	Number		Chi phí thức ăn đang cho ăn
feedingTime	Date		Ngày giờ cho ăn
notes	String		Ghi chú

Bảng 3.7: Quản lý sức khỏe:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
_id	ObjectId	PK	Mã quản lý sức khỏe
tankId	ObjectId	FK	Mã bể nuôi
disease	String		Tên bệnh
medicine	ObjectId	FK	Mã kho thuốc
medicineAmount	Number		Số lượng thuốc đã sử dụng
survivalRate	Number		Tỷ lệ sống sót
notes	String		Ghi chú
recordedAt	Date		Thời gian ghi nhận

Bảng 3.8: Quản lý thức ăn:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
_id	ObjectId	PK	Mã thức ăn
name	String		Tên thức ăn
type	String		Loại thức ăn
protein	String		Độ đạm trong thức ăn
unit	String		Đơn vị tính (kg, bao, tấn,…)
quantityImport	Number		Số lượng nhập
currentStock	Number		Số lượng tồn kho
pricePerUnit	Number		Giá nhập
totalCost	Number		Tổng chi phí nhập

supplierName	String		Tên người bán
supplierPhone	String		Số điện thoại người bán
source	String		Địa chỉ, nguồn gốc cửa hàng
importDate	Date		Ngày nhập hàng
expiryDate	Date		Hạn sử dụng
Notes	String		Ghi chú
createdAt	Date		Ngày giờ tạo phiếu nhập
updatedAt	Date		Ngày giờ chỉnh sửa phiếu

Bảng 3.9: Quản lý thuốc:

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
_id	ObjectId	PK	Mã thuốc
Name	String		Tên thuốc
Usage	String		Công dụng thuốc
Unit	String		Đơn vị tính
quantityImport	Number		Số lượng nhập
currentStock	Number		Số lượng tồn kho
pricePerUnit	Number		Giá nhập
totalCost	Number		Tổng chi phí
supplierName	String		Tên người bán
supplierPhone	String		Số điện thoại người bán

source	String		Nguồn nhập
importDate	Date		Ngày nhập
expiryDate	Date		Hạn sử dụng
notes	String		Ghi chú

Bảng 3.10: Quản lý xuất bán

Tên thuộc tính	**Kiểu dữ liệu**	**Khóa**	**Mô tả**
_id	ObjectId	PK	Mã xuất bán
tankId	ObjectId	FK	Mã bể nuôi
buyerName	String		Tên người mua
buyerPhone	String		Số điện thoại người mua
saleDatel	Date		Ngày bán
detail	Array		Grade: tên loại (Loại 1,…) Weight: số kg bán được của loại này Price: giá bán 1 kg của loại này Subtotal: thành tiền (weight * price)
totalWeight	Number		Tổng số lượng bán
totalRevenue	Number		Tổng doanh thu
quantitySold	Number		Số lượng con ước tính bán ra
isFinalHarvest	Boolean		Đánh dấu xem có Tát ao hay không
Notes	String		Ghi chú thêm
createdAt	Date		Ngày giờ tạo phiếu bán

3.4 Mô hình xử lý:

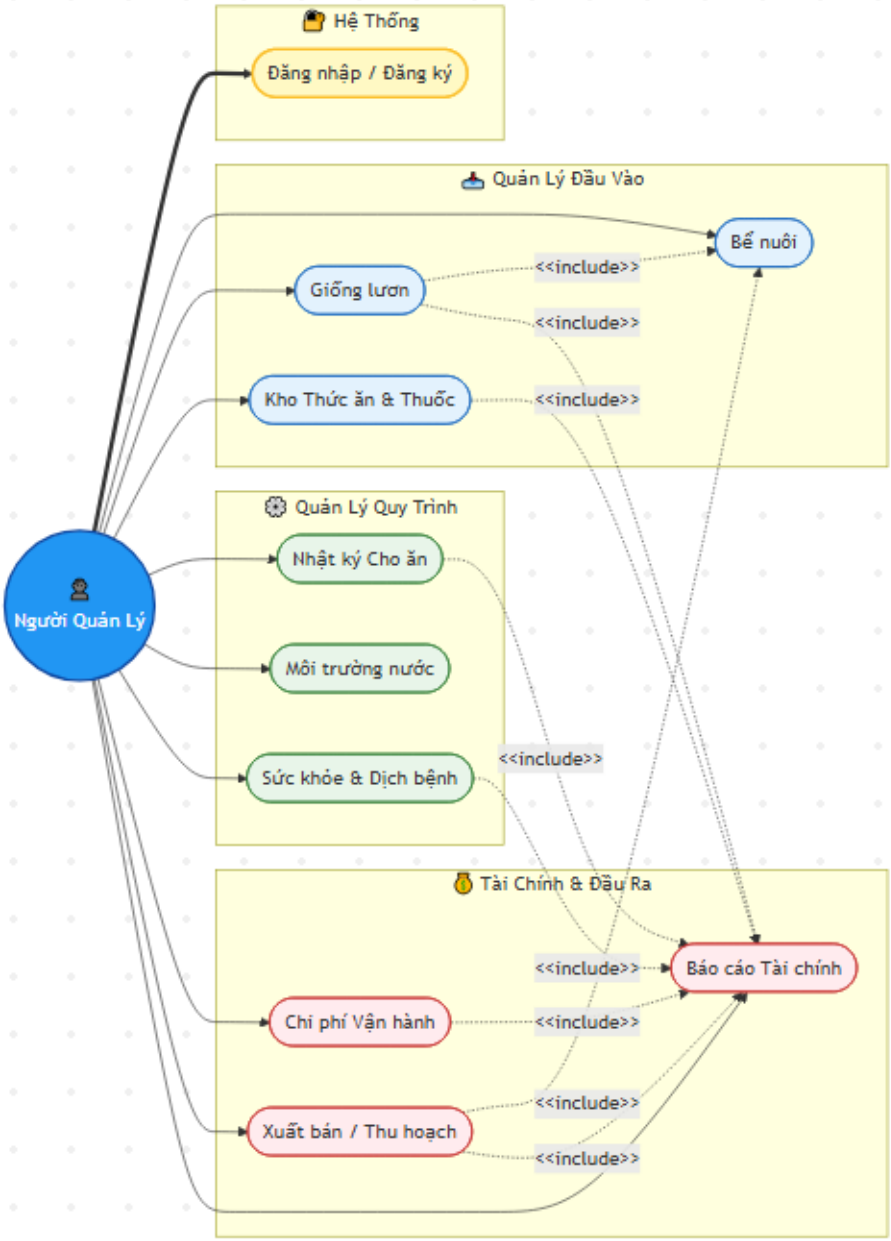
3.4.1. Sơ đồ Use case tổng quát

Hệ thống bao gồm một tác nhân là Người dùng.

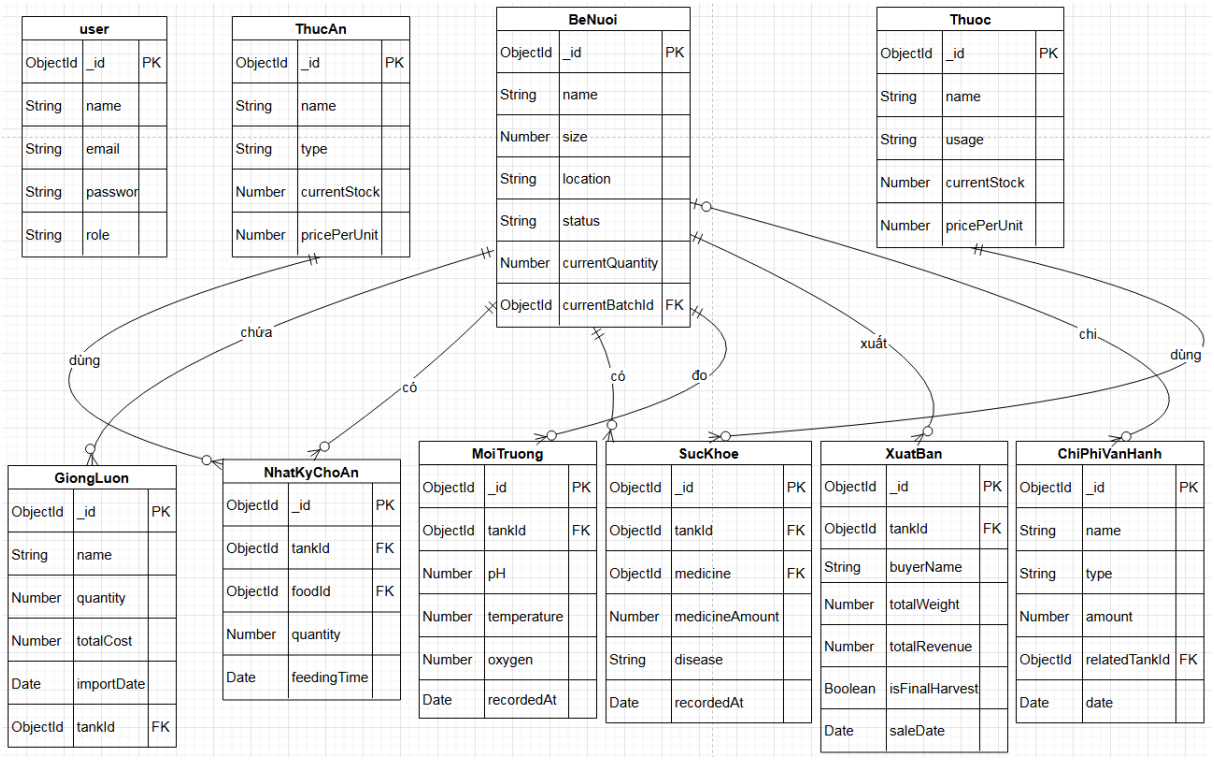
Tác nhân Người dùng:

Người dùng phải đăng ký và đăng nhập thành công vào hệ thống, có quyền sử dụng các chức năng của hệ thống như:

Đăng nhập, đăng ký: Là chức năng bắt buộc để người dùng có thể sử dụng hệ thống.
Quản lý giống lươn: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách các thông tin của giống lươn đã nhập hàng.
Quản lý bể nuôi: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách các thông tin của bể nuôi.
Quản lý thức ăn: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách thông tin các loại thức ăn đã nhập vào kho thức ăn.
Quản lý thuốc: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách thông tin các loại thuốc đã nhập vào kho thuốc.
Quản lý môi trường: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách thông tin môi trường nuôi của các bể nuôi.
Quản lý nhật ký cho ăn: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách thông tin cho ăn của các bể nuôi.
Quản lý sức khỏe: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách thông tin sức khỏe lươn của các bể.
Quản lý xuất bán: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách thông tin xuất bán của các bể nuôi.
Quản lý chi phí vận hành: Cho phép người dùng thêm, xóa, thay đổi, xem danh sách thông tin các loại chi phí vận hành khác như tiền điện, nước, vận chuyển,..



Hình 3.1 Sơ đồ Use case



Hình 3.2 Sơ đồ ERD

3.4.2 Thiết kế API:

Bảng 3.11: API đăng ký đăng nhập:

Method	**Endpoint**	**Mô tả**
POST	/api/users/register	Đăng ký tài khoản người dùng
POST	/api/users/login	Đăng nhập hệ thống
GET	/api/users/profile	Lấy thông tin cá nhân

Bảng 3.12: API quản lý giống lươn:

Method	**Endpoint**	**Mô tả**
POST	/api/GiongLuon	Tạo lô giống mới
GET	/api/GiongLuon	Lấy danh sách tất cả lô giống
GET	/api/GiongLuon/{id}	Xem chi tiết 1 lô
PUT	/api/GiongLuon/{id}	Cập nhật thông tin lô
DELETE	/api/GiongLuon/{id}	Xóa lô giống

Bảng 3.13: API quản lý bể nuôi:

Method	**Endpoint**	**Mô tả**
POST	/api/tank	Tạo bể nuôi mới

GET	/api/ tank	Lấy danh sách tất cả bể nuôi
GET	/api/ tank /{id}	Xem chi tiết 1 bể
PUT	/api/ tank /{id}	Cập nhật thông tin bể
DELETE	/api/ tank /{id}	Xóa bể nuôi

Bảng 3.14: API quản lý thức ăn:

Method	**Endpoint**	**Mô tả**
POST	/api/ThucAn	Tạo thức ăn mới
GET	/api/ ThucAn	Lấy danh sách tất cả thức ăn
GET	/api/ ThucAn /{id}	Xem chi tiết 1 thức ăn
PUT	/api/ ThucAn /{id}	Cập nhật thông tin thức ăn
DELETE	/api/ ThucAn /{id}	Xóa thức ăn

Bảng 3.15: API quản lý thuốc:

Method	**Endpoint**	**Mô tả**
POST	/api/Thuoc	Tạo thuốc mới
GET	/api/ Thuoc	Lấy danh sách tất cả thuốc
GET	/api/ Thuoc /{id}	Xem chi tiết 1 loại thuốc
PUT	/api/ Thuoc /{id}	Cập nhật thông tin thuốc
DELETE	/api/ Thuoc /{id}	Xóa thuốc

Bảng 3.16: API quản lý môi trường:

Method	**Endpoint**	**Mô tả**
POST	/api/MoiTruong	Tạo môi trường mới
GET	/api/ MoiTruong	Lấy danh sách tất cả môi trường
GET	/api/ MoiTruong /{id}	Xem chi tiết 1 môi trường
PUT	/api/ MoiTruong /{id}	Cập nhật thông tin môi trường
DELETE	/api/ MoiTruong /{id}	Xóa môi trường

Bảng 3.17: API quản lý nhật ký cho ăn:

Method	**Endpoint**	**Mô tả**
POST	/api/NhatKyChoAn	Tạo nhật ký cho ăn
GET	/api/ NhatKyChoAn	Lấy danh sách tất cả nhật ký

		cho ăn
GET	/api/NhatKyChoAn /tank/{tankId}	Xem chi tiết 1 nhật ký cho ăn
PUT	/api/ NhatKyChoAn /{id}	Cập nhật thông tin nhật ký cho ăn
DELETE	/api/ NhatKyChoAn /{id}	Xóa nhật ký cho ăn

Bảng 3.18: API quản lý sức khỏe:

Method	**Endpoint**	**Mô tả**
POST	/api/SucKhoe	Tạo nhật ký sức khỏe
GET	/api/ SucKhoe	Lấy danh sách tất cả nhật ký sức khỏe
GET	/api/ SucKhoe /{id}	Xem chi tiết 1 nhật ký sức khỏe
PUT	/api/ SucKhoe /{id}	Cập nhật thông tin sức khỏe
DELETE	/api/ SucKhoe /{id}	Xóa nhật ký sức khỏe

Bảng 3.19: API quản lý xuất bán:

Method	**Endpoint**	**Mô tả**
POST	/api/XuatBan	Tạo xuất bán
GET	/api/ XuatBan	Lấy danh sách tất cả xuất bán
GET	/api/ XuatBan /tank/{tankId}	Xem chi tiết 1 xuất bán theo bể
PUT	/api/ XuatBan /{id}	Cập nhật xuất bán
DELETE	/api/ XuatBan /{id}	Xóa xuất bán

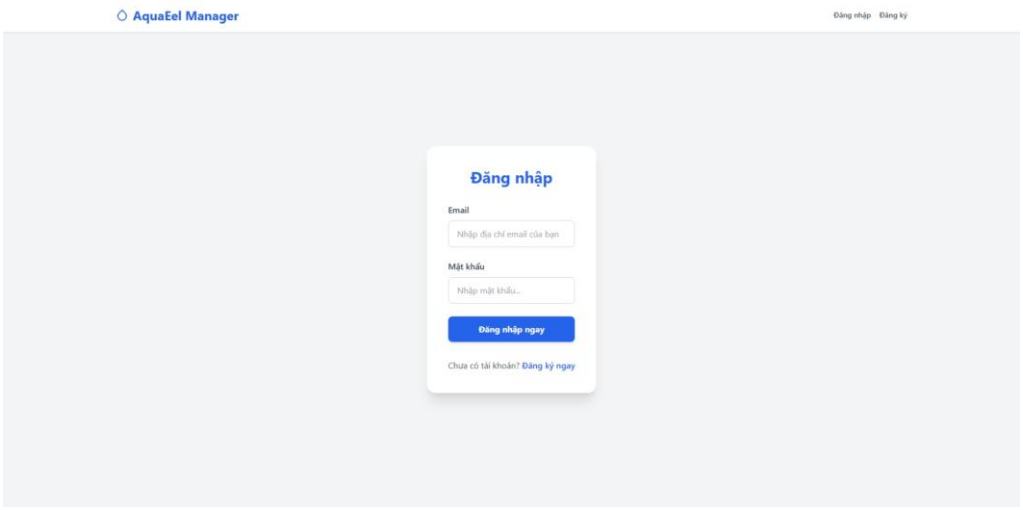
Bảng 3.20: API quản lý chi phí vận hành:

Method	**Endpoint**	**Mô tả**
POST	/api/ChiPhiVanHanh	Tạo chi phí vận hành
GET	/api/ ChiPhiVanHanh	Lấy danh sách tất cả chi phí vận hành
PUT	/api/ ChiPhiVanHanh /{id}	Cập nhật chi phí vận hành
DELETE	/api/ ChiPhiVanHanh /{id}	Xóa chi phí vận hành

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Giao diện trang đăng nhập:

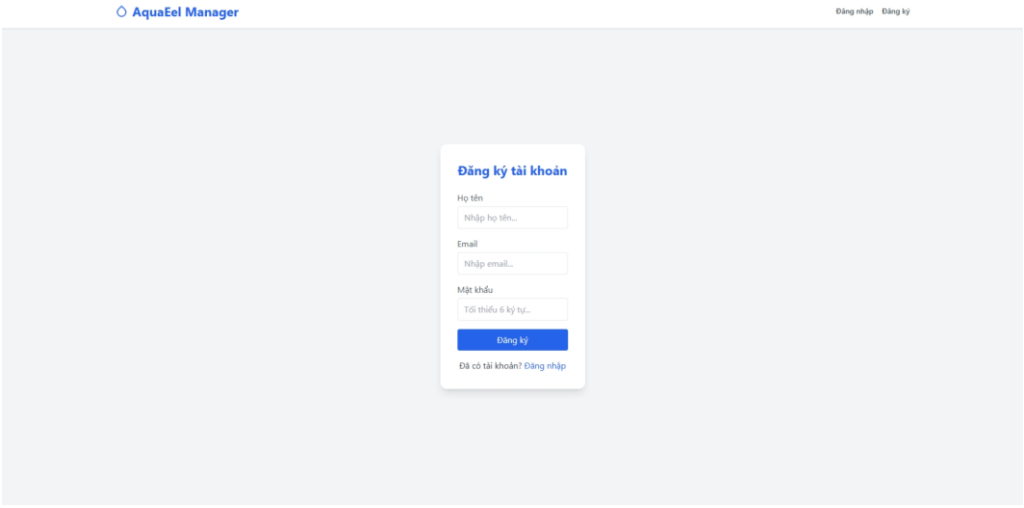
Tại giao diện này, người dùng cần dùng Email và mật khẩu để có thể đăng nhập vào tài khoản.



Hình 4.1: Giao diện trang đăng nhập

4.2. Giao diện đăng ký:

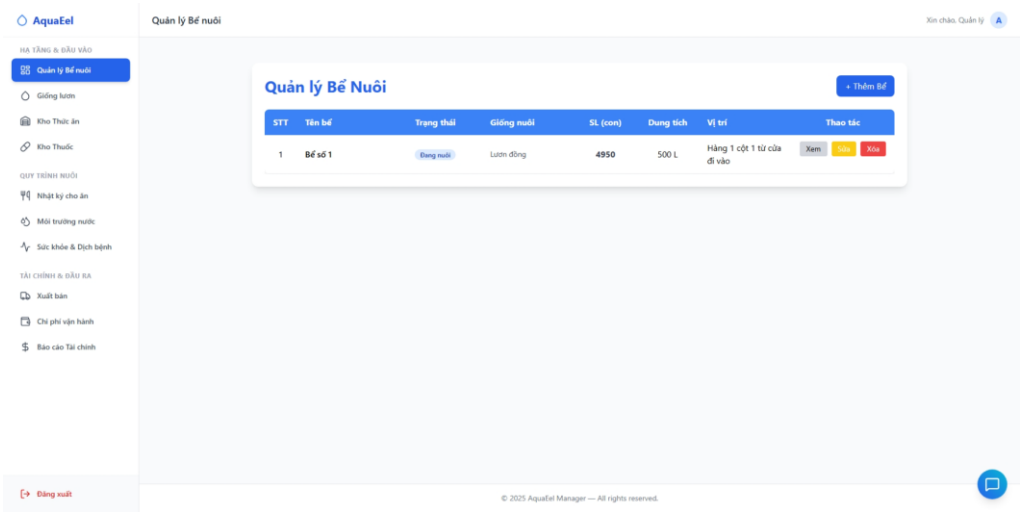
Tại giao diện này, người dùng cần cung cấp họ tên, email, mật khẩu để có thể đăng ký tài khoản.



Hình 4.2: Giao diện trang đăng ký

4.3. Giao diện quản lý bể nuôi:

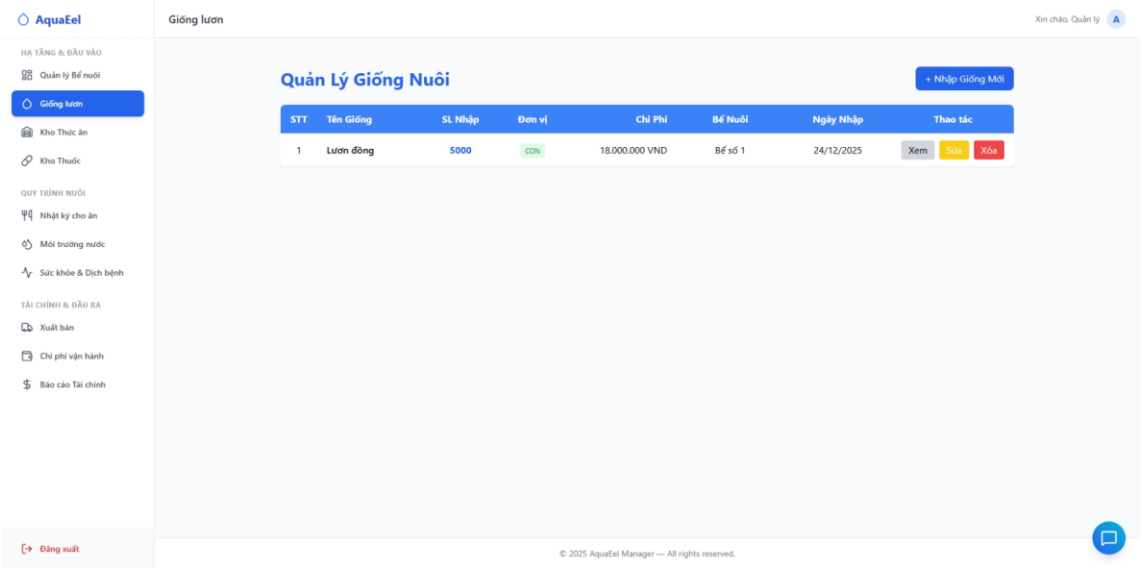
Ở giao diện này, người dùng có thể xem danh sách, có quyền thêm, xóa, sửa, xem chi tiết từng bể



Hình 4.3: Giao diện trang quản lý bể nuôi

4.4. Giao diện quản lý giống lươn:

Ở giao diện này, người quản trị có thể xem tất cả giống lươn, có quyền thêm, xóa, sửa, xem chi tiết từng giống lươn.



Hình 4.4: Giao diện trang quản lý giống lươn

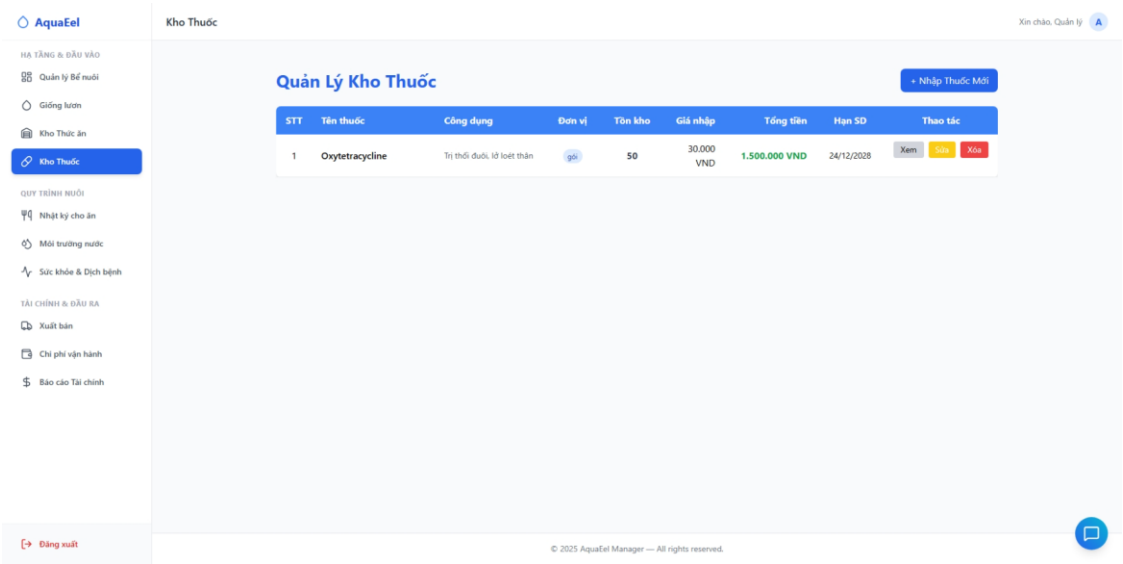
4.5. Giao diện quản lý kho thức ăn:

Tại giao diện này, người quản trị có thể xem tất cả thức ăn có trong kho, có quyền thêm, xóa, sửa, xem chi tiết từng thức ăn.

Hình 4.5: Giao diện trang quản lý kho thức ăn

4.6. Giao diện quản lý kho thuốc:

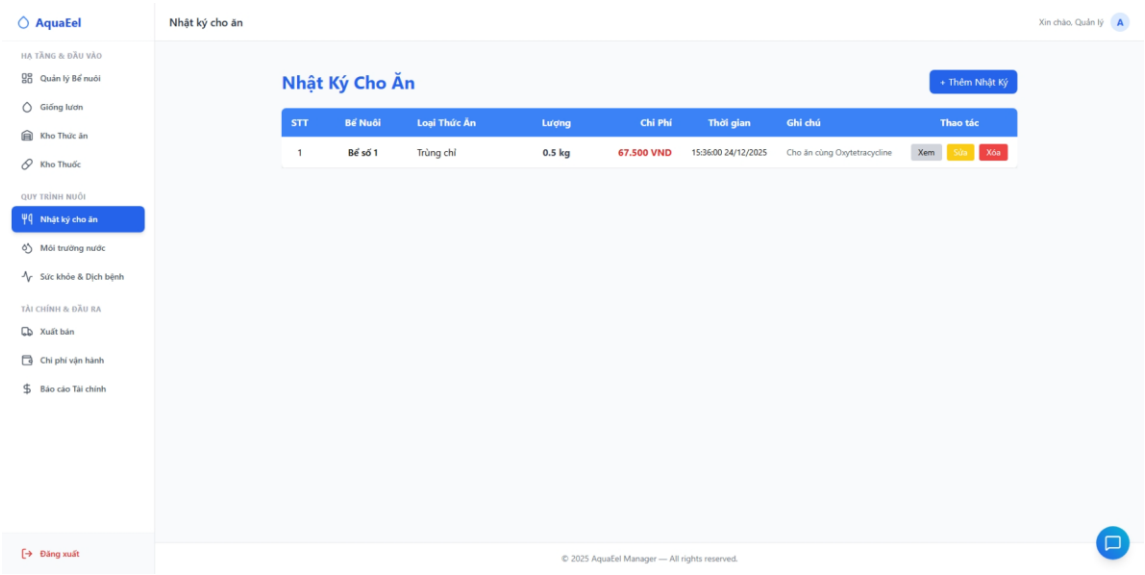
Ở giao diện này, người quản trị có thể xem tất cả loại thuốc, có quyền thêm, xóa, sửa, xem chi tiết từng loại thuốc.



Hình 4.6: Giao diện trang quản lý kho thuốc

4.7. Giao diện nhật ký cho ăn:

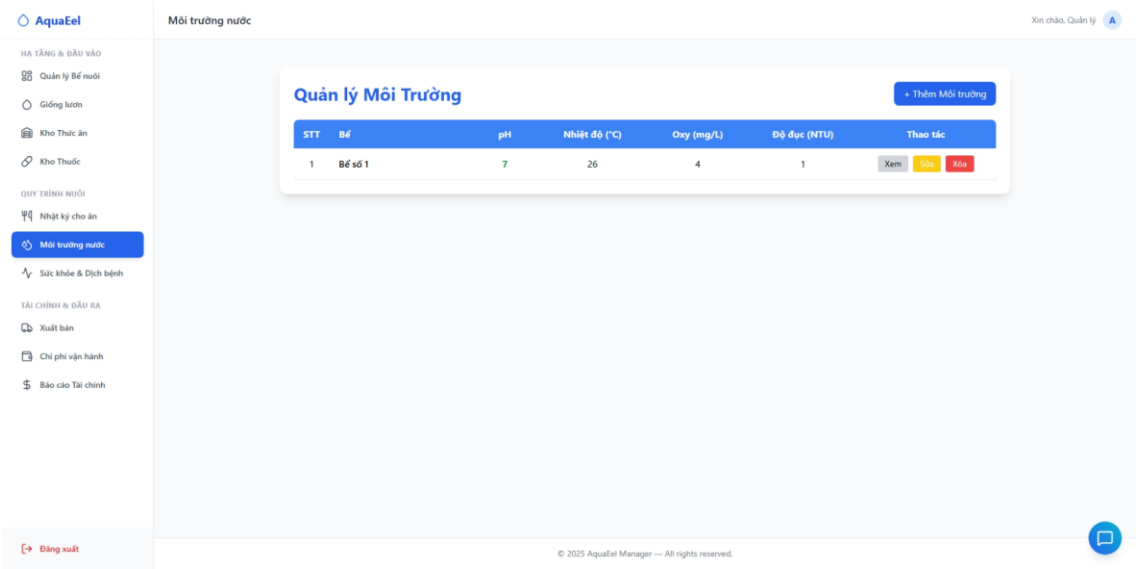
Ở giao diện này, người quản trị có thể xem tất cả các nhật ký cho ăn, có quyền thêm, xóa, sửa, xem chi tiết từng nhật ký cho ăn.



Hình 4.7: Giao diện trang nhật ký cho ăn.

4.8. Giao diện quản lý môi trường:

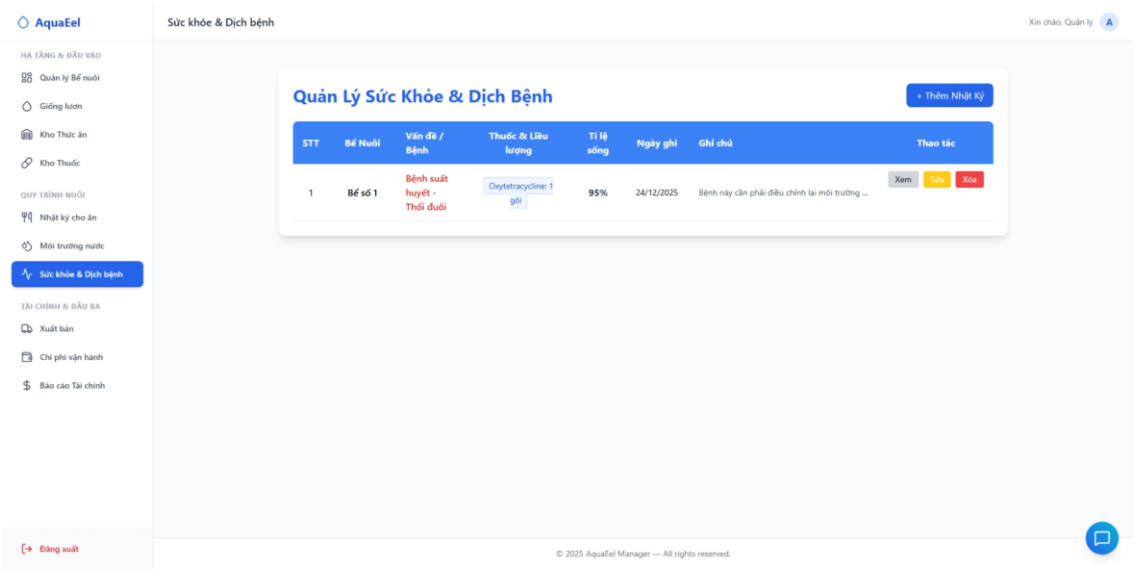
Ở giao diện này, người quản trị có thể xem tất cả các môi trường sống của lươn theo từng bể, có quyền thêm, xóa, sửa, xem chi tiết từng môi trường.



Hình 4.8: Giao diện trang quản lý môi trường

4.9. Giao diện sức khỏe:

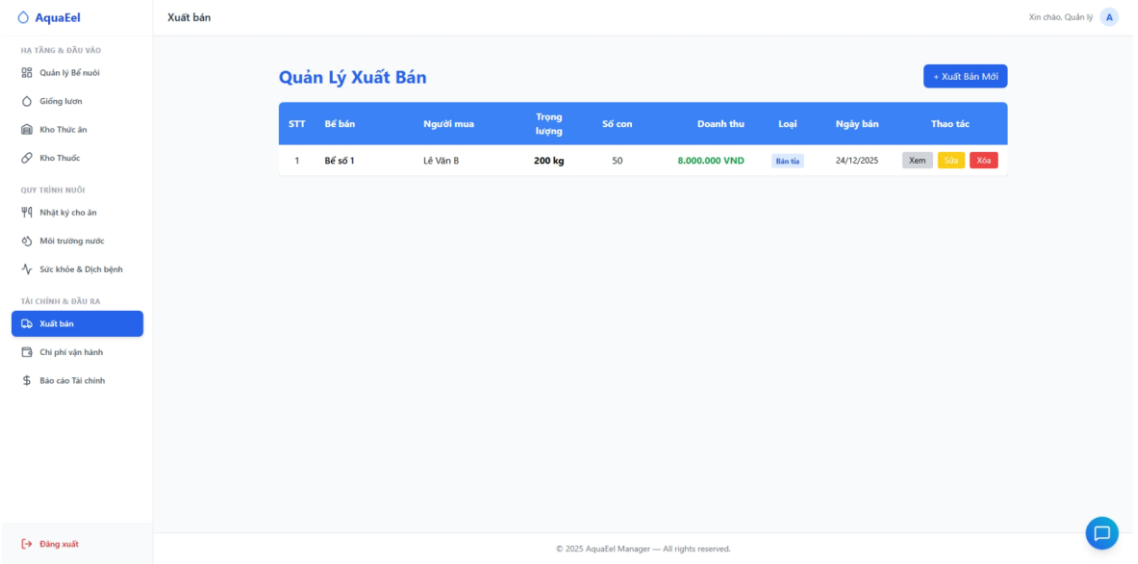
Ở giao diện này, người quản trị có thể xem tất cả nhật ký theo dõi sức khỏe của lươn theo từng bể, có quyền thêm, xóa, sửa, xem chi tiết từng nhật ký theo dõi sức khỏe của lươn.



Hình 4.9: Giao diện trang quản lý sức khỏe & dịch bệnh

4.10. Giao diện quản lý xuất bán:

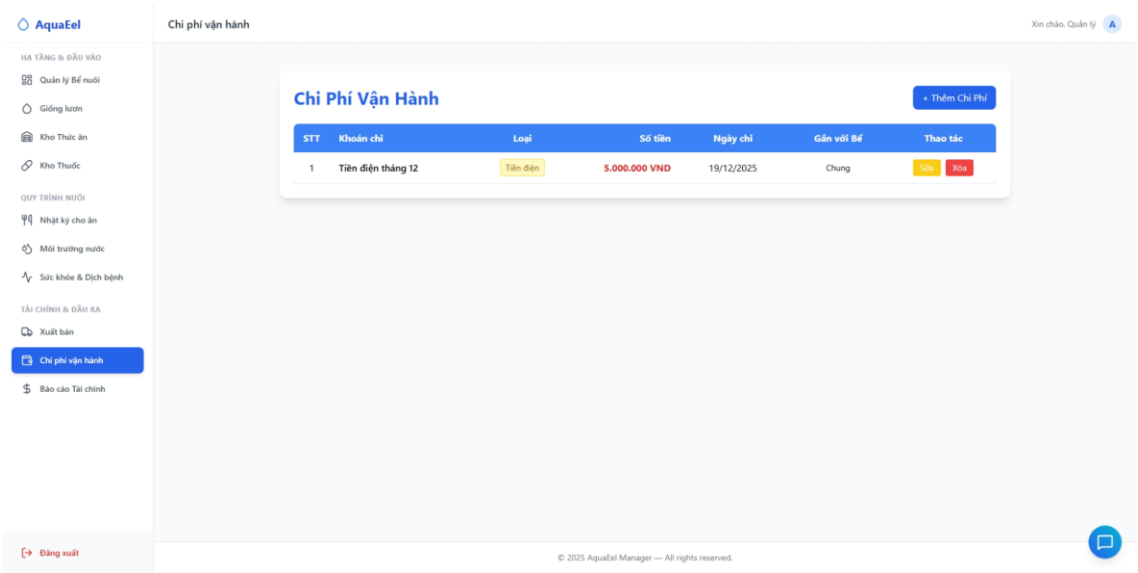
Ở giao diện này, người quản trị có thể xem tất cả xuất bán, có quyền thêm, xóa, sửa, xem chi tiết từng xuất bán.



Hình 4.10: Giao diện quản lý xuất bán

4.11. Giao diện quản lý chi phí vận hành:

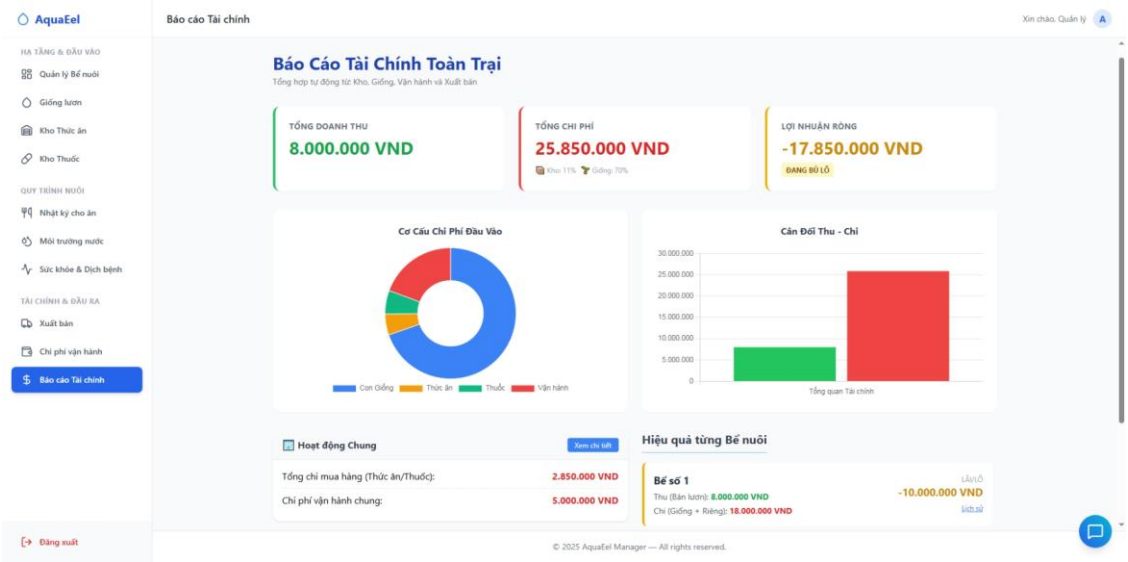
Ở giao diện này, người quản trị có thể xem tất cả chi phí vận hành khác như tiền điện, nước, vận chuyển,… Có quyền thêm, xóa, sửa, xem chi tiết từng chi phí vận hành.



Hình 4.11: Giao diện quản lý chi phí vận hành

4.12. Giao diện báo cáo tài chính:

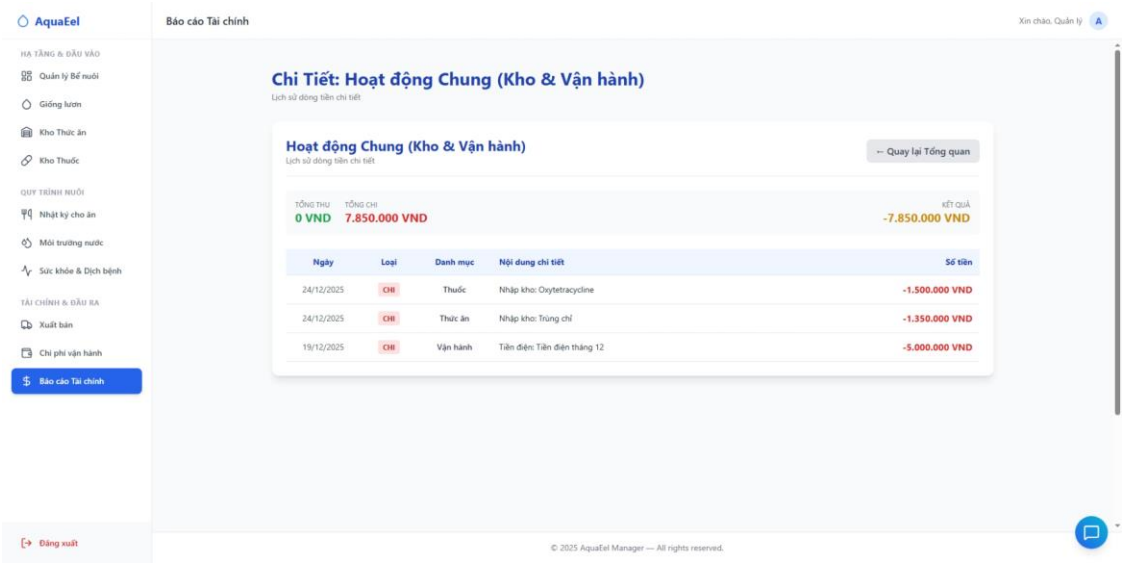
Ở giao diện này, người quản trị có thể xem tổng quan báo cáo thống kê của toàn trại, chi tiết theo từng bể, chi tiết theo chi phí vận hành. Từ đó có thể theo dõi được dòng tiền được lưu trữ trong hệ thống.



Hình 4.12: Giao diện trang báo cáo tài chính

4.13. Giao diện thống kê tài chính chi tiết theo hoạt động chung:

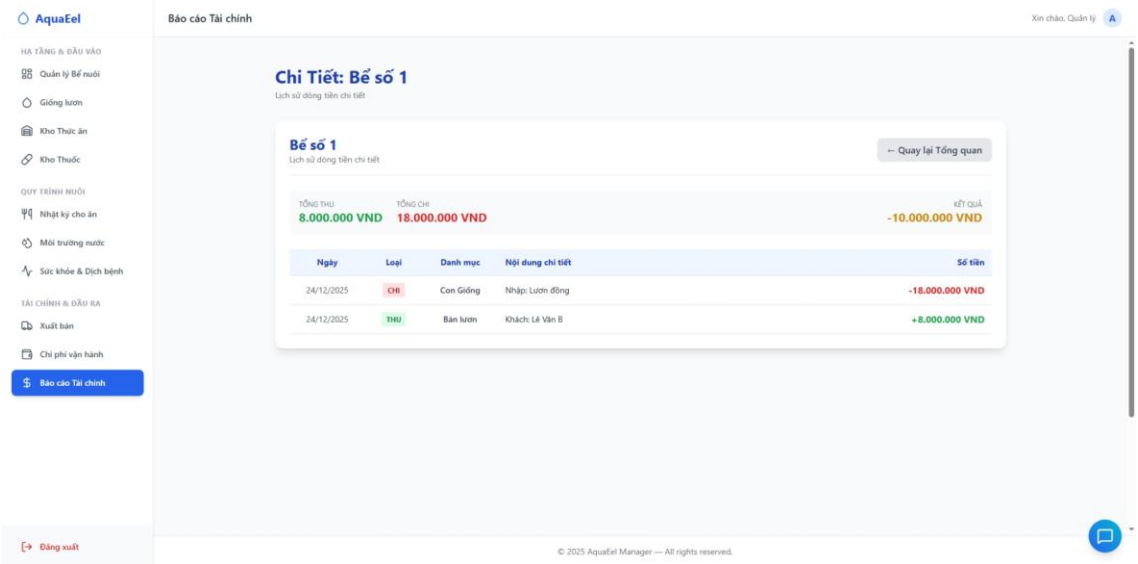
Tại giao diện này, người quản trị có thể xem chi tiết chi phí vận hành chung của trại.



Hình 4.13: Giao diện trang thống kê chi tiết theo hoạt động chung

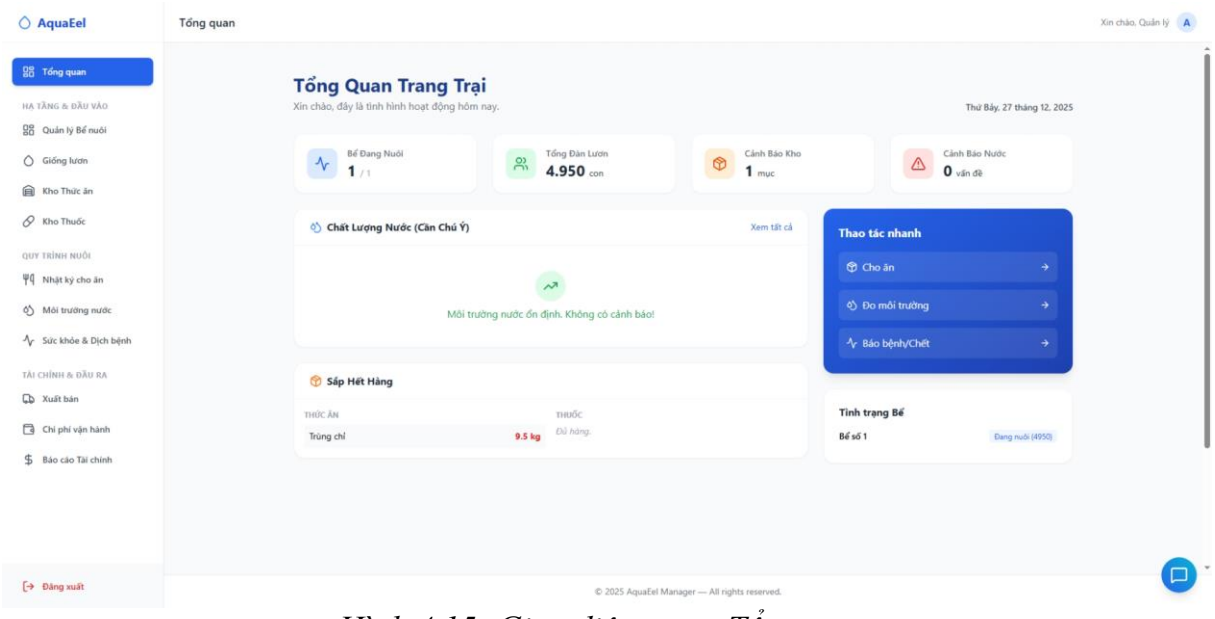
4.14. Giao diện thống kê tài chính chi tiết theo từng bể:

Tại giao diện này, người quản trị có thể xem báo cáo thống kê tài chính chi tiết theo từng bể.



Hình 4.14: Giao diện trang thống kê tài chính chi tiết theo từng bể

4.15. Giao diện tổng quan:



Hình 4.15: Giao diện trang Tổng quan

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được:

Sau quá trình nghiên cứu và thực hiện, đề tài "Xây dựng website quản lý trại chăn nuôi lươn" đã hoàn thành được các mục tiêu đề ra ban đầu, cụ thể:

- **Về mặt công nghệ:**
    - Xây dựng thành công hệ thống trên nền tảng **MERN Stack** (MongoDB, Express.js, React.js, Node.js), đảm bảo hiệu năng tốt và khả năng mở rộng.
    - Triển khai kiến trúc **RESTful API** chuẩn, giúp giao tiếp dữ liệu giữa Frontend và Backend mạch lạc, bảo mật thông qua cơ chế xác thực **JWT (JSON Web Token)**.
    - Tích hợp thành công **AI (Google Gemini)** để đóng vai trò trợ lý ảo, hỗ trợ tư vấn kỹ thuật nuôi lươn theo thời gian thực.
- **Về mặt chức năng và nghiệp vụ:**
    - **Quản lý hạ tầng & Đầu vào:** Đã số hóa toàn bộ quy trình nhập kho (Con giống, Thức ăn, Thuốc) và quản lý trạng thái Bể nuôi (Trống/Đang nuôi) một cách tự động.
    - **Quản lý vận hành:** Hệ thống cho phép ghi nhận chi tiết các hoạt động hàng ngày như: Nhật ký cho ăn (tự động trừ kho), Theo dõi chỉ số môi trường nước, và Nhật ký sức khỏe (tự động cập nhật số lượng hao hụt).
    - **Quản lý Tài chính & Báo cáo:** Đây là điểm mạnh của hệ thống với khả năng tự động tổng hợp dòng tiền từ tất cả các nguồn (Nhập giống, Mua vật tư, Chi phí vận hành, Xuất bán). Các biểu đồ (Chart.js) trực quan giúp người dùng nắm bắt nhanh tình hình lãi/lỗ của từng bể và toàn trại.
    - **Quy trình khép kín:** Đã giải quyết được bài toán logic phức tạp về sự liên kết dữ liệu giữa các bảng: *Nhập giống -> Kích hoạt Bể -> Chăm sóc -> Xuất bán -> Reset Bể -> Tính Lãi.*

5.2. Hạn chế:

Bên cạnh những kết quả đạt được, hệ thống vẫn còn một số hạn chế cần khắc phục trong tương lai:

Phụ thuộc vào nhập liệu thủ công: Các chỉ số môi trường (pH, nhiệt độ...) vẫn dựa vào việc người nuôi đo và nhập tay, dẫn đến độ trễ và nguy cơ sai sót số liệu.

Phân quyền chưa chuyên sâu: Hệ thống hiện tại mới chỉ tập trung vào quy mô vừa và nhỏ, nên phần phân quyền vẫn chưa được chuyên sâu.

5.3. Hướng phát triển:

Để hệ thống hoàn thiện và có tính ứng dụng cao hơn trong thực tế, các hướng phát triển tiếp theo được đề xuất bao gồm:

- **Tích hợp IoT (Internet of Things):** Kết nối trực tiếp với các cảm biến đo môi trường nước tự động. Khi chỉ số (pH, Oxy) vượt ngưỡng, hệ thống sẽ tự động cập nhật và gửi cảnh báo khẩn cấp (SMS/Zalo) cho người nuôi mà không cần nhập liệu.
- **Phát triển Mobile App:** Xây dựng phiên bản ứng dụng di động để tận dụng camera quét mã QR (truy xuất nguồn gốc thức ăn/thuốc).
- **Mở rộng quy mô hệ thống:** Nâng cấp hệ thống từ một website lưu trữ thông tin và thu thập kết quả, hệ thống sẽ lấy những kết quả thu thập được sẽ nâng cấp lên thành gợi ý các giống nuôi, cách nuôi, môi trường nuôi tạo ra lợi nhuận cao cho doanh nghiệp nói chung và người nông dân nói riêng.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] topdev.vn "Node.js là gì? Tổng hợp kiến thức NodeJS từ A-Z",[Node.js là gì? Tổng hợp kiến thức NodeJS thật chi tiết], truy cập ngày 29/11/2025.
[2] Viblo, "ReactJS là gì? Tổng quan về ReactJS,". Truy cập tại: https://viblo.asia/p/reactjs-tim-hieu-thong-qua-vi-du-gDVK2Oe2ZLj. Ngày truy cập: [12/11/2025].
[3] Viblo, "Tailwind CSS là gì? Giới thiệu và cách sử dụng Tailwind CSS,". Truy cập tại: https://viblo.asia/p/tailwind-css-la-gi-gioi-thieu-va-cach-su-dung-4dbZN8R4lYM. Ngày truy cập: [10/12/2025].
[4] Nguyễn Văn Tuấn, Lập trình Node.js cơ bản và nâng cao, Nhà xuất bản Thanh Niên, Hà Nội, 2021.

PHỤ LỤC