TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH

HỌC KỲ I, NĂM HỌC 2025-2026

Xây dựng website chia sẻ công thức nấu ăn

Giảng viên hướng dẫn:
ThS. Hà Thị Thúy Vi

Sinh viên thực hiện:
Họ tên: Nguyễn Thanh Duy
MSSV: 110122062
Lớp: DA22TTD

Vĩnh Long, tháng 12 năm 2025

TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH

HỌC KỲ I, NĂM HỌC 2025-2026

Xây dựng website chia sẻ công thức nấu ăn

Giảng viên hướng dẫn:
ThS. Hà Thị Thúy Vi

Sinh viên thực hiện:
Họ tên: Nguyễn Thanh Duy
MSSV: 110122062
Lớp: DA22TTD

Vĩnh Long, tháng 12 năm 2025

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

Trước khi đi sâu vào dự án, em xin gửi lời cảm ơn chân thành đến trường Đại học Trà Vinh đã tạo điều kiện cho em thực hiện dự án này, những cá nhân đã hỗ trợ và giúp đỡ em một cách tận tình. Cũng như là sự hỗ trợ và giúp đỡ của Cô Hà Thị Thúy Vi, người đã đóng vai trò quan trọng trong việc phát triển và hoàn thành dự án này.

Em rất cảm kích vì sự giúp đỡ của những người quanh em, những người đã dành thời gian, công sức và kiến thức của họ để giúp đỡ em. Các ý kiến đóng góp và sự hợp tác của mọi người là nguồn động lực giúp em phát triển bản thân.

Em rất quý trọng những người đã hỗ trợ và giúp đỡ em trong suốt thời gian qua, đặt biệt là sự giúp đỡ của Cô Hà Thị Thúy Vi, nhờ sscó sự giúp đỡ của Cô mà em mới có thể thực hiện và hoàn thành dự án. Một lần nữa, xin chân thành trường cảm ơn trường Đại học Trà Vinh và Cô Hà Thị Thúy Vi đã giúp đỡ và em mong rằng sẽ nhận được sự ủng hộ của mọi người trong những dự án sắp tới.

Trân trọng!

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN 1
1.1 Bối cảnh 1
1.2 Vấn đề đặt ra 1
1.3 Mục tiêu của hệ thống 1
1.4 Phạm vi nghiên cứu 2
1.5 Phương pháp và hướng tiếp cận 2
1.6 Kết quả mong đợi 2
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT 3
2.1 Kiến trúc và nguyên lý web hiện đại 3
2.2 Bảo mật và xác thực 3
2.2.1 JSON Web Token (JWT) 3
2.2.2 Bcrypt và bảo vệ mật khẩu 4
2.2.3 CORS (Cross-Origin Resource Sharing) 4
2.3 Công nghệ và kỹ thuật triển khai 4
2.3.1 Frontend: React, React Router, Axios, CSS Responsive 4
2.3.2 Backend: Node.js (Non-blocking I/O), Express Middleware 5
2.3.3 Dữ liệu và lưu trữ 5
2.4 Cấu trúc dữ liệu và thuật toán cho nested comments 5
2.4.1 Nested Comments (Bình luận lồng nhau) 5
2.4.2 Like System (Hệ thống thích) 6
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU 7
3.1 Đặc tả nhu cầu 7
3.2 Thiết kế hệ thống 8
3.2.1 Kiến trúc tổng thể 8
3.2.2 Mô hình dữ liệu 9
3.2.3 Thiết kế API (REST) 15
3.2.4 Thiết kế giao diện (UI/UX) 15
3.3 Hiện thực (Implementation) 15
3.3.1 Frontend (React) 15
3.3.2 Backend (Node.js + Express) 16
3.3.3 Database (MySQL) 19
3.3.4 Lưu trữ ảnh (Cloudinary) 19
3.4 Bảo mật và quản lý phiên 20
3.5 Thiết kế chi tiết nested comments và like system 20

3.5.1 Nested Comments 20
3.5.2 Like System 21
3.6 Kiểm thử 21
3.7 Đánh giá và ghi nhận 21
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU 22
4.1 Kết quả chức năng 22
4.2 Trải nghiệm người dùng (UX/UI) 24
4.3 Hiệu năng và kỹ thuật 24
4.4 Bảo mật 25
4.5 Giao diện minh họa (mô tả nhanh) 25
4.6 Đánh giá nhanh 25
4.7 Hướng phát triển 25
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 26
5.1 Kết luận 26
5.2 Hạn chế 26
5.2 Hướng phát triển 26
DANH MỤC TÀI LIỆU THAM KHẢO 27
PHỤ LỤC 28

DANH MỤC HÌNH ẢNH

Hình 3.1 Mô hình dữ liệu ERD....11
Hình 3.2 Mô hình cơ sở dữ liệu....12
Hình F.1 Trang đăng nhập và đăng ký....38
Hình F.2 Trang chủ với danh sách công thức....38
Hình F.3 Trang chủ với danh sách công thức....39
Hình F.4. Nested comments & Like button trên bình luận....39
Hình F.5 Trang Công thức của tôi....40
Hình F.6 Công thức đã lưu....40
Hình F.7 Form tạo công thức....41
Hình F.8: Trang quản trị (Admin Dashboard)....41
Hình F.9: Chế độ hạn chế dành cho Quản trị viên....42
Hình F.10: Trang cá nhân & Cài đặt tài khoảng....43
Hình F.11: Trang cá nhân & Cài đặt tài khoảng(Quản trị viên và Admin)....44
Hình F.11.1: Trang cá nhân & Cài đặt tài khoảng(User)....45
Hình F.12: Trang quản lý báo cáo bài viết dành cho Quản trị viên & Admin....45
Hình F.13 Tùy chỉnh giao diện....46

DANH MỤC BẢNG BIỂU

Bảng 2.1: Mã trạng thái HTTP 3
Bảng 2.2: So sánh phương pháp xác thực 4
Bảng 2.3: Công nghệ sử dụng 5
Bảng 3.1: Danh sách API endpoints 7
Bảng 3.2: Schema cơ sở dữ liệu 13
Bảng 3.3: Cấu trúc thư mục dữ liệu 17
Bảng 3.4: Phân quyền user vs admin 20
Bảng 4.1: Kết quả kiểm thử chức năng 22
Bảng 4.2: Đánh giá hiệu năng 24
Bảng PL.1: Môi trường phát triển 28

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

VẤN ĐỀ NGHIÊN CỨU

Trong thời đại số hóa, nhu cầu chia sẻ và trao đổi kiến thức về nấu ăn ngày càng tăng cao. Tuy nhiên, các nền tảng hiện tại thường thiếu tính tích hợp hoặc khó sử dụng. Đồ án này nhằm xây dựng một website chia sẻ công thức nấu ăn (CookShare) với đầy đủ chức năng:

- Cho phép người dùng tạo, chỉnh sửa, xóa công thức riêng của mình
- Hỗ trợ tương tác xã hội (bình luận, đánh giá, yêu thích)
- Cung cấp tìm kiếm và phân loại công thức hiệu quả
- Triển khai hệ thống quản lý admin để kiểm duyệt nội dung
- Tối ưu hóa giao diện đáp ứng (Responsive Design) trên mọi thiết bị

CÁC HƯỚNG TIẾP CẬN

1. Kiến trúc 3 tầng (Three-Layer Architecture):
   - Tầng Giao diện (Presentation Layer): React 19.x với Component-based architecture
   - Tầng Ứng dụng (Application Layer): Node.js + Express 4.x cung cấp REST API
   - Tầng Dữ liệu (Data Layer): MySQL 8.0+ lưu trữ dữ liệu quan hệ
2. Mô hình MVC (Model-View-Controller):
   - Model: Cơ sở dữ liệu MySQL (users, recipes, comments, ratings, favorites)
   - View: React components (Home, CreateRecipe, RecipeDetail, AdminDashboard, v.v.)
   - Controller: Express routes (auth.js, recipe.js, rating.js, favorite.js, admin.js)
3. Bảo mật & Xác thực:
   - Dùng Bcrypt (với Salt & Adaptive hashing) để mã hóa mật khẩu
   - Dùng JSON Web Token (JWT) để quản lý phiên người dùng (token hết hạn sau 7 ngày)
   - Cấu hình CORS để kiểm soát truy cập giữa frontend (localhost:3000) và backend (localhost:3001)
   - Middleware xác thực để kiểm tra quyền (user/admin)
4. Tối ưu hóa & Hiệu suất:
   - Non-blocking I/O: Node.js xử lý nhiều request đồng thời mà không chặn
   - Cloudinary: Lưu trữ ảnh trên đám mây, giảm tải server
   - CSS Responsive: Giao diện tự động thích ứng với mọi kích thước màn hình

CÁCH GIẢI QUYẾT VẤN ĐỀ

Phía Backend (Node.js + Express):

1. Xây dựng REST API với các endpoint:
   - /auth/register, /auth/login → quản lý xác thực
   - /recipe/list, /recipe/search, /recipe/create, /recipe/update/:id, /recipe/delete/:id → quản lý công thức
   - /rating, /favorite, /comment → quản lý tương tác
   - /admin → quản lý hệ thống
2. Sử dụng Middleware xác thực JWT để bảo vệ các route yêu cầu đăng nhập
3. Cấu hình CORS để cho phép frontend truy cập API
4. Kết nối MySQL, thực hiện CRUD (Create, Read, Update, Delete) trên cơ sở dữ liệu

Phía Frontend (React):

1. Xây dựng Component-based UI:
   - Navbar (điều hướng, đăng nhập/đăng xuất)
   - Home (danh sách công thức, thanh tìm kiếm)
   - CreateRecipe, RecipeDetail, MyRecipes, FavoriteRecipes (quản lý công thức)
   - AdminDashboard (quản lý người dùng, nội dung)
2. Dùng React Router để điều hướng giữa các trang
3. Dùng Axios gọi API backend
4. Lưu JWT token trong localStorage, gửi lại trong header Authorization cho các API yêu cầu xác thực
5. Thiết kế CSS responsive với media queries

Cơ sở dữ liệu:

1. Thiết kế schema gồm 5 bảng chính:
   - users (id, email, password, username, role, created_at)
   - recipes (id, user_id, title, ingredients, steps, image_url, created_at, updated_at)
   - comments (id, recipe_id, user_id, content, created_at)
   - ratings (id, recipe_id, user_id, rating, created_at)
   - favorites (id, recipe_id, user_id, created_at)
2. Thiết lập khóa ngoài (Foreign Key) để đảm bảo tính toàn vẹn dữ liệu

KẾT QUẢ ĐẠT ĐƯỢC

1. Chức năng Xác thực & Quản lý Người dùng:
   - Đăng ký tài khoản với xác thực email, mật khẩu
   - Đăng nhập với JWT token (hết hạn 7 ngày)
   - Hồ sơ người dùng, quên mật khẩu (gửi OTP qua email)
   - Phân biệt vai trò: user vs admin

2. Chức năng Quản lý Công Thức:
   - Tạo công thức (tiêu đề, nguyên liệu, bước, tải lên ảnh)
   - Danh sách công thức với phân trang
   - Xem chi tiết công thức
   - Chỉnh sửa công thức (chỉ chủ sở hữu)
   - Xóa công thức (chủ sở hữu hoặc admin)
   - Tìm kiếm công thức theo từ khóa

3. Chức năng Tương tác & Bình luận:
   - Bình luận trên công thức (tạo, xem, xóa của chủ sở hữu)
   - Đánh giá sao (1-5 sao, mỗi user chỉ được đánh giá 1 lần)
   - Yêu thích công thức (lưu vào danh sách yêu thích)
   - Thống kê đánh giá (trung bình rating, số lượng bình luận)

4. Chức năng Quản lý Admin:
   - Xem danh sách tất cả người dùng
   - Thay đổi vai trò người dùng (user ↔ admin)
   - Xóa người dùng hoặc công thức vi phạm

5. Giao diện & Trải nghiệm Người dùng:
   - Giao diện đáp ứng (Responsive Design) trên desktop, tablet, mobile
   - Thanh điều hướng (Navbar) với tính năng đăng nhập/đăng xuất
   - Trang chủ với danh sách công thức, thanh tìm kiếm
   - Bố cục lưới (grid layout) thích hợp cho việc hiển thị công thức
   - Trang quản lý công thức của bản thân (My Recipes)
   - Trang danh sách yêu thích (Favorite Recipes)

6. Bảo mật & Hiệu suất:
    - Mật khẩu được mã hóa bằng Bcrypt (adaptive hashing + salt)
    - JWT token với hết hạn 7 ngày
    - Middleware xác thực kiểm tra quyền trước khi cho phép truy cập
    - CORS cấu hình để chỉ cho phép frontend từ localhost:3000 truy cập
    - Ảnh được lưu trên Cloudinary (giảm tải server)

ĐÁNH GIÁ & HẠN CHẾ

1. Điểm mạnh:
    - Kiến trúc module hóa, dễ mở rộng
    - Bảo mật tốt (Bcrypt, JWT, CORS, Middleware)
    - Giao diện responsive, thân thiện với người dùng
    - Non-blocking I/O giúp xử lý nhiều request hiệu quả

2. Hạn chế & hướng phát triển trong tương lai:
    - Chưa triển khai Redis để cache dữ liệu thường xuyên truy cập
    - Chưa có hệ thống notification real-time (có thể dùng WebSocket)
    - Chưa triển khai unit test toàn diện
    - Chưa có CI/CD pipeline để tự động test & deploy
    - Có thể thêm Machine Learning để gợi ý công thức dựa trên sở thích người dùng

MỞ ĐẦU

1. Lý do chọn đề tài

Xu hướng số hóa trong lĩnh vực ẩm thực ngày càng mạnh, nhu cầu chia sẻ và tìm kiếm công thức nấu ăn trực tuyến rất lớn.

Các nền tảng hiện có thường phân tán, thiếu quản lý chất lượng nội dung hoặc chưa tối ưu trải nghiệm người dùng (UX/UI) trên đa thiết bị.

Mong muốn xây dựng một hệ thống tập trung, thân thiện, cho phép người dùng dễ dàng đăng tải, tìm kiếm, tương tác và quản trị nội dung công thức nấu ăn.

2. Mục đích nghiên cứu

Xây dựng website CookShare đáp ứng các chức năng cốt lõi: đăng ký/đăng nhập, tạo–chỉnh sửa–xóa công thức, tìm kiếm, đánh giá, bình luận, yêu thích.

Đảm bảo bảo mật dữ liệu người dùng (mã hóa mật khẩu, xác thực bằng JWT, kiểm soát quyền truy cập).

Thiết kế giao diện đáp ứng (responsive) cho cả desktop và mobile, tối ưu tốc độ và trải nghiệm.

Đề xuất kiến trúc mở rộng, dễ bảo trì (Node.js + Express REST API, React frontend, MySQL, lưu trữ ảnh cloud).

3. Đối tượng nghiên cứu

Người dùng cuối: cá nhân đam mê nấu ăn, muốn chia sẻ hoặc tìm kiếm công thức.

Nhà phát triển/nhóm vận hành: quản trị hệ thống, kiểm duyệt nội dung, mở rộng tính năng.

Công nghệ và giải pháp: React (UI), React Router, Axios, CSS responsive; Node.js/Express (API), JWT, Bcrypt; MySQL; Cloudinary (lưu trữ ảnh).

4. Phạm vi nghiên cứu

Chức năng người dùng: đăng ký, đăng nhập, quản lý công thức (tạo/sửa/xóa), bình luận, đánh giá, yêu thích, tìm kiếm.

Chức năng quản trị: quản lý người dùng, vai trò (user/admin), kiểm duyệt/xóa nội dung vi phạm.

Hạ tầng: chạy cục bộ trên môi trường phát triển (frontend: localhost:3000, backend: localhost:3001); lưu trữ ảnh trên dịch vụ đám mây.

Ngoài phạm vi: chưa triển khai realtime notifications, chưa có hệ thống khuyến nghị bằng ML, chưa triển khai CI/CD tự động và cache/Redis (đề xuất cho tương lai).

CHƯƠNG 1: TỔNG QUAN

1.1 Bối cảnh

- Xu hướng tìm kiếm và chia sẻ công thức nấu ăn trực tuyến tăng mạnh cùng với sự phổ biến của mạng xã hội và thiết bị di động.
- Người dùng cần một nền tảng tập trung, dễ dùng, tin cậy để:
  + Đăng tải, lưu trữ, chỉnh sửa công thức cá nhân.
  + Tìm kiếm, học hỏi từ cộng đồng.
  + Tương tác (bình luận, đánh giá, yêu thích) và được kiểm duyệt nội dung.

1.2 Vấn đề đặt ra

- Nội dung ẩm thực hiện có phân tán, chất lượng không đồng đều, thiếu kiểm duyệt.
- Trải nghiệm người dùng chưa tối ưu trên nhiều thiết bị (desktop, tablet, mobile).
- Bảo mật và quyền riêng tư (mật khẩu, phiên đăng nhập, phân quyền) chưa luôn được đảm bảo.
- Thiếu cơ chế quản trị/admin để xử lý nội dung vi phạm và quản lý người dùng.

1.3 Mục tiêu của hệ thống

- Xây dựng website CookShare với các chức năng cốt lõi:
  + Đăng ký/đăng nhập, xác thực an toàn (JWT), mã hóa mật khẩu (Bcrypt).
  + Quản lý công thức: tạo, chỉnh sửa, xóa, xem chi tiết, tìm kiếm.
  + Tương tác: bình luận, đánh giá sao, đánh dấu yêu thích.
  + Quản trị: phân quyền user/admin, kiểm duyệt nội dung, xóa tài khoản/công thức vi phạm.
- Đảm bảo trải nghiệm tốt trên đa thiết bị (responsive design).
- Lưu trữ ảnh trên đám mây, giảm tải cho server.
- Tương tác nâng cao: bình luận lồng nhau (nested comments), cho phép user trả lời trực tiếp bình luận của người khác.
- Thích bình luận (like system): user có thể thích bình luận, xem số lượt thích, tạo sự tương tác xã hội tích cực.

1.4 Phạm vi nghiên cứu

- Chức năng người dùng: quản lý tài khoản, công thức, tương tác.
- Chức năng quản trị: duyệt/xóa nội dung, thay đổi vai trò.
- Hạ tầng: frontend React (localhost:3000), backend Node/Express (localhost:3001), MySQL, Cloudinary cho ảnh.
- Ngoài phạm vi hiện tại: realtime notification, gợi ý cá nhân hóa bằng ML, cache/Redis, CI/CD tự động (đề xuất cho tương lai).
- Nested comments: bình luận trả lời, cấu trúc cây (tree structure).
- Like system: toggle like/unlike bình luận, hiển thị số lượt thích.

1.5 Phương pháp và hướng tiếp cận

- Kiến trúc 3 tầng: Giao diện (React) – Ứng dụng (Express REST API) – Dữ liệu (MySQL).
- Mô hình MVC trên backend: Model (MySQL), Controller (routes Express), View (JSON responses).
- Bảo mật: Bcrypt (Salt + adaptive hashing), JWT (phiên đăng nhập 7 ngày), CORS cấu hình nguồn gốc.
- Hiệu năng: Non-blocking I/O của Node.js, tách lưu trữ ảnh sang Cloudinary.

1.6 Kết quả mong đợi

- Một nền tảng web hoàn chỉnh cho chia sẻ công thức nấu ăn:
  + Người dùng đăng nhập, tạo và quản lý công thức, tương tác an toàn.
  + Quản trị viên có công cụ kiểm duyệt, quản lý người dùng/nội dung.
  + Giao diện thân thiện, phản hồi nhanh, chạy tốt trên nhiều thiết bị.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Kiến trúc và nguyên lý web hiện đại

Kiến trúc 3 tầng (Three-Layer Architecture)

- **Presentation:** React hiển thị UI, quản lý trạng thái client, gọi API.
- **Application:** Node.js/Express xử lý logic nghiệp vụ, định tuyến REST.
- **Data:** MySQL lưu trữ quan hệ (users, recipes, comments, ratings, favorites).

Mô hình RESTful API

- Tài nguyên được ánh xạ vào URL; thao tác CRUD qua HTTP methods (GET/POST/PUT/DELETE).
- Chuẩn hóa mã trạng thái HTTP (200/201/400/401/403/404/500).
- Phản hồi dạng JSON; stateless giữa các request.

Mô hình MVC trên backend Express

- **Model:** Tầng truy cập dữ liệu MySQL.
- **View:** JSON responses (thay cho HTML view truyền thống).
- **Controller:** Router/handler tổ chức logic và gọi Model.

Bảng 2.1: Mã trạng thái HTTP

Mã	Tên	Khi sử dụng	Ví dụ
200	OK	Request thành công	GET /recipe/list → trả danh sách
201	Created	Tạo tài nguyên mới	POST /auth/register → tạo user
400	Bad Request	Lỗi validation từ client	Thiếu trường email khi đăng ký
401	Unauthorized	Chưa xác thực/token sai	Không gửi JWT hoặc token hết hạn
403	Forbidden	Không đủ quyền	User thường gọi API admin
404	Not Found	Tài nguyên không tồn tại	GET /recipe/9999 (ID không có)
500	Internal Server Error	Lỗi server	Exception, lỗi database

2.2 Bảo mật và xác thực

2.2.1 JSON Web Token (JWT)

- Cấu trúc 3 phần: Header (thuật toán, kiểu token), Payload (claims: id, email, role, exp), Signature (chữ ký bảo toàn toàn vẹn).
- Quy trình: đăng nhập → sign token (exp 7 ngày) → client gửi **Authorization: Bearer <token>** → middleware verify.
- Ưu điểm: stateless, dễ mở rộng microservice.

Bảng 2.2: So sánh phương pháp xác thực

Tiêu chí	Session-based	JWT (Token-based)
Lưu trữ server	Có (session store)	Không (stateless)
Scalability	Khó (cần shared session)	Dễ (mỗi server verify độc lập)
Băng thông	Nhẹ (chỉ session ID)	Nặng hơn (toàn bộ token)
Bảo mật	Phụ thuộc session store	Token có thể bị đánh cắp
Hết hạn	Server kiểm soát	Client giữ đến khi exp
Phù hợp	Web truyền thống	API, microservices, mobile

2.2.2 Bcrypt và bảo vệ mật khẩu

- **Adaptive hashing:** điều chỉnh cost theo thời gian.
- **Salt:** chuỗi ngẫu nhiên gắn vào mật khẩu trước khi hash, chống rainbow table.
- **One-way:** không thể giải ngược hash; so sánh bằng **bcrypt.compare**.

2.2.3 CORS (Cross-Origin Resource Sharing)

- Kiểm soát truy cập giữa các origin (frontend http://localhost:3000 ↔ backend http://localhost:3001).
- Preflight (OPTIONS) kiểm tra Access-Control-Allow-Origin/ Methods/ Headers/Credentials.
- Cho phép gửi JWT qua header khi được cấu hình origin và credentials.

2.3 Công nghệ và kỹ thuật triển khai

2.3.1 Frontend: React, React Router, Axios, CSS Responsive

- React: Virtual DOM, component-based, state/props để quản lý UI.
- React Router: điều hướng client-side, bảo vệ tuyến qua ProtectedRoute.
- Axios: HTTP client với interceptor cho JWT.
- CSS Responsive: media queries, grid/flex để thích ứng đa màn hình.

Bảng 2.3: Công nghệ sử dụng

Thành phần	Công nghệ	Phiên bản	Vai trò
Frontend	React	19.x	Xây dựng UI component-based, Virtual DOM
	React Router	6.x	Điều hướng client-side, ProtectedRoute
	Axios	1.x	HTTP client, gọi API backend
	CSS3	-	Responsive design, layout
Backend	Node.js	16.x+	JavaScript runtime, non-blocking I/O
	Express	4.x	Web framework, REST API, middleware
	jsonwebtoken	9.x	Tạo và verify JWT token
	bcryptjs	2.x	Hash mật khẩu với Salt
Database	MySQL	8.0+	Cơ sở dữ liệu quan hệ, ACID
Cloud Storage	Cloudinary	-	Lưu trữ và CDN hóa ảnh
Tools	npm	-	Quản lý package dependencies

2.3.2 Backend: Node.js (Non-blocking I/O), Express Middleware

- Non-blocking I/O: event loop xử lý nhiều request đồng thời.
- Middleware: xử lý tuần tự (logging, CORS, body parsing, auth) trước handler.
- Validation: kiểm tra đầu vào, trả mã lỗi phù hợp.

2.3.3 Dữ liệu và lưu trữ

- MySQL: quan hệ, ACID; khóa ngoại giữa users, recipes, comments, ratings, favorites.
- Cloudinary: lưu trữ ảnh trên cloud, trả URL tối ưu để hiển thị trên frontend.

2.4 Cấu trúc dữ liệu và thuật toán cho nested comments

2.4.1 Nested Comments (Bình luận lồng nhau)

- Cách triển khai: Sử dụng cột `parent_comment_id` trong bảng `comments` để tạo liên kết cha-con.
- Lợi ích: Người dùng có thể trả lời trực tiếp bình luận của người khác; tổ chức cuộc trò chuyện theo chủ đề.
- Truy vấn recursion: Backend truy vấn theo `parent_comment_id = NULL` (top-level), sau đó lấy nested với `parent_comment_id = comment_id`.
- Frontend: Sử dụng recursive component (CommentItem gọi lại chính nó) để render tree structure.

2.4.2 Like System (Hệ thống thích)

- Bảng riêng `comment_likes` lưu trữ: `comment_id, user_id, created_at`.
- Ràng buộc UNIQUE: Mỗi user chỉ thích 1 lần mỗi comment (toggle like/unlike).
- Toggle endpoint: POST `/comment/:id/like` trả về `{liked: true/false}`.
- Query optimization: Include `like_count, user_liked` khi fetch comments.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Đặc tả nhu cầu

- **Tác nhân:**
  + Người dùng (user): đăng ký/đăng nhập, tạo–sửa–xóa công thức của mình, xem/tìm kiếm, bình luận, đánh giá, yêu thích.
  + Quản trị viên (admin): toàn bộ quyền của user + duyệt/xóa công thức vi phạm, đổi vai trò người dùng, xóa tài khoản.
- **Chức năng chính:**
  + Xác thực & phân quyền: đăng ký, đăng nhập, JWT 7 ngày, phân biệt user/admin.
  + Quản lý công thức: CRUD, tải ảnh lên Cloudinary, tìm kiếm theo tiêu đề/nguyên liệu.
  + Tương tác: bình luận, đánh giá sao (1 user/1 recipe/1 rating), đánh dấu yêu thích.
  + Quản trị: quản lý người dùng, thay đổi role, xóa nội dung vi phạm.
- **Phi chức năng:**
  + Bảo mật: Bcrypt (Salt + adaptive hashing), JWT, CORS giới hạn origin.
  + Hiệu năng: Non-blocking I/O, tách lưu trữ ảnh sang cloud.
  + UX/UI: Responsive, điều hướng mượt (React Router), thông báo lỗi/thành công rõ ràng.

Bảng 3.1: Danh sách API endpoints

Method	Endpoint	Auth	Mô tả	Response
POST	/auth/register	No	Đăng ký tài khoản	201: success / 400: validation error
POST	/auth/login	No	Đăng nhập, nhận JWT	200: {token, user} / 400: invalid
GET	/recipe/list	No	Danh sách công thức	200: [recipes]
GET	/recipe/search?q=	No	Tìm kiếm công thức	200: [filtered recipes]
POST	/recipe/create	JWT	Tạo công thức + upload ảnh	201: {recipeId} / 401: no token

PUT	/recipe/update/:id	JWT	Cập nhật công thức	200: success / 403: not owner
DELETE	/recipe/delete/:id	JWT	Xóa công thức	200: deleted / 403: not owner/admin
POST	/rating/:recipeId	JWT	Đánh giá sao	201: rated / 400: already rated
POST	/favorite/:recipeId	JWT	Thêm yêu thích	201: added
POST	/comment/:recipeId	JWT	Thêm bình luận	201: created
GET	/admin/users	JWT (admin)	Danh sách người dùng	200: [users] / 403: not admin
PUT	/admin/users/:id/role	JWT (admin)	Đổi vai trò user	200: updated / 403: not admin
POST	/comment/:recipeId	JWT	Thêm bình luận (hỗ trợ parent_comment_id)	201: created / 400: validation
GET	/recipe/comment/:id	JWT	Lấy danh sách bình luận + replies (nested)	200: [comments with replies]
POST	/comment/:id/like	JWT	Toggle like/unlike bình luận	200: {liked: true/false}
DELETE	/comment/:id	JWT	Xóa bình luận (owner/admin)	200: deleted / 403: forbidden
GET	/notifications	JWT	Lấy danh sách thông báo (bình luận mới, like, reply)	200: [notifications]

3.2 Thiết kế hệ thống

3.2.1 Kiến trúc tổng thể

- **tầng:** Frontend (React) ↔ Backend (Express REST) ↔ Database (MySQL); ảnh trên Cloudinary.
- **MVC (backend):** Model (MySQL queries), Controller (Express routes), View (JSON response).
- **Giao tiếp:** HTTP/HTTPS, JSON; JWT gửi qua header Authorization.

3.2.2 Mô hình dữ liệu

Cấu trúc bản:

- **Bảng users**
  - + id INT PK AUTO_INCREMENT
  - + username VARCHAR(50) UNIQUE NOT NULL
  - + email VARCHAR(100) UNIQUE NOT NULL
  - + password VARCHAR(255) NOT NULL
  - + role ENUM('user','admin') DEFAULT 'user'
  - + created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

- **Bảng recipes**
  - + id INT PK AUTO_INCREMENT
  - + user_id INT NOT NULL FK → users(id) ON DELETE CASCADE
  - + title VARCHAR(200) NOT NULL
  - + ingredients TEXT NOT NULL
  - + steps TEXT NOT NULL
  - + image_url VARCHAR(500) NULL
  - + created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  - + updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

- **Bảng comments (có nested)**
  - + id INT PK AUTO_INCREMENT
  - + recipe_id INT NOT NULL FK → recipes(id) ON DELETE CASCADE
  - + user_id INT NOT NULL FK → users(id) ON DELETE CASCADE
  - + content TEXT NOT NULL
  - + parent_comment_id INT NULL FK → comments(id) ON DELETE CASCADE
  - + created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

- **Bảng ratings**
  - + id INT PK AUTO_INCREMENT
  - + recipe_id INT NOT NULL FK → recipes(id) ON DELETE CASCADE
  - + user_id INT NOT NULL FK → users(id) ON DELETE CASCADE
  - + rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5)
  - + created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  - + UNIQUE (recipe_id, user_id)

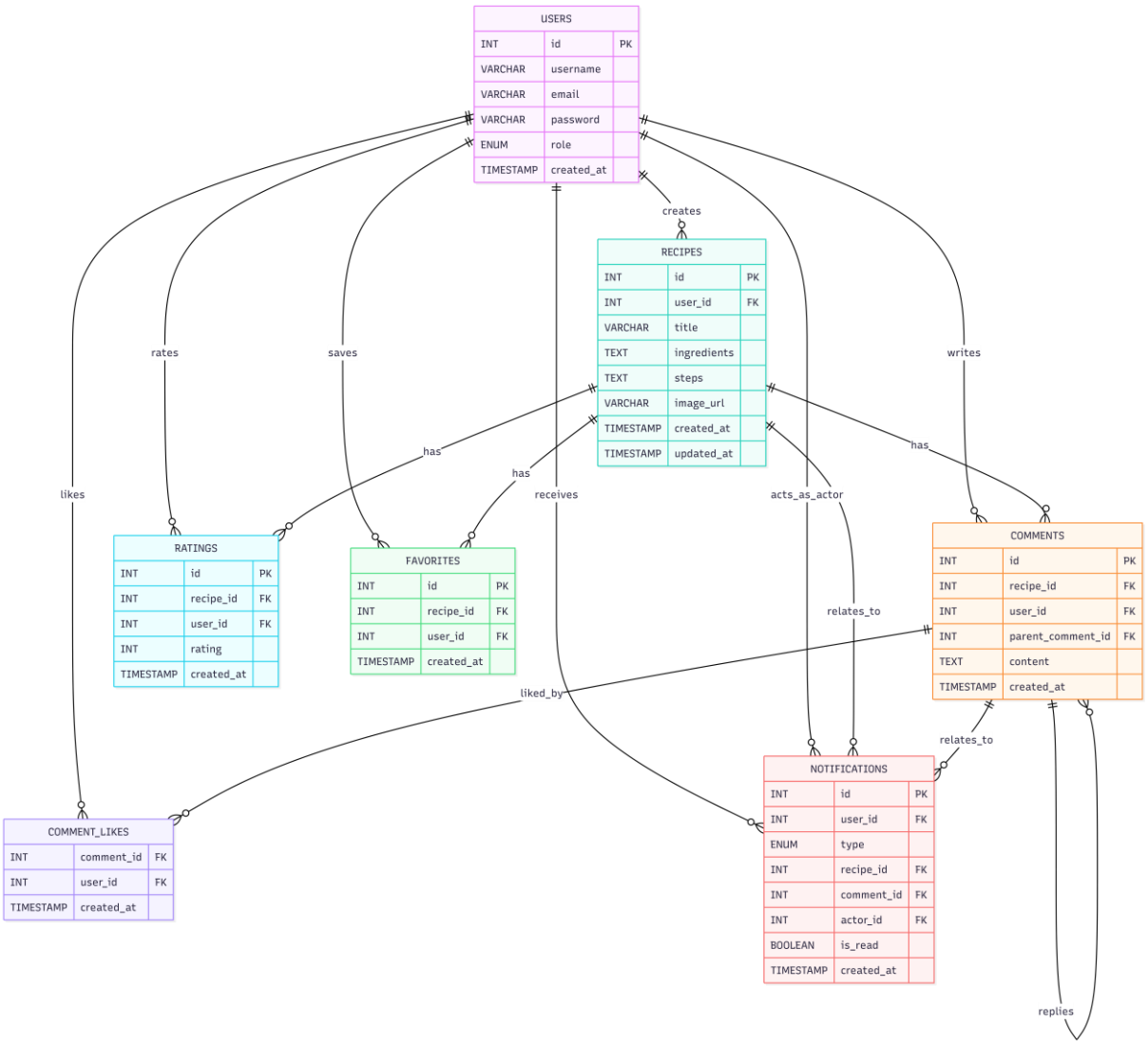
- **Bảng favorites**
  - + id INT PK AUTO_INCREMENT
  - + recipe_id INT NOT NULL FK → recipes(id) ON DELETE CASCADE
  - + user_id INT NOT NULL FK → users(id) ON DELETE CASCADE
  - + created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  - + UNIQUE (recipe_id, user_id)

- **Bảng comment_likes**
  - + comment_id INT NOT NULL FK → comments(id) ON DELETE CASCADE
  - + user_id INT NOT NULL FK → users(id) ON DELETE CASCADE
  - + created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  - + UNIQUE (comment_id, user_id)

- **Bảng notifications**
  - + id INT PK AUTO_INCREMENT
  - + user_id INT NOT NULL FK → users(id) ON DELETE CASCADE // người nhận thông báo
  - + type ENUM('comment','reply','like','recipe_update') NOT NULL
  - + recipe_id INT NULL FK → recipes(id) ON DELETE CASCADE
  - + comment_id INT NULL FK → comments(id) ON DELETE CASCADE
  - + actor_id INT NOT NULL FK → users(id) ON DELETE CASCADE // người gây ra sự kiện
  - + is_read BOOLEAN DEFAULT FALSE
  - + created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

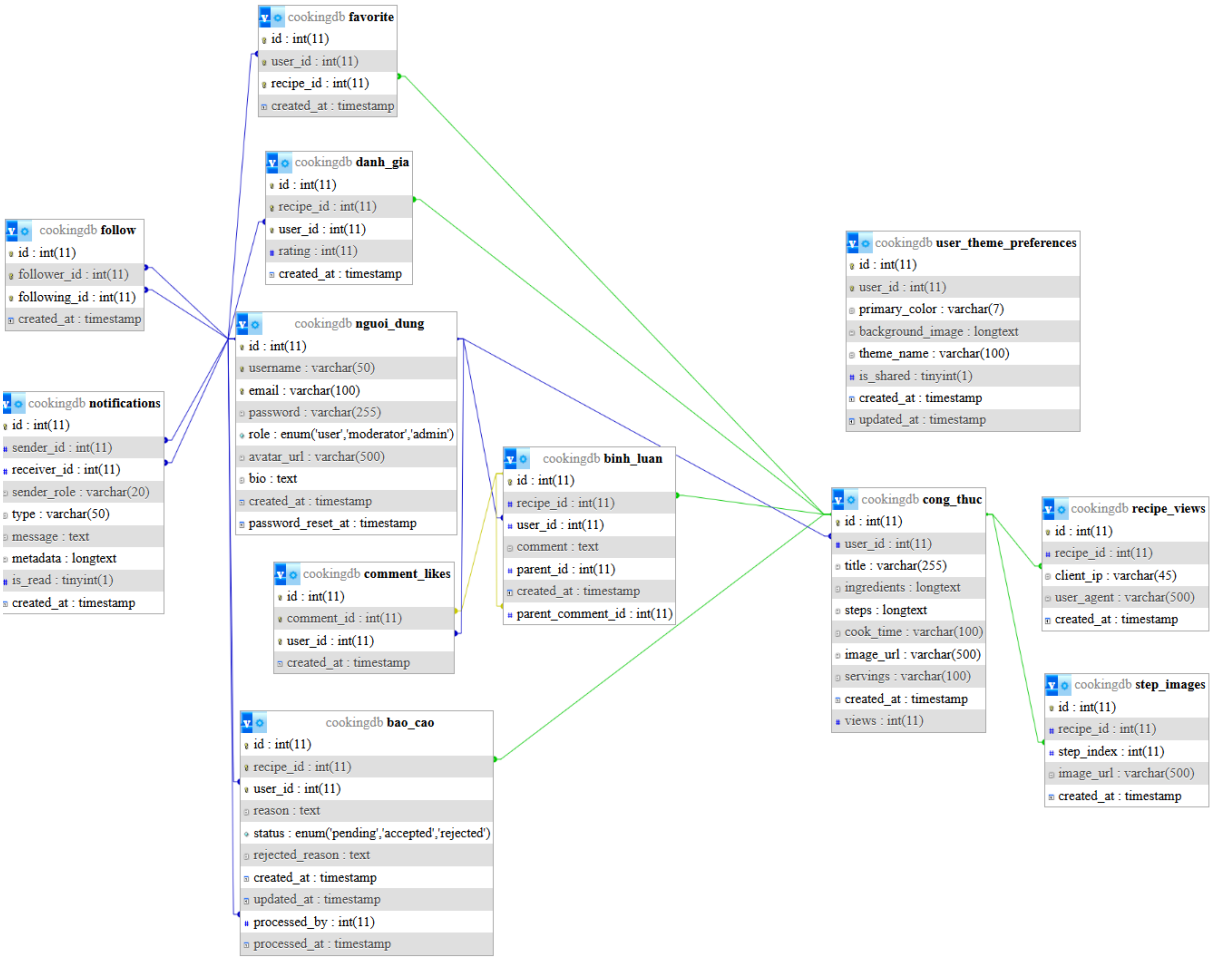
- **Chỉ mục gợi ý**
  - + Index: comments(recipe_id), comments(parent_comment_id)
  - + Index: recipes(title) hoặc FULLTEXT(title, ingredients) nếu bật fulltext
  - + Dùng UNIQUE như trên để ngăn trùng lặp (ratings, favorites, comment_likes)

Mô hình dữ liệu ERD:



Hình 3.1 Mô hình dữ liệu ERD

Mô hình cơ sở dữ liệu:



Hình 3.2 Mô hình cơ sở dữ liệu

Bảng 3.2: Schema cơ sở dữ liệu

Bảng	Trường chính	Kiểu dữ liệu	Ràng buộc
users	id	INT	PRIMARY KEY, AUTO_INCREMENT
	username	VARCHAR(50)	UNIQUE, NOT NULL
	email	VARCHAR(100)	UNIQUE, NOT NULL
	password	VARCHAR(255)	NOT NULL (Bcrypt hash)
	role	ENUM('user','admin')	DEFAULT 'user'
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
recipes	id	INT	PRIMARY KEY, AUTO_INCREMENT
	user_id	INT	FOREIGN KEY → users(id)
	title	VARCHAR(200)	NOT NULL
	ingredients	TEXT	NOT NULL
	steps	TEXT	NOT NULL
	image_url	VARCHAR(500)	NULL
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
	updated_at	TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
comments	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY → recipes(id)
	user_id	INT	FOREIGN KEY → users(id)
	content	TEXT	NOT NULL
	parent_comment_id	INT	FOREIGN KEY → comments(id), NULLABLE
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
ratings	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY →

			recipes(id)
	user_id	INT	FOREIGN KEY → users(id)
	rating	INT	CHECK (1-5)
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
			UNIQUE (recipe_id, user_id)
favorites	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY → recipes(id)
	user_id	INT	FOREIGN KEY → users(id)
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
			UNIQUE (recipe_id, user_id)
comment_likes	comment_id	INT	FOREIGN KEY → comments(id) ON DELETE CASCADE
	user_id	INT	FOREIGN KEY → users(id) ON DELETE CASCADE
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
			UNIQUE (comment_id, user_id)
notifications	id	INT	PRIMARY KEY, AUTO_INCREMENT
	user_id	INT	FOREIGN KEY → users(id) ON DELETE CASCADE
	type	ENUM	'comment', 'reply', 'like', 'recipe_update'
	recipe_id	INT	FOREIGN KEY → recipes(id), NULLABLE
	comment_id	INT	FOREIGN KEY →

			comments(id), NULLABLE
	actor_id	INT	FOREIGN KEY → users(id) (người gây action)
	is_read	BOOLEAN	DEFAULT FALSE
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

3.2.3 Thiết kế API (REST)

- Auth: POST /auth/register, POST /auth/login
- Recipe: GET /recipe/list, GET /recipe/search, POST /recipe/create, PUT /recipe/update/:id, DELETE /recipe/delete/:id
- Interaction: POST /rating/:recipeId, POST /favorite/:recipeId, POST /comment /:recipeId
- Admin: GET /admin/users, PUT /admin/users/:id/role, DELETE /admin/ recipes/:id
- Mã trạng thái: 200/201/400/401/403/404/500; thông điệp lỗi rõ ràng.

3.2.4 Thiết kế giao diện (UI/UX)

- Trang chính: danh sách công thức, thanh tìm kiếm, lưới ảnh.
- Navbar: logo, đăng nhập/đăng ký, tên người dùng + logout, link admin nếu role=admin.
- Trang chi tiết công thức: nguyên liệu, bước, ảnh, bình luận, đánh giá.
- Trang quản lý cá nhân: My Recipes, Favorite Recipes.
- Trang quản trị: danh sách người dùng, đổi role, xóa công thức.

3.3 Hiện thực (Implementation)

3.3.1 Frontend (React)

- Công nghệ: React 19.x, React Router 6.x, Axios 1.x, CSS responsive.
- Cấu trúc (rút gọn):
  + components/: Navbar.jsx, ProtectedRoute.jsx
  + pages/: Home.jsx, CreateRecipe.jsx, RecipeDetail.jsx, AdminDashboard.jsx, MyRecipes.jsx, FavoriteRecipes.jsx, Login.jsx, Register.jsx, v.v.
- Xử lý chính:

+ Lưu JWT trong localStorage; interceptor của Axios thêm Authorization header.
+ ProtectedRoute kiểm tra token trước khi vào các trang cần đăng nhập.
+ Form tạo/sửa công thức: upload ảnh (gửi đến backend, backend đẩy lên Cloudinary).

3.3.2 Backend (Node.js + Express)

- Công nghệ: Node.js 16+, Express 4.x, jsonwebtoken, bcryptjs, multer/cloudinary SDK, dotenv.
- Cấu trúc:
  + server.js: khởi tạo Express, CORS, body parser, mount routes, kết nối DB.
  + routes/: auth.js, recipe.js, rating.js, favorite.js, admin.js.
  + auth.js: verify JWT, check role.
  + config/: db.js (MySQL pool), cloudinary.js.

- Luồng chính:
  + Auth: hash mật khẩu với Bcrypt; login tạo JWT (exp 7d); middleware verify token.
  + Recipe: CRUD + upload ảnh lên Cloudinary, lưu URL vào DB; kiểm tra quyền owner/admin.
  + Interaction: rating (mỗi user 1 lần/recipe), comment, favorite; ràng buộc quyền đăng nhập.
  + Admin: đổi role, xóa công thức hoặc user vi phạm.

Bảng 3.3: Cấu trúc thư mục dữ liệu

Bảng	Trường chính	Kiểu dữ liệu	Ràng buộc
users	id	INT	PRIMARY KEY, AUTO_INCREMENT
	username	VARCHAR(50)	UNIQUE, NOT NULL
	email	VARCHAR(100)	UNIQUE, NOT NULL
	password	VARCHAR(255)	NOT NULL (Bcrypt hash)
	role	ENUM('user','admin')	DEFAULT 'user'
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
recipes	id	INT	PRIMARY KEY, AUTO_INCREMENT
	user_id	INT	FOREIGN KEY → users(id)
	title	VARCHAR(200)	NOT NULL
	ingredients	TEXT	NOT NULL
	steps	TEXT	NOT NULL
	image_url	VARCHAR(500)	NULL
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
	updated_at	TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
binh_luan	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY → recipes(id)
	user_id	INT	FOREIGN KEY → users(id)
	comment	TEXT	NOT NULL
	parent_comment_id	INT	FOREIGN KEY →

			binh_luan(id), NULL (cho top-level)
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
comment_likes	id	INT	PRIMARY KEY, AUTO_INCREMENT
	comment_id	INT	FOREIGN KEY → binh_luan(id) ON DELETE CASCADE
	user_id	INT	FOREIGN KEY → users(id) ON DELETE CASCADE
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
			UNIQUE (comment_id, user_id)
ratings	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY → recipes(id)
	user_id	INT	FOREIGN KEY → users(id)
	rating	INT	CHECK (1-5)
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
			UNIQUE (recipe_id, user_id)
favorites	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY → recipes(id)

	user_id	INT	FOREIGN KEY → users(id)
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
			UNIQUE (recipe_id, user_id)
notifications	id	INT	PRIMARY KEY, AUTO_INCREMENT
	sender_id	INT	FOREIGN KEY → users(id), NULL nếu system
	receiver_id	INT	FOREIGN KEY → users(id) NOT NULL
	type	ENUM('comment_reply', 'report_status', 'mention')	NOT NULL
	message	VARCHAR(500)	NOT NULL
	metadata	JSON	NULL (chứa comment_id, report_id, v.v.)
	is_read	BOOLEAN	DEFAULT FALSE
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

3.3.3 Database (MySQL)

- Bảng chính: users, recipes, comments, ratings, favorites.
- Ràng buộc khóa ngoại đảm bảo toàn vẹn; chỉ số (index) cho trường tìm kiếm (title, ingredients).

3.3.4 Lưu trữ ảnh (Cloudinary)

Backend nhận file (multer), upload lên Cloudinary, nhận URL trả về; lưu URL vào trường image_url của recipes.

3.4 Bảo mật và quản lý phiên

- **Bcrypt:** hash mật khẩu với Salt, cost ~10 (có thể tăng khi hạ tầng mạnh hơn).
- **JWT:** lưu trên client; gửi Bearer token; middleware verify; exp 7 ngày.
- **CORS:** origin http://localhost:3000, cho phép credentials; hạn chế header/method cần thiết.
- **Phân quyền:** middleware kiểm tra role cho route admin; kiểm tra owner trước khi sửa/xóa công thức.

Bảng 3.4: Phân quyền user vs admin

Hành động		User	Admin
Xem danh sách công thức		✓	✓
Tìm kiếm công thức		✓	✓
Tạo công thức mới		✓	✓
Sửa công thức của mình		✓	✓
Xóa công thức của mình		✓	✓
Xóa công thức của người khác		✗	✓
Bình luận, đánh giá		✓	✓
	Reply bình luận (tạo thread)	✓	✓
	Like/Unlike bình luận	✓	✓
Xóa bình luận của người khác		✗	✓
	Xóa reply của người khác	✗	✓
Xem danh sách người dùng		✗	✓
Đổi vai trò người dùng		✗	✓
Xóa tài khoản người dùng		✗	✓
	Xem/phản hồi báo cáo	✗	✓
	Nhận thông báo	✓	✓

3.5 Thiết kế chi tiết nested comments và like system

3.5.1 Nested Comments

- Khi user trả lời bình luận, gửi `POST /comment/:recipeId` với `parent_comment_id = comment_id`.

- Backend lưu vào bảng `comments` với trường `parent_comment_id` không NULL.
- Khi fetch: GET `/recipe/comment/:id` trả struct nested: `[{id, content, user, created_at, replies: [{...}], like_count, user_liked}]`.
- Frontend: Component `CommentItem` recursive, render reply form inline khi user click "Reply".
- UI: Reply hiển thị với indent/margin-left để phân biệt cấp độ.

3.5.2 Like System

- Database: Bảng riêng `comment_likes` (comment_id, user_id, UNIQUE).
- Endpoint: `POST /comment/:id/like` kiểm tra UNIQUE constraint; nếu tồn tại → delete (unlike), không tồn tại → insert (like); trả `{liked: true/false}`.
- Frontend: Fetch comments include `like_count, user_liked`; button "♥ (count)" hiển thị ♥ (filled) nếu `user_liked=true`, ♡ (outline) nếu false; onclick gọi POST like endpoint.
- Notifications: Khi user A like comment của user B, insert vào bảng `notifications` với type='like', actor_id=A, comment_id=..., user_id=B.

3.6 Kiểm thử

- **Chức năng:** đăng ký/đăng nhập; CRUD công thức; tìm kiếm; bình luận; đánh giá; yêu thích; đổi role; xóa nội dung.
- **Bảo mật:** từ chối request thiếu/invalid JWT; không cho user thường dùng route admin; kiểm tra upload chỉ qua route bảo vệ.
- **Hiệu năng cơ bản:** đo thời gian phản hồi API chính; kiểm tra độ trễ upload ảnh; duyệt trên desktop/mobile (responsive).

3.7 Đánh giá và ghi nhận

- Hệ thống đáp ứng đầy đủ chức năng cốt lõi; giao diện thân thiện; bảo mật cơ bản tốt (Bcrypt, JWT, CORS).
- Hiệu năng phù hợp với quy mô đề tài; có thể mở rộng bằng cache (Redis), CDN, hoặc tách dịch vụ.
- Hướng phát triển: realtime notification (WebSocket), gợi ý cá nhân hóa (ML), CI/CD tự động, test tự động (unit/integration), tối ưu SEO và accessibility.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1 Kết quả chức năng

- Hoàn thiện luồng xác thực: đăng ký, đăng nhập, JWT (hết hạn 7 ngày), phân quyền user/admin.
- Quản lý công thức: tạo, sửa, xóa, xem chi tiết; tìm kiếm theo từ khóa; upload ảnh lên Cloudinary.
- Tương tác người dùng: bình luận, đánh giá sao (1 user/recipe/1 rating), đánh dấu yêu thích.
- Quản trị: xem danh sách người dùng, đổi vai trò, xóa công thức/tài khoản vi phạm.
- API REST chuẩn: mã trạng thái 200/201/400/401/403/404/500; thông điệp phản hồi rõ ràng.

Bảng 4.1: Kết quả kiểm thử chức năng

ID	Chức năng	Input	Expected	Actual	Status
TC01	Đăng ký - hợp lệ	Valid email, password	201, success	201, user created	✓ Pass
TC02	Đăng ký - email trùng	Duplicate email	400, email exists	400, email exists	✓ Pass
TC03	Đăng nhập - đúng	Valid credentials	200, JWT token	200, token returned	✓ Pass
TC04	Đăng nhập - sai password	Invalid password	400, wrong password	400, wrong password	✓ Pass
TC05	Tạo công thức - có token	Valid data + JWT	201, created	201, recipe created	✓ Pass
TC06	Tạo công thức - không token	No token	401, unauthorized	401, unauthorized	✓ Pass
TC07	Xóa công thức - owner	Owner token	200, deleted	200, deleted	✓ Pass
TC08	Xóa công thức - not owner	Not owner	403, forbidden	403, forbidden	✓ Pass

TC09	Đánh giá - lần đầu	Valid rating 1-5	201, rated	201, rated	✓ Pass
TC10	Đánh giá - lần 2	Duplicate rating	400, already rated	400, already rated	✓ Pass
TC11	Bình luận - top-level	Valid comment + JWT	201, created	201, comment created	✓ Pass
TC12	Reply bình luận	Valid reply + parent_id + JWT	201, created	201, reply created	✓ Pass
TC13	Like bình luận - lần 1	comment_id + JWT	200, {liked: true}	200, liked: true	✓ Pass
TC14	Unlike bình luận	Unlike bình luận	200, {liked: false}	200, {liked: false}	✓ Pass
TC15	Like bình luận - 2 users	Diff users like same comment	like_count: 2	like_count: 2	✓ Pass
TC16	Lấy comments với likes	GET /recipe/comment/:id	200: [comments + like_count + user_liked]	200: nhận dữ liệu đầy đủ	✓ Pass
TC17	Notification - reply	Tác giả reply bình luận	Notification created	Notification created, type: comment_reply	✓ Pass
TC18	Notification - report status	Admin phản hồi báo cáo	201: notification sent	201, is_read: false	✓ Pass
TC19	Admin đổi role	Admin token	200, role changed	200, role changed	✓ Pass
TC120	User đổi role	User token	403, not admin	403, not admin	✓ Pass

4.2 Trải nghiệm người dùng (UX/UI)

- Giao diện responsive: hiển thị tốt trên desktop, tablet, mobile.
- Navbar động: hiển thị nút đăng nhập/đăng ký khi chưa login; tên người dùng + logout khi đã login; link admin cho admin.
- Trang chủ: lưới công thức, thanh tìm kiếm, nút tạo công thức khi đã đăng nhập.
- Trang chi tiết: ảnh, nguyên liệu, bước nấu; bình luận và đánh giá trực tiếp.
- Trang cá nhân: My Recipes, Favorite Recipes; CRUD công thức của riêng mình.
- Trang quản trị: bảng người dùng, đổi role, xóa công thức vi phạm.

4.3 Hiệu năng và kỹ thuật

- Non-blocking I/O (Node.js + Express) xử lý đồng thời nhiều request; không ghi nhận nghẽn ở tải thử nghiệm cục bộ.
- Upload ảnh chuyển sang Cloudinary, giảm tải đĩa và băng thông máy chủ.
- CORS cấu hình cho front (localhost:3000) ↔ back (localhost:3001) an toàn; JWT qua header Authorization.
- MySQL đáp ứng truy vấn danh sách/tìm kiếm công thức; ràng buộc khóa ngoại đảm bảo toàn vẹn.

Bảng 4.2: Đánh giá hiệu năng

API Endpoint	Method	Avg Response Time	Note
/auth/login	POST	~150ms	Bao gồm Bcrypt compare
/auth/register	POST	~200ms	Bao gồm Bcrypt hash
/recipe/list	GET	~80ms	Query 50 recipes
/recipe/search	GET	~120ms	Full-text search
/recipe/create	POST	~800ms	Bao gồm upload Cloudinary
/recipe/update/:id	PUT	~600ms	Với upload ảnh mới
/recipe/delete/:id	DELETE	~50ms	Xóa từ DB
/rating/:id	POST	~60ms	Insert rating
/recipe/comment/:recipeId	GET	~100ms	Build nested structure
/recipe/comment	POST	~80ms	Insert comment/reply
/comment/:commentId/like	POST	~70ms	Toggle like (query + insert/delete)
/notification/list	GET	~90ms	Query user notifications

/notification/:id/reply	POST	~100ms	Insert notification
/recipe/comment/:id	GET	~150ms	Lấy comments + nested (2-level deep, ~50 comments)
/comment/:id/like	POST	~80ms	Toggle like, insert/delete comment_likes
/comment/:recipeId	POST	~100ms	Create comment với parent_comment_id
/notifications	GET	~120ms	Lấy 20 notifications gần đây

4.4 Bảo mật

- Mật khẩu được hash với Bcrypt (Salt + adaptive hashing).
- JWT có hạn, kiểm tra ở middleware; phân quyền rõ (user vs admin).
- Không lưu mật khẩu gốc; từ chối truy cập khi thiếu/invalid token; hạn chế origin qua CORS.

4.5 Giao diện minh họa (mô tả nhanh)

- **Navbar:** Logo/tên app, đăng nhập/đăng ký hoặc tên người dùng + logout, link Admin (nếu admin).
- **Home:** Lưới công thức, thẻ ảnh + tiêu đề; thanh tìm kiếm.
- **Create/Edit Recipe:** Form nhập tiêu đề, nguyên liệu, bước nấu, upload ảnh; nút lưu.
- **Recipe Detail:** Ảnh, nội dung, bình luận, đánh giá; nút yêu thích.
- **Admin Dashboard:** Bảng người dùng (username, email, role), nút đổi role/xóa; danh sách công thức để gỡ bỏ nếu vi phạm.

4.6 Đánh giá nhanh

- Đạt được: Đầy đủ chức năng cốt lõi; giao diện thân thiện và responsive; bảo mật cơ bản tốt (hash mật khẩu, JWT, phân quyền); hiệu năng phù hợp quy mô đề tài.
- Hạn chế: Chưa có cache (Redis), chưa có realtime notification, chưa triển khai CI/CD và bộ test tự động; chưa có gợi ý cá nhân hóa.

4.7 Hướng phát triển

- Thêm realtime notification (WebSocket) cho bình luận/đánh giá mới.
- Thêm cache (Redis) cho danh sách/tìm kiếm công thức.
- Viết test tự động (unit/integration) và thiết lập CI/CD.
- Mở rộng gợi ý công thức dựa trên hành vi người dùng (recommendation).

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

- Hoàn thành website CookShare với đầy đủ chức năng cốt lõi: đăng ký/đăng nhập (JWT), quản lý công thức (tạo/sửa/xóa/tìm kiếm), bình luận, đánh giá, yêu thích, và phân quyền user/admin.
- Áp dụng kiến trúc 3 tầng, RESTful API, bảo mật Bcrypt + JWT, CORS, và lưu trữ ảnh qua Cloudinary; giao diện React responsive vận hành ổn định.
- Đảm bảo cơ sở dữ liệu quan hệ MySQL với ràng buộc toàn vẹn; trải nghiệm người dùng mạch lạc, quy trình nghiệp vụ rõ ràng.
- Đóng góp: một giải pháp chia sẻ công thức nấu ăn có kiểm duyệt, dễ mở rộng, bảo mật cơ bản tốt, tổ chức mã theo chuẩn (MVC, middleware, phân tầng).
- Tích hợp nested comments cho phép cuộc trò chuyện chi tiết hơn; like system tăng tương tác xã hội và feedback từ cộng đồng.
- Thiết kế phân tầng: truy vấn recursion, bảng riêng cho likes, notification queue.

5.2 Hạn chế

- Chưa có pagination cho threads bình luận sâu; dữ liệu nhiều cấp sẽ chậm khi tải.
- Chưa có hệ thống strike/violation (tự ẩn bình luận khi vi phạm nhiều lần).
- Notifications chưa realtime (chưa dùng WebSocket, mới ở mức polling/thủ công).
- Chưa dùng cache (Redis) cho danh sách/tìm kiếm; dễ bị chậm khi tải lớn.
- Chưa có test tự động (unit/integration) và chưa thiết lập CI/CD.
- Chưa có hệ thống gợi ý cá nhân hóa (recommendation/ML) và tối ưu SEO/accessibility.

5.2 Hướng phát triển

- Thêm realtime notifications (WebSocket) cho bình luận/like/reply.
- Bổ sung cache (Redis) cho danh sách và tìm kiếm để giảm độ trễ.
- Viết test tự động (unit/integration) và thiết lập CI/CD để tăng độ tin cậy.
- Pagination/virtualization cho bình luận nhiều cấp để tối ưu hiệu năng.
- Xây dựng strike/violation system: tự ẩn hoặc khóa bình luận khi vượt ngưỡng vi phạm.
- Gợi ý công thức cá nhân hóa (recommendation/ML); tối ưu SEO và accessibility.

DANH MỤC TÀI LIỆU THAM KHẢO

- RESTful API: Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine, 2000.
- JSON Web Token (JWT): Auth0. *JSON Web Token Introduction*. https://jwt.io/introduction
- Bcrypt: Niels Provos, David Mazieres. *A Future-Adaptable Password Scheme*. Proceedings of the 1999 USENIX Annual Technical Conference.
- CORS: MDN Web Docs. *HTTP access control (CORS)*. https://developer.mozilla.org/docs/Web/HTTP/CORS
- React: React Team. *React Documentation*. https://react.dev
- Node.js & Express: Express.js Team. *Express Guide*. https://expressjs.com
- MySQL: Oracle. *MySQL 8.0 Reference Manual*. https://dev.mysql.com/doc
- Cloudinary: Cloudinary Docs. *Image Upload API*. https://cloudinary.com/documentation/image_upload_api
- OWASP: *OWASP Cheat Sheet Series* (Authentication, Password Storage, REST Security). https://cheatsheetseries.owasp.org

PHỤ LỤC

PHỤ LỤC A: Cấu hình môi trường

File .env (Backend)

Bảng PL.1: Môi trường phát triển

Thành phần	Phiên bản/Cấu hình	Ghi chú
OS	Windows 11 / macOS / Linux	Đa nền tảng
Node.js	16.20.0+	LTS
npm	8.19.0+	Package manager
MySQL	8.0.33	Database server
IDE	VS Code 1.85+	Với extension ESLint, Prettier
Browser	Chrome 120+	DevTools support
Postman	10.0+	API testing

```
PORT=3001
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=yourpassword
DB_NAME=cookshare

SECRET_KEY=your-jwt-secret-key-here
JWT_SECRET=your-jwt-secret-key-here

CLOUDINARY_CLOUD_NAME=your-cloud-name
CLOUDINARY_API_KEY=your-api-key
CLOUDINARY_API_SECRET=your-api-secret

EMAIL_USER=your-email@gmail.com
EMAIL_PASSWORD=your-app-password
```

Yêu cầu hệ thống:

- Node.js: 16.x trở lên
- MySQL: 8.0 trở lên
- npm: 8.x trở lên
- Trình duyệt: Chrome 90+, Firefox 88+, Safari 14+

PHỤ LỤC B: Hướng dẫn cài đặt

Bước 1: Clone repository

git clone <repository-url>

cd DoAnChuyenNganh

Bước 2: Cài đặt Backend

cd src/backend

npm install

Tạo file .env theo mẫu ở Phụ lục A

Tạo database MySQL

mysql -u root -p < database/database.sql

npm start

Bước 3: Cài đặt Frontend

cd src/cookshare

npm install

npm start

Bước 4: Truy cập ứng dụng

Frontend: **http://localhost:3000**

Backend API: **http://localhost:3001**

PHỤ LỤC C: Code mẫu quan trọng

C.1 Middleware xác thực JWT (middleware/auth.js)

```
const jwt = require("jsonwebtoken");
const SECRET_KEY = process.env.SECRET_KEY || "SECRET_KEY";
const verifyToken = (req, res, next) => {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];
  if (!token) {
    return res.status(401).json({ message: "Access token required" });
  }
  jwt.verify(token, SECRET_KEY, (err, user) => {
    if (err) {
      return res.status(403).json({ message: "Invalid or expired token" });
    }
    req.user = user;
    next();
  });
};
```

```
module.exports = { verifyToken };
```

C.2 Tạo JWT khi đăng nhập (routes/auth.js)

```
const token = jwt.sign({ id: user.id }, SECRET_KEY, { expiresIn: "7d" });
return res.json({
  message: "Đăng nhập thành công!",
  token,
  username: user.username,
  role: user.role,
  userId: user.id
});
```

C.3 Hash mật khẩu với Bcrypt (routes/auth.js)

```
const bcrypt = require("bcrypt");
// Khi đăng ký
const hashed = await bcrypt.hash(password, 10);
// Khi đăng nhập
const match = await bcrypt.compare(password, user.password);
```

C.4 Axios interceptor (Frontend)

```
import axios from 'axios';
const api = axios.create({
  baseURL: 'http://localhost:3001'
});
// Thêm token vào mọi request
api.interceptors.request.use(config => {
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
export default api;
```

C.5 Like comment handler (routes/recipe.js)

```javascript
// POST /comment/:id/like - Toggle like/unlike
router.post("/comment/:id/like", verifyToken, async (req, res) => {
  const commentId = req.params.id;
  const userId = req.user.id;
  try {
    // Check nếu đã like
    const checkLike = await db.query(
      "SELECT * FROM comment_likes WHERE comment_id = ? AND user_id = ?",
      [commentId, userId]
    );
    let liked = false;
    if (checkLike.length > 0) {
      // Unlike
      await db.query(
        "DELETE FROM comment_likes WHERE comment_id = ? AND user_id = ?",
        [commentId, userId]
      );
      liked = false;
    } else {
      // Like
      await db.query(
        "INSERT INTO comment_likes (comment_id, user_id) VALUES (?, ?)",
        [commentId, userId]
      );
      liked = true;
    }
    res.json({ message: "Success", liked });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

C.6 Nested comments fetch (routes/recipe.js)

```javascript
// GET /recipe/comment/:recipeId - Get comments with nested replies
router.get("/recipe/comment/:recipeId", verifyToken, async (req, res) => {
  const recipeId = req.params.recipeId;
  const userId = req.user.id;
  try {
    // Get top-level comments
    const topComments = await db.query(
      `SELECT c.id, c.content, c.user_id, u.username, c.created_at,
              COUNT(DISTINCT cl.user_id) as like_count,
              MAX(CASE WHEN cl.user_id = ? THEN 1 ELSE 0 END) as user_liked
       FROM comments c
       JOIN users u ON c.user_id = u.id
       LEFT JOIN comment_likes cl ON c.id = cl.comment_id
       WHERE c.recipe_id = ? AND c.parent_comment_id IS NULL
       GROUP BY c.id
       ORDER BY c.created_at DESC`,
      [userId, recipeId]
    );
    // For each top comment, get nested replies
    for (let comment of topComments) {
      const replies = await db.query(
        `SELECT c.id, c.content, c.user_id, u.username, c.created_at,
                COUNT(DISTINCT cl.user_id) as like_count,
                MAX(CASE WHEN cl.user_id = ? THEN 1 ELSE 0 END) as user_liked
         FROM comments c
         JOIN users u ON c.user_id = u.id
         LEFT JOIN comment_likes cl ON c.id = cl.comment_id
         WHERE c.parent_comment_id = ?
         GROUP BY c.id
         ORDER BY c.created_at ASC`,
        [userId, comment.id]
      );
      comment.replies = replies;
    }
```

```
    res.json(topComments);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

C.7 Frontend - CommentItem recursive component

```javascript
// components/CommentItem.jsx - Recursive component
const CommentItem = ({ comment, recipeId, onCommentAdd, userId, handleLike }) => {
  const [showReplyForm, setShowReplyForm] = useState(false);
  const [replyText, setReplyText] = useState("");
  const handleReplySubmit = async () => {
    if (!replyText.trim()) return;
    try {
      await api.post(`/comment/${recipeId}`, {
        content: replyText,
        parent_comment_id: comment.id
      });
      setReplyText("");
      setShowReplyForm(false);
      onCommentAdd(); // Refresh comments
    } catch (error) {
      console.error("Reply failed:", error);
    }
  };
  return (
    <div className="comment-item">
      <div className="comment-header">
        <strong>{comment.username}</strong>
<span className="comment-date">{new Date(comment.created_at) .toLocale
DateString()}</span>
      </div>
      <p className="comment-content">{comment.content}</p>
      <div className="comment-actions">
        <button onClick={() => handleLike(comment.id)} className="btn-like">
          {comment.user_liked ? "♥" : "♡"} {comment.like_count}
```

```
          </button>
          {userId === comment.user_id && (
            <button onClick={() => setShowReplyForm(!showReplyForm)}>
              Reply
            </button>
          )}
        </div>
        {showReplyForm && (
          <div className="reply-form">
            <textarea
              value={replyText}
              onChange={(e) => setReplyText(e.target.value)}
              placeholder="Write a reply..."
            />
            <button onClick={handleReplySubmit}>Post Reply</button>
          </div>
        )}
        {/* Nested replies */}
        {comment.replies && comment.replies.length > 0 && (
          <div className="nested-replies">
            {comment.replies.map((reply) => (
              <CommentItem
                key={reply.id}
                comment={reply}
                recipeId={recipeId}
                onCommentAdd={onCommentAdd}
                userId={userId}
                handleLike={handleLike}
              />
            ))}
          </div>
        )}
      </div>
    );
};
export default CommentItem;
```

PHỤ LỤC D: Schema cơ sở dữ liệu chi tiết

Bảng users

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) UNIQUE NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  role ENUM('user', 'admin') DEFAULT 'user',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Bảng recipes

```
CREATE TABLE recipes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  title VARCHAR(200) NOT NULL,
  ingredients TEXT NOT NULL,
  steps TEXT NOT NULL,
  image_url VARCHAR(500),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Bảng comments

```
CREATE TABLE comments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  recipe_id INT NOT NULL,
  user_id INT NOT NULL,
  content TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (recipe_id) REFERENCES recipes(id) ON DELETE CASCADE,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Bảng ratings

```
CREATE TABLE comments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  recipe_id INT NOT NULL,
  user_id INT NOT NULL,
  content TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (recipe_id) REFERENCES recipes(id) ON DELETE CASCADE,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Bảng favorites

```
CREATE TABLE favorites (
  id INT AUTO_INCREMENT PRIMARY KEY,
  recipe_id INT NOT NULL,
  user_id INT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY unique_favorite (recipe_id, user_id),
  FOREIGN KEY (recipe_id) REFERENCES recipes(id) ON DELETE CASCADE,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

PHỤ LỤC E: API Documentation chi tiết

E.1 Authentication Endpoints

POST /auth/register

- Body: **{ username, email, password, confirmPassword }**
- Response: **{ message: "Đăng ký thành công!" }**
- Errors: 400 (validation), 500 (server)

POST /auth/login

- Body: **{ email, password }**
- Response: **{ message, token, username, role, userId }**
- Errors: 400 (invalid credentials), 500 (server)

E.2 Recipe Endpoints

GET /recipe/list

- Response: [{ id, title, ingredients, steps, image_url, username, ... }]

GET /recipe/search?q=keyword

- Query: q (search keyword)
- Response: [{ matching recipes }]

POST /recipe/create

- Headers: Authorization: Bearer <token>
- Body: FormData with title, ingredients, steps, image
- Response: { message, recipeId }

PUT /recipe/update/:id

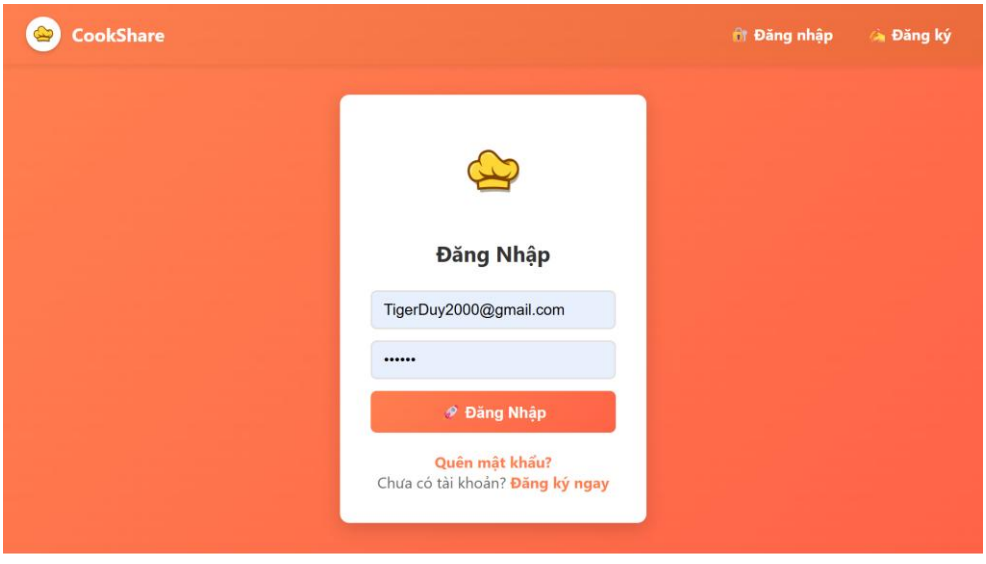
- Headers: Authorization: Bearer <token>
- Body: FormData
- Response: { message }
- Errors: 403 (not owner), 404 (not found)

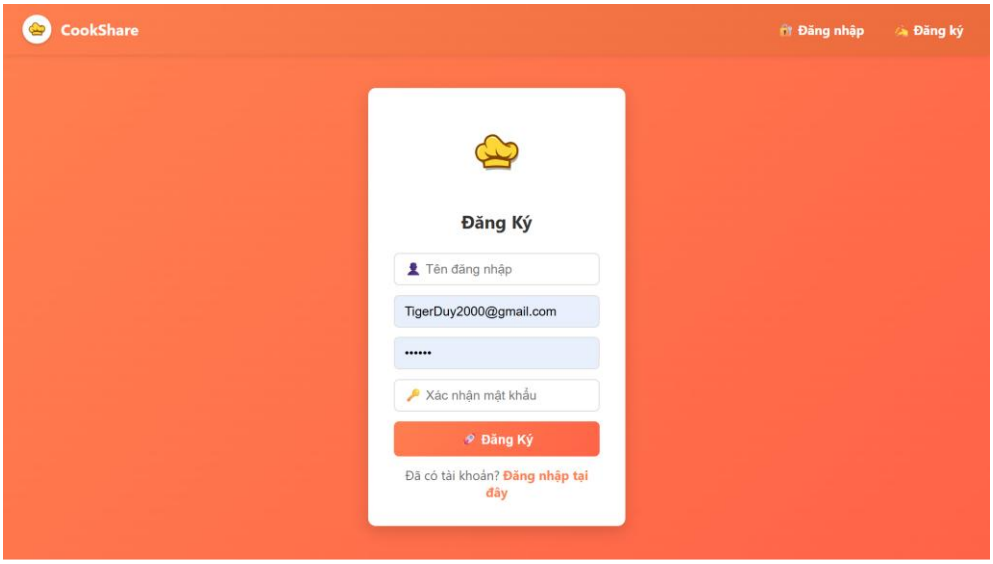
DELETE /recipe/delete/:id

- Headers: Authorization: Bearer <token>
- Response: { message }
- Errors: 403 (not owner/admin), 404 (not found)

PHỤ LỤC F: Screenshots giao diện

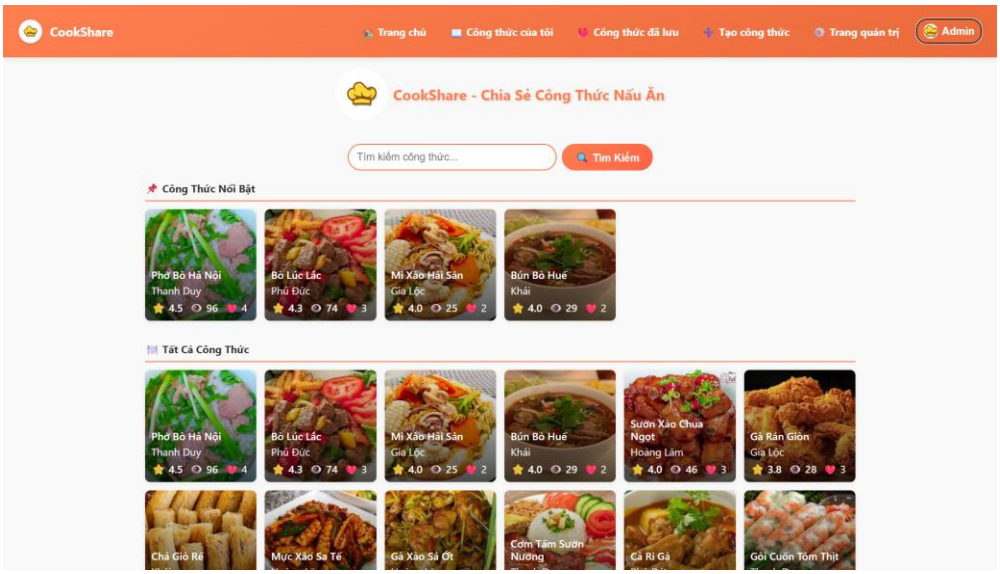
F.1: Trang đăng nhập và đăng ký





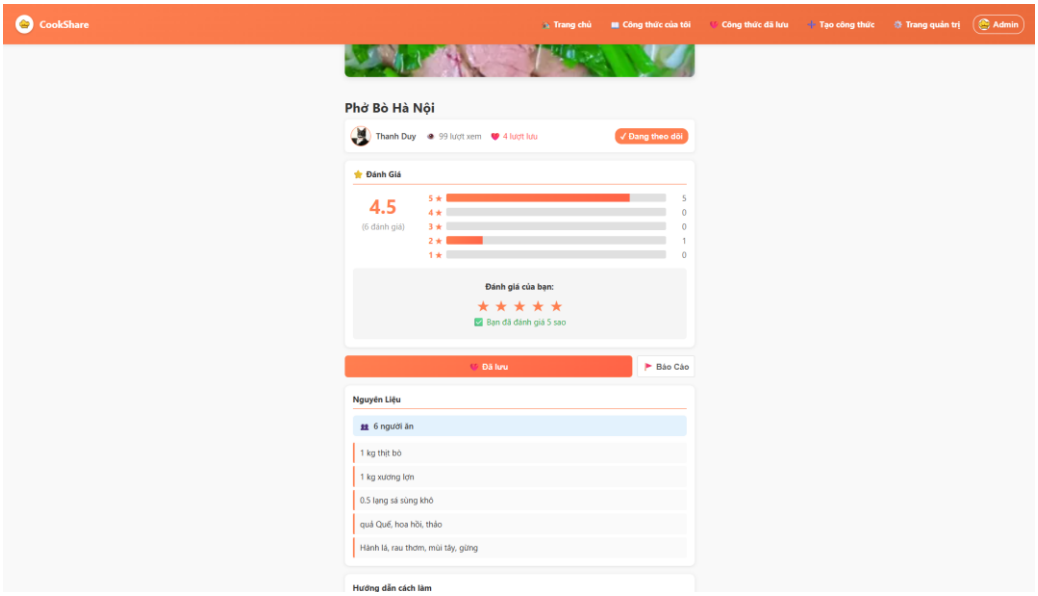
Hình F.1 Trang đăng nhập và đăng ký

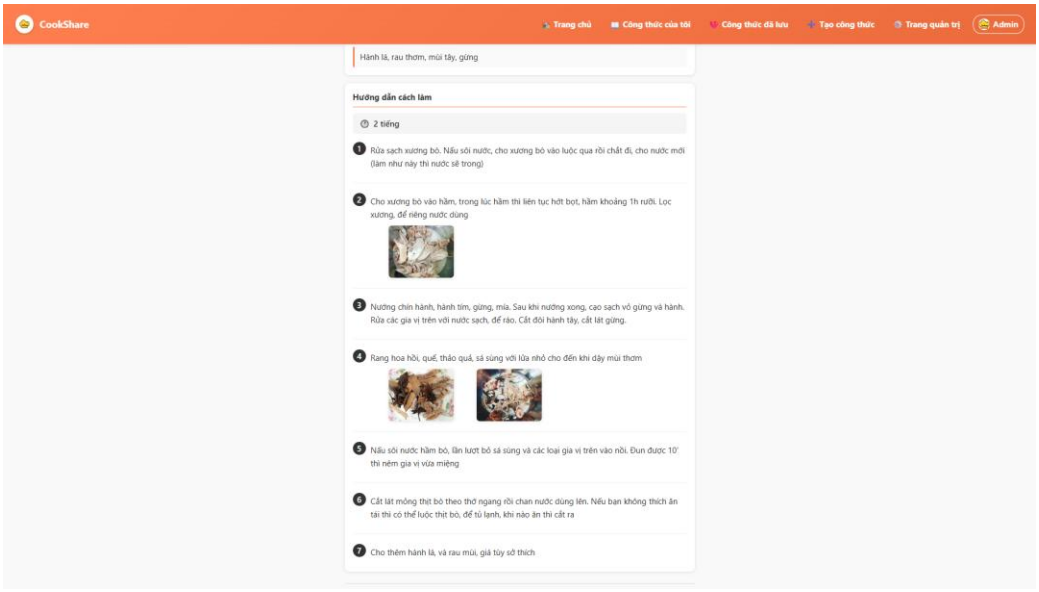
F.2: Trang chủ với danh sách công thức



Hình F.2 Trang chủ với danh sách công thức

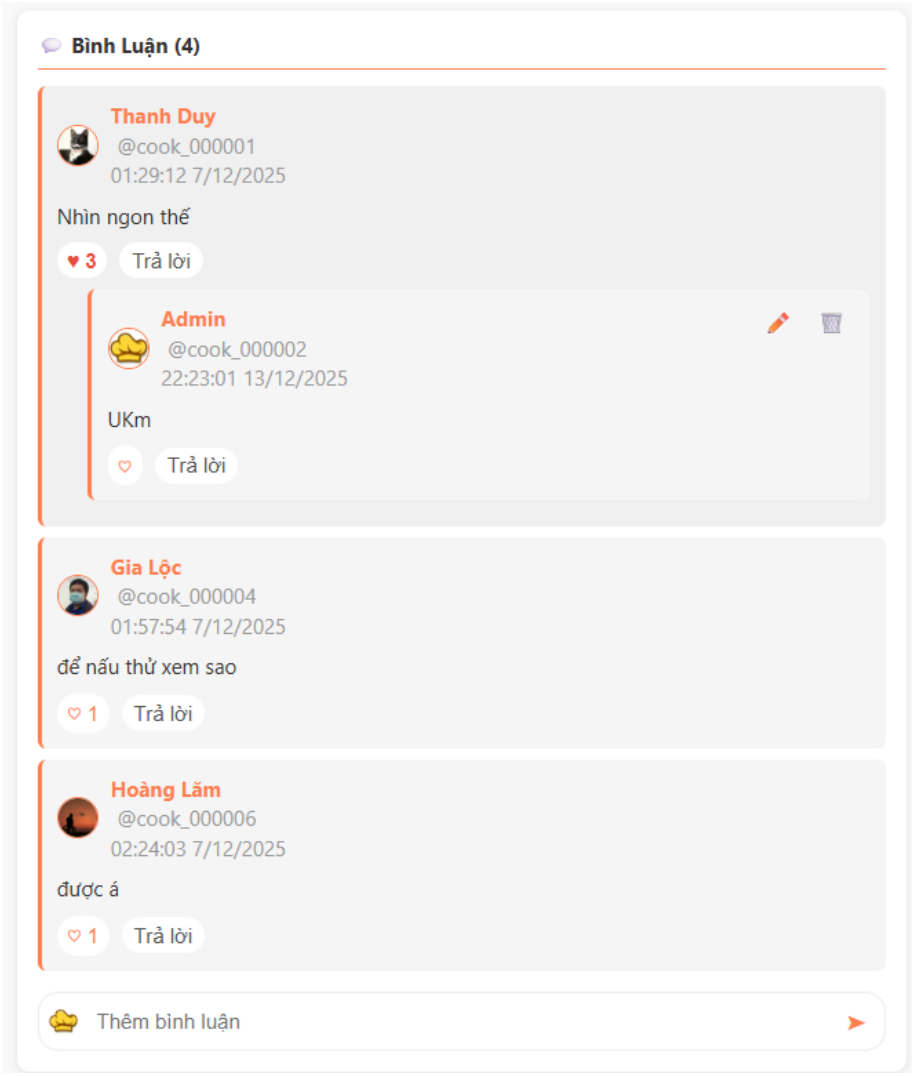
F.3: Trang chi tiết công thức





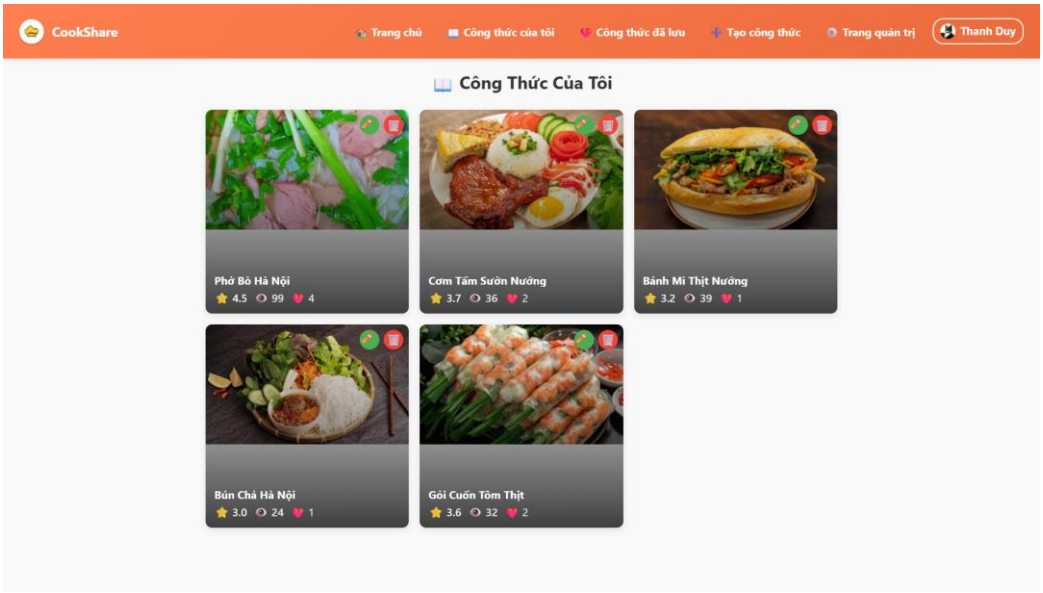
Hình F.3 Trang chủ với danh sách công thức

F.4: Nested comments & Like button trên bình luận (bình luận lồng nhau với indent và nút Reply & ♥/♡ icon với số lượt thích)



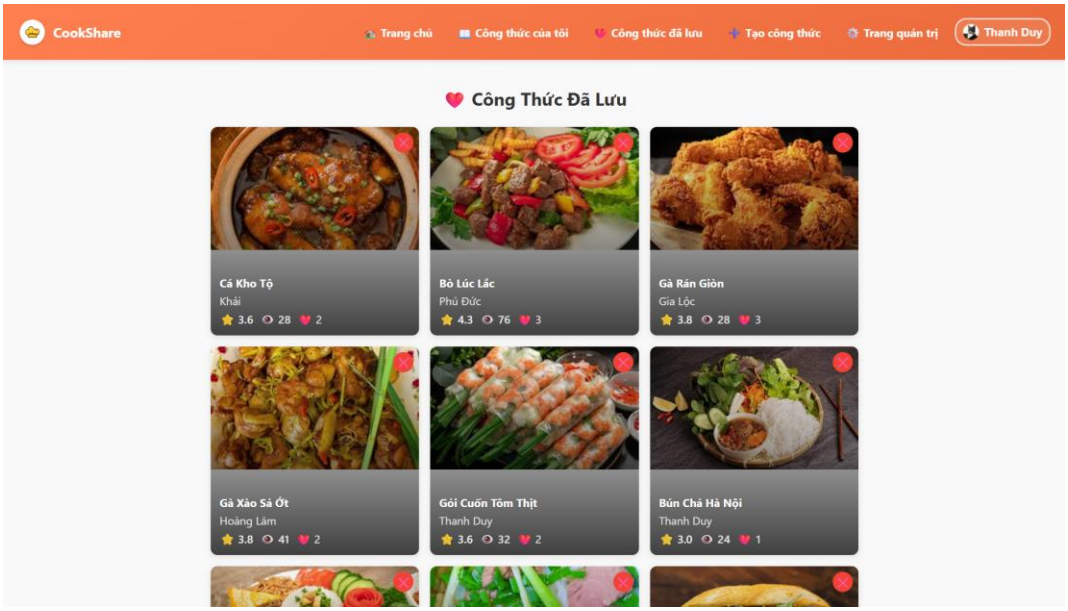
Hình F.4. Nested comments & Like button trên bình luận

F.5 Trang Công thức của tôi



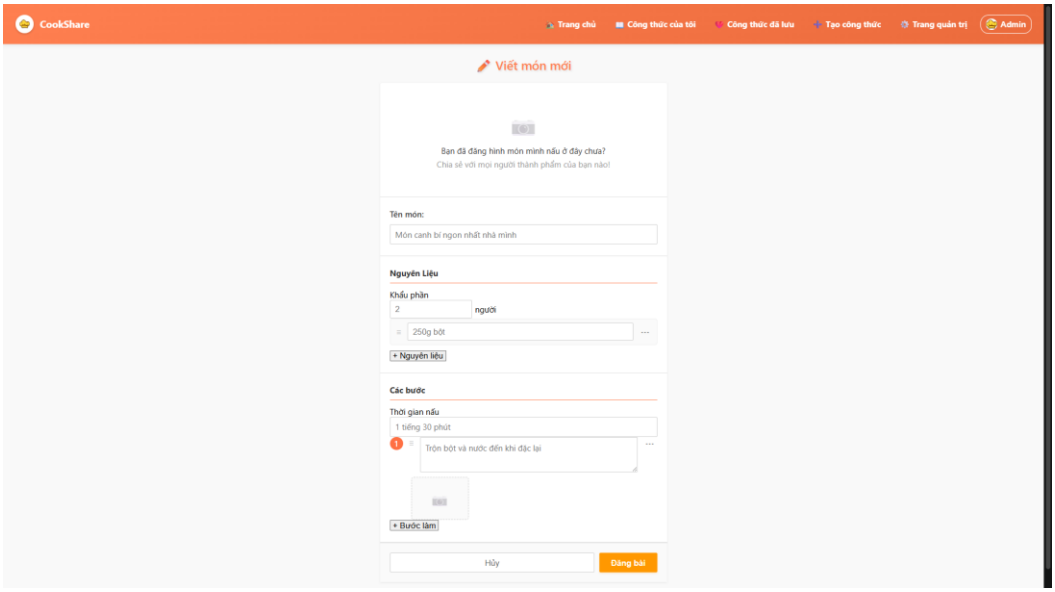
Hình F.5 Trang Công thức của tôi

F.6 Công thức đã lưu



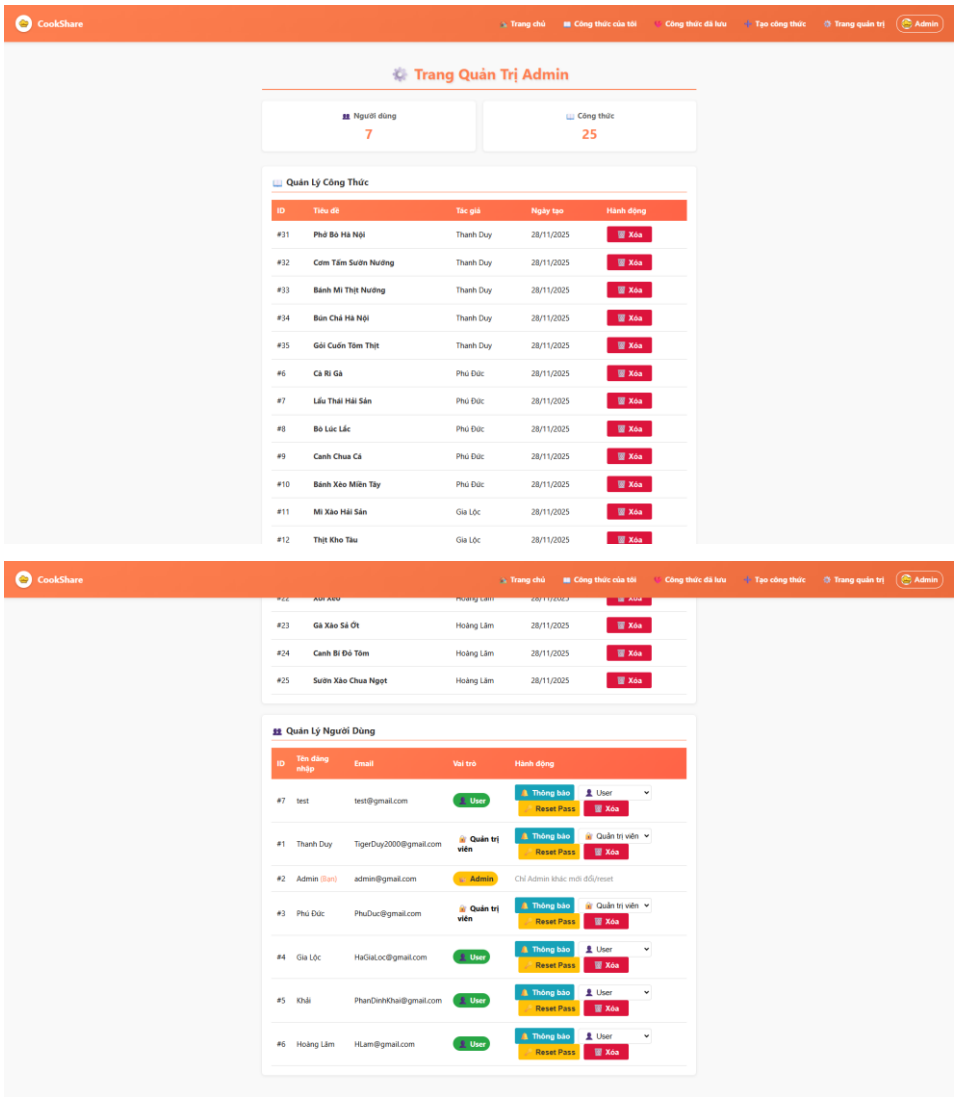
Hình F.6 Công thức đã lưu

F.7: Form tạo công thức



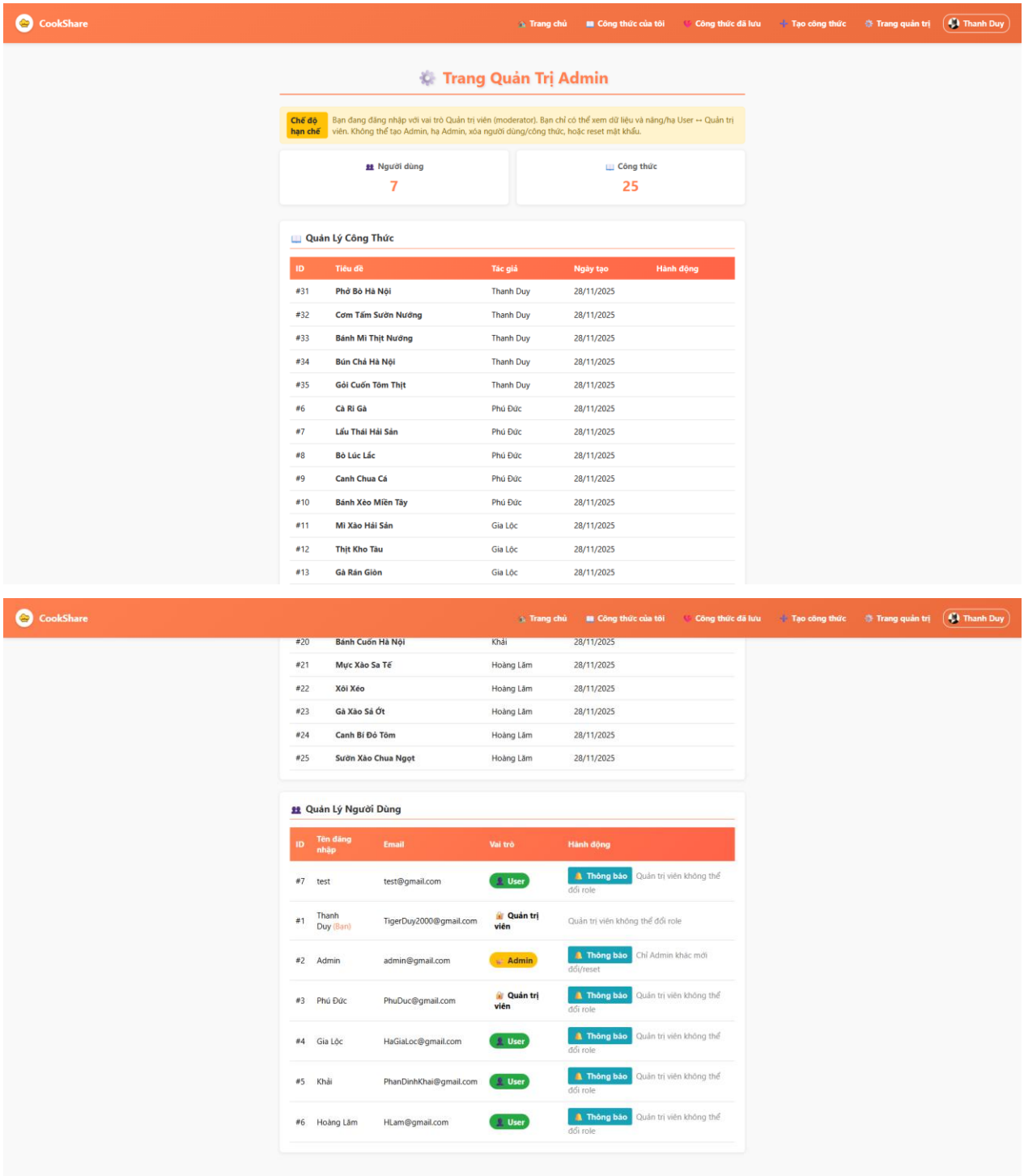
Hình F.7 Form tạo công thức

F.8: Trang quản trị (Admin Dashboard)



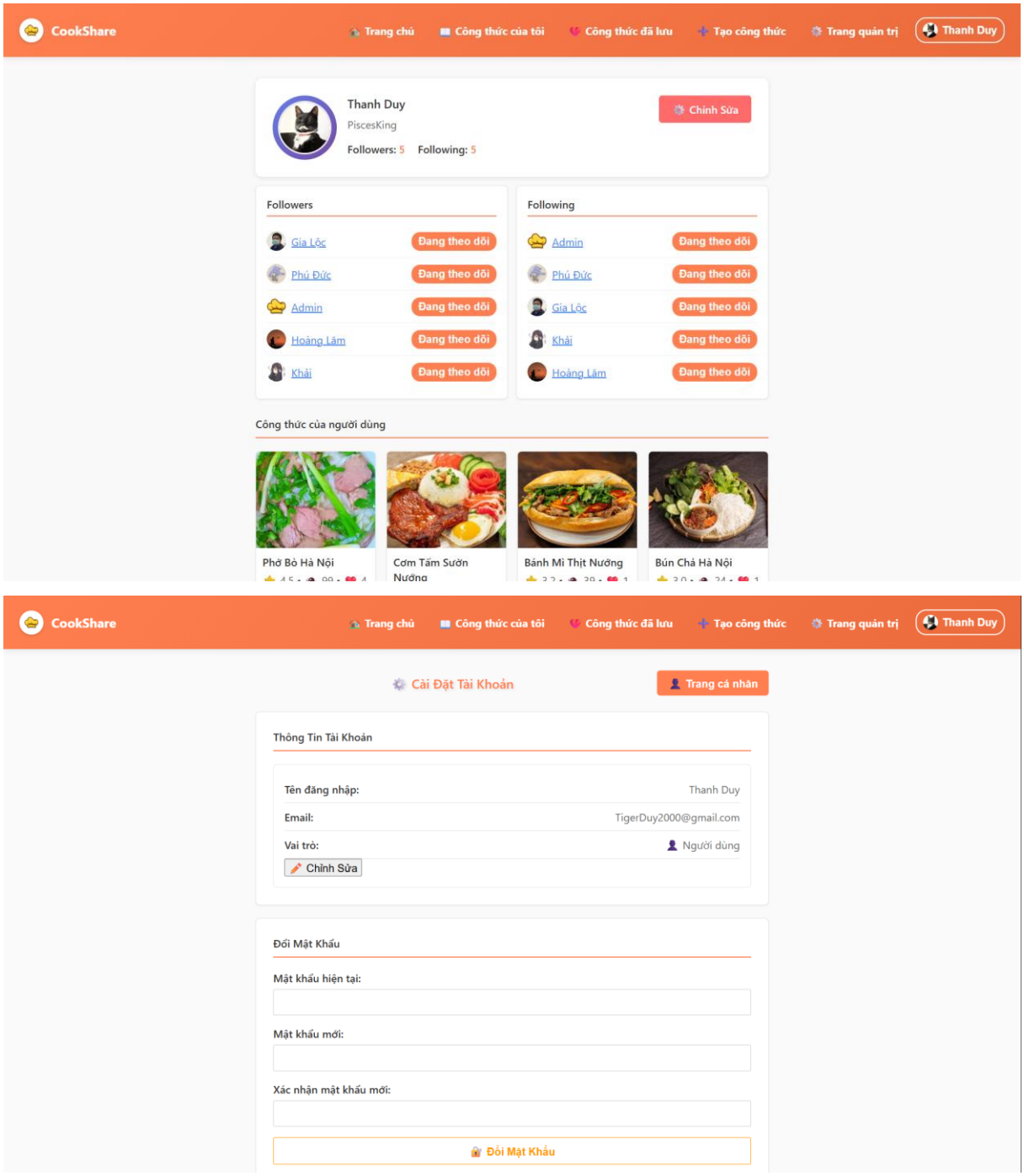
Hình F.8: Trang quản trị (Admin Dashboard)

F.9: Chế độ hạn chế dành cho Quản trị viên



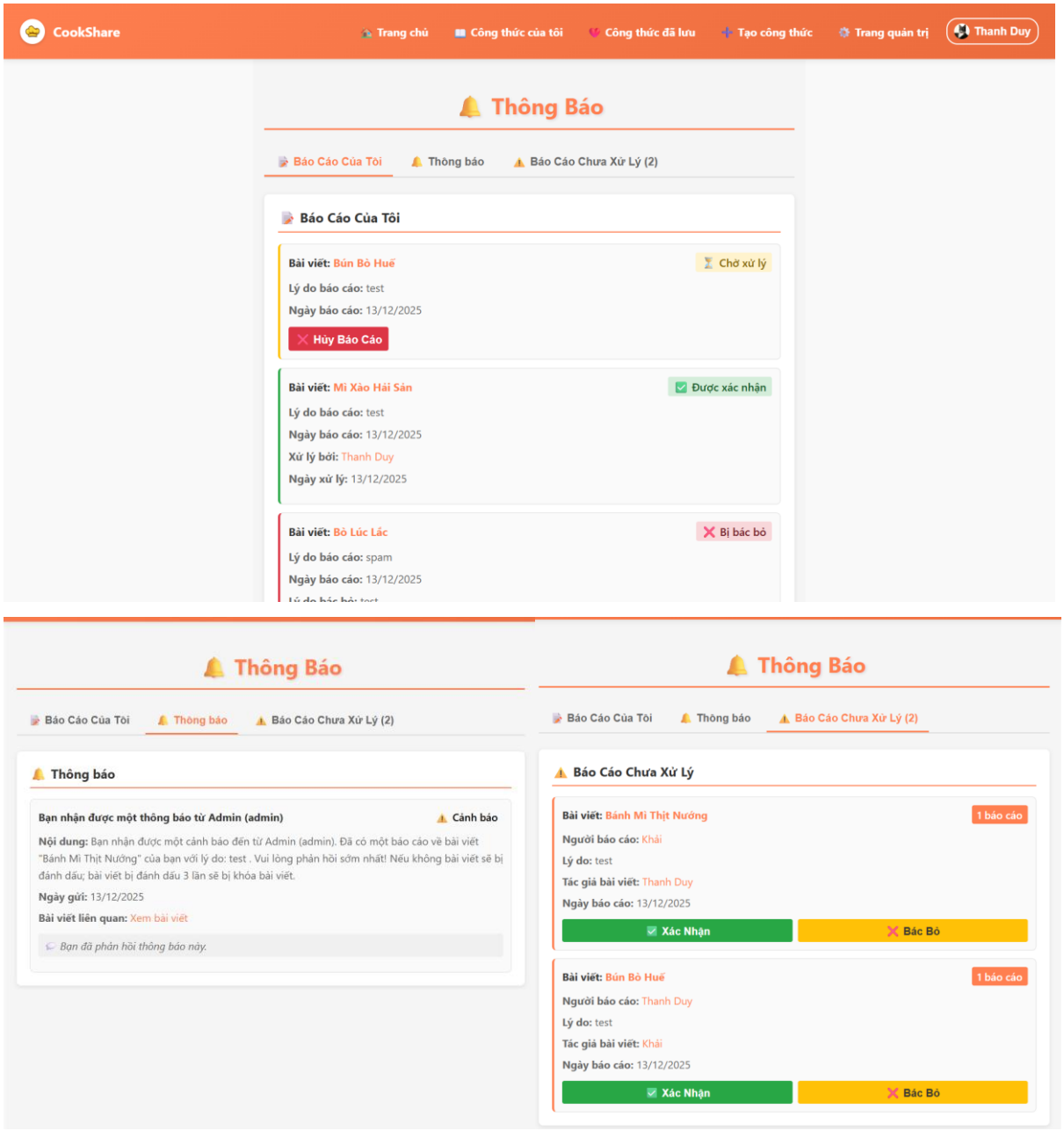
Hình F.9: Chế độ hạn chế dành cho Quản trị viên

F.10: Trang cá nhân & Cài đặt tài khoảng

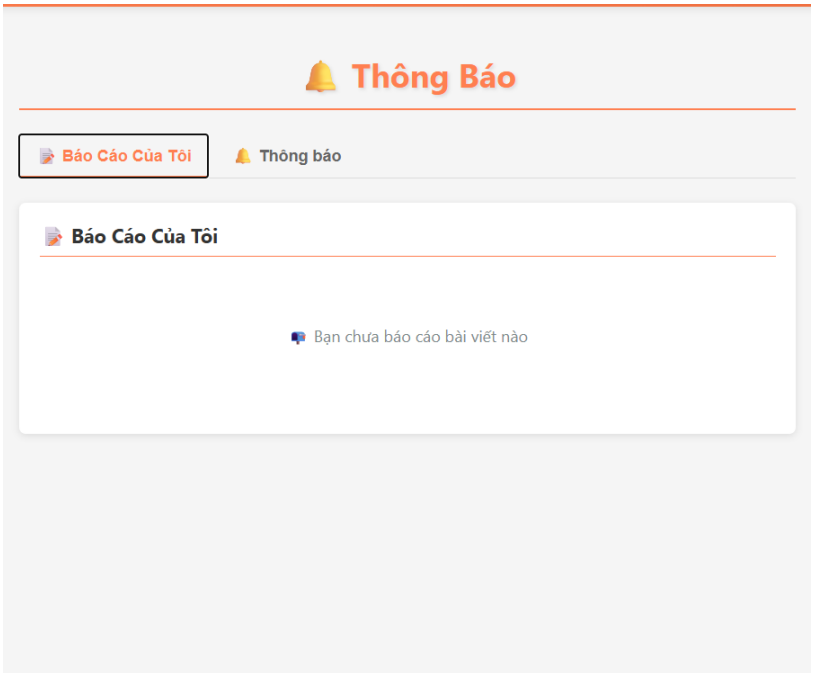


Hình F.10: Trang cá nhân & Cài đặt tài khoảng

F.11 Trang thông báo(Quản trị viên và Admin)

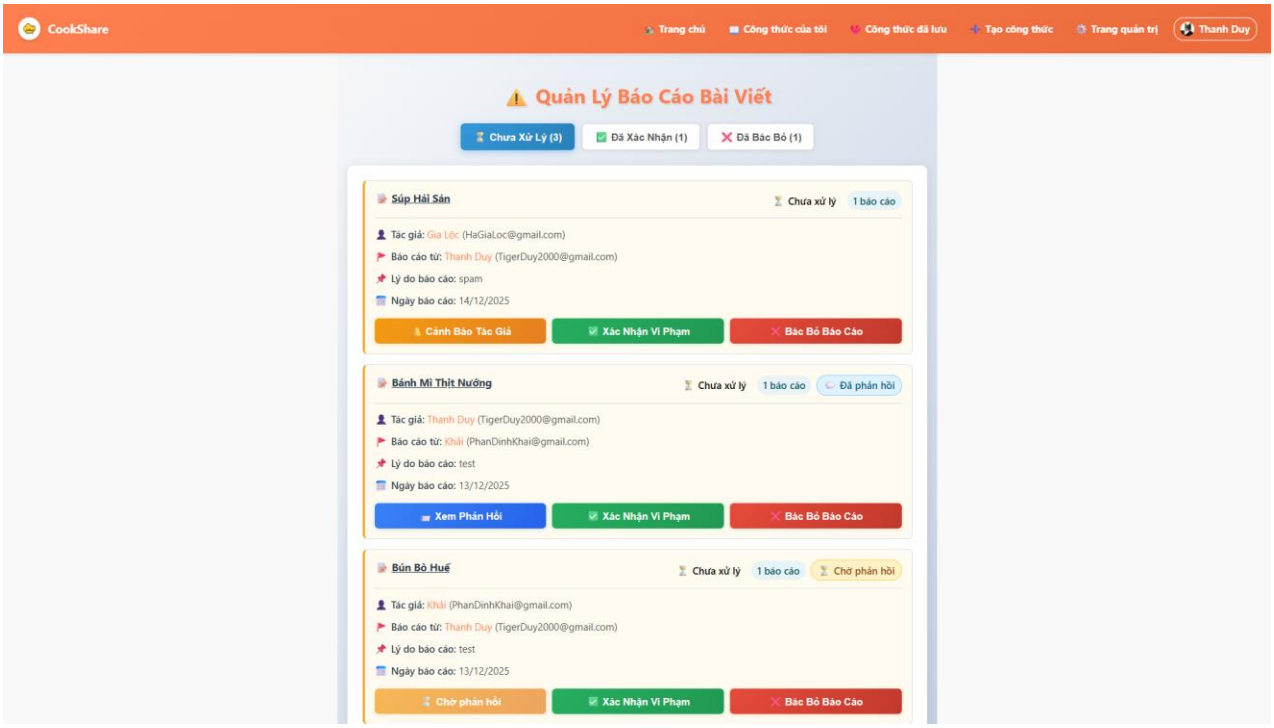


Hình F.11: Trang cá nhân & Cài đặt tài khoảng(Quản trị viên và Admin)



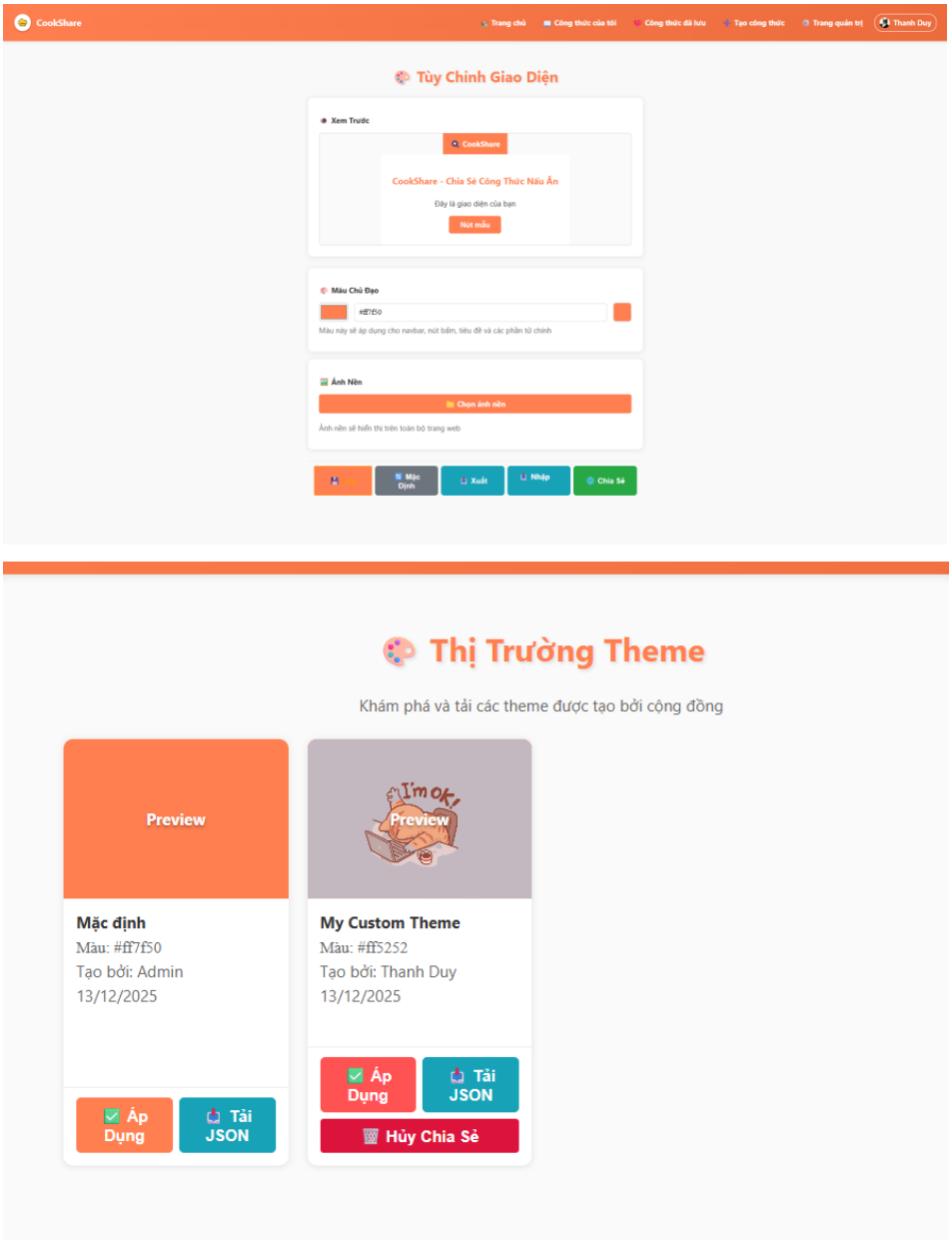
Hình F.11.1: Trang cá nhân & Cài đặt tài khoảng(User)

F.12: Trang quản lý báo cáo bài viết dành cho Quản trị viên & Admin



Hình F.12: Trang quản lý báo cáo bài viết dành cho Quản trị viên & Admin

F.13 Tùy chỉnh giao diện



Hình F.13 Tùy chỉnh giao diện

PHỤ LỤC G: Kiểm thử

G.1 Test cases chính

ID	Chức năng	Input	Expected Output	Kết quả
TC01	Đăng ký	Valid email, password	201, message success	✓ Pass
TC02	Đăng ký	Duplicate email	400, email exists	✓ Pass
TC03	Đăng nhập	Valid credentials	200, JWT token	✓ Pass
TC04	Đăng nhập	Invalid password	400, wrong password	✓ Pass
TC05	Tạo công thức	Valid data + token	201, recipe created	✓ Pass
TC06	Tạo công thức	No token	401, unauthorized	✓ Pass
TC07	Xóa công thức	Owner	200, deleted	✓ Pass
TC08	Xóa công thức	Not owner	403, forbidden	✓ Pass
TC09	Đánh giá	Valid rating 1-5	201, rated	✓ Pass
TC10	Đánh giá lần 2	Duplicate rating	400, already rated	✓ Pass
TC11	Admin đổi role	Admin token	200, role changed	✓ Pass
TC12	User đổi role	User token	403, not admin	✓ Pass
TC13	Nested comment	Reply với parent_id	201, reply created	✓ Pass
TC14	Nested invalid	Parent không tồn tại	400, invalid parent	✓ Pass
TC15	Like comment lần đầu	Valid comment_id	200, {liked: true}	✓ Pass
TC16	Unlike comment	Like lại	200, {liked: false}	✓ Pass
TC17	Xóa reply - owner	Owner xóa reply	200, deleted	✓ Pass
TC18	Xóa reply - admin	Admin xóa reply	200, deleted	✓ Pass
TC19	Lấy nested comments	GET /recipe/comment	200, [nested array]	✓ Pass
TC20	Notification on reply	User A reply → B	201, noti created	✓ Pass