

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 Thành Duy
MSSV: 110122062
Lớp: DA22TTD

Vĩnh Long, tháng 12 năm 2025

TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐO 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 Thành 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, đặc biệt là sự giúp đỡ của Cô Hà Thị Thúy Vi, nhờ có sự giúp đỡ của Cô mà em mới có thể thực hiện và hoàn thành dự án. Một lần nữa, xin chân thành 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	1
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).....	5
2.4.3 Follow System (Hệ thống theo dõi).....	6
2.4.4 Report Quota System (Hệ thống quota báo cáo)	6
2.4.5 Violation Auto-Block System (Hệ thống tự động khóa khi vi phạm)	6
2.4.6 Broadcast Notification System (Hệ thống thông báo hàng loạt)	6
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU	7
3.1 Đặc tả nhu cầu.....	7
3.2 Thiết kế hệ thống	10
3.2.1 Kiến trúc tổng thể	10
3.2.2 Mô hình dữ liệu.....	10
3.2.3 Thiết kế API (REST)	20
3.2.4 Thiết kế giao diện (UI/UX).....	21
3.3 Hiện thực hóa (Implementation).....	21
3.3.1 Frontend (React)	21
3.3.2 Backend (Node.js + Express)	22

3.3.3 Database (MySQL).....	23
3.4 Bảo mật và quản lý phiên	24
3.5 Thiết kế chi tiết nested comments và like system.....	25
3.5.1 Nested Comments	25
3.5.2 Like System	25
3.5.3 Follow System	25
3.5.4 Report System.....	26
3.5.5 Notification System	26
3.5.6 Theme System.....	26
3.6 Kiểm thử	26
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU.....	27
4.1 Kết quả chức năng	27
4.2 Trải nghiệm người dùng (UX/UI).....	30
4.3 Hiệu năng và kỹ thuật	30
4.4 Bảo mật	31
4.5 Giao diện minh họa (mô tả nhanh)	31
4.6 Đánh giá nhanh	32
4.7 Hướng phát triển	32
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	33
5.1 Kết luận.....	33
5.2 Hạn chế	33
5.2 Hướng phát triển	33
DANH MỤC TÀI LIỆU THAM KHẢO.....	34
PHỤ LỤC	35

DANH MỤC HÌNH ẢNH

Hình 3.1 Mô hình dữ liệu ERD.....	15
Hình F.1 Trang đăng nhập	47
Hình F.2 Trang đăng ký	47
Hình F.2 Trang chủ.....	47
Hình F.3 Trang chi tiết công thức & Nested comments & Like button trên bình luận	48
Hình F.5 Trang Công thức của tôi	48
Hình F.6 Công thức đã lưu	49
Hình F.7 Form tạo công thức.....	49
Hình F.8: Trang quản trị (Admin Dashboard)	50
Hình F.9: Chế độ hạn chế dành cho Quản trị viên.....	51
Hình F.10: Trang cá nhân	52
Hình F.11: Cài đặt tài khoản.....	53
Hình F.11: Trang cá nhân & Cài đặt tài khoản(Quản trị viên và Admin)	53
Hình F.12: Trang cá nhân & Cài đặt tài khoản(User).....	54
Hình F.13: Trang quản lý báo cáo bài viết dành cho Quản trị viên & Admin.....	54
Hình F.14 Tùy chỉnh giao diện	55
Hình F.15: Quy tắc cộng đồng.....	56

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.....	4
Bảng 3.1: Danh sách API endpoints AUTH	7
Bảng 3.2: Danh sách API endpoints RECIPE	8
Bảng 3.3: Danh sách API endpoints COMMENT	8
Bảng 3.4: Danh sách API endpoints RATING	8
Bảng 3.5: Danh sách API endpoints FAVORITE	8
Bảng 3.6: Danh sách API endpoints FOLLOW	8
Bảng 3.7: Danh sách API endpoints REPORT.....	8
Bảng 3.7: Danh sách API endpoints NOTIFICATION	9
Bảng 3.8: Danh sách API endpoints THEME	9
Bảng 3.9: Danh sách API endpoints ADMIN.....	9
Bảng 3.10: Schema cơ sở dữ liệu	15
Bảng 3.11: Phân quyền user vs admin.....	24
Bảng 4.1: Kết quả kiểm thử chức năng.....	27
Bảng 4.2: Đánh giá hiệu năng.....	30
Bảng PL.1: Môi trường phát triển.....	35
Bảng G.1: Test cases chính.....	56

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

VĂN 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, theo dõi người dùng)
- Cung cấp tìm kiếm và phân loại công thức hiệu quả
- Triển khai hệ thống quản lý admin/moderator để kiểm duyệt nội dung
- Hệ thống báo cáo vi phạm đa dạng (bài viết, bình luận, người dùng)
- Tùy chỉnh giao diện cá nhân và chia sẻ theme
- 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 (nguoi_dung, cong_thuc, binh_luan, danh_gia, favorite, follow, bao_cao, notifications, ...)
- View: React components (Home, CreateRecipe, RecipeDetail, AdminDashboard, ThemeCustomization, v.v.)
- Controller: Express routes (auth.js, recipe.js, rating.js, favorite.js, admin.js, follow.js, report.js, notification.js, theme.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 và backend
- Middleware xác thực để kiểm tra quyền (user/moderator/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
- Scheduled Tasks: Tác vụ tự động chạy định kỳ (reset quota, xóa bài vi phạm)

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 với avatar, bio
- Quên mật khẩu (gửi OTP qua email)
- Phân biệt vai trò: user / moderator / admin

2. Chức năng Quản lý Công Thức:

- Tạo công thức (tiêu đề, nguyên liệu, bước nấu, ảnh, khẩu phần, thời gian nấu)
- Ảnh cho từng bước nấu (step images)
- Danh sách công thức với phân trang
- Xem chi tiết công thức với đếm lượt xem
- Chính sửa công thức (chỉ chủ sở hữu)
- Xóa công thức (chủ sở hữu hoặc admin/moderator)
- Tìm kiếm công thức theo từ khóa

3. Chức năng Tương tác & Bình luận:

- Bình luận trên công thức (tạo, xem, xóa)
- Bình luận lồng nhau (nested comments) trả lời bình luận
- Thích bình luận (like system)
- Đánh giá sao (1-5 sao, mỗi user chỉ được đánh giá 1 lần)
- Yêu thích công thức (lưu vào danh sách yêu thích)

4. Chức năng Theo dõi (Follow System):

- Theo dõi/hủy theo dõi người dùng khác
- Xem danh sách followers/following
- Đếm số lượng followers/following
- Xem profile người dùng khác

5. Chức năng Báo cáo Vi phạm:

- Báo cáo bài viết vi phạm
- Báo cáo bình luận vi phạm
- Báo cáo người dùng vi phạm
- Hệ thống quota báo cáo (giới hạn số lần báo cáo)
- Upload ảnh bằng chứng khi báo cáo
- Tự động khóa tính năng khi vi phạm nhiều lần
- Gửi email thông báo khi xử lý báo cáo

6. Chức năng Thông báo:

- Thông báo cá nhân từ admin/moderator
- Thông báo hàng loạt (broadcast) đến tất cả người dùng
- Trả lời thông báo
- Đánh dấu đã đọc
- Đếm thông báo chưa đọc

7. Chức năng Tùy chỉnh Giao diện:

- Thay đổi màu chủ đạo
- Thay đổi hình nền
- Lưu và export theme
- Chia sẻ theme lên marketplace
- Tải theme từ marketplace

8. Chức năng Quản lý Admin/Moderator:

- Xem danh sách tất cả người dùng
- Thay đổi vai trò người dùng (user ↔ moderator ↔ admin)
- Xóa người dùng hoặc công thức vi phạm
- Ân bài viết thủ công
- Xử lý báo cáo (chấp nhận/bác bỏ)
- Gửi thông báo đến người dùng
- Gửi thông báo hàng loạt

ĐÁNH GIÁ & HẠN CHẾ

Điểm mạnh:

- Kiến trúc module hóa, dễ mở rộng
- Bảo mật tốt (Bcrypt, JWT, CORS, Middleware)
- Giao diện responsive, thân thiện với người dùng
- Hệ thống báo cáo và kiểm duyệt hoàn chỉnh
- Tính năng xã hội phong phú (follow, like, comment)
- Tùy chỉnh giao diện cá nhân

Hạn chế & hướng phát triển trong tương lai:

- Chưa triển khai Redis để cache dữ liệu
- Chưa có hệ thống notification real-time (WebSocket)
- Chưa triển khai unit test toàn diện
- Chưa có CI/CD pipeline
- Có thể thêm Machine Learning để gợi ý công thức

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 cơ bản: đăng ký/đăng nhập, tạo–chỉnh sửa–xóa công thức, tìm kiếm, đánh giá, bình luận, yêu thích, theo dõi người dùng
- 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
- Xây dựng hệ thống báo cáo vi phạm và kiểm duyệt nội dung hoàn chỉnh
- Đề xuất kiến trúc mở rộng, dễ bảo trì

3. Đối tượng nghiên cứu

- **Người dùng cuối:** Cá nhân đam mê nấu ăn, muốn chia sẻ hoặc tìm kiếm công thức
- **Kiểm duyệt viên (moderator):** Hỗ trợ admin kiểm duyệt nội dung, xử lý báo cáo
- **Quản trị viên (admin):** Quản trị hệ thống, quản lý người dùng, phân quyền
- **Công nghệ và giải pháp:** React (UI), React Router, Axios, CSS responsive; Node.js/Express (API), JWT, Bcrypt; MySQL; Cloudinary; Nodemailer

4. Phạm vi nghiên cứu

Chức năng người dùng:

- Đăng ký, đăng nhập, quản lý profile (avatar, bio)
- Quản lý công thức (tạo/sửa/xóa) với ảnh từng bước
- Bình luận nested, đánh giá, yêu thích
- Theo dõi người dùng khác
- Báo cáo vi phạm (bài viết/bình luận/người dùng)
- Tùy chỉnh giao diện cá nhân
- Nhận thông báo

Chức năng kiểm duyệt viên (moderator):

- Xử lý báo cáo vi phạm
- Ẩn bài viết/xóa bình luận vi phạm
- Gửi thông báo đến người dùng

Chức năng quản trị viên (admin):

- Toàn bộ quyền của moderator
- Quản lý người dùng, đổi vai trò
- Xóa tài khoản

Hệ thống:

- Frontend: React (localhost:3000)
- Backend: Node/Express (localhost:3001)
- Database: MySQL
- Lưu trữ ảnh: Cloudinary
- Gửi email: Nodemailer

Ngoài phạm vi:

- Realtime notifications (WebSocket)
- Gợi ý cá nhân hóa bằng ML
- Cache/Redis
- CI/CD tự động

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, theo dõi)
 - + Được kiểm duyệt nội dung và báo cáo vi phạm

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
- Bảo mật và quyền riêng tư chưa luôn được đảm bảo
- Thiếu cơ chế quản trị để xử lý nội dung vi phạm
- Thiếu tính năng tương tác xã hội (theo dõi, thích bình luận)
- Thiếu khả năng tùy chỉnh giao diện cá nhân

1.3 Mục tiêu của hệ thống

Xây dựng website CookShare với các chức năng:

- **Xác thực & Phân quyền:** Đăng ký/đăng nhập, JWT, mã hóa mật khẩu, phân quyền user/moderator/admin
- **Quản lý công thức:** Tạo, chỉnh sửa, xóa, xem chi tiết, tìm kiếm, ảnh từng bước
- **Tương tác:** Bình luận (nested), đánh giá sao, yêu thích, thích bình luận
- **Theo dõi:** Follow/unfollow người dùng, xem profile
- **Báo cáo:** Báo cáo bài viết/bình luận/người dùng, hệ thống quota
- **Thông báo:** Thông báo cá nhân, broadcast, trả lời thông báo
- **Tùy chỉnh giao diện:** Theme cá nhân, marketplace chia sẻ theme
- **Quản trị:** Kiểm duyệt nội dung, xử lý báo cáo, quản lý người dùng

1.4 Phạm vi nghiên cứu

Trong phạm vi:

- **Chức năng người dùng:** quản lý tài khoản, công thức, tương tác, theo dõi
- **Chức năng báo cáo:** báo cáo bài viết/bình luận/người dùng
- **Chức năng thông báo:** cá nhân và hàng loạt
- **Chức năng tùy chỉnh giao diện**
- **Chức năng quản trị:** duyệt/xóa nội dung, thay đổi vai trò

- **Hệ tầng:** frontend React, backend Node/Express, MySQL, Cloudinary

Ngoài phạm vi:

- Realtime notification (WebSocket)
- Gợi ý cá nhân hóa bằng ML
- Cache/Redis
- CI/CD tự động

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, JWT (7 ngày), CORS
- **Hiệu năng:** Non-blocking I/O, Cloudinary, Scheduled Tasks

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
- Theo dõi người dùng khác, xem profile
- Báo cáo nội dung vi phạm
- Tùy chỉnh giao diện cá nhân
- 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

Mô hình RESTful API

- Tài nguyên được ánh xạ vào URL; thao tác CRUD qua HTTP methods
- 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
- **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)
409	Conflict	Xung đột dữ liệu	Báo cáo đã tồn tại
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	Nhỏ (chỉ session ID)	Năng 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
- Preflight (OPTIONS) kiểm tra Access-Control-Allow-Origin/ Methods/ Headers/Credentials.
- Cho phép gửi JWT qua header khi được cấu hình.

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

Thành phần	Công nghệ	Phiên bản	Vai trò
	Express	4.x	Web framework, REST API, middleware
	jsonwebtoken	9.x	Tạo và verify JWT token
	bcryptjs	2.x	Hash mật khẩu với Salt
	Multer	1.x	Upload file
	nodemailer	6.x	Gửi email
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 các bảng.
- 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 `binh_luan`
- Lợi ích: Người dùng có thể trả lời trực tiếp bình luận của người khác.
- 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.

2.4.3 Follow System (Hệ thống theo dõi)

- Mô hình quan hệ nhiều-nhiều: Một user có thể follow nhiều user khác và được follow bởi nhiều user
- Bảng `follow` với 2 khóa ngoại: follower_id (người theo dõi) và following_id (người được theo dõi)
- Ràng buộc UNIQUE (follower_id, following_id) đảm bảo không follow trùng
- Kiểm tra không cho phép follow chính mình
- Query đếm followers/following sử dụng COUNT với điều kiện phù hợp

2.4.4 Report Quota System (Hệ thống quota báo cáo)

- Mục đích: Chống spam báo cáo, đảm bảo người dùng báo cáo có trách nhiệm
- Mỗi user có 3 lượt báo cáo cho mỗi loại (recipe/comment/user)
- Quota được hoàn lại khi báo cáo được xử lý (accept hoặc reject)
- Nếu báo cáo bị reject quá nhiều (3 lần/tuần), user bị khóa tính năng báo cáo 30 ngày
- Bảng `user_report_quota` theo dõi số lượt còn lại theo từng loại

2.4.5 Violation Auto-Block System (Hệ thống tự động khóa khi vi phạm)

- Theo dõi số vi phạm trong khoảng thời gian (7 ngày cho bài viết, 1 tháng cho bình luận)
- Nguồn gốc khóa:
 - + 3 bài viết bị ẩn trong tháng → khóa đăng bài 30 ngày
 - + 3 bình luận vi phạm trong tháng → khóa bình luận 30 ngày
 - + 3 báo cáo bị bác bỏ trong tuần → khóa báo cáo 30 ngày
- Tự động mở khóa khi hết hạn (kiểm tra tại middleware hoặc scheduled task)
- Gửi email thông báo khi bị khóa

2.4.6 Broadcast Notification System (Hệ thống thông báo hàng loạt)

- Khác với thông báo cá nhân (1-1), broadcast gửi đến tất cả user
- Bảng riêng `broadcast_notifications` lưu nội dung broadcast
- Bảng `user_broadcast_read` theo dõi user nào đã đọc broadcast nào
- Khi fetch thông báo: kết hợp personal notifications và broadcast chưa đọc
- Tối ưu: không tạo N bản ghi cho N user, chỉ tạo 1 bản ghi broadcast

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, theo dõi người dùng khác, báo cáo vi phạm, tùy chỉnh giao diện
- **Kiểm duyệt viên (moderator):** Toàn bộ quyền của user + xử lý báo cáo, ẩn bài viết vi phạm, gửi thông báo
- **Quản trị viên (admin):** Toàn bộ quyền của moderator + đổi vai trò người dùng, xóa tài khoản, quản lý toàn hệ thống

Chức năng chính:

- **Xác thực & phân quyền:** đăng ký, đăng nhập, JWT 7 ngày, phân biệt user/moderator/admin
- **Quản lý công thức:** CRUD, tải ảnh lên Cloudinary, tìm kiếm, ảnh từng bước
- **Tương tác:** bình luận (nested), đánh giá sao, yêu thích, thích bình luận
- **Theo dõi:** follow/unfollow, xem profile người dùng
- **Báo cáo:** báo cáo bài viết/bình luận/người dùng, quota system
- **Thông báo:** cá nhân, broadcast, reply
- **Tùy chỉnh giao diện:** theme cá nhân, marketplace
- **Quản trị:** quản lý người dùng, xử lý báo cáo, ẩn bài viết

Phi chức năng:

- Bảo mật: Bcrypt, JWT, CORS, middleware phân quyền
- Hiệu năng: Non-blocking I/O, tách lưu trữ ảnh sang cloud
- UX/UI: Responsive, điều hướng mượt, thông báo rõ ràng

Bảng 3.1: Danh sách API endpoints AUTH

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	/auth/profile	JWT	Lấy thông tin profile	200: {user}
PUT	/auth/profile	JWT	Cập nhật profile	200: success
POST	/auth/forgot-password	No	Gửi OTP reset password	200: success

Bảng 3.2: Danh sách API endpoints RECIPE

Method	Endpoint	Auth	Mô tả	Response
GET	/recipe/list	No	Danh sách công thức	200: [recipes]
GET	/recipe/search?q=	No	Tìm kiếm công thức	200: [filtered recipes]
GET	/recipe/:id	No	Chi tiết công thức	200: {recipe}
POST	/recipe/create	JWT	Tạo công thức + upload ảnh	201: {recipeId} / 401
PUT	/recipe/update/:id	JWT	Cập nhật công thức	200 / 403
DELETE	/recipe/delete/:id	JWT	Xóa công thức	200 / 403

Bảng 3.3: Danh sách API endpoints COMMENT

Method	Endpoint	Auth	Mô tả	Response
POST	/recipe/comment/:recipeId	JWT	Thêm bình luận (nested)	201
GET	/recipe/comment/:recipeId	No	Lấy bình luận + replies	200
DELETE	/recipe/comment/:id	JWT	Xóa bình luận	200 / 403
POST	/recipe/comment/:id/like	JWT	Like / Unlike bình luận	200

Bảng 3.4: Danh sách API endpoints RATING

Method	Endpoint	Auth	Mô tả	Response
POST	/rating/:recipeId	JWT	Đánh giá sao	201 / 400
GET	/rating/:recipeId	No	Lấy rating công thức	200

Bảng 3.5: Danh sách API endpoints FAVORITE

Method	Endpoint	Auth	Mô tả	Response
POST	/favorite/:recipeId	JWT	Thêm / xóa yêu thích	200
GET	/favorite/list	JWT	Danh sách yêu thích	200

Bảng 3.6: Danh sách API endpoints FOLLOW

Method	Endpoint	Auth	Mô tả	Response
POST	/follow/:userId	JWT	Theo dõi người dùng	200 / 400
DELETE	/follow/:userId	JWT	Hủy theo dõi	200
GET	/follow/is-following/:userId	JWT	Kiểm tra theo dõi	200
GET	/follow/counts/:userId	No	Đếm followers/following	200
GET	/follow/followers/:userId	No	Danh sách followers	200
GET	/follow/following/:userId	No	Danh sách following	200

Bảng 3.7: Danh sách API endpoints REPORT

Method	Endpoint	Auth	Mô tả	Response
POST	/report/recipe/:id	JWT	Báo cáo bài viết	200 / 403

Method	Endpoint	Auth	Mô tả	Response
POST	/report/comment/:id	JWT	Báo cáo bình luận	200
POST	/report/user/:id	JWT	Báo cáo người dùng	200
DELETE	/report/recipe/:id	JWT	Hủy báo cáo	200
GET	/report/quota	JWT	Lấy quota báo cáo	200
GET	/report/my-reports	JWT	Báo cáo của tôi	200
GET	/report	JWT (admin/mod)	Tất cả báo cáo	200
PUT	/report/:id/status	JWT (admin/mod)	Xử lý báo cáo	200

Bảng 3.7: Danh sách API endpoints NOTIFICATION

Method	Endpoint	Auth	Mô tả	Response
GET	/notification/my	JWT	Thông báo của tôi	200
PUT	/notification/:id/read	JWT	Đánh dấu đã đọc	200
PUT	/notification/broadcast/:id/read	JWT	Đánh dấu broadcast	200
POST	/notification/:id/reply	JWT	Trả lời thông báo	200
GET	/notification/unread-count	JWT	Đếm chưa đọc	200
POST	/notification/send	JWT (admin/mod)	Gửi thông báo	200
POST	/notification/broadcast	JWT (admin/mod)	Gửi broadcast	200

Bảng 3.8: Danh sách API endpoints THEME

Method	Endpoint	Auth	Mô tả	Response
GET	/theme/preferences	JWT	Lấy theme	200
POST	/theme/preferences	JWT	Lưu theme	200
POST	/theme/share	JWT	Chia sẻ theme	200
GET	/theme/marketplace	JWT	Marketplace	200
DELETE	/theme/share/:id	JWT	Xóa theme chia sẻ	200

Bảng 3.9: Danh sách API endpoints ADMIN

Method	Endpoint	Auth	Mô tả	Response
GET	/admin/users	JWT (admin)	Danh sách user	200
PUT	/admin/users/:id/role	JWT (admin)	Đổi vai trò	200
DELETE	/admin/users/:id	JWT (admin)	Xóa tài khoản	200
POST	/admin/recipes/:id/hide	JWT (admin/mod)	Ẩn bài viết	200

Method	Endpoint	Auth	Mô tả	Response
POST	/admin/recipes/:id/unhide	JWT (admin/mod)	Bỏ ẩn bài viết	200

3.2 Thiết kế hệ thống

3.2.1 Kiến trúc tổng thể

- **3 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

Bảng nguoi_dung (Users)

```sql

```
CREATE TABLE nguoi_dung (
 id INT PRIMARY KEY AUTO_INCREMENT,
 username VARCHAR(50) UNIQUE NOT NULL,
 email VARCHAR(100) UNIQUE NOT NULL,
 password VARCHAR(255) NOT NULL,
 role ENUM('user', 'moderator', 'admin') DEFAULT 'user',
 avatar_url VARCHAR(500) NULL,
 bio TEXT NULL,
 is_posting_blocked BOOLEAN DEFAULT FALSE,
 posting_blocked_until DATETIME NULL,
 is_commenting_blocked BOOLEAN DEFAULT FALSE,
 commenting_blocked_until DATETIME NULL,
 is_reporting_blocked BOOLEAN DEFAULT FALSE,
 reporting_blocked_until DATETIME NULL,
 monthly_post_violations INT DEFAULT 0,
 monthly_comment_violations INT DEFAULT 0,
 monthly_rejected_reports INT DEFAULT 0,
 lastViolationReset DATETIME DEFAULT CURRENT_TIMESTAMP,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```

Bảng cong_thuc (Recipes)

```
```sql
CREATE TABLE cong_thuc (
 id INT PRIMARY KEY AUTO_INCREMENT,
 user_id INT NOT NULL,
 title VARCHAR(255) NOT NULL,
 ingredients LONGTEXT NOT NULL,
 steps LONGTEXT NOT NULL,
 image_url VARCHAR(500) NULL,
 servings VARCHAR(100) NULL,
 cook_time VARCHAR(100) NULL,
 views INT DEFAULT 0,
 violation_count INT DEFAULT 0,
 is_hidden BOOLEAN DEFAULT FALSE,
 hidden_at DATETIME NULL,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 FOREIGN KEY (user_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE
);
```

```

Bảng binh_luan (Comments với nested)

```
```sql
CREATE TABLE binh_luan (
 id INT PRIMARY KEY AUTO_INCREMENT,
 recipe_id INT NOT NULL,
 user_id INT NOT NULL,
 comment TEXT NOT NULL,
 parent_comment_id INT NULL,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 FOREIGN KEY (recipe_id) REFERENCES cong_thuc(id) ON DELETE CASCADE,
 FOREIGN KEY (user_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE,
 FOREIGN KEY (parent_comment_id) REFERENCES binh_luan(id) ON DELETE
CASCADE
);
```

```

Bảng follow (Theo dõi)

```
```sql
CREATE TABLE follow (
 id INT PRIMARY KEY AUTO_INCREMENT,
 follower_id INT NOT NULL,
 following_id INT NOT NULL,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 FOREIGN KEY (follower_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE,
 FOREIGN KEY (following_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE,
 UNIQUE KEY unique_follow (follower_id, following_id)
);
```

```

Bảng bao_cao (Reports)

```
```sql
CREATE TABLE bao_cao (
 id INT PRIMARY KEY AUTO_INCREMENT,
 user_id INT NOT NULL,
 reason TEXT NOT NULL,
 image_url VARCHAR(500) DEFAULT NULL,
 status ENUM('pending','accepted','rejected') DEFAULT 'pending',
 rejected_reason TEXT DEFAULT NULL,
 processed_by INT DEFAULT NULL,
 processed_at DATETIME DEFAULT NULL,
 target_type ENUM('recipe','comment','user') DEFAULT 'recipe',
 recipe_id INT DEFAULT NULL,
 comment_id INT DEFAULT NULL,
 reported_user_id INT DEFAULT NULL,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 FOREIGN KEY (user_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE,
 FOREIGN KEY (recipe_id) REFERENCES cong_thuc(id) ON DELETE CASCADE,
```

```
 FOREIGN KEY (comment_id) REFERENCES binh_luan(id) ON DELETE CASCADE,
 FOREIGN KEY (reported_user_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE,
 FOREIGN KEY (processed_by) REFERENCES nguoi_dung(id) ON DELETE SET NULL
);
...
;
```

### Bảng user\_report\_quota (Quota báo cáo)

```
```sql  
CREATE TABLE user_report_quota (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    user_id INT NOT NULL,  
    report_type ENUM('recipe','comment','user') NOT NULL,  
    remaining_reports INT DEFAULT 3,  
    last_reset DATETIME DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (user_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE,  
    UNIQUE KEY unique_user_report_type (user_id, report_type)  
);  
...  
;
```

Bảng notifications (Thông báo)

```
```sql  
CREATE TABLE notifications (
 id INT PRIMARY KEY AUTO_INCREMENT,
 sender_id INT NOT NULL,
 receiver_id INT NOT NULL,
 sender_role VARCHAR(20) DEFAULT NULL,
 type VARCHAR(50) DEFAULT 'manual',
 message TEXT NOT NULL,
 image_url VARCHAR(500) DEFAULT NULL,
 metadata JSON DEFAULT NULL,
 is_read BOOLEAN DEFAULT FALSE,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 FOREIGN KEY (sender_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE,
...
;
```

```
FOREIGN KEY (receiver_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE
);
```
```

```

### Bảng broadcast\_notifications (Thông báo hàng loạt)

```
```sql
CREATE TABLE broadcast_notifications (
    id INT PRIMARY KEY AUTO_INCREMENT,
    sender_id INT NOT NULL,
    message TEXT NOT NULL,
    image_url VARCHAR(500) DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (sender_id) REFERENCES nguoi_dung(id) ON DELETE CASCADE
);
```
```

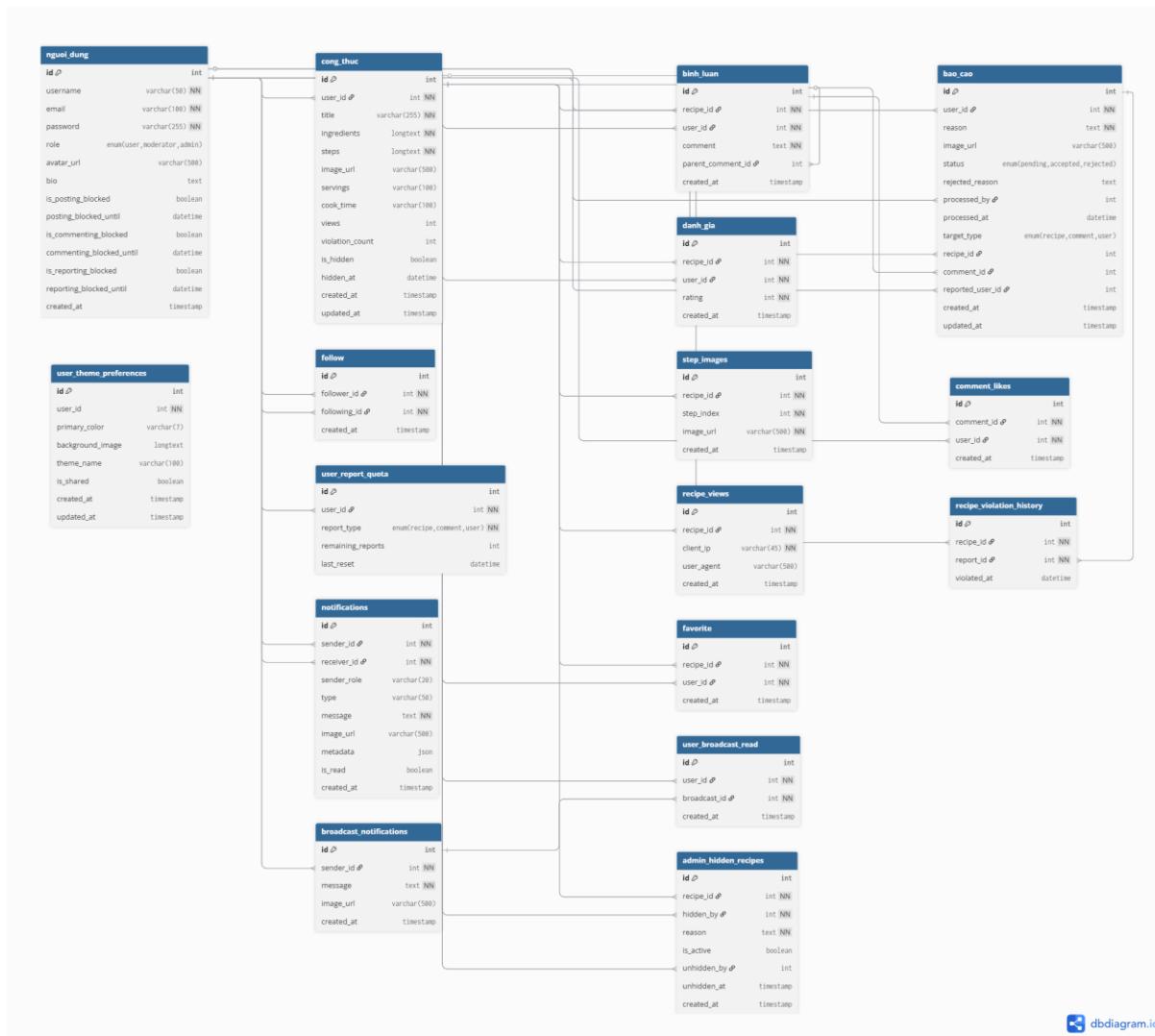
```

Bảng user_theme_preferences (Tùy chỉnh giao diện)

```
```sql
CREATE TABLE user_theme_preferences (
 id INT PRIMARY KEY AUTO_INCREMENT,
 user_id INT NOT NULL,
 primary_color VARCHAR(7) DEFAULT '#ff7f50',
 background_image LONGTEXT,
 theme_name VARCHAR(100),
 is_shared BOOLEAN DEFAULT FALSE,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 UNIQUE KEY unique_user_id (user_id)
);
```
```

```

### Mô hình dữ liệu ERD:



Hình 3.1 Mô hình dữ liệu ERD

Bảng 3.10: Schema cơ sở dữ liệu

Bảng	Trường	Kiểu dữ liệu	Ràng buộc
nguo_dung	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)

Bảng	Trường	Kiểu dữ liệu	Ràng buộc
cong_thuc	role	ENUM	'user', 'moderator', 'admin'
	avatar_url	VARCHAR(500)	NULL
	bio	TEXT	NULL
	is_posting_blocked	BOOLEAN	DEFAULT FALSE
	is_commenting_blocked	BOOLEAN	DEFAULT FALSE
	is_reporting_blocked	BOOLEAN	DEFAULT FALSE
	id	INT	PRIMARY KEY, AUTO_INCREMENT
	user_id	INT	FOREIGN KEY → nguo_dung(id)
	title	VARCHAR(255)	NOT NULL
	ingredients	LONGTEXT	NOT NULL
binh_luan	steps	LONGTEXT	NOT NULL
	image_url	VARCHAR(500)	NULL
binh_luan	servings	VARCHAR(100)	NULL
	cook_time	VARCHAR(100)	NULL
	views	INT	DEFAULT 0
	violation_count	INT	DEFAULT 0
	is_hidden	BOOLEAN	DEFAULT FALSE
	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN

Bảng	Trường	Kiểu dữ liệu	Ràng buộc
comment_likes			KEY → cong_thuc(id)
	user_id	INT	FOREIGN KEY → nguoi_dung(id)
	comment	TEXT	NOT NULL
	parent_comment_id	INT	FOREIGN KEY → binh_luan(id), NULL
danh_gia	id	INT	PRIMARY KEY, AUTO_INCREMENT
	comment_id	INT	FOREIGN KEY → binh_luan(id)
	user_id	INT	FOREIGN KEY → nguoi_dung(id) UNIQUE (comment_id, user_id)
danh_gia	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY → cong_thuc(id)
	user_id	INT	FOREIGN KEY → nguoi_dung(id)

Bảng	Trường	Kiểu dữ liệu	Ràng buộc
favorite	rating	INT	CHECK (1–5), UNIQUE (recipe_id, user_id)
	id	INT	PRIMARY KEY, AUTO_INCREMENT
	recipe_id	INT	FOREIGN KEY → cong_thuc(id)
	user_id	INT	FOREIGN KEY → nguo_dung(id) UNIQUE (recipe_id, user_id)
follow	id	INT	PRIMARY KEY, AUTO_INCREMENT
	follower_id	INT	FOREIGN KEY → nguo_dung(id)
	following_id	INT	FOREIGN KEY → nguo_dung(id) UNIQUE (follower_id, following_id)
	bao_cao	INT	PRIMARY KEY, AUTO_INCREMENT

Bảng	Trường	Kiểu dữ liệu	Ràng buộc
user_report_quota			MENT
	user_id	INT	FOREIGN KEY → nguo_dung(id)
	target_type	ENUM	'recipe', 'comment', 'user'
	status	ENUM	'pending', 'accepted', 'rejected'
	recipe_id	INT	FOREIGN KEY, NULL
	comment_id	INT	FOREIGN KEY, NULL
	reported_user_id	INT	FOREIGN KEY, NULL
	image_url	VARCHAR(500)	NULL (ảnh bằng chứng)
	processed_by	INT	FOREIGN KEY → nguo_dung(id), NULL
user_report_quota	user_id	INT	FOREIGN KEY → nguo_dung(id)
	report_type	ENUM	'recipe', 'comment', 'user'
	remaining_reports	INT	DEFAULT 3 UNIQUE (user_id, report_type)
notifications	id	INT	PRIMARY KEY, AUTO_INCRE

Bảng	Trường	Kiểu dữ liệu	Ràng buộc
broadcast_notifications			MENT
	sender_id	INT	FOREIGN KEY → nguoi_dung(id)
	receiver_id	INT	FOREIGN KEY → nguoi_dung(id)
	type	VARCHAR(50)	DEFAULT 'manual'
	message	TEXT	NOT NULL
	is_read	BOOLEAN	DEFAULT FALSE
user_theme_preferences	id	INT	PRIMARY KEY, AUTO_INCREMENT
	sender_id	INT	FOREIGN KEY → nguoi_dung(id)
	message	TEXT	NOT NULL
user_theme_preferences	user_id	INT	UNIQUE, FOREIGN KEY → nguoi_dung(id)
	primary_color	VARCHAR(7)	DEFAULT '#ff7f50'
	is_shared	BOOLEAN	DEFAULT FALSE

### 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, lướt 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 hóa (Implementation)

### 3.3.1 Frontend (React)

Công nghệ: React 19.x, React Router 6.x, Axios 1.x, CSS responsive.

Cấu trúc thư mục:

```
src/cookshare/src/
 ├── components/
 | ├── Navbar.jsx # Thanh điều hướng
 | ├── ProtectedRoute.jsx # Bảo vệ route cần đăng nhập
 | ├── FollowButton.jsx # Nút theo dõi
 | ├── FollowersList.jsx # Danh sách followers/following
 | ├── ReportButton.jsx # Nút báo cáo
 | ├── ProfileHeader.jsx # Header trang profile
 | ├── ImageLightbox.jsx # Xem ảnh phóng to
 | ├── BroadcastNotification.jsx # Thông báo hàng loạt
 | ├── AvatarCropper.jsx # Crop avatar
 | ├── Footer.jsx # Footer
 | └── RulesModal.jsx # Modal quy định
 └── pages/
 ├── Home.jsx # Trang chủ
 └── Login.jsx # Đăng nhập
```

```
| └── Register.jsx # Đăng ký
| └── ForgotPassword.jsx # Quên mật khẩu
| └── CreateRecipe.jsx # Tạo công thức
| └── EditRecipe.jsx # Sửa công thức
| └── RecipeDetail.jsx # Chi tiết công thức
| └── MyRecipes.jsx # Công thức của tôi
| └── FavoriteRecipes.jsx # Công thức yêu thích
| └── Search.jsx # Tìm kiếm
| └── Profile.jsx # Profile cá nhân
| └── UserProfile.jsx # Profile người khác
| └── AccountSettings.jsx # Cài đặt tài khoản
| └── Notifications.jsx # Thông báo
| └── ThemeCustomization.jsx # Tùy chỉnh giao diện
| └── ThemeMarketplace.jsx # Marketplace theme
| └── AdminDashboard.jsx # Quản trị người dùng
| └── AdminReports.jsx # Quản lý báo cáo
| └── Rules.jsx # Quy định cộng đồng
| └── styles/ # CSS files
| └── hooks/ # Custom hooks
| └── utils/ # Utility functions
````
```

3.3.2 Backend (Node.js + Express)

Công nghệ: Node.js 16+, Express 4.x, jsonwebtoken, bcryptjs, multer/cloudinary SDK, dotenv.

Cấu trúc:

```
```
src/backend/
| └── config/
| | └── db.js # Kết nối MySQL
| | └── cloudinary.js # Cấu hình Cloudinary
| | └── mailer.js # Cấu hình Nodemailer
| └── middleware/
| | └── auth.js # Middleware xác thực JWT, phân quyền
| └── routes/
```

```
| └── auth.js # Xác thực (register, login, profile)
| └── recipe.js # Quản lý công thức, bình luận
| └── rating.js # Đánh giá
| └── favorite.js # Yêu thích
| └── follow.js # Theo dõi người dùng
| └── report.js # Báo cáo vi phạm
| └── notification.js # Thông báo
| └── theme.js # Tùy chỉnh giao diện
| └── admin.js # Quản trị hệ thống
|
| └── scripts/
| └── scheduled_tasks.js # Tác vụ tự động định kỳ
| └── [migration scripts] # Scripts tạo bảng, migrate
|
| └── uploads/ # Thư mục lưu file tạm
|
| └── server.js # Entry point
````
```

Luồng chính:

- **Auth:** Hash mật khẩu với Bcrypt, tạo JWT (exp 7 ngày), middleware verify token và kiểm tra role
- **Recipe:** CRUD công thức + upload ảnh lên Cloudinary, kiểm tra quyền owner/admin/moderator
- **Follow:** Follow/unfollow người dùng, kiểm tra không cho follow chính mình, đếm followers/following
- **Report:** Báo cáo bài viết/bình luận/người dùng, hệ thống quota, tự động khóa tính năng khi vi phạm
- **Notification:** Gửi thông báo cá nhân, broadcast, trả lời thông báo
- **Theme:** Lưu/lấy theme preferences, chia sẻ theme lên marketplace

3.3.3 Database (MySQL)

Bảng chính:

- `nguo_dung` Người dùng với các trường khóa tính năng
- `cong_thuc` Công thức với views, violation_count, is_hidden
- `step_images` Ảnh từng bước nấu
- `binh_luan` Bình luận với parent_comment_id cho nested
- `comment_likes` Like bình luận
- `danh_gia` Đánh giá sao

- `favorite` Yêu thích
- `follow` Theo dõi người dùng
- `bao_cao` Báo cáo vi phạm (đa loại)
- `user_report_quota` Quota báo cáo
- `recipeViolationHistory` Lịch sử vi phạm
- `notifications` Thông báo cá nhân
- `broadcast_notifications` Thông báo hàng loạt
- `user_broadcast_read` Đánh dấu đã đọc broadcast
- `user_theme_preferences` Tùy chỉnh giao diện
- `admin_hidden_recipes` Ân bài viết thủ công
- `recipe_views` Theo dõi lượt xem

3.4 Bảo mật và quản lý phiên

- **Bcrypt:** Hash mật khẩu với Salt, cost 10
- **JWT:** Lưu trên client; gửi Bearer token; middleware verify; exp 7 ngày
- **CORS:** Cho phép credentials; hạn chế header/method cần thiết
- **Phân quyền:** Middleware kiểm tra role cho route admin/moderator; kiểm tra owner trước khi sửa/xóa
- **Khóa tính năng:** Tự động khóa đăng bài/bình luận/báo cáo khi vi phạm nhiều lần

Bảng 3.11: Phân quyền user vs admin

| Hành động | User | Moderator | 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 (nested) | ✓ | ✓ | ✓ |
| Like / Unlike bình luận | ✓ | ✓ | ✓ |
| Xóa bình luận của người khác | ✗ | ✓ | ✓ |

| Hành động | User | Moderator | Admin |
|-------------------------------------|------|-----------|-------|
| Theo dõi người dùng | ✓ | ✓ | ✓ |
| Báo cáo vi phạm | ✓ | ✓ | ✓ |
| Xem danh sách báo cáo | ✗ | ✓ | ✓ |
| Xử lý báo cáo | ✗ | ✓ | ✓ |
| Ân bài viết thủ công | ✗ | ✓ | ✓ |
| Gửi thông báo đến người dùng | ✗ | ✓ | ✓ |
| Gửi thông báo hàng loạt (broadcast) | ✗ | ✓ | ✓ |
| Xem danh sách người dùng | ✗ | ✗ | ✓ |
| Đổi vai trò người dùng | ✗ | ✗ | ✓ |
| Xóa tài khoản người dùng | ✗ | ✗ | ✓ |
| Tùy chỉnh giao diện | ✓ | ✓ | ✓ |
| Chia sẻ theme | ✓ | ✓ | ✓ |
| Xóa theme của người khác | ✗ | ✗ | ✓ |
| 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 với `parent_comment_id`
- Backend lưu vào bảng `binh_luan` với trường `parent_comment_id` không NULL
- Frontend: Component recursive, render reply form inline

3.5.2 Like System

- Bảng riêng `comment_likes` (comment_id, user_id, UNIQUE)
- Toggle endpoint: kiểm tra UNIQUE constraint; nếu tồn tại → delete, không tồn tại → insert

3.5.3 Follow System

- Bảng `follow` với `follower_id` và `following_id`
- Không cho phép follow chính mình
- Endpoints: follow, unfollow, check is-following, counts, list followers/following

3.5.4 Report System

- 3 loại báo cáo: recipe, comment, user
- Quota system: Mỗi user có 3 lượt báo cáo mỗi loại
- Upload ảnh bằng chứng khi báo cáo
- Tự động khóa tính năng khi vi phạm nhiều lần

3.5.5 Notification System

- Thông báo cá nhân: Admin/moderator gửi đến user cụ thể
- Thông báo hàng loạt (broadcast): Gửi đến tất cả user
- Hỗ trợ đính kèm ảnh, trả lời thông báo

3.5.6 Theme System

- Lưu preferences: primary_color, background_image, theme_name
- Chia sẻ theme lên marketplace (is_shared = true)
- Marketplace: Danh sách theme công khai, có thể áp dụng

3.6 Kiểm thử

- Chức năng: Đăng ký/đăng nhập, CRUD công thức, tìm kiếm, bình luận nested, đánh giá, yêu thích, theo dõi, báo cáo, thông báo, tùy chỉnh giao diện
- Bảo mật: Từ chối request thiếu/invalid JWT, kiểm tra owner, kiểm tra quota
- Hiệu năng: Đo thời gian phản hồi API, kiểm tra responsive

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/moderator/admin
- Quản lý công thức: tạo, sửa, xóa, xem chi tiết với ảnh từng bước; tìm kiếm; đếm lượt xem; upload ảnh lên Cloudinary
- Tương tác người dùng: bình luận nested, like bình luận, đánh giá sao, yêu thích
- Theo dõi người dùng: follow/unfollow, xem profile, đếm followers/following
- Báo cáo vi phạm: 3 loại (bài viết/bình luận/người dùng), quota system, upload bằng chứng, tự động khóa tính năng
- Thông báo: cá nhân, broadcast, reply, đếm chưa đọc
- Tùy chỉnh giao diện: theme cá nhân, marketplace chia sẻ
- Quản trị: quản lý người dùng, xử lý báo cáo, ẩn bài viết, gửi thông báo
- API REST chuẩn: mã trạng thái 200/201/400/401/403/404/409/500

Bảng 4.1: Kết quả kiểm thử chức năng

| ID | Chức năng | Input | Expected Result | Actual Result | Status |
|------|-----------------------------|------------------------|---------------------|----------------------|--------|
| TC01 | Đăng ký – hợp lệ | Email, password hợp lệ | 201, success | 201, user created | ✓ Pass |
| TC02 | Đăng ký – email trùng | Email đã tồn tại | 400, email exists | 400, email exists | ✓ Pass |
| TC03 | Đăng nhập – đúng | Thông tin hợp lệ | 200, JWT token | 200, token returned | ✓ Pass |
| TC04 | Đăng nhập – sai mật khẩu | Password sai | 400, wrong password | 400, wrong password | ✓ Pass |
| TC05 | Tạo công thức – có token | Dữ liệu hợp lệ + JWT | 201, created | 201, recipe created | ✓ Pass |
| TC06 | Tạo công thức – không token | Không có JWT | 401, unauthorized | 401, unauthorized | ✓ Pass |
| TC07 | Tạo công thức – bị khóa | User bị khóa đăng bài | 403, blocked | 403, posting blocked | ✓ Pass |
| TC08 | Xóa công thức – owner | Token chủ sở hữu | 200, deleted | 200, deleted | ✓ Pass |

| ID | Chức năng | Input | Expected Result | Actual Result | Status |
|------|----------------------------------|----------------------|--------------------------|-------------------------|--------|
| TC09 | Xóa công thức – không phải owner | Token khác | 403, forbidden | 403, forbidden | ✓ Pass |
| TC10 | Xóa công thức – moderator | Token moderator | 200, deleted | 200, deleted | ✓ Pass |
| TC11 | Đánh giá – lần đầu | Rating 1–5 | 201, rated | 201, rated | ✓ Pass |
| TC12 | Đánh giá – lần 2 | Đánh giá trùng | 400, already rated | 400, already rated | ✓ Pass |
| TC13 | Bình luận – top-level | Nội dung hợp lệ | 201, created | 201, comment created | ✓ Pass |
| TC14 | Reply bình luận | Nội dung + parent_id | 201, created | 201, reply created | ✓ Pass |
| TC15 | Like bình luận – lần 1 | comment_id + JWT | 200, liked: true | 200, liked: true | ✓ Pass |
| TC16 | Unlike bình luận | Like lại | 200, liked: false | 200, liked: false | ✓ Pass |
| TC17 | Follow user | user_id hợp lệ | 200, success | 200, followed | ✓ Pass |
| TC18 | Follow chính mình | Own user_id | 400, error | 400, cannot follow self | ✓ Pass |
| TC19 | Unfollow user | user_id hợp lệ | 200, success | 200, unfollowed | ✓ Pass |
| TC20 | Đếm followers | user_id | 200, followers/following | 200, counts returned | ✓ Pass |
| TC21 | Báo cáo bài viết | reason + JWT | 200, success | 200, reported | ✓ Pass |
| TC22 | Báo cáo – hết quota | Quá 3 lần | 403, quota exceeded | 403, no quota | ✓ Pass |
| TC23 | Báo cáo – bị khóa | User bị khóa | 403, blocked | 403, reporting blocked | ✓ Pass |
| TC24 | Báo cáo bình luận | comment_id + reason | 200, success | 200, reported | ✓ Pass |
| TC25 | Báo cáo người dùng | user_id + reason | 200, success | 200, reported | ✓ Pass |

| ID | Chức năng | Input | Expected Result | Actual Result | Status |
|------|-------------------------|--------------------|-------------------|--------------------------|--------|
| TC26 | Báo cáo admin | admin user_id | 403, forbidden | 403, cannot report admin | ✓ Pass |
| TC27 | Xử lý báo cáo – accept | Admin/Mod token | 200, accepted | 200, accepted | ✓ Pass |
| TC28 | Xử lý báo cáo – reject | Admin/Mod + reason | 200, rejected | 200, rejected | ✓ Pass |
| TC29 | Gửi thông báo | Admin/Mod → User | 200, sent | 200, notification sent | ✓ Pass |
| TC30 | Gửi broadcast | Admin/Mod | 200, sent | 200, broadcast sent | ✓ Pass |
| TC31 | Reply thông báo | Nội dung reply | 200, sent | 200, reply sent | ✓ Pass |
| TC32 | Reply thông báo – trùng | Đã reply trước | 409, conflict | 409, already replied | ✓ Pass |
| TC33 | Lưu theme | primary_color | 200, saved | 200, theme saved | ✓ Pass |
| TC34 | Chia sẻ theme | theme_name | 200, shared | 200, theme shared | ✓ Pass |
| TC35 | Xóa theme – owner | Token owner | 200, deleted | 200, theme unshared | ✓ Pass |
| TC36 | Xóa theme – không owner | Token khác | 403, forbidden | 403, forbidden | ✓ Pass |
| TC37 | Moderator xử lý báo cáo | Token moderator | 200, success | 200, processed | ✓ Pass |
| TC38 | Moderator đổi role | Token moderator | 403, not admin | 403, not admin | ✓ Pass |
| TC39 | Admin đổi role | Token admin | 200, role changed | 200, role changed | ✓ Pass |
| TC40 | Bình luận – bị khóa | User bị khóa | 403, blocked | 403, commenting blocked | ✓ 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 + badge thông báo khi đã login; link admin cho admin/moderator
- **Trang chủ:** lưới công thức, thanh tìm kiếm, nút tạo công thức khi đã đăng nhập
- **Trang chi tiết:** ảnh, nguyên liệu, bước nấu với ảnh từng bước; bình luận nested và đánh giá; nút yêu thích, báo cáo
- **Trang profile:** avatar, bio, số followers/following, danh sách công thức, nút follow
- **Trang cá nhân:** My Recipes, Favorite Recipes, Account Settings
- **Trang thông báo:** danh sách thông báo với nút trả lời, đánh dấu đã đọc
- **Trang tùy chỉnh giao diện:** chọn màu, upload hình nền, preview
- **Theme Marketplace:** danh sách theme, nút áp dụng
- **Trang quản trị:** bảng người dùng, đổi role; danh sách báo cáo, xử lý; gửi thông báo
- Lightbox xem ảnh phóng to
- Modal quy định cộng đồng

4.3 Hiệu năng và kỹ thuật

- Non-blocking I/O (Node.js + Express) xử lý đồng thời nhiều request
- Upload ảnh chuyển sang Cloudinary, giảm tải server
- CORS cấu hình an toàn; JWT qua header Authorization
- MySQL đáp ứng truy vấn; ràng buộc khóa ngoại đảm bảo toàn vẹn
- Scheduled tasks tự động xử lý định kỳ

Bảng 4.2: Đánh giá hiệu năng

| API Endpoint | Method | Thời gian phản hồi trung bình | Ghi chú |
|--------------------|--------|-------------------------------|------------------------|
| /auth/login | POST | ~150 ms | Bao gồm so sánh Bcrypt |
| /auth/register | POST | ~200 ms | Bao gồm Bcrypt hash |
| /recipe/list | GET | ~80 ms | Truy vấn ~50 công thức |
| /recipe/search | GET | ~120 ms | Full-text search |
| /recipe/create | POST | ~800 ms | Upload ảnh Cloudinary |
| /recipe/update/:id | PUT | ~600 ms | Upload ảnh mới |

| API Endpoint | Method | Thời gian phản hồi trung bình | Ghi chú |
|--------------------------|--------|-------------------------------|-------------------------------|
| /recipe/delete/:id | DELETE | ~50 ms | Xóa dữ liệu trong DB |
| /recipe/comment/:id | GET | ~150 ms | Lấy bình luận lồng (nested) |
| /recipe/comment/:id/like | POST | ~80 ms | Toggle like / unlike |
| /rating/:id | POST | ~60 ms | Ghi dữ liệu đánh giá |
| /follow/:userId | POST | ~70 ms | Ghi quan hệ follow |
| /follow/counts/:userId | GET | ~50 ms | Truy vấn COUNT |
| /report/recipe/:id | POST | ~100 ms | Kiểm tra quota + insert |
| /report | GET | ~120 ms | Admin xem báo cáo |
| /report/:id/status | PUT | ~200 ms | Cập nhật + gửi email |
| /notification/my | GET | ~100 ms | Thông báo cá nhân + broadcast |
| /notification/broadcast | POST | ~150 ms | Gửi thông báo hàng loạt |
| /theme/preferences | GET | ~50 ms | Truy vấn đơn giản |
| /theme/marketplace | GET | ~80 ms | Danh sách theme chia sẻ |

4.4 Bảo mật

- Mật khẩu được hash với Bcrypt (Salt + adaptive hashing)
- JWT có hạn 7 ngày, kiểm tra ở middleware
- Phân quyền rõ ràng: user / moderator / admin
- Không lưu mật khẩu gốc
- Từ chối truy cập khi thiếu/invalid token
- Hạn chế origin qua CORS
- Kiểm tra owner trước khi sửa/xóa
- Quota system chống spam báo cáo
- Tự động khóa tính năng khi vi phạm

4.5 Giao diện minh họa (mô tả nhanh)

- **Navbar:** Logo/tên app, đăng nhập/đăng ký hoặc tên người dùng + logout + badge thông báo, link Admin (nếu admin/moderator)
- **Home:** Lưới công thức, thẻ ảnh + tiêu đề + rating; thanh tìm kiếm

- **Create/Edit Recipe:** Form nhập tiêu đề, nguyên liệu, bước nấu với ảnh từng bước, upload ảnh chính, khẩu phần, thời gian nấu
- **Recipe Detail:** Ảnh, nội dung, bước nấu với ảnh; bình luận nested với like; đánh giá; nút yêu thích, báo cáo
- **Profile:** Avatar, bio, followers/following count, danh sách công thức, nút follow (nếu xem profile người khác)
- **Notifications:** Danh sách thông báo cá nhân và broadcast, nút reply, đánh dấu đã đọc
- **Theme Customization:** Color picker, upload hình nền, preview, nút lưu/chia sẻ
- **Theme Marketplace:** Grid các theme, tên tác giả, nút áp dụng
- **Admin Dashboard:** Bảng người dùng (username, email, role), nút đổi role/xóa
- **Admin Reports:** Danh sách báo cáo theo status, chi tiết báo cáo, nút accept/reject

4.6 Đánh giá nhanh

Đạt được:

- Đầy đủ chức năng cốt lõi và nâng cao
- Giao diện thân thiện và responsive
- Bảo mật tốt (hash mật khẩu, JWT, phân quyền 3 cấp)
- Hệ thống báo cáo và kiểm duyệt hoàn chỉnh
- Tính năng xã hội phong phú (follow, like, nested comments)
- Tùy chỉnh giao diện cá nhân với marketplace
- Hiệu năng phù hợp quy mô đê tài

Hạn chế:

- Chưa có cache (Redis)
- Chưa có realtime notification (WebSocket)
- Chưa triển khai CI/CD và bộ test tự động
- Chưa có gợi ý cá nhân hóa (ML)

4.7 Hướng phát triển

- Thêm realtime notifications (WebSocket) cho bình luận/like/follow mới
- Bổ sung cache (Redis) cho danh sách và tìm kiếm
- Viết test tự động (unit/integration) và thiết lập CI/CD
- Pagination/virtualization cho bình luận nhiều cấp
- Gợi ý công thức cá nhân hóa (recommendation/ML)
- Tối ưu SEO và accessibility

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

- Hoàn thành website CookShare với đầy đủ chức năng cốt lõi và nâng cao: đăng ký/đăng nhập (JWT), quản lý công thức (tạo/sửa/xóa/tìm kiếm với ảnh từng bước), bình luận nested, đánh giá, yêu thích, theo dõi người dùng, và phân quyền user/moderator/admin
- Xây dựng hệ thống báo cáo vi phạm hoàn chỉnh với 3 loại báo cáo (bài viết/bình luận/người dùng), quota system, upload bằng chứng, tự động khóa tính năng khi vi phạm nhiều lần, gửi email thông báo kết quả
- Triển khai hệ thống thông báo đa dạng: thông báo cá nhân, thông báo hàng loạt (broadcast), trả lời thông báo, đếm số chưa đọc
- Tích hợp tính năng tùy chỉnh giao diện cá nhân với marketplace chia sẻ theme
- 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 đa cấp (user/moderator/admin), dễ mở rộng, bảo mật tốt, tổ chức mã theo chuẩn (MVC, middleware, phân tầng)

5.2 Hạn chế

- Chưa có pagination cho threads bình luận sâu; dữ liệu nhiều cấp sẽ chậm khi tải
- 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/follow/reply mới
- 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
- Gợi ý công thức cá nhân hóa (recommendation/ML) dựa trên hành vi người dùng
- Tối ưu SEO và accessibility
- Mở rộng marketplace theme với tính năng rating, download count
- Thêm tính năng chat/message giữa người dùng

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] paessler, "REST," [Online]. Available: <https://www.paessler.com/it-explained/rest>. [Accessed 08 12 2025].
- [2] "JWT Debugger," Introduction to JSON Web Tokens, [Online]. Available: <https://www.jwt.io/introduction#what-is-json-web-token>. [Accessed 10 12 2025].
- [3] dominhhai, "Hai's Blog," [Node.js] Mã hoá mật khẩu với Bcrypt, 02 02 2016. [Online]. Available: <https://dominhhai.github.io/vi/2016/02/nodejs-ecrypt-password-with-bcrypt/>. [Accessed 14 12 2025].
- [4] "M mdn_," Cross-Origin Resource Sharing (CORS), [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS>. [Accessed 09 12 2025].
- [5] M. O. Source, "React," React The library for web and native user interfaces, [Online]. Available: <https://react.dev/>. [Accessed 25 11 2025].
- [6] "Express5.2.1," Fast, unopinionated, minimalist web framework for Node.js, [Online]. Available: <https://expressjs.com>. [Accessed 08 12 2025].
- [7] O. Foundation, "nodejs.org," [Online]. Available: <https://nodejs.org/en/>. [Accessed 08 12 2025].
- [8] MySQL, "MySQL HeatWave," [Online]. Available: <https://dev.mysql.com/doc>. [Accessed 13 12 2025].
- [9] clouinary, "Image and Video API Platform," [Online]. Available: <https://clouinary.com/documentation>. [Accessed 22 12 2025].
- [10] OWASP, "OWASP Cheat Sheet Series," OWASP: OWASP Cheat Sheet Series (Authentication, Password Storage, REST Security), [Online]. Available: <https://cheatsheetseries.owasp.org/>. [Accessed 24 12 2025].

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 |

File .env (Backend)

...

PORt=3001

DB_HOST=localhost

DB_USER=root

DB_PASSWORD=yourpassword

DB_NAME=cookingdb

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

```
```bash
git clone <repository-url>
cd DoAnChuyenNganh
````
```

Bước 2: Cài đặt Backend

```
```bash
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

```
```bash
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)

```
```javascript
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();
 });
};
const verifyAdminOrModerator = (db) => (req, res, next) => {
 verifyToken(req, res, () => {
 db.query(
 "SELECT role FROM nguoi_dung WHERE id = ?",
 [req.user.id],
 (err, rows) => {
 if (err || rows.length === 0) {
 return res.status(500).json({ message: "Lỗi kiểm tra quyền" });
 }
 const role = rows[0].role;
 if (role === "admin" || role === "moderator") {
 req.user.role = role;
 next();
 } else {
 return res.status(403).json({ message: "Không đủ quyền truy cập" });
 }
 }
);
 });
};

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

C.2 Follow handler (routes/follow.js)

```
```javascript
// POST /:userId - Theo dõi người dùng
router.post("/:userId", verifyToken, (req, res) => {
 const followingId = req.params.userId;
 const followerId = req.user.id;
 if (parseInt(followingId) === followerId) {
```

```
 return res.status(400).json({ message: "✗ Không thể theo dõi chính mình!" });
 }
 db.query(
 "INSERT INTO follow (follower_id, following_id) VALUES (?, ?)",
 [followerId, followingId],
 (err) => {
 if (err) {
 if (err.code === "ER_DUP_ENTRY") {
 return res.status(400).json({ message: "⚠ Bạn đã theo dõi người dùng này!" });
 }
 return res.status(500).json({ message: "✗ Lỗi khi theo dõi!" });
 }
 res.json({ message: "✓ Đã theo dõi!" });
 }
);
});
```
// GET /counts/:userId - Dém followers/following
router.get("/counts/:userId", (req, res) => {
    const userId = req.params.userId;
    db.query(
        `SELECT
            (SELECT COUNT(*) FROM follow WHERE following_id = ?) as followers,
            (SELECT COUNT(*) FROM follow WHERE follower_id = ?) as following`,
        [userId, userId],
        (err, result) => {
            if (err) return res.status(500).json({ message: "✗ Lỗi!" });
            res.json({
                followers: result[0].followers || 0,
                following: result[0].following || 0
            });
        }
    );
});
```
});
```
```

C.3 Report với quota system (routes/report.js)

```
```javascript
// Kiểm tra và lấy quota báo cáo
const getReportQuota = (userId, reportType, callback) => {
 db.query(
 `SELECT * FROM user_report_quota WHERE user_id = ? AND report_type = ?`,
 [userId, reportType],
 (err, rows) => {
 if (err) return callback(err);
 if (rows.length === 0) {
 // Tạo quota mới
 db.query(
 `INSERT INTO user_report_quota (user_id, report_type, remaining_reports)
VALUES (?, ?, 3)`,
 [userId, reportType],
 (err2) => {
```

```
 if (err2) return callback(err2);
 callback(null, { remaining_reports: 3 });
 }
);
} else {
 callback(null, rows[0]);
}
);
};

// POST /recipe/:id - Báo cáo bài viết
router.post("/recipe/:id", verifyToken, upload.single("image"), (req, res) => {
 const recipeId = req.params.id;
 const userId = req.user.id;
 const { reason } = req.body;
 const imageUrl = req.file ? `/uploads/reports/${req.file.filename}` : null;

 // Kiểm tra quota
 getReportQuota(userId, "recipe", (err, quota) => {
 if (quota.remaining_reports <= 0) {
 return res.status(403).json({
 message: "❌ Bạn đã hết lượt báo cáo bài viết!"
 });
 }
 // Tạo báo cáo và giảm quota
 db.query(
 `INSERT INTO bao_cao (recipe_id, user_id, reason, image_url, target_type)
 VALUES (?, ?, ?, ?, 'recipe')`,
 [recipeId, userId, reason, imageUrl],
 (err, result) => {
 if (err) return res.status(500).json({ message: "❌ Lỗi tạo báo cáo!" });

 // Giảm quota
 db.query(
 `UPDATE user_report_quota SET remaining_reports = remaining_reports - 1
 WHERE user_id = ? AND report_type = 'recipe'`,
 [userId]
);
 res.json({ message: "✅ Báo cáo thành công!", reportId: result.insertId });
 }
);
 });
});
```

```

C.4 Broadcast notification (routes/notification.js)

```
```javascript
// POST /broadcast - Gửi thông báo hàng loạt
router.post("/broadcast", verifyAdminOrModerator(db), upload.single("image"), (req, res) => {
 const senderId = req.user.id;
 const { message } = req.body;
```

```
const imageUrl = req.file ? `/uploads/notifications/${req.file.filename}` : null;
if (!message || message.trim() === "") {
 return res.status(400).json({ message: " ✗ Vui lòng nhập nội dung thông báo" });
}
db.query(
 `INSERT INTO broadcast_notifications (sender_id, message, image_url) VALUES (?, ?, ?)`,
 [senderId, message, imageUrl],
 (err, result) => {
 if (err) {
 return res.status(500).json({ message: " ✗ Lỗi gửi thông báo hàng loạt" });
 }
 res.json({
 message: " ✓ Đã gửi thông báo đến tất cả người dùng",
 broadcastId: result.insertId
 });
 }
);
});
// GET /my - Lấy thông báo của user (bao gồm broadcast)
router.get("/my", verifyToken, (req, res) => {
 const userId = req.user.id;
 // Lấy thông báo cá nhân
 const personalQuery = `
 SELECT n.*, u.username as sender_name, 'personal' as notification_type
 FROM notifications n
 JOIN nguoi_dung u ON n.sender_id = u.id
 WHERE n.receiver_id = ?
 `;
 // Lấy broadcast chưa đọc
 const broadcastQuery = `
 SELECT
 bn.id, bn.sender_id, bn.message, bn.image_url, bn.created_at,
 u.username as sender_name, 'broadcast' as notification_type,
 CASE WHEN ubr.id IS NOT NULL THEN TRUE ELSE FALSE END as is_read
 FROM broadcast_notifications bn
 JOIN nguoi_dung u ON bn.sender_id = u.id
 LEFT JOIN user_broadcast_read ubr ON bn.id = ubr.broadcast_id AND ubr.user_id = ?
 `;
 db.query(personalQuery, [userId], (err, personalNotifs) => {
 if (err) return res.status(500).json({ message: " ✗ Lỗi lấy thông báo" });

 db.query(broadcastQuery, [userId], (err2, broadcastNotifs) => {
 if (err2) return res.status(500).json({ message: " ✗ Lỗi lấy thông báo" });
 // Kết hợp và sắp xếp theo thời gian
 const allNotifs = [...personalNotifs, ...broadcastNotifs].sort(
 (a, b) => new Date(b.created_at) - new Date(a.created_at)
);
 res.json(allNotifs);
 });
 });
});
```

## C.5 Theme marketplace (routes/theme.js)

```
```javascript
// POST /share - Chia sẻ theme lên marketplace
router.post("/share", verifyToken, (req, res) => {
  const user_id = req.user.id;
  const { theme_name } = req.body;
  if (!theme_name) {
    return res.status(400).json({ message: "❌ Vui lòng cung cấp tên theme!" });
  }
  db.query(
    "UPDATE user_theme_preferences SET is_shared = TRUE, theme_name = ? WHERE user_id = ?",
    [theme_name, user_id],
    (err) => {
      if (err) {
        return res.status(500).json({ message: "❌ Lỗi chia sẻ giao diện!" });
      }
      res.json({ message: "✅ Đã chia sẻ giao diện! Mọi người có thể tải theme của bạn" });
    }
  );
});

// GET /marketplace - Lấy danh sách theme công khai
router.get("/marketplace", verifyToken, (req, res) => {
  db.query(
    `SELECT
      utp.id,
      utp.primary_color,
      utp.background_image,
      utp.theme_name,
      utp.created_at,
      u.username as created_by,
      u.id as owner_id
    FROM user_theme_preferences utp
    JOIN nguoi_dung u ON utp.user_id = u.id
    WHERE utp.is_shared = TRUE
    ORDER BY utp.created_at DESC
    LIMIT 50`,
    (err, result) => {
      if (err) {
        return res.status(500).json({ message: "❌ Lỗi tải danh sách theme!" });
      }
      res.json(result || []);
    }
  );
});
```

```

## 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("");
        } catch (error) {
            console.error(error);
        }
    };
}
```

```
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) .toLocaleDateString()}</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

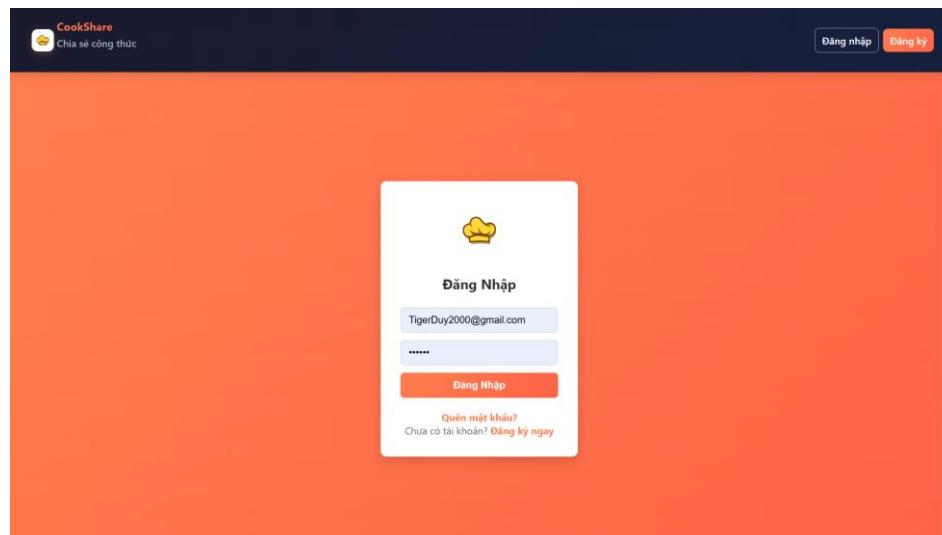
- Headers: Authorization: Bearer <token>
- Body: FormData
- Response: { message }
- Errors: 403 (not owner), 404 (not found)

DELETE /recipe/delete/:id

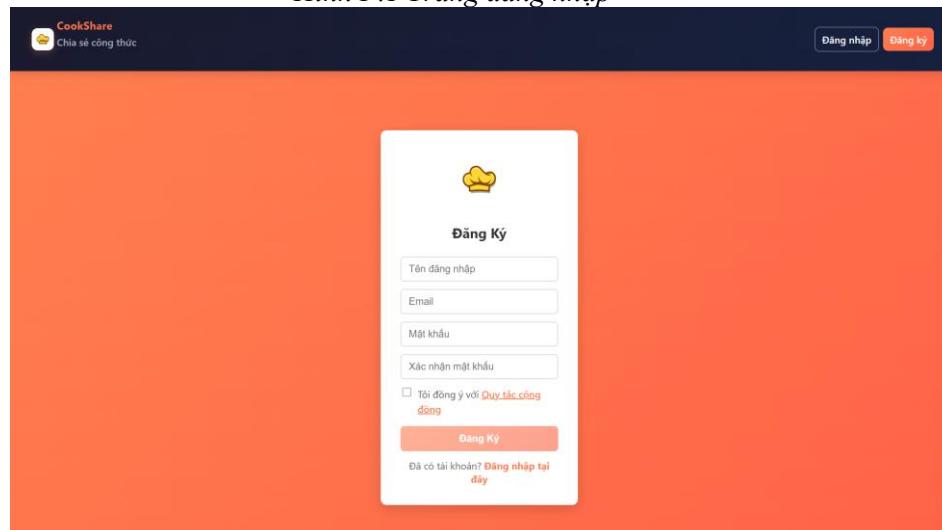
- Headers: Authorization: Bearer <token>
- Response: { message }
- Errors: 403 (not owner/admin), 404 (not found)

PHỤ LỤC F: Screenshots giao diện

F.1: Trang đăng nhập và đăng ký

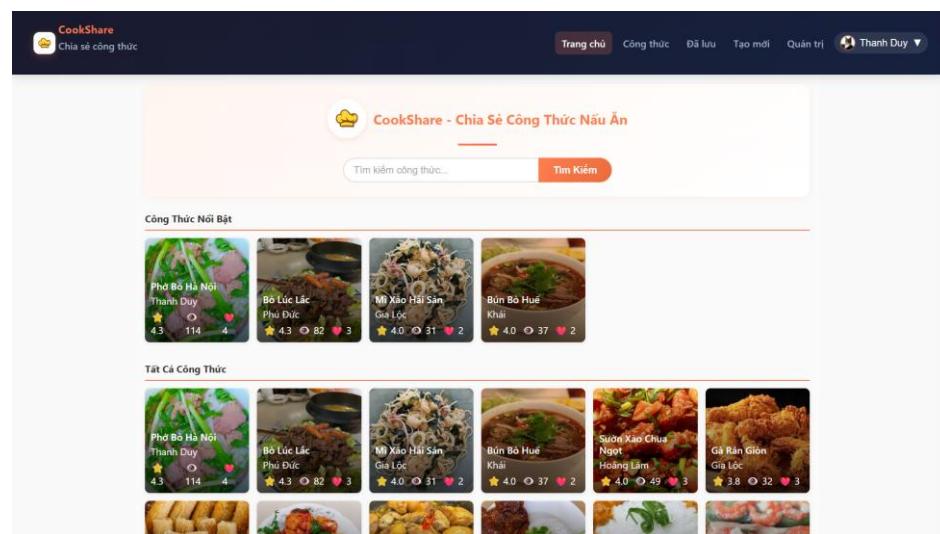


Hình F.1 Trang đăng nhập



Hình F.2 Trang đăng ký

F.2: Trang chủ



Hình F.2 Trang chủ

F.3: Trang chi tiết công thức & Nested comments & Like button trên bình luận

Phở Bò Hà Nội

Thanh Duy • 115 lượt xem • 4 lượt lưu

Đánh Giá

4.3
(6 đánh giá)

Đánh giá của bạn:
★★★★★
 Bạn đã đánh giá 5 sao

Đã lưu

Nguyên Liệu

6 người ăn

- 1 kg thịt bò
- 1 kg xương lợn
- 0.5 lạng sả sùng khô
- quả Quế, hoa hồi, thảo
- Hành lá, rau thơm, mùi tây, gừng

Hướng dẫn cách làm

2 tiếng

- Rửa sạch xương bò. Nấu sôi nước, cho xương bò vào luộc qua rồi chắt đi, cho nước mới (làm như này thì nước sẽ trong)
- Cho xương bò vào hầm, trong lúc hầm thì liên tục hớt bọt, hầm khoảng 1h rưỡi. Lọc xương.
- Nướng chín hành, hành tím, gừng, miè. Sau khi nướng xong, cạo sạch vỏ gừng và hành. Rửa các gia vị trên với nước sạch, để ráo. Cắt đôi hành tây, cắt lát gừng.
- Rang hoa hồi, quế, thảo quả, sả sùng với lửa nhỏ cho đến khi dậy mùi thơm
- Nấu sôi nước hầm bò, lần lượt bỏ sả sùng và các loại gia vị trên vào nồi. Đun được 10' thì nếm gia vị vừa miệng
- Cắt lát mỏng thịt bò theo thớ ngang rồi chan nước dùng lên. Nếu bạn không thích ăn tái thì có thể luộc thịt bò, để tủ lạnh, khi nào ăn thì cắt ra
- Cho thêm hành lá, và rau mùi, giá tùy sở thích

Bình Luận (2)

Phú Đức @cool_000003 20:07:39 23/12/2025

Ăn được không

1 Trả lời Báo cáo

Thanh Duy @cool_000001 01:40:25 24/12/2025

Sao lại không nhỉ

Trả lời

Thêm bình luận

Hình F.3 Trang chi tiết công thức & Nested comments & Like button trên bình luận

F.5 Trang Công thức của tôi

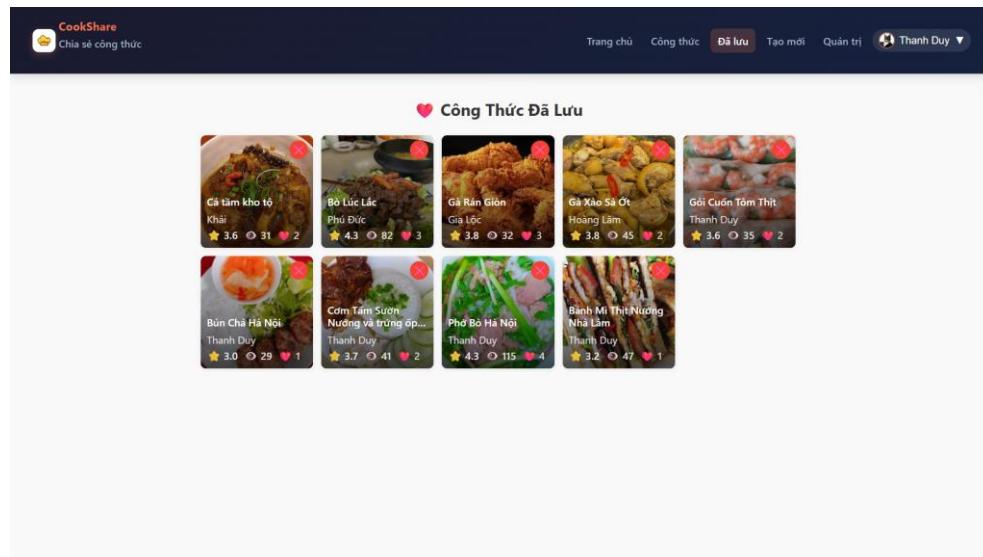
Công Thức Của Tôi

Tạo Công Thức Mới

Phở Bò Hà Nội	Cơm Tấm Günde Nướng và trứng ốp...	Bánh Mì Thịt Nướng Nhà Lãm	Bún Chả Hà Nội	Gỏi Cuốn Tôm Thịt
★★★★★ 4.3 115 lượt lưu	★★★★ 3.7 41 lượt lưu	★★★★ 3.2 47 lượt lưu	★★★★ 3.0 29 lượt lưu	★★★★ 3.6 35 lượt lưu

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

A screenshot of the 'Viết món mới' (Create New Recipe) form on the CookShare website. The form is divided into several sections: 1. 'Tên món:' (Name of the dish) with a text input field containing 'Món canh bí ngon nhất nhà mình'. 2. 'Nguyên Liệu' (Ingredients) section: 'Khẩu phần' (Portion) is set to '2 người' (2 people) and '250g bột' (250g flour). There is a '+ Nguyên liệu' (Add ingredient) button. 3. 'Các bước' (Steps) section: 'Thời gian nấu' (Cooking time) is listed as '1 tiếng 30 phút' (1 hour 30 minutes). Step 1 is described as 'Trộn bột và nước đến khi đặc lại' (Mix flour and water until it becomes thick). There is a '+ Bước làm' (Add step) button. 4. A large dashed box area for adding images or descriptions. 5. A footer with 'Hủy' (Cancel) and 'Đăng bài' (Post) buttons.

Hình F.7 Form tạo công thức

F.8: Trang quản trị (Admin Dashboard)

The screenshot displays the Admin Dashboard with two main sections:

- Quản Lý Công Thức (Recipe Management):**
 - Count: 26 Recipes
 - Table Headers: ID, TIÊU ĐỀ (Title), TÁC GIẢ (Author), TRẠNG THÁI (Status), NGÀY TẠO L (Created Date), HÀNH ĐỘNG (Actions).
 - Table Data: A list of 10 recipes including "Test chức năng đăng bài viết", "Phở Bò Hà Nội", "Cơm Tấm Sườn Nướng và trứng ốp la", etc.
 - Pagination: Page 1 of 3.
- Quản Lý Người Dùng (User Management):**
 - Count: 7 Users
 - Table Headers: ID, TÊN ĐĂNG NHẬP (Login Name), EMAIL, VAI TRÔ (Role), HÀNH ĐỘNG (Actions).
 - Table Data: A list of users including "test", "Thanh Duy", "Admin (Ban)", etc.
 - Pagination: Page 1 of 1.

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

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

The screenshot displays the 'Trang Quản Trị Admin' (Admin Control Panel) interface. It features two main sections: 'QUẢN LÝ CÔNG THỨC' (Recipe Management) and 'QUẢN LÝ NGƯỜI DÙNG' (User Management).

QUẢN LÝ CÔNG THỨC:

- CHẾ ĐỘ HẠN CHẾ:** Ban đang đăng nhập với vai trò Moderator (moderator). Bạn chỉ có thể xem dữ liệu và nâng/ hạ User -- Moderator. Không thể tạo Admin, hạ Admin, xóa người dùng/công thức, hoặc reset mật khẩu.
- Tổng số:** 7 công thức
- Hiển thị:** 26 công thức
- Danh sách công thức:**

ID	Tiêu Đề	Tác Giả	Trạng Thái	Ngày Tạo	Hành Động
#37	Test chức năng ẩn bài viết	Admin	Đã ẩn (2 vi phạm)	15/12/2025	Bỏ ẩn
#31	Phở Bò Hà Nội	Thanh Duy	Hiển thị	28/11/2025	Ẩn
#32	Cơm Tấm Suôn Nướng và trứng ốp la	Thanh Duy	Hiển thị	28/11/2025	Ẩn
#33	Bánh Mì Thịt Nướng Nhà Làm	Thanh Duy	Hiển thị	28/11/2025	Ẩn
#34	Bún Chả Hà Nội	Thanh Duy	Hiển thị	28/11/2025	Ẩn
#35	Gỏi Cuốn Tôm Thịt	Thanh Duy	Hiển thị	28/11/2025	Ẩn
#6	Cơm cá ri gá	Phú Đức	Hiển thị	28/11/2025	Ẩn
#7	Lẩu Thái Hải Sản	Phú Đức	Hiển thị	28/11/2025	Ẩn
#8	Bò Lát Lắc	Phú Đức	Hiển thị	28/11/2025	Ẩn
#9	Cánh Chua Cá Hú	Phú Đức	Hiển thị	28/11/2025	Ẩn
- Trang:** ← Trước | 1 | 2 | 3 | Tiếp →

QUẢN LÝ NGƯỜI DÙNG:

- Gửi Thông Báo Hàng Loạt:**
- Tổng số:** 7 người dùng
- Danh sách người dùng:**

ID	TÊN ĐĂNG NHẬP	EMAIL	VAI TRÒ	HÀNH ĐỘNG
#7	test	test@gmail.com	User	Thông báo
#1	Thanh Duy (Bạn)	TigerDuy2000@gmail.com	Moderator	
#2	Admin	admin@gmail.com	Admin	Thông báo Chỉ Admin khác mới đổi/reset
#3	Phú Đức	PhuDuc@gmail.com	Moderator	Thông báo
#4	Gia Lộc	HaGiaLoc@gmail.com	User	Thông báo
#5	Khai	PhanDinhKhai@gmail.com	User	Thông báo
#6	Hoàng Lâm	HLam@gmail.com	User	Thông báo

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

F.10: Trang cá nhân



Thanh Duy
PiscesKing

Followers: 5 Following: 5

Chỉnh Sửa

Followers		Following			
	Gia Lộc	Đang theo dõi		Admin	Đang theo dõi
	Phú Đức	Đang theo dõi		Phú Đức	Đang theo dõi
	Admin	Đang theo dõi		Gia Lộc	Đang theo dõi
	Hoàng Lãm	Đang theo dõi		Khải	Đang theo dõi
	Khải	Đang theo dõi		Hoàng Lãm	Đang theo dõi

Công thức của người dùng



Phở Bò Hà Nội
⭐ 4.3 • ⚪ 115 • ❤️ 4



Cơm Tấm Sườn Nướng và trứng ốp la
⭐ 3.7 • ⚪ 41 • ❤️ 2



Bánh Mì Thịt Nướng Nhà Làm
⭐ 3.2 • ⚪ 47 • ❤️ 1



Bún Chả Hà Nội
⭐ 3.0 • ⚪ 29 • ❤️ 1



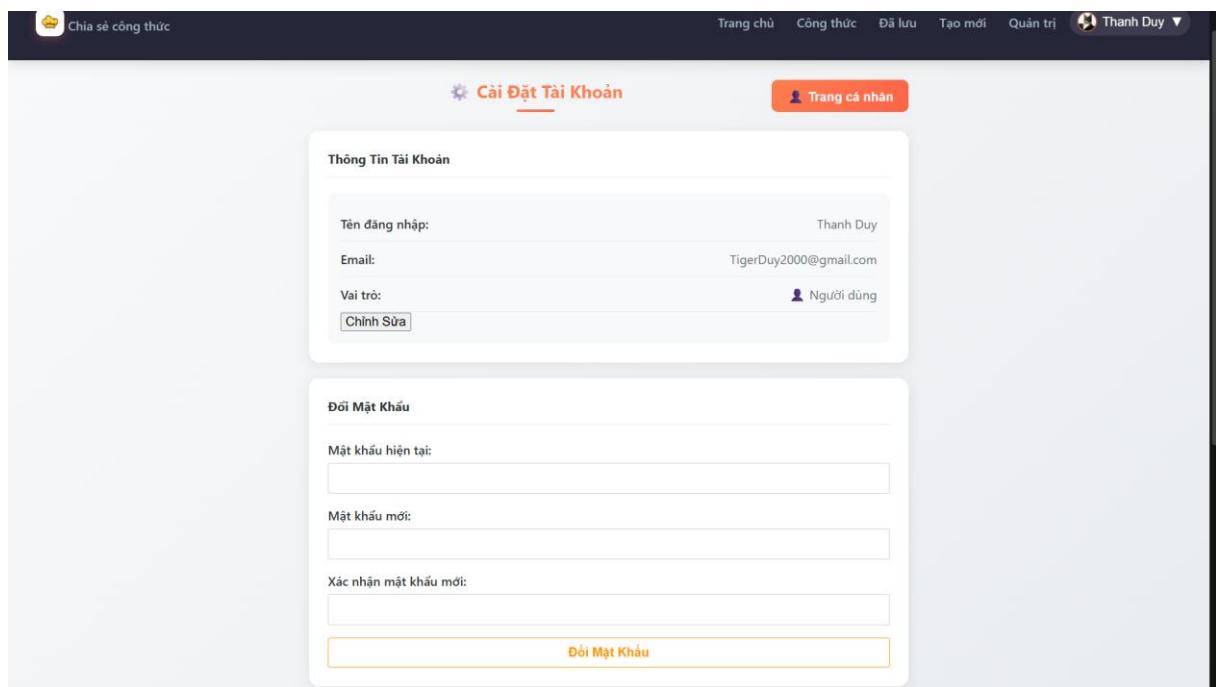
Gỏi Cuốn Tôm Thịt
⭐ 3.6 • ⚪ 35 • ❤️ 2

Hình F.10: Trang cá nhân

Nguyễn Thanh Duy

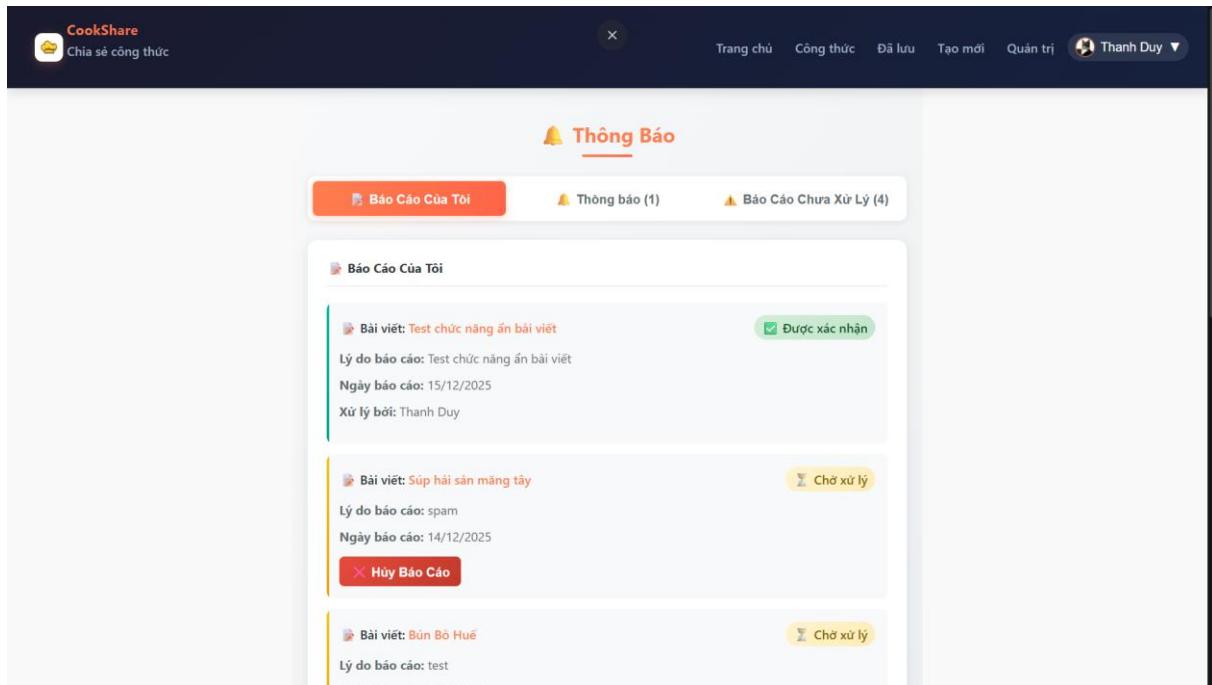
52

F.11: Cài đặt tài khoản



Hình F.11: Cài đặt tài khoản

F.12 Trang thông báo(Quản trị viên và Admin)



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

F.12 Trang thông báo(Quản trị viên và Admin)

The screenshot shows the 'Thông Báo' (Report) section of the CookShare website. It lists three reports:

- Báo cáo của Tôi:
 - Bài viết: Test chức năng ẩn bài viết
 - Lý do báo cáo: Test chức năng ẩn bài viết
 - Ngày báo cáo: 15/12/2025
 - Xử lý bởi: Thanh Duy
- Bài viết: Súp hải sản măng tây
 - Lý do báo cáo: spam
 - Ngày báo cáo: 14/12/2025
 - Chờ xử lý
- Bài viết: Bún Bò Huế
 - Lý do báo cáo: test
 - Chờ xử lý

Hình F.12: Trang cá nhân & Cài đặt tài khoản(User)

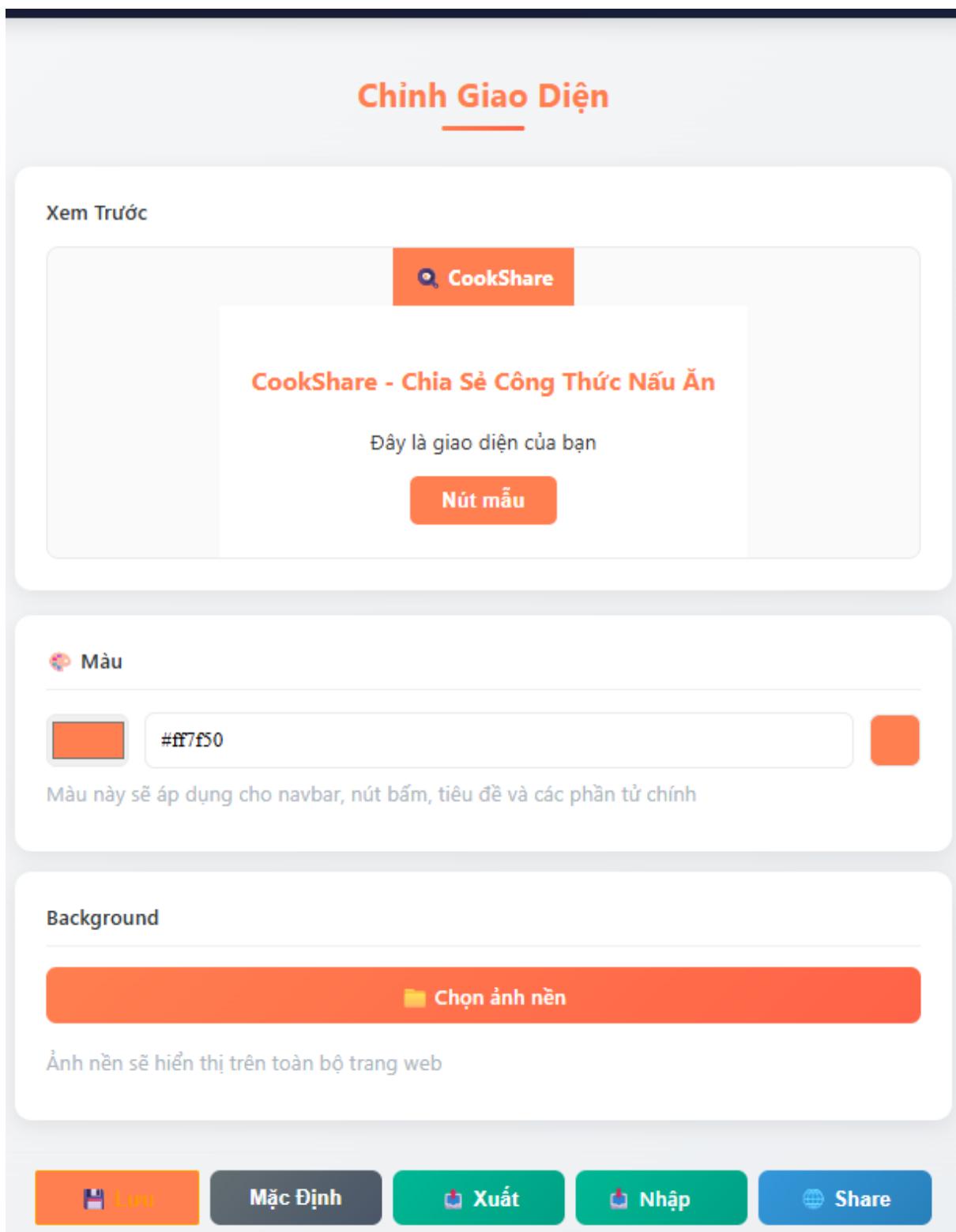
F.13: Trang quản lý báo cáo bài viết dành cho Quản trị viên & Admin

The screenshot shows the 'Quản Lý Báo Cáo' (Report Management) section of the CookShare website. It displays reports categorized by status:

- Tất cả (4):
 - Bài viết
 - Bài viết: Mì Xào Hải Sản
 - Tác giả: Gia Lộc (HaGiaLoc@gmail.com)
 - Báo cáo từ: Khải (PhanDinhKhai@gmail.com)
 - Lý do báo cáo: test
 - Ngày báo cáo: 14/12/2025
 - Chờ phản hồi
 - Xác Nhận Ví Phạm
 - Báo Bó Báo Cáo
 - Bài viết
 - Bài viết: Súp hải sản măng tây
 - Tác giả: Gia Lộc (HaGiaLoc@gmail.com)
 - Báo cáo từ: Thanh Duy (TigerDuy2000@gmail.com)
 - Lý do báo cáo: spam
 - Đã Xác Nhận (6)
 - Đã Báo Bó (1)

Hình F.13: Trang quản lý báo cáo bài viết dành cho Quản trị viên & Admin

F.14 Tùy chỉnh giao diện



Hình F.14 Tùy chỉnh giao diện

F.15 Quy tắc cộng đồng

Hình F.15: Quy tắc cộng đồng

PHỤ LỤC G: Kiêm thử

G.1 Test cases chính

Bảng 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	Reply với parent_id	201, reply created	✓ Pass

ID	Chức năng	Input	Expected Output	Kết quả
	comment			
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