BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP VIỆT – HUNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: CÔNG NGHỆ THÔNG TIN

TÊN ĐỀ TÀI: TRANG WEB QUẢN LÝ DỊCH VỤ SPA TÍCH HỢP ĐẶT LỊCH LÀM ĐẸP TRỰC TUYẾN

Người hướng dẫn : *TS. Nguyễn Hồng Quân*

Họ tên sinh viên : *Đỗ Kim Ngọc Anh*

Mã sinh viên : *2100876*

Khóa : *45*

Hà Nội – Năm 2025

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP VIỆT – HUNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: CÔNG NGHỆ THÔNG TIN

TÊN ĐỀ TÀI: TRANG WEB QUẢN LÝ DỊCH VỤ SPA TÍCH HỢP ĐẶT LỊCH LÀM ĐẸP TRỰC TUYẾN

Người hướng dẫn : *TS. Nguyễn Hồng Quân*

Họ tên sinh viên : *Đỗ Kim Ngọc Anh*

Mã sinh viên : *2100876*

Khóa : *45*

Hà Nội – Năm 2025

MỤC LỤC

LỜI CẢM ƠN

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc tới Thầy TS. Nguyễn Hồng Quân, giảng viên Khoa Công nghệ Thông tin – Trường Đại học Công nghiệp Việt – Hung, người đã trực tiếp hướng dẫn, tận tình chỉ bảo và định hướng cho em trong suốt quá trình thực hiện đề tài: "Trang web quản lý dịch vụ Spa tích hợp đặt lịch làm đẹp trực tuyến."

Thầy không chỉ giúp em củng cố kiến thức chuyên môn mà còn truyền đạt cho em tinh thần làm việc nghiêm túc, phương pháp nghiên cứu khoa học, cũng như sự cẩn trọng và tỉ mỉ trong từng chi tiết. Những góp ý quý báu của Thầy đã giúp em hoàn thiện đề tài một cách khoa học, thực tiễn và có tính ứng dụng cao.

Em cũng xin gửi lời cảm ơn chân thành đến toàn thể quý Thầy, Cô trong Khoa Công nghệ Thông tin đã tận tình giảng dạy, cung cấp nền tảng kiến thức vững chắc và tạo điều kiện thuận lợi cho em trong suốt quá trình học tập và nghiên cứu tại trường.

Bên cạnh đó, em xin cảm ơn gia đình và bạn bè – những người luôn ở bên động viên, chia sẻ và hỗ trợ em cả về tinh thần lẫn vật chất trong suốt thời gian thực hiện đồ án.

Mặc dù đã cố gắng hoàn thiện đề tài một cách tốt nhất, nhưng do hạn chế về thời gian và kinh nghiệm thực tế, bài làm chắc chắn vẫn còn những thiếu sót. Em rất mong nhận được sự góp ý của quý Thầy, Cô để đề tài được hoàn thiện hơn trong tương lai.

Một lần nữa, em xin chân thành cảm ơn Thầy TS. Nguyễn Hồng Quân, cùng toàn thể quý Thầy, Cô Khoa Công nghệ Thông tin – Trường Đại học Công nghiệp Việt – Hung.

Em xin trân trọng cảm ơn!

Hà Nội, tháng năm 2025

Sinh viên thực hiện

(ký và ghi rõ họ tên)

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC SƠ ĐỒ, HÌNH VẼ

DANH MỤC CÁC TỪ VIẾT TẮT – THUẬT NGỮ ANH – VIỆT

TT	Từ viết tắt	Nội dung	Ghi chú
1			
2			
3			
4			
5			

LỜI MỞ ĐẦU

1. Tính cấp thiết của đề tài

Trong bối cảnh nhu cầu chăm sóc sức khỏe và sắc đẹp ngày càng tăng, các cơ sở Spa cần hệ thống quản lý hiện đại để tối ưu hóa quy trình đặt lịch, quản lý khách hàng và dịch vụ. Việc xây dựng hệ thống quản lý Spa là cần thiết nhằm nâng cao hiệu quả vận hành và trải nghiệm khách hàng.

2. Mục đích nghiên cứu của đề tài

Xây dựng một hệ thống web quản lý Spa giúp tự động hóa các quy trình chính như quản lý dịch vụ, nhân viên, khách hàng, lịch hẹn và báo cáo doanh thu, đồng thời đảm bảo giao diện thân thiện và dễ sử dụng.

3. Đối tượng nghiên cứu

Hệ thống quản lý Spa ứng dụng công nghệ web, hướng đến người quản lý, nhân viên và khách hàng sử dụng trực tuyến.

4. Nhiệm vụ nghiên cứu

Phân tích, thiết kế và xây dựng hệ thống quản lý Spa với đầy đủ chức năng đặt lịch, phân quyền, thống kê và gửi thông báo tự động

5. Phương pháp nghiên cứu

Áp dụng phương pháp phân tích hệ thống thông tin, mô hình hóa nghiệp vụ bằng UML, thiết kế cơ sở dữ liệu, và triển khai theo quy trình phát triển phần mềm **Agile/Scrum**.

6. Các kết quả đạt được của đề tài

Đề tài đã xây dựng thành công hệ thống quản lý Spa hoạt động ổn định, có giao diện thân thiện, hỗ trợ quản lý hiệu quả và dễ dàng mở rộng trong tương lai.

7. Kết cấu của đồ án

Đồ án gồm 4 chương như sau:

Chương 1 : Tổng quan về đề tài

Chương 2 : Cơ sở lý thuyết

Chương 3 : Phân tích hệ thống

Chương 4 : Thiết kế hệ thống

CHƯƠNG 1
TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu đề tài

1.1.1. Tên đề tài

Đề tài có tên là **"Trang web quản lý dịch vụ spa tích hợp đặt lịch làm đẹp trực tuyến"**, được định hướng nhằm phát triển một nền tảng quản lý toàn diện cho các cơ sở Spa trong bối cảnh chuyển đổi số hiện nay.

1.1.2. Bối cảnh và lý do chọn đề tài

Trong thời đại công nghệ số, việc áp dụng các giải pháp công nghệ thông tin vào quản lý và kinh doanh là xu hướng tất yếu đối với mọi lĩnh vực dịch vụ, trong đó có ngành Spa và chăm sóc sắc đẹp. Ngành dịch vụ này đang phát triển nhanh chóng với nhu cầu ngày càng lớn của người dân về chăm sóc sức khỏe và làm đẹp. Tuy nhiên, phần lớn các cơ sở Spa hiện nay vẫn còn quản lý theo phương pháp truyền thống như ghi chép lịch hẹn bằng sổ tay, quản lý khách hàng thủ công, chưa có hệ thống thống kê và báo cáo doanh thu hiệu quả, cũng như thiếu các cơ chế thông báo tự động. Việc không có nền tảng trực tuyến cũng gây khó khăn cho khách hàng khi muốn đặt lịch hẹn, đặc biệt trong bối cảnh dịch bệnh COVID-19, khi nhu cầu đặt lịch trực tuyến để hạn chế tiếp xúc tăng cao.

Xuất phát từ thực trạng đó, đề tài "Trang web quản lý dịch vụ spa tích hợp đặt lịch làm đẹp trực tuyến" được lựa chọn với mong muốn mang lại một giải pháp công nghệ hiện đại, giúp các cơ sở Spa tối ưu hóa quy trình hoạt động, nâng cao trải nghiệm khách hàng và tăng cường khả năng quản lý, giám sát hoạt động kinh doanh một cách hiệu quả.

Tiêu chí	**Phương pháp truyền thống**	**Phương pháp hiện đại**
Ghi chép lịch hẹn	Sổ sách thủ công	Hệ thống điện tử
Quản lý khách hàng	Thông tin rời rạc	Cơ sở dữ liệu tập trung
Thống kê doanh thu	Tính toán thủ công	Tự động hóa
Thông báo	Gọi điện / Email thủ công	Gửi thông báo tự động
Báo cáo	Lập báo cáo thủ công	Dashboard real-time

Bảng 1.1. So sánh phương pháp quản lý truyền thống và hiện đại

1.1.3. Ý nghĩa thực tiễn

Việc xây dựng hệ thống quản lý Spa trực tuyến có ý nghĩa thiết thực đối với cả chủ cơ sở, nhân viên và khách hàng. Đối với chủ Spa, hệ thống giúp quản lý lịch hẹn chính xác, tránh trùng lặp; theo dõi doanh thu và tình hình kinh doanh; quản lý thông tin khách hàng và nhân viên một cách tập trung; đồng thời tự động hóa quy trình thông báo và nhắc nhở, góp phần nâng cao hiệu quả vận hành.

Về phía khách hàng, việc đặt lịch trực tuyến 24/7 giúp họ chủ động trong việc lựa chọn thời gian, dịch vụ và nhân viên phục vụ, đồng thời có thể xem lại lịch sử dịch vụ và nhận các thông báo nhắc nhở tự động.

Đối với nhân viên, hệ thống giúp quản lý lịch làm việc rõ ràng, theo dõi hiệu suất công việc và hỗ trợ giao diện thao tác thân thiện, dễ sử dụng, giảm thiểu sai sót trong quá trình phục vụ.

Đối tượng	**Lợi ích chính**	**Mô tả chi tiết**
Chủ cơ sở Spa	Tăng hiệu quả quản lý	Giảm khoảng **70% thời gian quản lý**, dễ dàng theo dõi doanh thu và hoạt động
Khách hàng	Trải nghiệm tốt hơn	**Đặt lịch 24/7**, nhận thông báo tự động, xem lịch sử dịch vụ tiện lợi
Nhân viên	Quản lý lịch rõ ràng	Theo dõi hiệu suất công việc, sử dụng **giao diện thân thiện và trực quan**

Bảng 1.2. Phân tích lợi ích của hệ thống

1.2. Mục tiêu nghiên cứu

1.2.1. Mục tiêu chính

Mục tiêu tổng quát của đề tài là xây dựng một hệ thống quản lý Spa trực tuyến hoàn chỉnh, đáp ứng đầy đủ nhu cầu quản lý, vận hành và phục vụ khách hàng trong các cơ sở Spa hiện đại.

1.2.2. Mục tiêu cụ thể

Trước hết, đề tài hướng đến việc phân tích và thiết kế hệ thống dựa trên yêu cầu thực tế của các cơ sở Spa. Quá trình này bao gồm việc tìm hiểu nghiệp vụ, thiết kế kiến trúc phần mềm phù hợp và xây dựng cơ sở dữ liệu tối ưu, đảm bảo tính mở rộng và bảo mật.

Tiếp theo, hệ thống được phát triển với các chức năng cốt lõi như đăng ký, đăng nhập và phân quyền người dùng; quản lý thông tin khách hàng, nhân viên và dịch vụ; hỗ trợ

đặt lịch hẹn trực tuyến; quản lý lịch hẹn theo trạng thái; cung cấp cơ chế thông báo tự động; và xây dựng trang tổng quan (dashboard) phục vụ báo cáo, thống kê hoạt động kinh doanh.

Ngoài ra, đề tài tập trung vào việc thiết kế giao diện người dùng (UI/UX) hiện đại, gồm hai phần chính: giao diện quản trị dành cho admin và giao diện đặt lịch dành cho khách hàng. Cả hai giao diện đều được xây dựng theo hướng responsive, bảo đảm khả năng hiển thị tốt trên nhiều thiết bị.

Cuối cùng, hệ thống được kiểm thử toàn diện nhằm bảo đảm chất lượng, độ ổn định, tính bảo mật và hiệu suất vận hành.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là hệ thống quản lý Spa trực tuyến, bao gồm các quy trình nghiệp vụ đặc thù trong hoạt động kinh doanh Spa, cùng với việc ứng dụng các công nghệ web hiện đại như Node.js, React và MongoDB.

1.3.2. Phạm vi nghiên cứu

Về mặt chức năng, hệ thống tập trung vào các nghiệp vụ chính như quản lý người dùng (admin, nhân viên, khách hàng), quản lý dịch vụ và danh mục, quản lý lịch hẹn và trạng thái, hệ thống thông báo tự động, cùng các chức năng báo cáo và thống kê.

Về mặt kỹ thuật, hệ thống sử dụng Node.js và Express.js cho phần backend, React.js cho frontend, MongoDB làm cơ sở dữ liệu chính. Quá trình xác thực và bảo mật người dùng được triển khai bằng JWT (JSON Web Token), trong khi việc giao tiếp giữa frontend và backend được thực hiện thông qua RESTful API.

Về mặt đối tượng sử dụng, hệ thống được thiết kế phục vụ ba nhóm người dùng chính: Admin (quản trị toàn hệ thống), nhân viên (quản lý lịch hẹn và khách hàng), và khách hàng (thực hiện đặt lịch và xem thông tin dịch vụ).

1.4. Phương pháp nghiên cứu

Đề tài được thực hiện dựa trên ba nhóm phương pháp chính.

Thứ nhất là phương pháp thu thập dữ liệu, bao gồm việc nghiên cứu tài liệu, tìm hiểu các hệ thống Spa trực tuyến hiện có, khảo sát thực tế quy trình hoạt động của các cơ sở Spa và nghiên cứu các công nghệ web phù hợp.

Thứ hai là phương pháp phát triển hệ thống, trong đó đề tài áp dụng phương pháp hướng đối tượng để phân tích và thiết kế phần mềm, kết hợp với mô hình phát triển

linh hoạt Agile nhằm đảm bảo khả năng thích ứng trong từng giai đoạn phát triển. Các hoạt động kiểm thử được tiến hành song song để đảm bảo chất lượng của từng module và toàn bộ hệ thống.

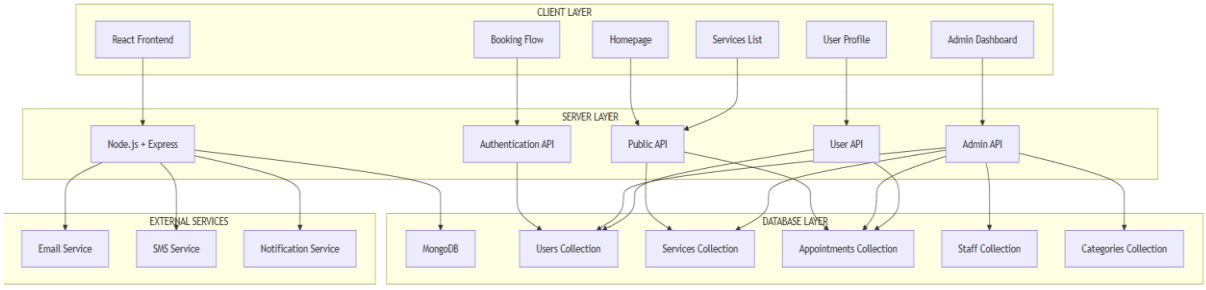
Thứ ba là phương pháp và công cụ hỗ trợ, bao gồm việc sử dụng Visual Studio Code và Git trong phát triển phần mềm, MongoDB Compass trong quản lý cơ sở dữ liệu, Postman để kiểm thử API, và Figma hoặc Adobe XD cho thiết kế giao diện người dùng.

1.5. Kết quả đạt được

1.5.1. Sản phẩm chính

Kết quả của đề tài là một hệ thống quản lý Spa trực tuyến hoàn chỉnh, bao gồm đầy đủ các chức năng nghiệp vụ thiết yếu. Hệ thống hỗ trợ xác thực và phân quyền người dùng một cách an toàn thông qua JWT, cho phép quản lý khách hàng, nhân viên và lịch làm việc, cũng như tổ chức dữ liệu dịch vụ theo danh mục và gói combo.

Hệ thống đặt lịch hẹn trực tuyến hoạt động 24/7, có khả năng kiểm tra xung đột lịch tự động và theo dõi trạng thái từng cuộc hẹn. Ngoài ra, hệ thống được tích hợp module thông báo qua email hoặc tin nhắn SMS, giúp nhắc nhở khách hàng về lịch hẹn sắp tới. Trang dashboard tổng hợp dữ liệu và hiển thị các chỉ số quan trọng như doanh thu, lượng khách hàng, hiệu suất nhân viên và tình trạng đặt lịch, giúp người quản lý có cái nhìn tổng quan và ra quyết định kịp thời.



Hình 1.1. Sơ đồ tổng quan hệ thống

1.5.2. Đóng góp khoa học

Đề tài góp phần ứng dụng các công nghệ web hiện đại như Node.js, React và MongoDB vào lĩnh vực quản lý Spa, tạo nên một mô hình hệ thống hiệu quả, dễ mở rộng và có khả năng triển khai thực tế cao. Ngoài ra, việc thiết kế kiến trúc hướng đối tượng và cơ sở dữ liệu tối ưu giúp chuẩn hóa quy trình nghiệp vụ, tăng tính ổn định và bảo mật cho hệ thống.

1.5.3. Ý nghĩa thực tiễn

Sản phẩm của đề tài mang lại giải pháp tổng thể cho các cơ sở Spa trong việc số hóa quy trình hoạt động. Việc triển khai hệ thống không chỉ nâng cao hiệu quả quản lý và tiết kiệm thời gian mà còn giúp cải thiện trải nghiệm khách hàng, tăng tính chuyên nghiệp và tạo lợi thế cạnh tranh cho doanh nghiệp. Đồng thời, hệ thống được thiết kế mở, có thể dễ dàng mở rộng hoặc tích hợp với các nền tảng khác trong tương lai.

1.6. Cấu trúc của đồ án

Đồ án được cấu trúc thành bốn chương chính.

Chương 1 trình bày tổng quan về đề tài, bao gồm tên đề tài, bối cảnh, lý do lựa chọn, mục tiêu, phạm vi và phương pháp nghiên cứu.

Chương 2 giới thiệu cơ sở lý thuyết và các công nghệ được sử dụng, làm nền tảng cho quá trình thiết kế và phát triển hệ thống.

Chương 3 tập trung vào thiết kế hệ thống, bao gồm kiến trúc tổng thể, thiết kế cơ sở dữ liệu, giao diện người dùng và các sơ đồ nghiệp vụ.

Chương 4 trình bày quá trình phân tích, kiểm thử và đánh giá hệ thống, đồng thời đưa ra định hướng phát triển trong tương lai.

1.7. Kết luận chương

Chương 1 đã khái quát toàn bộ bối cảnh, mục tiêu và ý nghĩa của đề tài "Xây dựng hệ thống quản lý Spa trực tuyến". Việc lựa chọn đề tài xuất phát từ nhu cầu thực tế của ngành Spa trong việc hiện đại hóa và tự động hóa quy trình quản lý. Hệ thống được định hướng phát triển trên nền tảng công nghệ web hiện đại như Node.js, React và MongoDB, đảm bảo tính linh hoạt, bảo mật và khả năng mở rộng.

Kết quả của chương này là cơ sở cho các nội dung tiếp theo, đặc biệt là phần cơ sở lý thuyết và thiết kế hệ thống ở các chương sau, nhằm hướng tới mục tiêu cuối cùng là xây dựng một hệ thống quản lý Spa trực tuyến hoàn chỉnh, có khả năng áp dụng trong thực tế và mang lại hiệu quả thiết thực cho người sử dụng.

CHƯƠNG 2
CƠ SỞ LÝ THUYẾT

2.1. Lý thuyết về hệ thống quản lý Spa

2.1.1. Khái niệm hệ thống quản lý Spa

Hệ thống quản lý Spa là một hệ thống thông tin được thiết kế nhằm hỗ trợ quản trị toàn diện hoạt động của cơ sở Spa, bao gồm quản lý khách hàng, nhân viên, dịch vụ, lịch hẹn và các nghiệp vụ tài chính liên quan. Mục tiêu cốt lõi của hệ thống là tự động hóa quy trình, giảm phụ thuộc vào thao tác thủ công, đồng thời nâng cao hiệu quả vận hành, độ chính xác dữ liệu và chất lượng trải nghiệm của khách hàng.

2.1.2. Các thành phần chính của hệ thống

Thành phần **quản lý khách hàng** đảm nhiệm lưu trữ hồ sơ cá nhân, lịch sử sử dụng dịch vụ, phân hạng khách hàng (ví dụ VIP), cũng như ghi nhận sở thích và nhu cầu để phục vụ cá nhân hóa; dữ liệu về điểm tích lũy và khuyến mãi được quản lý thống nhất nhằm triển khai các chương trình chăm sóc khách hàng hiệu quả. Thành phần **quản lý nhân viên** bao gồm hồ sơ cá nhân và chuyên môn, lịch làm việc theo ca, theo dõi hiệu suất và doanh thu, cùng quản lý bộ kỹ năng và chứng chỉ nhằm phân công phù hợp. Thành phần **quản lý dịch vụ** tổ chức danh mục dịch vụ, giá, gói combo và add-ons; đồng thời quản trị thời lượng, tài nguyên, phòng/giường và mức độ phổ biến để tối ưu lịch. Thành phần **quản lý lịch hẹn** đảm bảo đặt lịch trực tuyến, phát hiện xung đột thời gian, theo dõi trạng thái phiên hẹn theo vòng đời, và phát hành thông báo nhắc nhở tự động. Cuối cùng, **báo cáo và thống kê** cung cấp dashboard tổng quan, báo cáo doanh thu theo kỳ, phân tích xu hướng khách hàng và thống kê hiệu suất nhân viên phục vụ việc ra quyết định.

2.1.3. Lợi ích của hệ thống quản lý Spa

Đối với chủ cơ sở, hệ thống giúp tăng hiệu quả quản trị, tối ưu chi phí vận hành, theo dõi tình hình kinh doanh theo thời gian thực và chuẩn hóa quy trình. Đối với khách hàng, trải nghiệm được cải thiện nhờ đặt lịch thuận tiện, nhắc hẹn tự động, theo dõi lịch sử dịch vụ và thụ hưởng các ưu đãi phù hợp. Đối với nhân viên, lịch làm việc minh bạch, chỉ số hiệu suất rõ ràng và giao diện thao tác thân thiện góp phần nâng cao động lực và chất lượng phục vụ.

2.2. Công nghệ sử dụng

2.2.1. Công nghệ Backend

Node.js là runtime cho JavaScript trên môi trường server với cơ chế event-driven và non-blocking I/O, nhờ đó đạt hiệu năng cao và phù hợp mô hình mở rộng theo chiều ngang. Việc sử dụng JavaScript xuyên suốt full-stack giúp giảm chi phí chuyển ngữ và tận dụng hệ sinh thái NPM phong phú. **Express.js** đóng vai trò web framework tối giản nhưng linh hoạt, cung cấp routing, middleware và xử lý lỗi một cách có cấu trúc, cho phép mở rộng từng lớp chức năng mà không làm hệ thống cồng kềnh. **MongoDB** là cơ sở dữ liệu NoSQL định hướng document, lưu trữ dữ liệu dạng BSON với schema linh hoạt, hỗ trợ mở rộng ngang, truy vấn đa dạng và **aggregation pipeline** mạnh mẽ cho phân tích dữ liệu. **Mongoose** được sử dụng như ODM để định nghĩa schema, ràng buộc và validate dữ liệu, tổ chức middleware (hooks) trước/sau thao tác, hỗ trợ population giữa các collection và tự động ép kiểu, qua đó cân bằng tính linh hoạt của NoSQL với kỷ luật dữ liệu cần thiết.

2.2.2. Công nghệ Frontend

React.js cho phép xây dựng giao diện theo kiến trúc component-based, tối ưu render nhờ Virtual DOM và luồng dữ liệu một chiều giúp đơn giản hóa việc gỡ lỗi. JSX nâng cao tính diễn đạt của UI, trong khi hệ sinh thái phong phú giúp rút ngắn thời gian phát triển. **Vite** đảm nhiệm vai trò build tool hiện đại với HMR nhanh, sử dụng ES Modules gốc và tối ưu gói build cho môi trường production, đồng thời hỗ trợ TypeScript và hệ plugin đa dạng. **Tailwind CSS** áp dụng cách tiếp cận utility-first, cho phép tạo style nhanh, đảm bảo tính nhất quán và loại bỏ CSS không dùng, nhờ đó tối ưu kích thước bundle. **Ant Design** cung cấp thư viện UI ở mức enterprise với hệ component phong phú, thiết kế nhất quán, hỗ trợ accessibility và quốc tế hóa, góp phần rút ngắn thời gian hiện thực giao diện quản trị.

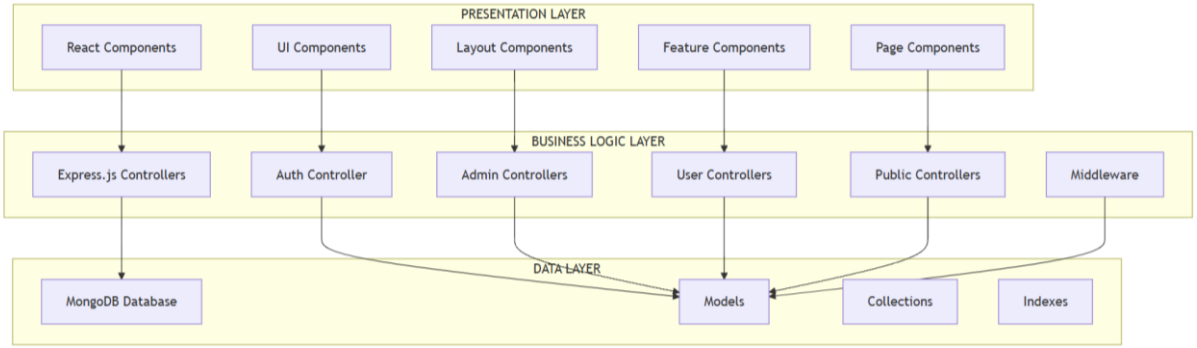
2.2.3. Công nghệ Authentication

JWT (JSON Web Token) hiện thực hóa mô hình xác thực stateless: token tự chứa thông tin cần thiết, kích thước gọn và có thể hoạt động cross-domain; chữ ký số đảm bảo tính toàn vẹn, kết hợp thời hạn hiệu lực và cơ chế refresh token để cân bằng an toàn – trải nghiệm. **bcrypt** được sử dụng để băm mật khẩu với salt và độ khó có thể điều chỉnh, chủ đích chậm để chống tấn công brute force, là lựa chọn phổ biến và an toàn trong thực tiễn.

2.3. Kiến trúc hệ thống

2.3.1. Kiến trúc Client–Server ba tầng

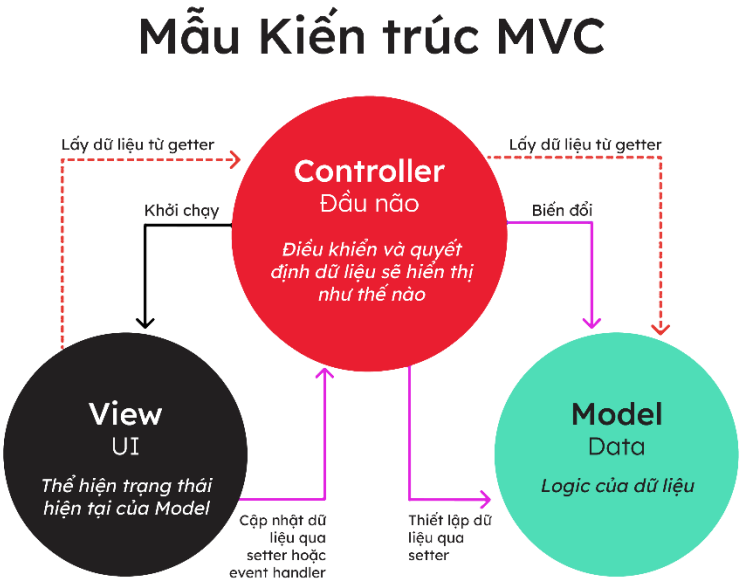
Tầng trình bày sử dụng **React.js** làm frontend, thiết kế responsive để đáp ứng đa thiết bị, quản lý trạng thái một cách có tổ chức và tích hợp RESTful API để trao đổi dữ liệu. Tầng nghiệp vụ sử dụng **Express.js** hiện thực business logic, áp dụng middleware cho xác thực – phân quyền, kiểm tra – chuẩn hóa dữ liệu đầu vào và xử lý lỗi nhất quán. Tầng dữ liệu dựa trên **MongoDB** với **Mongoose** làm cầu nối object–document; chỉ số (index) được thiết lập theo mẫu truy vấn thực tế nhằm tối ưu hiệu năng và bảo toàn tính toàn vẹn dữ liệu ở mức ứng dụng.



Hình 2.1. Kiến trúc Client–Server ba tầng

2.3.2. Mô hình MVC (Model–View–Controller)

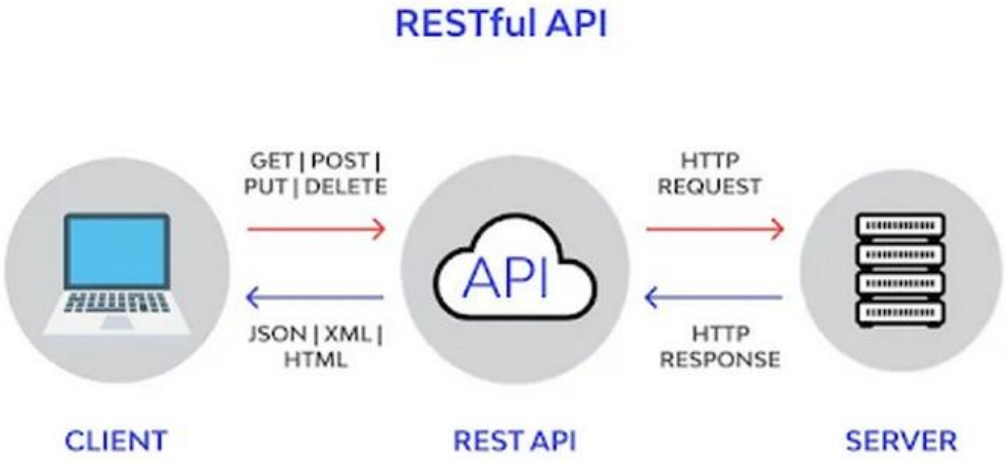
Lớp **Model** định nghĩa schema Mongoose, quy tắc nghiệp vụ gắn với dữ liệu và các thao tác CRUD. Lớp **View** được thể hiện bằng hệ **React components** với JSX và hệ thống style dựa trên Tailwind CSS kết hợp Ant Design, bảo đảm bố cục responsive và tính nhất quán. Lớp **Controller** định nghĩa tuyến **Express routes**, tiếp nhận HTTP request, điều phối luồng dữ liệu đến service/repository và trả về response đúng chuẩn, kèm theo xử lý ngoại lệ trung tâm.



Hình 2.2. Mô hình MVC (Model–View–Controller)

2.3.3. Thiết kế RESTful API

Giao diện REST duy trì tính **stateless**, tách biệt rõ client–server, hỗ trợ cache khi phù hợp và tuân thủ **uniform interface**. Các phương thức HTTP được sử dụng theo đúng ngữ nghĩa (GET/POST/PUT/PATCH/DELETE), kết hợp mã trạng thái 2xx/4xx/5xx phản ánh chính xác kết quả xử lý, giúp client dễ dàng diễn giải và xử lý tiếp.



Hình 2.3. Thiết kế RESTful API

2.4. Các mô hình thiết kế phần mềm

Repository Pattern đóng vai trò tách biệt logic truy cập dữ liệu khỏi business logic, cải thiện khả năng kiểm thử và cho phép thay thế nguồn dữ liệu mà không ảnh hưởng tầng nghiệp vụ. Repository interface quy định hợp đồng, còn implementation hiện thực cho MongoDB; **Service Layer** sử dụng repository để ghép nối nghiệp vụ thống nhất. **Service Layer Pattern** tập trung hóa business logic trong các lớp service, thuận lợi

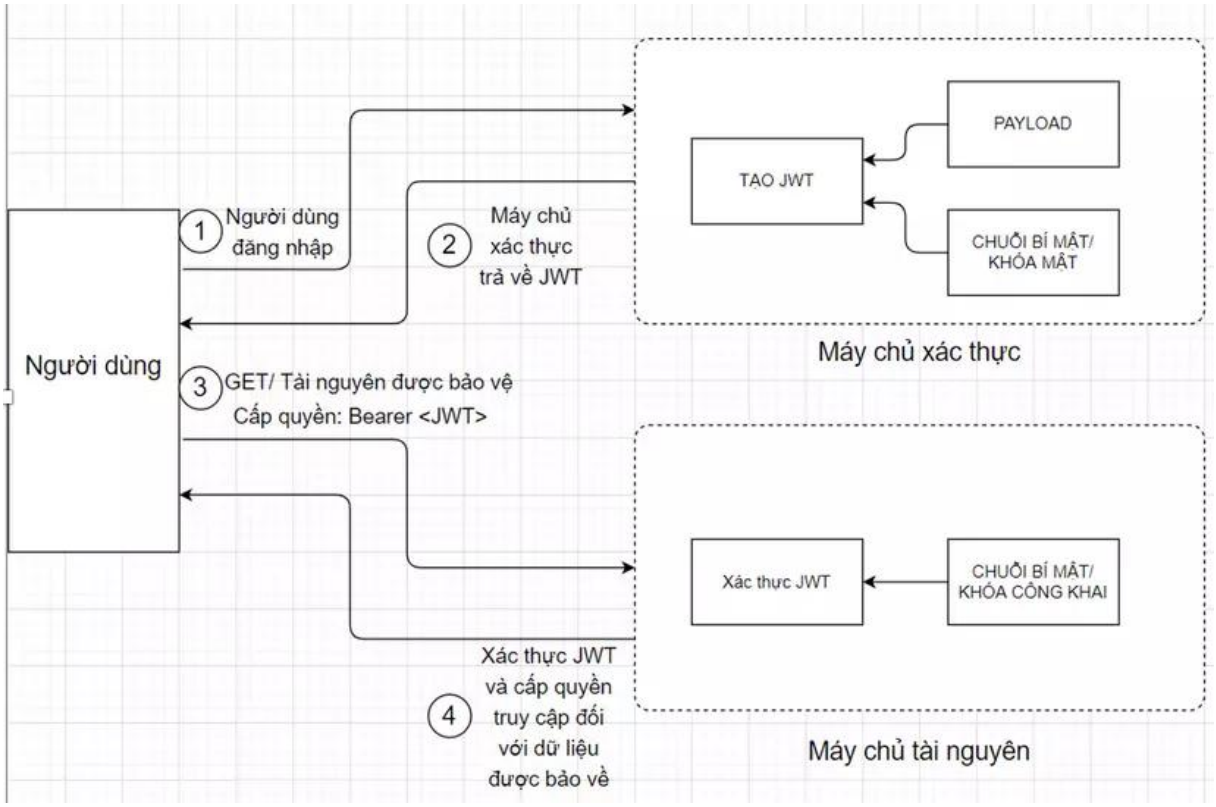
cho tái sử dụng, kiểm soát giao dịch và xử lý các mối quan tâm cắt ngang (logging, caching, bảo mật). **Middleware Pattern** tổ chức pipeline xử lý request/response theo thứ tự có kiểm soát, tái sử dụng và cô lập các chức năng như authentication, authorization, validation, logging và error handling. Ở frontend, **Component Pattern** giúp phân rã UI thành các đơn vị tái sử dụng, phân tách rõ giữa presentational và container components, tận dụng higher-order components và custom hooks để chia sẻ logic trạng thái một cách tinh gọn.

2.5. Database Design Patterns

Về **schema design**, hệ thống kết hợp linh hoạt giữa **embedded documents** để đồng vị trí dữ liệu truy cập cùng lúc và **references** (ObjectId) cho quan hệ phức tạp hoặc có khả năng tái sử dụng cao; cách tiếp cận lai (hybrid) giúp cân bằng hiệu năng đọc/ghi và tính chuẩn hóa. Mô hình one-to-many có thể dùng mảng embedded khi dữ liệu con nhỏ và gắn chặt ngữ nghĩa, trong khi many-to-many ưu tiên reference arrays để tránh phình to document. Với **indexing strategy**, hệ thống sử dụng single-field, compound, text và sparse index tùy theo mẫu truy vấn; chiến lược lập chỉ mục dựa trên quan sát hành vi truy vấn thực tế, cân bằng chi phí ghi, sử dụng bộ nhớ và kế hoạch bảo trì chỉ mục.

2.6. Security Patterns

Authentication dựa trên JWT với quy trình tạo, xác minh chữ ký, quản lý thời hạn và cơ chế refresh token; **password security** áp dụng bcrypt, quy tắc mật khẩu và quy trình reset an toàn. **Authorization** sử dụng **RBAC** để định nghĩa vai trò và quyền, gán vai trò cho người dùng và kiểm tra quyền truy cập theo ngữ cảnh; với tài nguyên nhạy cảm, hệ thống bổ sung **resource-based authorization** dựa trên quyền sở hữu, điều kiện truy cập và lưu **audit trail** phục vụ truy vết.



Hình 2.4. Security Patterns

2.7. Performance Optimization Patterns

Caching được áp dụng đa lớp: phía client khai thác browser cache cho tài nguyên tĩnh và local/session storage cho dữ liệu phù hợp; phía server có application cache, cache kết quả truy vấn cơ sở dữ liệu, kết hợp CDN cache cho static assets và có thể sử dụng Redis như external cache. Về tối ưu cơ sở dữ liệu, thiết kế truy vấn tuân theo chỉ mục, áp dụng aggregation pipeline khi cần, sử dụng phân trang hiệu quả; quản trị kết nối thông qua connection pooling, giới hạn, timeout hợp lý và cơ chế phục hồi lỗi.

2.8. Error Handling Patterns

Chiến lược xử lý lỗi bắt đầu từ phân loại lỗi (validation, authentication, authorization, business, system), sau đó hiện thực try–catch cục bộ, **global error handler** ở mức ứng dụng và **structured logging** để ghi nhận ngữ cảnh. Hệ thống phân cấp mức log (error, warn, info, debug), tập trung và xoay vòng log, đồng thời phân tích log nhằm phát hiện sớm xu hướng bất thường và cải thiện chất lượng vận hành.

2.9. Testing Patterns

Kiểm thử đơn vị (**unit testing**) tuân theo cấu trúc Arrange–Act–Assert, bao phủ function, React components, Mongoose models và service layer. Kiểm thử tích hợp (**integration testing**) tập trung vào API endpoints, luồng authentication–authorization,

dòng chảy dữ liệu xuyên tầng và kiểm thử cơ sở dữ liệu cho CRUD, quan hệ, ràng buộc cũng như hiệu năng. Cách tiếp cận này giúp phát hiện sớm lỗi hồi quy và bảo đảm tính ổn định khi mở rộng chức năng.

2.10. Kết luận chương

2.10.1. Tổng kết lý thuyết

Chương này đã xây dựng nền tảng lý thuyết cho hệ thống quản lý Spa, khẳng định vai trò của từng thành phần trong tổng thể kiến trúc và làm rõ lợi ích đối với các bên liên quan. Hệ sinh thái công nghệ lựa chọn—Node.js, Express.js, MongoDB, Mongoose ở backend; React.js, Vite, Tailwind CSS, Ant Design ở frontend—tạo thành nền tảng vững chắc cho một **web application** hiện đại. Bên cạnh đó, cơ chế bảo mật bằng JWT và bcrypt, kiến trúc Client–Server ba tầng, mô hình MVC và thiết kế RESTful API, cùng các design patterns (Repository, Service Layer, Middleware, Component) là bộ khung phương pháp luận để hiện thực hệ thống một cách bài bản. Các thực hành về cơ sở dữ liệu, bảo mật, hiệu năng, xử lý lỗi và kiểm thử tạo nên bức tranh hoàn thiện cho khả năng triển khai thực tế.

2.10.2. Lựa chọn công nghệ

Các công nghệ được lựa chọn dựa trên tiêu chí hiệu năng, khả năng mở rộng, trải nghiệm phát triển thuận tiện và mức độ hỗ trợ cộng đồng. Node.js và MongoDB cho phép xử lý tải lớn với mô hình mở rộng ngang; JavaScript toàn stack giảm độ phức tạp tích hợp; frontend hiện đại với React.js và Vite đẩy nhanh chu kỳ phát triển; Tailwind CSS và Ant Design đảm bảo chất lượng giao diện nhất quán ở cấp độ enterprise.

2.10.3. Kiến trúc tối ưu

Kiến trúc được định hướng theo nguyên tắc separation of concerns và modularity, từ đó tăng khả năng bảo trì, kiểm thử và mở rộng. Bảo mật được đưa vào từ khâu thiết kế, hiệu năng được tối ưu bằng chiến lược cache nhiều lớp và truy vấn theo chỉ mục, trong khi chất lượng được đảm bảo bằng pipeline kiểm thử toàn diện. Toàn bộ cơ sở lý thuyết tại Chương 2 là nền móng trực tiếp cho **Chương 3 – Thiết kế hệ thống**, nơi các mô hình, sơ đồ và cấu trúc dữ liệu sẽ được cụ thể hóa để tiến tới hiện thực hóa sản phẩm.

CHƯƠNG 3
PHÂN TÍCH HỆ THỐNG

3.1. Phân tích yêu cầu chức năng

3.1.1. Yêu cầu chức năng tổng quan

Hệ thống quản lý Spa được định hình như một nền tảng web thống nhất, nơi các bên liên quan (khách hàng, nhân viên và quản trị viên) tương tác thông qua các quy trình nghiệp vụ số hóa. Ở lớp người dùng, hệ thống cho phép đăng ký và đăng nhập bằng email và mật khẩu, kèm theo khả năng cập nhật hồ sơ cá nhân, thay đổi mật khẩu và tải ảnh đại diện. Cơ chế phân quyền hai cấp **Admin/User** đảm bảo mỗi vai trò chỉ được truy cập vào phạm vi chức năng phù hợp. Để hỗ trợ hoạt động chăm sóc khách hàng, hồ sơ được phân loại tự động theo mức độ VIP dựa trên tổng chi tiêu, qua đó giúp cá nhân hóa ưu đãi và phục vụ.

Khối quản lý dịch vụ cung cấp đầy đủ chu trình CRUD, cho phép hình thành danh mục đa cấp dưới dạng cấu trúc cây và mở rộng linh hoạt thông qua **combo** ưu đãi hoặc **add-ons** đi kèm. Ở khối nhân sự, hệ thống lưu trữ thông tin cá nhân, kỹ năng và vai trò của nhân viên; đồng thời quản lý ca làm việc theo ngày trong tuần, theo dõi hiệu suất qua số lịch hẹn và doanh thu, cũng như đánh dấu ngày nghỉ để làm cơ sở kiểm tra tính khả dụng khi đặt lịch.

Chức năng lịch hẹn là trọng tâm của hệ thống. Khách hàng có thể đặt chỗ trực tuyến 24/7; mọi yêu cầu đều được kiểm tra xung đột theo thời gian, trạng thái được quản lý xuyên suốt vòng đời (từ *pending* đến *completed*), và hỗ trợ đổi lịch, hủy lịch kèm lý do. Lớp thông báo hoạt động như một dịch vụ nền, phát hành **email** và **SMS** theo mẫu (**template**) được quản trị, đồng thời ghi nhận lịch sử gửi để phục vụ kiểm tra. Ở tầng quan sát vận hành, **dashboard** trình bày các KPI cốt lõi như số lịch hẹn theo ngày/tháng, doanh thu, tỷ lệ hủy, điểm hài lòng và danh sách dịch vụ phổ biến; dữ liệu được tổng hợp theo thời gian thực từ cơ sở dữ liệu.

3.1.2. Phân tích chi tiết các chức năng

3.1.2.1. Chức năng đặt lịch hẹn

Chức năng đặt lịch hẹn cho phép khách hàng lựa chọn dịch vụ mong muốn, tùy chọn chỉ định nhân viên hoặc để hệ thống tự động gán theo nguyên tắc tối ưu tài nguyên. Người dùng lựa chọn ngày và giờ phù hợp; ở thời điểm xác nhận, hệ thống kiểm tra tính khả dụng của nhân viên và ngăn chặn mọi xung đột với các lịch đã xác nhận trước

đó. Nếu người dùng chưa đăng nhập, các trường thông tin cơ bản (họ tên, số điện thoại, email) được yêu cầu nhằm bảo đảm liên lạc và xác thực tối thiểu. Sau khi người dùng xác nhận, hệ thống khởi tạo một bản ghi **appointment** ở trạng thái khởi đầu (*pending*), đồng thời phát hành thông báo xác nhận qua email/SMS và cập nhật lịch làm việc của nhân viên tương ứng.

Việc đặt lịch chỉ hợp lệ khi dịch vụ đang hoạt động (*isActive* = *true*), nhân viên rảnh trong khung giờ chọn, không có xung đột với phiên hẹn khác và dữ liệu khách hàng đáp ứng các ràng buộc định dạng. Kết quả của một yêu cầu hợp lệ là một lịch hẹn được lưu bền vững cùng mã định danh duy nhất, trạng thái ban đầu phù hợp chính sách, thông báo đã gửi thành công và lịch nhân viên đã được chặn khung giờ.

3.1.2.2. Chức năng quản lý dashboard

Dashboard dành cho quản trị viên cung cấp cái nhìn tổng quan về tình hình kinh doanh. Các thành phần hiển thị bao gồm tổng số lịch hẹn theo ngày hiện tại, theo tháng và tích lũy; doanh thu ở các mốc thời gian tương ứng; tỷ lệ hủy được tính trên tổng số lịch; điểm hài lòng khách hàng (CSAT) tổng hợp từ phản hồi; danh sách năm dịch vụ được đặt nhiều nhất; và năm lịch hẹn sắp diễn ra. Về phương thức tính toán, hệ thống xác định doanh thu dựa trên các lịch đã hoàn tất, tổng hợp bằng các phép *filter* theo trạng thái và khung thời gian, sau đó sử dụng **aggregation pipeline** để trích xuất KPI theo nhóm, đảm bảo tính cập nhật và hiệu năng khi dữ liệu tăng trưởng.

3.1.2.3. Chức năng xác thực và phân quyền

Cơ chế xác thực dựa trên **JWT** kết hợp **cookie httpOnly** nhằm giảm thiểu rủi ro XSS. Quy trình đăng nhập bao gồm tìm kiếm người dùng theo email, so sánh mật khẩu sau băm **bcrypt**, tạo token chứa thông tin nhận diện (id, role, thời hạn) và lưu token ở cookie an toàn trước khi trả về hồ sơ người dùng đã được ẩn trường nhạy cảm. Chính sách phân quyền được hiện thực bằng **middleware** kiểm tra token và vai trò: người dùng với vai trò **admin** có quyền truy cập đầy đủ các tác vụ quản trị; người dùng thông thường chỉ có quyền thao tác trong phạm vi cá nhân (đặt lịch, xem lịch sử, cập nhật hồ sơ). Mọi tuyến đường (*route*) yêu cầu đặc quyền đều đi qua lớp bảo vệ trung gian để ngăn truy cập trái phép và ghi nhận vi phạm (nếu có).

3.2. Phân tích yêu cầu phi chức năng

3.2.1. Yêu cầu hiệu suất

Hệ thống được thiết kế hướng tới độ trễ thấp và khả năng đáp ứng ổn định dưới tải trung bình–cao. Ở tầng API, mục tiêu thời gian phản hồi cho các tác vụ đơn giản nằm dưới 200ms trong điều kiện vận hành bình thường; các truy vấn đọc/ghi ở mức nghiệp vụ tiêu chuẩn phải hoàn tất trong khoảng 100ms tại lớp cơ sở dữ liệu. Trải nghiệm ban đầu của người dùng được bảo đảm bằng thời gian tải trang đầu tiên dưới 3 giây, trong khi các thao tác tìm kiếm dịch vụ có thể xử lý trong vòng 500ms nhờ tối ưu chỉ mục và giới hạn phạm vi dữ liệu.

Năng lực thông lượng được xác lập cho tối thiểu 100 người dùng đồng thời với mức 1.000 yêu cầu/phút mà không suy giảm đáng kể về chất lượng dịch vụ. Việc sử dụng connection pooling ở **MongoDB** duy trì từ 10 đến 20 kết nối hoạt động, cân bằng giữa chi phí tài nguyên và nhu cầu truy vấn. Khả năng mở rộng theo chiều ngang được dự liệu bằng việc bổ sung instance ứng dụng khi tải tăng; dữ liệu tĩnh và tài nguyên giao diện được phân phối qua **CDN**, còn cơ sở dữ liệu hỗ trợ **replica set** và **sharding** để mở rộng khi cần.

3.2.2. Yêu cầu bảo mật

Cơ chế xác thực và phân quyền tuân theo nguyên tắc **security by design**. **JWT** được ký bằng **secret key** mạnh và đặt thời hạn phù hợp, lưu trữ trong **cookie httpOnly** kết hợp cờ *secure* và *sameSite* để giảm thiểu nguy cơ XSS và CSRF. Mật khẩu được băm bằng **bcrypt** với *salt rounds* ở mức 12, cân bằng giữa độ an toàn và chi phí tính toán. Mọi tuyến đường yêu cầu đặc quyền đi qua middleware kiểm tra vai trò theo mô hình **role-based access**, bảo đảm chỉ những người dùng hợp lệ mới có thể thực hiện các thao tác quản trị.

Bảo vệ dữ liệu đầu vào được thực hiện xuyên suốt bằng **express-validator** nhằm ngăn các mẫu tấn công chèn mã và biến dạng dữ liệu. Dữ liệu trao đổi giữa client và server luôn đi qua **HTTPS**, bảo vệ trước nguy cơ nghe lén và tấn công xen giữa. Trên giao diện, các trường có khả năng chứa HTML được làm sạch, tránh kịch bản XSS phản chiếu hoặc lưu trữ. Chính sách **SameSite** đối với cookie cùng quy ước CORS chặt chẽ tiếp tục thu hẹp bề mặt tấn công trong các tương tác liên miền.

Về quyền riêng tư, hệ thống không bao giờ lưu trữ mật khẩu dạng thô; dữ liệu nhận diện cá nhân được xử lý theo nguyên tắc tối thiểu và chỉ hiển thị khi có mục đích

nghiệp vụ rõ ràng. Nhật ký truy cập ở phạm vi quản trị được lưu trữ để phục vụ truy vết và kiểm toán, tuân thủ các quy định nội bộ về an ninh thông tin.

3.2.3. Yêu cầu khả dụng

Mức khả dụng mục tiêu của hệ thống là 99,5% theo tháng, tương đương khoảng 3,6 giờ dừng dịch vụ, với tỷ lệ lỗi không vượt quá 0,1% tổng số yêu cầu. Khi xảy ra sự cố, mục tiêu khôi phục (RTO) được giới hạn dưới 30 phút nhờ quy trình phục hồi chuẩn hóa và cơ chế giám sát chủ động. Tính tin cậy của dữ liệu được bảo đảm bằng sao lưu **hàng ngày** ở cấp cơ sở dữ liệu; các lỗi ứng dụng được xử lý mềm dẻo (graceful) và phản hồi thân thiện, tránh để người dùng rơi vào trạng thái không xác định. Hệ thống giám sát theo thời gian thực kết hợp cảnh báo sớm và **health check endpoint** giúp phát hiện, cô lập và khắc phục sự cố nhanh chóng.

3.2.4. Yêu cầu khả năng sử dụng

Giao diện người dùng được phát triển theo định hướng **responsive** để vận hành mượt mà trên desktop, tablet và mobile. Cấu trúc điều hướng và bố cục theo chuẩn tối giản, ưu tiên khả năng nhận biết, giúp người dùng hoàn thành tác vụ với số bước tối thiểu. Trong suốt vòng đời tương tác, các trạng thái *loading*, *success* và *error* được hiển thị rõ ràng, hướng dẫn cách khắc phục khi dữ liệu không hợp lệ.

Tiêu chí **accessibility** được lồng ghép ngay từ khâu thiết kế: khả năng điều hướng bằng bàn phím, tương thích **screen reader**, độ tương phản màu đáp ứng ngưỡng khuyến nghị và cỡ chữ có thể điều chỉnh. Các thành phần biểu mẫu, bảng và nút lệnh tuân thủ hệ thống ký hiệu nhất quán để người dùng nhận biết trạng thái và ưu tiên hành động.

3.2.5. Yêu cầu tương thích

Hệ thống hướng đến khả năng hoạt động ổn định trên các trình duyệt hiện đại như Chrome, Firefox, Safari và Edge ở các phiên bản tương ứng mới, đồng thời duy trì lộ trình **progressive enhancement** để chức năng nền tảng vẫn khả dụng trên môi trường kém hiện đại hơn. Về thiết bị, ứng dụng được tối ưu cho Windows, macOS và Linux ở khối desktop; iOS và Android ở khối di động; cùng dải kích thước màn hình phổ biến từ 320px đến 2560px. Trên mobile, các cử chỉ chạm và cuộn được tối ưu để đảm bảo thao tác thuận tiện, trong khi trên màn hình lớn, thông tin dữ liệu và biểu đồ có thể hiển thị song song để hỗ trợ ra quyết định nhanh.

3.3. Phân tích các module chính

Hệ thống quản lý Spa được thiết kế theo hướng **modular architecture**, trong đó mỗi module đảm nhận một nhóm chức năng cụ thể và có khả năng hoạt động độc lập. Các module này giao tiếp thông qua **RESTful API**, tuân theo nguyên tắc "separation of concerns" để dễ bảo trì, kiểm thử và mở rộng. Cấu trúc tổng thể bao gồm năm module cốt lõi: **Authentication**, **Admin Management**, **Appointment Management**, **User Management** và **Notification System**.

3.3.1. Module Authentication

3.3.1.1. Cấu trúc module

```
auth/
├── controllers/
│   └── authController.js
├── middleware/
│   └── auth.js
├── routes/
│   └── authRoutes.js
└── utils/
    └── jwt.js
```

3.3.1.2. Chức năng chính

Module Authentication chịu trách nhiệm xác thực, phân quyền và quản lý danh tính người dùng. Nó bao gồm các tính năng đăng ký tài khoản mới, đăng nhập, đăng xuất, đặt lại mật khẩu, lấy lại thông tin tài khoản và cập nhật hồ sơ cá nhân. Token xác thực được phát hành dưới dạng **JWT (JSON Web Token)**, được lưu trong cookie **httpOnly** để đảm bảo an toàn. Hệ thống hỗ trợ cơ chế **role-based access control (RBAC)**, giúp phân biệt quyền truy cập giữa người dùng thông thường và quản trị viên.

3.3.1.3. Business Logic

Quy trình xác thực bắt đầu khi người dùng nhập email và mật khẩu. Hệ thống kiểm tra thông tin, mã hóa mật khẩu bằng **bcrypt**, sau đó tạo **JWT token** chứa thông tin định danh. Middleware auth.js được áp dụng để xác minh token trong mọi request. Các chức năng quản lý hồ sơ cá nhân cho phép người dùng cập nhật thông tin cơ bản, thay đổi mật khẩu, hoặc khôi phục tài khoản nếu quên.

3.3.1.4. API Endpoints

Phương thức	Endpoint	Mô tả
POST	/api/auth/register	Đăng ký tài khoản mới
POST	/api/auth/login	Đăng nhập hệ thống
POST	/api/auth/logout	Đăng xuất
POST	/api/auth/forgot-password	Gửi yêu cầu đặt lại mật khẩu
POST	/api/auth/reset-password	Đặt lại mật khẩu qua token
GET	/api/auth/me	Lấy thông tin người dùng hiện tại
PUT	/api/auth/profile	Cập nhật hồ sơ người dùng

Bảng 3.1. API Endpoints

3.3.2. Module Admin Management

3.3.2.1. Cấu trúc module

```
admin/
├── controllers/
│   ├── dashboardController.js
│   ├── servicesController.js
│   ├── staffController.js
│   ├── appointmentsController.js
│   ├── categoriesController.js
│   ├── userController.js
│   └── notificationsController.js
├── routes/
│   ├── dashboard.js
│   ├── services.js
│   ├── staff.js
│   ├── appointments.js
│   ├── categories.js
│   ├── user.js
│   └── notifications.js
└── middleware/
    └── requireAdmin.js
```

3.3.2.2. Chức năng chính

Module này cung cấp giao diện và API cho quản trị viên thực hiện các nghiệp vụ quản lý hệ thống. Bao gồm:

- **Dashboard**: Tổng hợp dữ liệu KPI, doanh thu và lịch hẹn sắp tới.
- **Service Management**: Thêm, sửa, xóa và phân loại dịch vụ.
- **Staff Management**: Quản lý nhân viên, lịch làm việc, hiệu suất và tình trạng hoạt động.
- **Appointment Management**: Theo dõi, điều chỉnh hoặc hủy lịch hẹn.
- **User Management**: Theo dõi khách hàng, cập nhật hồ sơ, phân loại theo mức độ VIP.
- **Notifications Management**: Quản lý mẫu thông báo, lịch gửi và lịch sử gửi thông báo.

3.3.2.3. Business Logic

Hệ thống dashboard sử dụng **MongoDB Aggregation Pipeline** để tổng hợp dữ liệu đa chiều, bao gồm tổng doanh thu, tỷ lệ hủy, dịch vụ phổ biến và số lượng khách hàng mới.

Các bộ điều khiển (**controller**) tuân theo mẫu **Service–Repository pattern**, trong đó các thao tác nghiệp vụ phức tạp (ví dụ: phân tích KPI, tổng hợp lịch hẹn) được xử lý ở tầng **service**, giúp đảm bảo tách biệt giữa logic và truy cập dữ liệu. Middleware requireAdmin được sử dụng để giới hạn quyền truy cập vào các endpoint thuộc khu vực quản trị.

3.3.3. Module Appointment Management

3.3.3.1. Cấu trúc module

```
appointments/
├── models/
│   └── Appointment.js
├── controllers/
│   └── appointmentsController.js
├── routes/
│   └── appointments.js
└── services/
    └── conflictDetection.js
```

3.3.3.2. Chức năng chính

Module này là trung tâm của hệ thống, đảm nhận việc tạo, cập nhật, hủy và theo dõi các lịch hẹn giữa khách hàng và nhân viên. Các chức năng chính bao gồm:

- **Tạo lịch hẹn mới** với thông tin dịch vụ, nhân viên, khách hàng.
- **Kiểm tra xung đột lịch** bằng thuật toán kiểm tra giao thoa thời gian.
- **Cập nhật trạng thái** theo quy trình: pending → confirmed → in-progress → completed → archived.
- **Gửi thông báo tự động** khi có thay đổi trạng thái hoặc hủy lịch.
- **Lưu lịch sử và phản hồi** của khách hàng sau dịch vụ.

3.3.3.3. Business Rules

Mỗi lịch hẹn có mã định danh duy nhất (**appointmentNumber**). Hệ thống ngăn chặn xung đột lịch bằng cách kiểm tra khung giờ của nhân viên trước khi xác nhận. Trạng thái lịch hẹn được giới hạn trong tập hợp giá trị hợp lệ và thay đổi có kiểm soát. Khi lịch bị hủy, lý do và người thực hiện (khách hàng hay quản trị viên) được ghi lại. Sau khi dịch vụ hoàn tất, khách hàng có thể gửi đánh giá để cập nhật chỉ số hài lòng.

4.3.3.4. Data Model

Trường	Kiểu dữ liệu	Mô tả
appointmentNumber	String	Mã lịch hẹn duy nhất
customerId	ObjectId (User)	Tham chiếu khách hàng
serviceId	ObjectId (Service)	Tham chiếu dịch vụ
staffId	ObjectId (Staff)	Tham chiếu nhân viên
appointmentDate	Date	Ngày thực hiện lịch hẹn
startTime / endTime	String	Giờ bắt đầu / kết thúc
status	Enum	Trạng thái: pending, confirmed, completed, cancelled
totalAmount	Number	Tổng tiền dịch vụ
note	String	Ghi chú của khách hàng
rating	Number	Đánh giá sau khi hoàn thành

Bảng 3.2. Data Model

3.3.4. Module User Management

3.3.4.1. Cấu trúc module

user/

```
├── models/
│   └── User.js
├── controllers/
│   ├── profileController.js
│   └── bookingsController.js
├── routes/
│   ├── profile.js
│   └── bookings.js
└── services/
    └── userService.js
```

3.3.4.2. Chức năng chính

Module này tập trung vào việc quản lý hồ sơ khách hàng, lịch sử đặt lịch và hệ thống điểm thưởng.

Người dùng có thể xem và chỉnh sửa thông tin cá nhân, tra cứu các lịch hẹn trước đây, thiết lập sở thích dịch vụ, và theo dõi tổng chi tiêu để tích lũy điểm thưởng. Hệ thống tự động gán **level** dựa trên ngưỡng chi tiêu, từ đó phân loại khách hàng thành **Thường**, **Loyal**, **Premium** hoặc **VIP**.

3.3.4.3. Business Logic

Cơ chế phân loại được triển khai thông qua **middleware pre-save** của Mongoose: mỗi khi giá trị tổng chi tiêu thay đổi, hệ thống tự động tính toán lại cấp độ. Phần thưởng hoặc quyền lợi được cập nhật tương ứng, chẳng hạn khách hàng VIP nhận ưu đãi giảm giá hoặc lịch ưu tiên. Các controller trong module sử dụng userService để tương tác với database, giảm thiểu trùng lặp và đảm bảo tính nhất quán.

3.3.5. Module Notification System

3.3.5.1. Cấu trúc module

```
notifications/
├── models/
│   ├── NotificationTemplate.js
│   └── NotificationLog.js
├── controllers/
│   └── notificationsController.js
├── services/
```

```
│   ├── emailService.js
│   └── smsService.js
└── routes/
    └── notifications.js
```

3.3.5.2. Chức năng chính

Module này đảm nhận toàn bộ quá trình gửi và ghi nhận thông báo trong hệ thống. Các mẫu thông báo (**templates**) được quản lý tập trung; khi có sự kiện nghiệp vụ như "đặt lịch", "xác nhận", "hủy" hoặc "hoàn thành", hệ thống tự động phát hành email/SMS đến người liên quan. Các thông báo được lưu lại trong **NotificationLog** để phục vụ kiểm tra hoặc gửi lại khi cần.

3.3.5.3. Các loại thông báo

Loại thông báo	**Mô tả**
Booking Confirmation	Xác nhận lịch hẹn thành công
Reminder	Nhắc nhở trước giờ hẹn
Cancellation	Thông báo hủy lịch hẹn
Completion	Xác nhận hoàn thành dịch vụ
Follow-up	Gợi ý đánh giá hoặc đặt lại dịch vụ

Bảng 3.3. Các loại thông báo

3.3.5.4. Business Logic

Module sử dụng hai lớp dịch vụ gửi thông báo: **emailService** (qua SMTP) và **smsService** (qua API đối tác). Các thao tác gửi được thực hiện bất đồng bộ, giúp giảm thời gian phản hồi của API chính. Lịch sử gửi (thời gian, người nhận, trạng thái, lỗi) được lưu để phục vụ thống kê và khắc phục sự cố.

3.4. Phân tích luồng xử lý

Phần này mô tả chi tiết cách các module trong hệ thống tương tác với nhau để thực hiện các nghiệp vụ chính, bao gồm: **đặt lịch hẹn**, **xác thực người dùng**, và **tổng hợp dữ liệu dashboard**. Các luồng xử lý được thiết kế theo nguyên tắc **tách biệt frontend – backend**, trong đó phía client chịu trách nhiệm thu thập và hiển thị dữ liệu, còn backend xử lý nghiệp vụ và truy xuất cơ sở dữ liệu. Mỗi luồng đều có cơ chế **xử lý lỗi (error handling)** và **kiểm tra điều kiện (validation)** để đảm bảo tính chính xác, nhất quán và độ tin cậy của hệ thống.

3.4.1. Luồng xử lý đặt lịch hẹn

3.4.1.1. Frontend Flow

Luồng xử lý đặt lịch hẹn trên giao diện người dùng bắt đầu khi khách hàng chọn dịch vụ mong muốn. Giao diện hiển thị danh sách nhân viên khả dụng, thời gian làm việc và các khung giờ trống tương ứng. Người dùng nhập thông tin cá nhân (nếu chưa đăng nhập), chọn nhân viên, ngày và giờ, sau đó xác nhận đặt lịch. Trước khi gửi yêu cầu đến máy chủ, frontend thực hiện bước **kiểm tra hợp lệ (client-side validation)** để đảm bảo dữ liệu có định dạng đúng, đầy đủ và hợp lệ.

Khi thông tin hợp lệ, frontend gửi yêu cầu **POST /api/appointments** đến server. Sau khi nhận phản hồi thành công, giao diện hiển thị thông báo xác nhận, đồng thời chuyển hướng người dùng đến trang "xác nhận lịch hẹn" (**confirmation page**) với mã định danh lịch hẹn (appointmentNumber). Trong trường hợp lỗi, như nhân viên không khả dụng hoặc dữ liệu bị thiếu, frontend hiển thị thông báo tương ứng và giữ nguyên dữ liệu đã nhập để người dùng chỉnh sửa.

3.4.1.2. Backend Flow

Phía server, controller **appointmentsController** tiếp nhận yêu cầu từ client và thực hiện chuỗi kiểm tra logic như sau:

1. **Validate Input:** Kiểm tra các trường bắt buộc như serviceId, appointmentDate, startTime, staffId, và thông tin khách hàng.
2. **Check Conflicts:** Gọi service conflictDetection.js để xác minh nhân viên có bị trùng lịch hay không. Nếu có, trả về lỗi "Conflict Detected".
3. **Get Service Details:** Lấy thông tin dịch vụ tương ứng để xác định thời lượng và giá.
4. **Calculate EndTime:** Tự động tính giờ kết thúc dựa trên thời lượng dịch vụ.
5. **Create Appointment:** Tạo bản ghi mới trong cơ sở dữ liệu với trạng thái mặc định **pending**.
6. **Send Notification:** Gọi module **Notification System** để gửi email/SMS xác nhận đặt lịch cho khách hàng và nhân viên liên quan.
7. **Return Response:** Trả về thông tin lịch hẹn kèm mã định danh duy nhất.
8. **Error Handling:** Nếu có lỗi (ví dụ: không tìm thấy nhân viên hoặc lỗi lưu cơ sở dữ liệu), hệ thống trả về mã lỗi thích hợp (4xx hoặc 5xx) cùng thông điệp chi tiết.

Kết quả cuối cùng của quy trình là một lịch hẹn được ghi nhận trong hệ thống, khách hàng nhận thông báo xác nhận và lịch làm việc của nhân viên được cập nhật tức thì.

3.4.2. Luồng xử lý xác thực

3.4.2.1. Login Flow

Quy trình đăng nhập của hệ thống được thiết kế theo cơ chế **JWT Authentication** kết hợp **cookie httpOnly**, đảm bảo tính bảo mật và trải nghiệm người dùng liền mạch.

1. **Find User:** Khi người dùng gửi yêu cầu đăng nhập, hệ thống tìm kiếm bản ghi User theo email.
2. **Check Password:** Mật khẩu được so sánh với bản mã hóa **bcrypt** trong cơ sở dữ liệu.
3. **Generate JWT Token:** Nếu thông tin hợp lệ, hệ thống tạo token chứa ID và vai trò của người dùng.
4. **Set Cookie:** Token được lưu trong cookie httpOnly, có thời hạn 7 ngày, giúp duy trì phiên đăng nhập an toàn.
5. **Update Last Login:** Trường lastLogin của người dùng được cập nhật để ghi nhận thời điểm truy cập gần nhất.
6. **Return Response:** API trả về thông tin người dùng cơ bản (không bao gồm mật khẩu) cùng trạng thái đăng nhập thành công.
7. **Error Handling:** Nếu sai mật khẩu, tài khoản không tồn tại hoặc bị khóa, hệ thống trả về thông báo lỗi tương ứng.

Luồng xử lý này đảm bảo tính bảo mật và ổn định trong quá trình xác thực, đồng thời cung cấp thông tin tối thiểu cần thiết cho frontend để hiển thị giao diện phù hợp với vai trò của người dùng.

3.4.2.2. Authentication Middleware Flow

Middleware auth.js là thành phần trung gian giúp kiểm tra và xác thực người dùng trong mọi yêu cầu API cần bảo vệ. Cơ chế hoạt động của middleware bao gồm các bước:

1. **Get Token:** Trích xuất JWT token từ cookie gửi kèm request.
2. **Verify Token:** Giải mã token và kiểm tra chữ ký với secret key.
3. **Find User:** Xác định người dùng từ cơ sở dữ liệu dựa trên ID trong token.
4. **Check Status:** Kiểm tra người dùng có đang hoạt động (isActive) hay không.

5. **Attach User:** Nếu hợp lệ, thông tin người dùng được gắn vào request để các controller phía sau có thể sử dụng.
6. **Continue Flow:** Cho phép request tiếp tục xử lý.
7. **Error Handling:** Nếu token hết hạn, không hợp lệ hoặc người dùng bị khóa, middleware trả về mã lỗi 401 Unauthorized.

Cơ chế này giúp duy trì phân quyền rõ ràng và đảm bảo rằng chỉ những người dùng đã được xác thực mới có thể truy cập các tài nguyên bảo mật trong hệ thống.

3.4.3. Luồng xử lý Dashboard

3.4.3.1. Data Aggregation Flow

Dashboard là điểm trung tâm giúp quản trị viên theo dõi hiệu quả kinh doanh. Dữ liệu hiển thị được tổng hợp thông qua quy trình phân tích nhiều tầng, sử dụng **MongoDB Aggregation Framework** để tối ưu tốc độ và độ chính xác.

1. **Get Appointments:** Hệ thống truy xuất toàn bộ lịch hẹn trong khoảng thời gian xác định, bao gồm cả thông tin dịch vụ, khách hàng và nhân viên.
2. **Calculate Statistics:** Các chỉ số như tổng số lịch hẹn, lịch hoàn thành, lịch đang chờ được tính toán và phân loại theo ngày, tuần hoặc tháng.
3. **Calculate Today's Data:** Tách riêng dữ liệu của ngày hiện tại để hiển thị nổi bật trên dashboard.
4. **Calculate Revenue:** Tính tổng doanh thu dựa trên các lịch hẹn đã hoàn tất, đồng thời lọc theo khung thời gian để xác định doanh thu ngày, tháng, năm.
5. **Get Counts:** Đếm số lượng khách hàng, dịch vụ và nhân viên đang hoạt động.
6. **Get Upcoming Appointments:** Trích xuất danh sách năm lịch hẹn gần nhất sắp diễn ra để quản trị viên dễ theo dõi.
7. **Get Top Services:** Sử dụng **aggregation pipeline** để xác định năm dịch vụ được đặt nhiều nhất trong giai đoạn gần nhất.
8. **Calculate Cancellation Rate:** Tính tỷ lệ hủy lịch dựa trên tổng số lịch hẹn.
9. **Compile Data:** Tổng hợp tất cả dữ liệu trên thành đối tượng JSON duy nhất trả về frontend.
10. **Error Handling:** Trong trường hợp lỗi truy vấn, hệ thống ghi log chi tiết và trả về dữ liệu mặc định để không gián đoạn giao diện dashboard.

CHƯƠNG 4
THIẾT KẾ HỆ THỐNG

4.1. Kiến trúc tổng thể hệ thống

4.1.1. Kiến trúc Client–Server

Hệ thống quản lý Spa trực tuyến được thiết kế theo mô hình **Client–Server** với kiến trúc **3-tier (ba tầng)** bao gồm: tầng trình bày (Client Tier), tầng ứng dụng (Application Tier) và tầng cơ sở dữ liệu (Database Tier). Kiến trúc này giúp tách biệt rõ ràng giữa giao diện người dùng, xử lý nghiệp vụ và quản lý dữ liệu, qua đó nâng cao khả năng mở rộng, bảo trì và tái sử dụng của hệ thống.

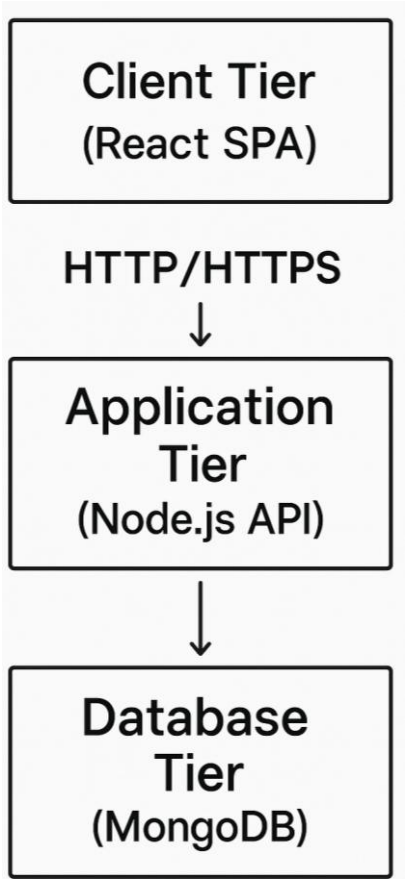
Tầng **Client** được xây dựng bằng **React.js** dưới dạng Single Page Application (SPA). Ứng dụng React được triển khai cùng công cụ **Vite** để tối ưu hiệu suất build và khả năng hot reload trong quá trình phát triển. Giao diện người dùng được thiết kế bằng sự kết hợp giữa **Ant Design** và **Tailwind CSS**, bảo đảm tính thẩm mỹ, nhất quán và khả năng phản hồi (responsive) trên nhiều loại thiết bị. Quản lý trạng thái trong ứng dụng được thực hiện thông qua **React Hooks** như useState, useEffect và useContext, cùng với hệ thống định tuyến linh hoạt của **React Router DOM**. Việc giao tiếp với máy chủ được thực hiện qua thư viện **Axios**, có cấu hình sẵn các interceptors để kiểm soát yêu cầu, phản hồi và xử lý lỗi tập trung.

Tầng **Application** được xây dựng bằng **Node.js** kết hợp với **Express.js**, hiện thực hóa mô hình **MVC (Model–View–Controller)** và chuẩn **RESTful API**. Tầng này chịu trách nhiệm xử lý toàn bộ nghiệp vụ, xác thực người dùng và điều phối dữ liệu giữa client và cơ sở dữ liệu. Hệ thống xác thực sử dụng **JWT (JSON Web Token)** kết hợp với cơ chế cookie-based storage để duy trì phiên làm việc an toàn và nhẹ. Việc kiểm tra và xác thực dữ liệu đầu vào được thực hiện thông qua thư viện **express-validator**, giúp đảm bảo dữ liệu hợp lệ trước khi lưu trữ. Cấu hình **CORS (Cross-Origin Resource Sharing)** được thiết lập chặt chẽ, chỉ cho phép các miền đáng tin cậy truy cập vào tài nguyên API, góp phần bảo vệ hệ thống khỏi các cuộc tấn công cross-domain.

Tầng **Database** sử dụng **MongoDB** với **Mongoose ODM** làm cầu nối giữa dữ liệu và đối tượng trong ứng dụng. Cơ sở dữ liệu được tổ chức linh hoạt theo mô hình document-oriented, cho phép lưu trữ dữ liệu có cấu trúc phong phú, dễ dàng mở rộng và thay đổi theo yêu cầu nghiệp vụ. Hệ thống áp dụng chiến lược **connection pooling**

để tối ưu hóa kết nối và giảm độ trễ khi truy cập dữ liệu, đồng thời sử dụng các **compound indexes** và **text indexes** nhằm cải thiện hiệu năng truy vấn, đặc biệt trong các thao tác tìm kiếm và lọc dữ liệu lớn. Toàn bộ schema trong Mongoose được định nghĩa với **schema-level validation**, đảm bảo dữ liệu luôn nhất quán và hợp lệ trong mọi thao tác CRUD.

Kiến trúc tổng thể này được minh họa qua sơ đồ sau:



Hình 4.1. Kiến trúc Client–Server

Nhờ việc áp dụng mô hình phân tầng này, hệ thống có khả năng mở rộng linh hoạt, dễ bảo trì và đáp ứng tốt các yêu cầu phát triển trong tương lai. Mỗi tầng có thể được triển khai, kiểm thử hoặc nâng cấp độc lập mà không ảnh hưởng đến các tầng khác, giúp giảm thiểu rủi ro và tăng hiệu quả trong quy trình phát triển phần mềm.

4.1.2. Kiến trúc module

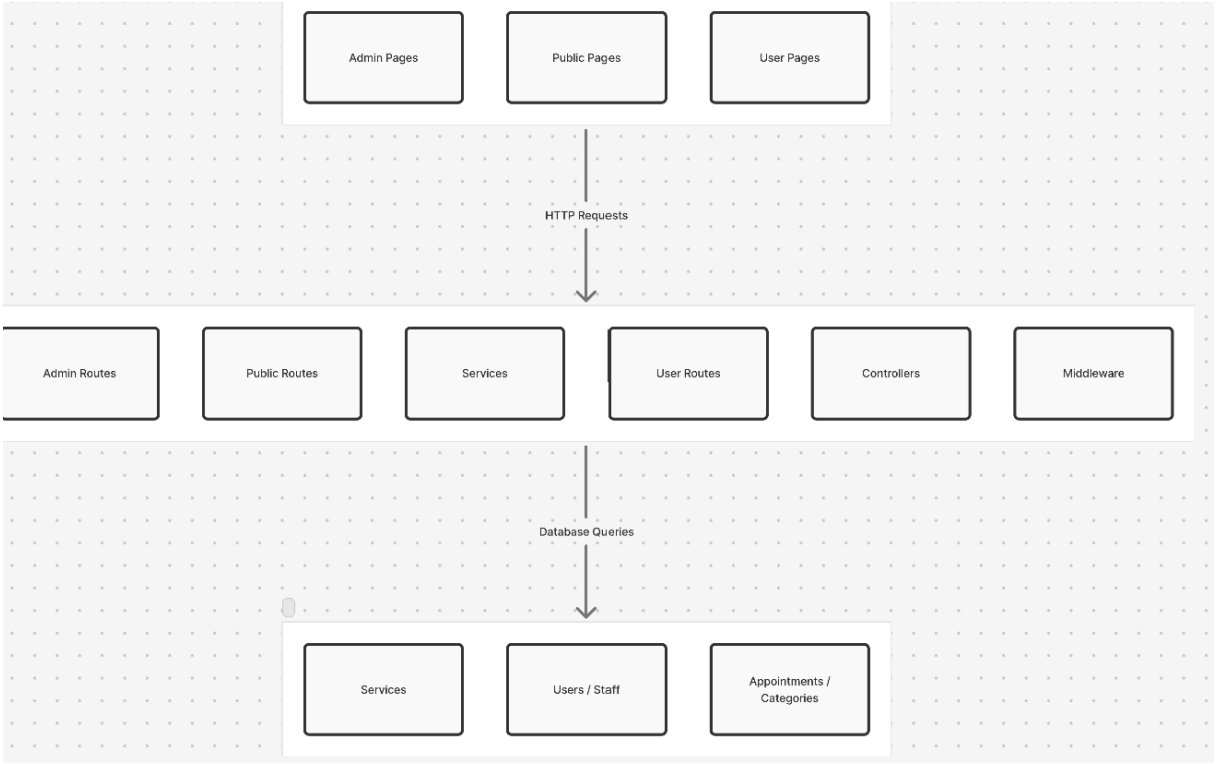
Trên cơ sở kiến trúc ba tầng, hệ thống được chia thành các **module chức năng** độc lập, đảm nhiệm từng phần của quy trình nghiệp vụ Spa. Phía client gồm ba nhóm trang chính: **Public Pages** dành cho khách truy cập (giới thiệu, dịch vụ, liên hệ), **User Pages** dành cho người dùng đã đăng nhập (đặt lịch, hồ sơ cá nhân, lịch sử sử dụng dịch vụ)

và **Admin Pages** dành cho quản trị viên (quản lý dịch vụ, nhân viên, khách hàng, lịch hẹn và thống kê doanh thu).

Tương ứng ở tầng server, các **API routes** được tổ chức thành ba nhóm chính: public routes (truy cập công khai), user routes (người dùng thường) và admin routes (quản trị hệ thống). Mỗi nhóm route liên kết với các **controllers** chịu trách nhiệm xử lý logic nghiệp vụ, **middleware** dùng để xác thực, kiểm tra quyền truy cập và xử lý lỗi, cùng với các **services** thực hiện thao tác dữ liệu thông qua repository hoặc trực tiếp với cơ sở dữ liệu MongoDB.

Bên dưới tầng API là tầng cơ sở dữ liệu gồm các **collections** chính như Users, Appointments, Services, Staff và Categories. Mỗi collection đại diện cho một thực thể trong hệ thống và có mối quan hệ rõ ràng, bảo đảm luồng dữ liệu thống nhất giữa các tầng.

Sơ đồ sau minh họa mô hình module tổng thể của hệ thống:



Hình 4.2. Kiến trúc module

Nhờ cách tổ chức này, việc mở rộng thêm các module mới như quản lý khuyến mãi, tích điểm hoặc thanh toán trực tuyến có thể được thực hiện dễ dàng mà không làm thay đổi cấu trúc tổng thể.

4.1.3. Luồng xử lý request

Luồng xử lý yêu cầu trong hệ thống được vận hành theo quy trình chuẩn **HTTP request–response cycle** giữa client và server. Khi người dùng gửi yêu cầu (ví dụ đặt lịch hẹn), React client sẽ gửi request thông qua **Axios** đến một endpoint REST API cụ thể. Tại backend, **Express.js** tiếp nhận yêu cầu, đi qua chuỗi **middleware** bao gồm kiểm tra CORS, xác thực JWT, phân quyền truy cập, kiểm tra dữ liệu và ghi log hoạt động. Sau đó, request được chuyển đến **controller** tương ứng, nơi gọi các **service** để xử lý nghiệp vụ, truy vấn dữ liệu thông qua **Mongoose models** và trả về response có cấu trúc JSON.

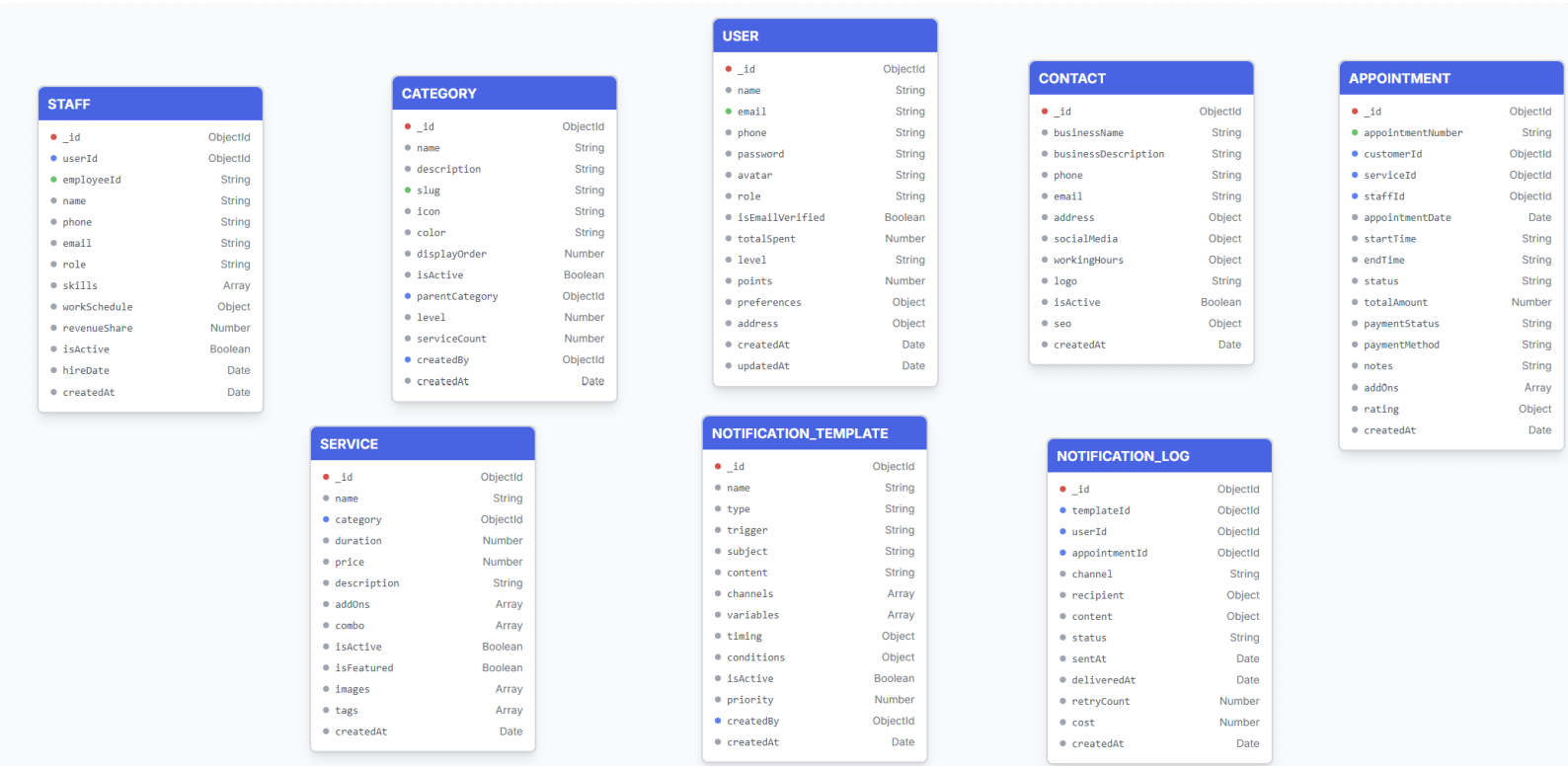
Để minh họa quy trình này, đoạn cấu hình server dưới đây cho thấy cách thiết lập pipeline xử lý yêu cầu trong ứng dụng Node.js:

```
import express from "express";
import cors from "cors";
import dotenv from "dotenv";
import cookieParser from "cookie-parser";

dotenv.config();
connectDB();

const app = express();
const port = config.PORT;

// Middleware pipeline
app.use(
  cors({
    origin: config.CORS.ALLOWED_ORIGINS,
    credentials: true,
  })
);
app.use(express.json({ limit: "10mb" }));
app.use(express.urlencoded({ extended: true, limit: "10mb" }));
app.use(cookieParser());
```

```
// Logging middleware
app.use((req, res, next) => {
  console.log(`${new Date().toISOString()} - ${req.method} ${req.path}`);
  next();
});

// Route definitions
app.use("/api/auth", authRoutes);
app.use("/api/admin", adminRoutes);
app.use("/api/user/bookings", userBookingsRoutes);
app.use("/api/user/profile", userProfileRoutes);
app.use("/api/public/contact", publicContactRoutes);
app.use("/api/public/services", publicServicesRoutes);
app.use("/api/public/categories", publicCategoriesRoutes);
app.use("/api/public/staff", publicStaffRoutes);
```

Luồng xử lý này đảm bảo rằng mọi yêu cầu đều được kiểm tra, xác thực, ghi log và phản hồi một cách thống nhất. Mỗi tầng trong pipeline đều có trách nhiệm rõ ràng, giúp việc kiểm thử, giám sát và xử lý lỗi trở nên dễ dàng hơn. Đây chính là nền tảng vận hành của hệ thống, đảm bảo sự ổn định, bảo mật và khả năng mở rộng khi triển khai trong môi trường thực tế.

4.2. Thiết kế cơ sở dữ liệu



Hình 4.3. Mongodb Scheme Diagram

4.2.1. Bảng: User

Tên trường	Kiểu dữ liệu	Ràng buộc / Validation	Mô tả & ghi chú
_id	ObjectId	PK	Khóa chính do MongoDB sinh
name	String	required, trim, min 2, max 50	Họ tên người dùng
email	String	required, unique, lowercase, isEmail	Email đăng nhập
phone	String	optional, regex ^[0-9]{9,11}$	SĐT (nếu có)
password	String	required,	Mật khẩu đã băm

		min 6, select:false	(bcrypt)
role	String	enum: user, admin, default user	Vai trò hệ thống
isEmailVerified	Boolean	default false	Trạng thái xác minh email
resetPasswordToken	String	optional	Token đặt lại mật khẩu
resetPasswordExpires	Date	optional	Hạn token reset
emailVerificationToken	String	optional	Token xác minh email
emailVerificationExpires	Date	optional	Hạn token xác minh
dateOfBirth	Date	optional	Ngày sinh
gender	String	enum: male, female, other	Giới tính
totalSpent	Number	min 0, default 0	Tổng chi tiêu
totalAppointments	Number	min 0, default 0	Tổng số lịch hẹn
level	String	enum: Thường, Loyal, Premium, VIP, default Thường	Phân hạng khách hàng
points	Number	min 0, default 0	Điểm tích lũy
preferences.services	[ObjectId→Service]	optional	Dịch vụ ưa thích
preferences.staff	[ObjectId→Staff]	optional	Nhân viên ưa thích

preferences.timeSlots	[String]	optional	Khung giờ ưa thích
preferences.communication	Object	defaults	Opt-in SMS/Email/Phone
address	Object	optional	Địa chỉ (street, city, state, zipCode)
emergencyContact	Object	optional	Liên hệ khẩn cấp
isActive	Boolean	default true	Trạng thái tài khoản
lastLogin	Date	default Date.now	Lần đăng nhập gần nhất
lastBooking	Date	optional	Lần đặt lịch gần nhất
lastVisit	Date	optional	Lần đến Spa gần nhất
source	String	enum	Kênh tiếp cận (website, referral, …)

Bảng 4.1. Bảng user

4.2.2. Bảng: Staff

Tên trường	**Kiểu dữ liệu**	**Ràng buộc / Validation**	**Mô tả & ghi chú**
_id	ObjectId	PK	Khóa chính
userId	ObjectId→User	optional	Liên kết tài khoản người dùng (nếu có)
employeeId	String	unique, uppercase	Mã nhân viên nội bộ
name	String	required, max 50	Họ tên nhân viên
phone	String	regex ^[0-9]{10,11}$	SĐT
email	String	isEmail,	Email công việc

		lowercase	
role	String	enum	Vị trí (Massage Therapist, Receptionist, …)
skills	[String]	optional	Danh sách kỹ năng
workSchedule	Object	required theo ngày	Lịch làm việc (mon–sun: start, end, isWorking)
daysOff	[Date]	optional	Ngày nghỉ
revenueShare	Number	min 0, max 1, default 0.3	Tỷ lệ chia doanh thu
hourlyRate	Number	min 0, default 0	Lương theo giờ
commission	Number	min 0, default 0	Hoa hồng
isActive	Boolean	default true	Trạng thái hoạt động
hireDate	Date	default Date.now	Ngày vào làm
terminationDate	Date	optional	Ngày nghỉ việc
emergencyContact	Object	optional	Liên hệ khẩn cấp
address	Object	optional	Địa chỉ
notes	String	max 500	Ghi chú nội bộ

Bảng 4.2. Bảng Staff

4.2.3. Bảng: Category

Tên trường	**Kiểu dữ liệu**	**Ràng buộc / Validation**	**Mô tả & ghi chú**
_id	ObjectId	PK	Khóa chính
name	String	required, trim, max 100	Tên danh mục
description	String	max 500	Mô tả
slug	String	unique, lowercase	Định danh URL
icon	String	isURL	Biểu tượng (URL)
color	String	hex, default #3B82F6	Mã màu hiển thị
displayOrder	Number	min 0, default 0	Thứ tự hiển thị
isActive	Boolean	default true	Trạng thái

parentCategory	ObjectId→Category	optional	Quan hệ cha–con
level	Number	min 0, default 0	Cấp độ danh mục
path	String	optional	Đường dẫn phân cấp
serviceCount	Number	min 0, default 0	Số dịch vụ thuộc danh mục
seoTitle	String	max 60	SEO title
seoDescription	String	max 160	SEO description
createdBy	ObjectId→User	required	Người tạo
updatedBy	ObjectId→User	optional	Người chỉnh sửa

Bảng 4.3. Bảng Category

4.2.4. Bảng: Service

Tên trường	**Kiểu dữ liệu**	**Ràng buộc / Validation**	**Mô tả & ghi chú**
_id	ObjectId	PK	Khóa chính
name	String	required, trim, max 100	Tên dịch vụ
category	ObjectId→Category	required	Danh mục
duration	Number	required, min 15, max 480	Thời lượng (phút)
price	Number	required, min 0	Giá niêm yết
description	String	max 1000	Mô tả
addOns	[Object]	each: name required, price ≥0	Dịch vụ bổ sung
combo	[Object]	price ≥0, originalPrice ≥0	Gói combo/giá ưu đãi
displayOrder	Number	default 0	Thứ tự hiển thị
isActive	Boolean	default true	Trạng thái
isFeatured	Boolean	default false	Gắn nổi bật
images	[String]	isURL	Hình ảnh minh họa
tags	[String]	optional	Thẻ tìm kiếm
requirements	String	max 500	Yêu cầu trước khi làm dịch vụ

notes	String	max 500	Ghi chú nội bộ

Bảng 4.4. Bảng Service

4.2.5. Bảng: Appointment

Tên trường	**Kiểu dữ liệu**	**Ràng buộc / Validation**	**Mô tả & ghi chú**
_id	ObjectId	PK	Khóa chính
appointmentNumber	String	required, unique	Mã lịch hẹn
customerId	ObjectId→User	required	Khách hàng
serviceId	ObjectId→Service	required	Dịch vụ
staffId	ObjectId→Staff	required	Nhân viên thực hiện
appointmentDate	Date	required	Ngày hẹn
startTime	String	HH:mm, required	Giờ bắt đầu
endTime	String	HH:mm, required	Giờ kết thúc
duration	Number	required, min 15	Thời lượng (phút)
status	String	enum: pending, confirmed, in-progress, completed, cancelled, no-show, default pending	Trạng thái lịch
totalAmount	Number	required, min 0	Tổng tiền

discount	Number	min 0, default 0	Giảm giá
finalAmount	Number	required, min 0	Số tiền cuối cùng
paymentStatus	String	enum: pending, paid, partial, refunded, default pending	Trạng thái thanh toán
paymentMethod	String	enum: cash, card, bank_transfer, points, combo	Phương thức thanh toán
notes	String	max 500	Ghi chú
specialRequests	String	max 500	Yêu cầu đặc biệt
addOns	[Object]	name/price/qty	Add-ons áp dụng
combo	[Object]	name/price/originalPrice	Combo áp dụng
reminders	[Object]	type sms/email/phone, status pending/sent/failed	Lịch sử nhắc hẹn
cancellationReason	String	optional	Lý do hủy
cancelledBy	String	enum customer/staff/admin/system	Ai hủy
cancelledAt	Date	optional	Thời điểm hủy
completedAt	Date	optional	Thời

			điểm hoàn thành
rating.score	Number	1–5	Điểm đánh giá
rating.comment	String	max 500	Nhận xét
rating.ratedAt	Date	optional	Thời điểm đánh giá
followUp	Object	flags & dates	Theo dõi sau dịch vụ

Bảng 4.5. Bảng Appointment

4.2.6. Bảng: Contact

Tên trường	**Kiểu dữ liệu**	**Ràng buộc / Validation**	**Mô tả & ghi chú**
_id	ObjectId	PK	Khóa chính
name	String	required, max 100	Họ tên người liên hệ
email	String	required, isEmail	Email liên hệ
phone	String	optional, regex ^[0-9]{9,11}$	SĐT
subject	String	required, max 150	Chủ đề
message	String	required, max 2000	Nội dung liên hệ
status	String	enum: new, in_progress, resolved, closed, default new	Trạng thái xử lý
handledBy	ObjectId→User	optional	Nhân sự tiếp nhận
handledAt	Date	optional	Thời điểm xử lý
notes	String	max 500	Ghi chú nội bộ

source	String	enum	Kênh gửi (web, social, …)

Bảng 4.6. Bảng Contact

4.2.7. Bảng: NotificationTemplate

Tên trường	Kiểu dữ liệu	Ràng buộc / Validation	Mô tả & ghi chú
_id	ObjectId	PK	Khóa chính
name	String	required, max 100	Tên mẫu thông báo
code	String	required, unique, lowercase	Mã/slug template (vd: booking_confirm)
channel	String	enum: email, sms, required	Kênh gửi
subject	String	optional (email), max 150	Tiêu đề email
content	String	required	Nội dung (có placeholder)
variables	[String]	optional	Danh sách biến khả dụng ({{...}})
isActive	Boolean	default true	Trạng thái mẫu
createdBy	ObjectId→User	required	Người tạo
updatedBy	ObjectId→User	optional	Người sửa

Bảng 4.7. Bảng NotificationTemplate

4.2.8. Bảng: NotificationLog

Tên trường	Kiểu dữ liệu	Ràng buộc / Validation	Mô tả & ghi chú
_id	ObjectId	PK	Khóa chính
templateId	ObjectId→NotificationTemplate	optional	Mẫu đã dùng (nếu có)
channel	String	enum: email, sms, required	Kênh đã gửi
to	String	required	Địa chỉ nhận (email/SĐT)

cc / bcc	[String]	optional	Email CC/BCC (email)
subject	String	optional	Tiêu đề (email)
content	String	required	Nội dung thực tế đã gửi
status	String	enum: queued, sent, failed, default queued	Trạng thái gửi
sentAt	Date	optional	Thời điểm gửi
error	String	optional	Thông điệp lỗi (nếu thất bại)
meta	Object	optional	Payload phản hồi từ gateway
recipientUserId	ObjectId→User	optional	Liên kết người nhận (nếu là user)
appointmentId	ObjectId→Appointment	optional	Liên kết lịch hẹn (nếu liên quan)

Bảng 4.8. Bảng NotificationLog

4.3. Thiết kế API Endpoints

4.3.1. Nguyên tắc thiết kế API

Hệ thống được xây dựng trên nền tảng **RESTful API**, cho phép client (ứng dụng React) giao tiếp với server (Node.js/Express) thông qua các yêu cầu HTTP chuẩn như **GET, POST, PUT, PATCH, DELETE**. Toàn bộ endpoint đều trả dữ liệu ở định dạng **JSON**, đảm bảo tính tương thích, gọn nhẹ và dễ xử lý ở phía client.

Các nguyên tắc chính gồm:

- **Stateless:** Mỗi yêu cầu được xử lý độc lập, không lưu trạng thái trên server.
- **Uniform Interface:** Mọi endpoint tuân thủ chuẩn về cấu trúc URL, phương thức và phản hồi.
- **Separation of Concerns:** Tầng API chỉ chịu trách nhiệm xử lý logic và trả dữ liệu, không can thiệp UI.

- **Security by Design:** Mọi thao tác nhạy cảm đều được xác thực và phân quyền qua **JWT Token**.
- **Versioning:** Hỗ trợ mở rộng trong tương lai qua tiền tố /api/v1/ (nếu triển khai nhiều version).

4.3.2. Cấu trúc phân nhóm API

Hệ thống API được tổ chức thành **4 nhóm chính**, tương ứng với các phạm vi truy cập khác nhau:

Nhóm API	Mục đích chính	Đối tượng sử dụng	Đường dẫn gốc
Auth API	Xác thực & quản lý tài khoản	Người dùng & quản trị	/api/auth
Admin API	Quản trị hệ thống, quản lý dữ liệu	Admin	/api/admin
User API	Các thao tác của người dùng đã đăng nhập	User	/api/user
Public API	Truy cập công khai (dịch vụ, danh mục, liên hệ)	Tất cả	/api/public

Bảng 4.9. Cấu trúc phân nhóm API

4.3.3. API Xác thực (Authentication API)

Các API xác thực chịu trách nhiệm đăng ký, đăng nhập, quản lý hồ sơ và bảo mật tài khoản. Dữ liệu phản hồi bao gồm thông tin người dùng và token xác thực.

Phương thức	Endpoint	Mô tả	Xác thực	Trả về
POST	/api/auth/register	Đăng ký tài khoản mới	Không	Thông tin user + token
POST	/api/auth/login	Đăng nhập hệ thống	Không	JWT Token + user info
POST	/api/auth/logout	Đăng xuất	Có	Trạng thái thành công
POST	/api/auth/forgot-password	Gửi yêu cầu đặt lại mật khẩu	Không	Token reset

POST	/api/auth/reset-password	Đặt lại mật khẩu	Không	Trạng thái
GET	/api/auth/me	Lấy thông tin người dùng hiện tại	Có	User profile
PUT	/api/auth/profile	Cập nhật thông tin cá nhân	Có	User cập nhật

Bảng 4.10. Danh sách API xác thực

4.3.4. API Quản trị (Admin API)

Các endpoint nhóm **Admin** hỗ trợ toàn bộ chức năng quản trị: dịch vụ, nhân viên, danh mục, lịch hẹn, người dùng và thông báo.

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Trả về**
GET	/api/admin/dashboard	Lấy tổng quan doanh thu, lịch hẹn, khách hàng	Admin	Số liệu dashboard

Bảng 4.11. API get dashboard

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/admin/services	Danh sách dịch vụ	Admin	Có lọc, phân trang
GET	/api/admin/services/:id	Lấy chi tiết dịch vụ	Admin	–
POST	/api/admin/services	Thêm dịch vụ mới	Admin	Validate dữ liệu
PUT	/api/admin/services/:id	Cập nhật dịch vụ	Admin	–
PATCH	/api/admin/services/:id/toggle	Bật/tắt trạng thái	Admin	–
DELETE	/api/admin/services/:id	Xóa dịch vụ	Admin	Xóa mềm
PUT	/api/admin/services/reorder	Thay đổi thứ	Admin	Dùng cho UI

		tự hiển thị		

Bảng 4.12. API quản lý dịch vụ

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/admin/staff	Danh sách nhân viên	Admin	Có tìm kiếm, lọc
GET	/api/admin/staff/:id	Chi tiết nhân viên	Admin	–
POST	/api/admin/staff	Thêm nhân viên	Admin	Validate role
PUT	/api/admin/staff/:id	Cập nhật thông tin	Admin	–
DELETE	/api/admin/staff/:id	Xóa nhân viên	Admin	Xóa mềm
PATCH	/api/admin/staff/:id/toggle	Bật/tắt trạng thái	Admin	–
GET	/api/admin/staff/:id/stats	Thống kê doanh thu, lịch hẹn	Admin	Cho KPI

Bảng 4.13. API quản lý nhân viên

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/admin/appointments	Danh sách lịch hẹn	Admin	Có lọc ngày/trạng thái
GET	/api/admin/appointments/calendar	Dữ liệu lịch dạng calendar	Admin	Render UI
GET	/api/admin/appointments/:id	Chi tiết lịch hẹn	Admin	–
POST	/api/admin/appointments	Tạo lịch hẹn mới	Admin	Kiểm tra xung đột
PUT	/api/admin/appointments/:id	Cập nhật lịch hẹn	Admin	–
PATCH	/api/admin/appointments/:id/status	Cập nhật trạng thái	Admin	–
PATCH	/api/admin/appointments/:id/reschedule	Đổi lịch	Admin	Kiểm tra

				trùng
PATCH	/api/admin/appointments/:id/assign-staff	Gán nhân viên	Admin	–
DELETE	/api/admin/appointments/:id	Xóa lịch hẹn	Admin	Xóa mềm

Bảng 4.14. API quản lý lịch hẹn

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/admin/categories	Danh sách danh mục	Admin	Có phân cấp
GET	/api/admin/categories/:id	Chi tiết danh mục	Admin	–
POST	/api/admin/categories	Tạo danh mục	Admin	–
PUT	/api/admin/categories/:id	Cập nhật danh mục	Admin	–
PATCH	/api/admin/categories/:id/toggle	Bật/tắt danh mục	Admin	–
DELETE	/api/admin/categories/:id	Xóa danh mục	Admin	Có kiểm tra dịch vụ con

Bảng 4.15. API quản lý danh mục

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/admin/users	Danh sách người dùng	Admin	Có lọc role, trạng thái
GET	/api/admin/users/:id	Chi tiết người dùng	Admin	–
POST	/api/admin/users	Thêm người dùng	Admin	–
PUT	/api/admin/users/:id	Cập nhật người dùng	Admin	–

PATCH	/api/admin/users/:id/tags	Cập nhật tags	Admin	Dán nhãn CRM
PATCH	/api/admin/users/:id/notes	Cập nhật ghi chú	Admin	Ghi chú nội bộ
DELETE	/api/admin/users/:id	Xóa người dùng	Admin	Soft delete

Bảng 4.16. API quản lý người dùng

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/admin/notifications/templates	Danh sách mẫu thông báo	Admin	–
POST	/api/admin/notifications/templates	Thêm mẫu mới	Admin	–
PUT	/api/admin/notifications/templates/:id	Cập nhật mẫu	Admin	–
PATCH	/api/admin/notifications/templates/:id/toggle	Bật/tắt mẫu	Admin	–
DELETE	/api/admin/notifications/templates/:id	Xóa mẫu	Admin	–
GET	/api/admin/notifications/logs	Nhật ký gửi thông báo	Admin	–
POST	/api/admin/notifications/send	Gửi thông báo thử	Admin	Test gửi

Bảng 4.17. API quản lý thông báo

4.3.5. API Người dùng (User API)

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/user/bookings	Danh sách lịch hẹn của user	Có	Có phân trang
GET	/api/user/bookings/:id	Chi tiết lịch hẹn	Có	–
POST	/api/user/bookings	Tạo lịch hẹn	Có	Kiểm tra trùng

PUT	/api/user/bookings/:id	Cập nhật lịch	Có	–
DELETE	/api/user/bookings/:id	Hủy lịch	Có	–
PATCH	/api/user/bookings/:id/status	Cập nhật trạng thái	Có	–
GET	/api/user/profile	Lấy hồ sơ cá nhân	Có	–
PUT	/api/user/profile	Cập nhật hồ sơ	Có	–
POST	/api/user/profile/avatar	Cập nhật ảnh đại diện	Có	Upload
PUT	/api/user/profile/password	Đổi mật khẩu	Có	Validate bcrypt

Bảng 4.18. API của user

4.3.6. API Công khai (Public API)

Phương thức	**Endpoint**	**Mô tả**	**Xác thực**	**Ghi chú**
GET	/api/public/services	Danh sách dịch vụ	Không	Có lọc danh mục
GET	/api/public/services/:id	Chi tiết dịch vụ	Không	–
GET	/api/public/services/search	Tìm kiếm dịch vụ	Không	full-text
GET	/api/public/categories	Danh mục dịch vụ	Không	–
GET	/api/public/categories/:id	Chi tiết danh mục	Không	–
GET	/api/public/categories/tree	Cây danh mục	Không	Hiển thị UI
GET	/api/public/staff	Danh sách nhân viên	Không	Hiển thị profile
GET	/api/public/staff/:id	Chi tiết nhân viên	Không	–
GET	/api/public/staff/available	Lấy nhân viên trống ca	Không	Theo thời gian
POST	/api/public/contact	Gửi liên hệ	Không	Form liên hệ

GET	/api/public/contact/info	Thông tin liên hệ Spa	Không	Dữ liệu tĩnh

Bảng 4.19. API cho khách truy cập

4.3.7. Chuẩn phản hồi API

Để thống nhất phản hồi, hệ thống áp dụng 3 cấu trúc JSON chuẩn:

(1) Thành công (Success Response)

```
{
  "success": true,
  "data": { ... },
  "pagination": {
    "currentPage": 1,
    "totalPages": 5,
    "totalItems": 50,
    "itemsPerPage": 10
  },
  "message": "Thực hiện thành công"
}
```

(2) Lỗi chung (Error Response)

```
{
  "success": false,
  "message": "Đã xảy ra lỗi trong quá trình xử lý",
  "error": "Chi tiết lỗi nội bộ",
  "code": "ERR_SERVER"
}
```

(3) Lỗi kiểm tra dữ liệu (Validation Error)

```
{
  "success": false,
  "message": "Dữ liệu không hợp lệ",
  "errors": [
    { "field": "email", "message": "Email không hợp lệ" },
    { "field": "password", "message": "Mật khẩu quá ngắn" }
  ]
```

}

4.3.8. Cơ chế bảo mật API

- **Xác thực:** Sử dụng **JWT Token** được gắn trong **HTTP-Only Cookie**, đảm bảo an toàn trước XSS.
- **Phân quyền:** Mọi route nhóm /admin chỉ cho phép role = "admin". Các route /user yêu cầu token hợp lệ.
- **Rate limiting:** Giới hạn số request trên mỗi IP nhằm chống brute-force.
- **CORS policy:** Chỉ cho phép origin frontend đáng tin cậy.
- **Input validation:** Toàn bộ request qua middleware express-validator.

4.4. Thiết kế giao diện người dùng

4.4.1. Tổng quan

Giao diện người dùng (UI – *User Interface*) của hệ thống quản lý Spa được thiết kế dựa trên nguyên tắc **Single Page Application (SPA)**, sử dụng **React.js** kết hợp với **Ant Design** và **Tailwind CSS**. Mục tiêu chính là đảm bảo trải nghiệm người dùng (*User Experience – UX*) mượt mà, thống nhất giữa các vai trò (khách hàng, nhân viên và quản trị viên), đồng thời duy trì khả năng mở rộng và dễ bảo trì.

Giao diện được xây dựng theo hướng **component-based**, tức là mỗi phần tử giao diện (form, bảng, modal, thẻ dịch vụ, biểu đồ,…) được thiết kế thành các **component độc lập, tái sử dụng được**, nhằm giảm trùng lặp mã và tăng hiệu suất phát triển.

4.4.2. Nguyên tắc thiết kế UX/UI

Việc thiết kế giao diện tuân theo các nguyên tắc cơ bản của UX/UI hiện đại, đảm bảo tính dễ dùng, thẩm mỹ và khả năng truy cập.

Nguyên tắc	Mô tả	Ứng dụng trong hệ thống
Tính nhất quán (Consistency)	Duy trì phong cách, màu sắc và cấu trúc điều hướng thống nhất	Giao diện admin, khách hàng và công khai đều sử dụng Ant Design và bảng màu thương hiệu Spa
Phản hồi người dùng (Feedback)	Cung cấp thông báo hoặc trạng thái rõ ràng sau mỗi hành động	Các form hiển thị toast "Thành công", "Thất bại" khi thao tác CRUD
Tối giản (Minimalism)	Loại bỏ yếu tố dư thừa, tập trung nội dung chính	Các biểu mẫu đăng ký, đặt lịch được thiết kế ngắn gọn, dễ hiểu
Khả năng truy cập	Màu sắc, kích thước chữ	Áp dụng font dễ đọc, tương phản

(Accessibility)	và độ tương phản phù hợp với người dùng phổ thông	cao, hỗ trợ điều hướng bàn phím
Tính phản hồi (Responsive Design)	Giao diện tự động điều chỉnh kích thước theo thiết bị	Tối ưu hiển thị cho màn hình di động, máy tính bảng và desktop
Trực quan (Visual Hierarchy)	Ưu tiên thông tin quan trọng bằng kích cỡ và màu sắc	Sử dụng card, bảng và biểu đồ có tiêu đề nổi bật trong Dashboard

Bảng 4.20. Nguyên tắc thiết kế UX/UI

4.4.3. Hệ thống thiết kế (Design System)

Giao diện sử dụng **Ant Design** làm framework UI chính, kết hợp **Tailwind CSS** để tùy biến chi tiết. Một số yếu tố cốt lõi trong hệ thống thiết kế gồm:

- **Bảng màu chủ đạo:**
  - Màu thương hiệu chính: #3B82F6 (xanh dương nhẹ, biểu tượng sự tin cậy và thư giãn)
  - Màu phụ trợ: #F9FAFB (xám sáng – nền trung tính)
  - Màu cảnh báo: #F97316 (cam) và #DC2626 (đỏ lỗi)
  - Màu thành công: #16A34A (xanh lá – hoàn tất)
- **Typography (Kiểu chữ):**
  - Font: “Inter”, “Roboto”, sans-serif
  - Heading: đậm (bold), body: trung bình (medium)
  - Kích thước: từ text-xs đến text-2xl theo hierarchy
- **Spacing (Khoảng cách):**
  - Sử dụng lưới 8px; padding và margin đồng nhất trên toàn hệ thống.
- **Components chuẩn hóa:**
  - Button (primary, secondary, ghost)
  - Input, Select, DatePicker, Modal, Table, Card
  - Notification (thông báo toàn cục)

4.4.4. Kiến trúc giao diện (UI Architecture)

4.4.4.1. Cấu trúc layout tổng thể

Hệ thống được chia thành 3 loại layout chính:

Loại	Mô tả	Thành phần chính

layout		
Public Layout	Dành cho khách truy cập (trang chủ, dịch vụ, liên hệ)	Header, Banner, ServiceList, Footer
User Layout	Dành cho người dùng đăng nhập (đặt lịch, hồ sơ, lịch sử)	Navbar, Sidebar, ContentArea
Admin Layout	Dành cho quản trị viên (dashboard, quản lý dữ liệu)	Sidebar quản trị, Header thống kê, Table View, Form Modal

Bảng 4.21. Cấu trúc layout tổng thể

3.4.4.2. Cấu trúc thư mục component

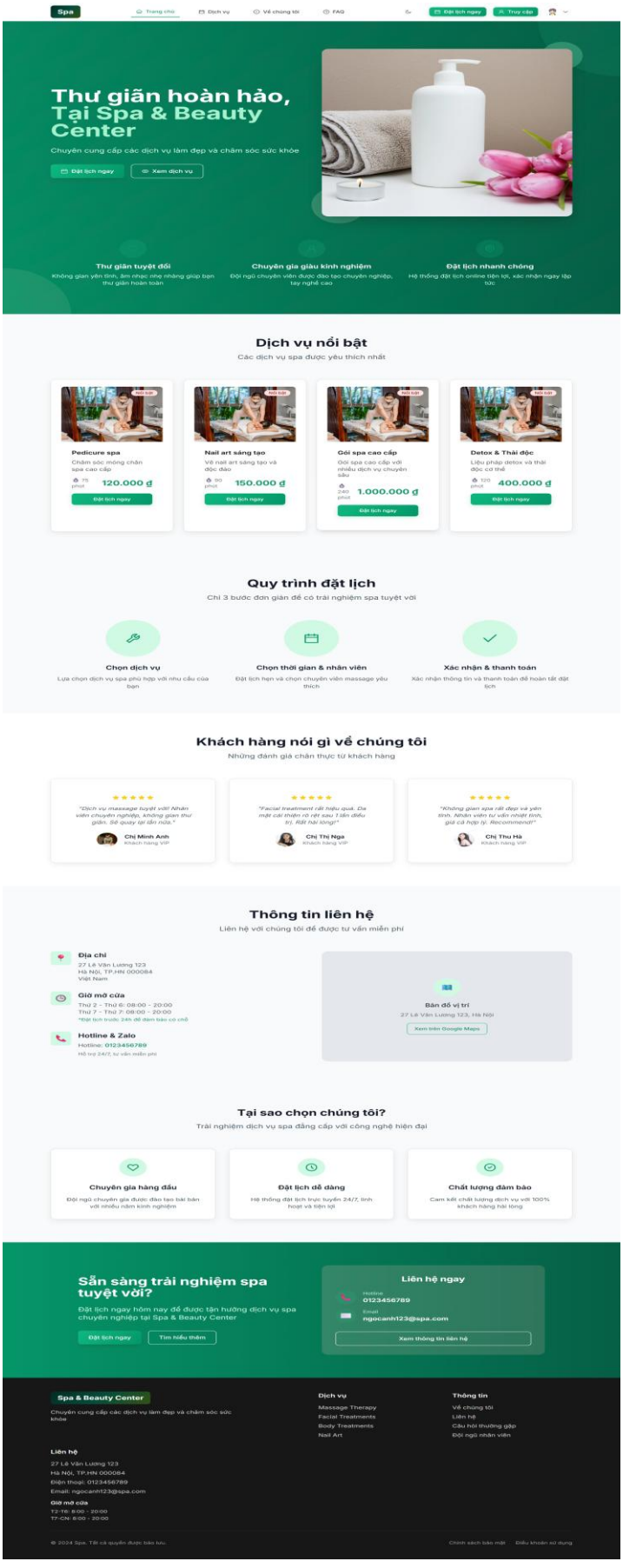
```
/src/components/
├── layout/
│   ├── PublicLayout.jsx
│   ├── AdminLayout.jsx
│   ├── AuthLayout.jsx
│   └── HeaderFooter/
│       ├── Header.jsx
│       ├── Footer.jsx
│       └── Sidebar.jsx
├── ui/
│   ├── Button.jsx
│   ├── Input.jsx
│   ├── Table.jsx
│   ├── Modal.jsx
│   ├── Card.jsx
│   ├── FormControl.jsx
│   └── Spinner.jsx
└── features/
    ├── Booking/
    ├── Profile/
    ├── Dashboard/
    ├── Services/
    └── Staff/
```

4.4.5. Các màn hình chức năng chính

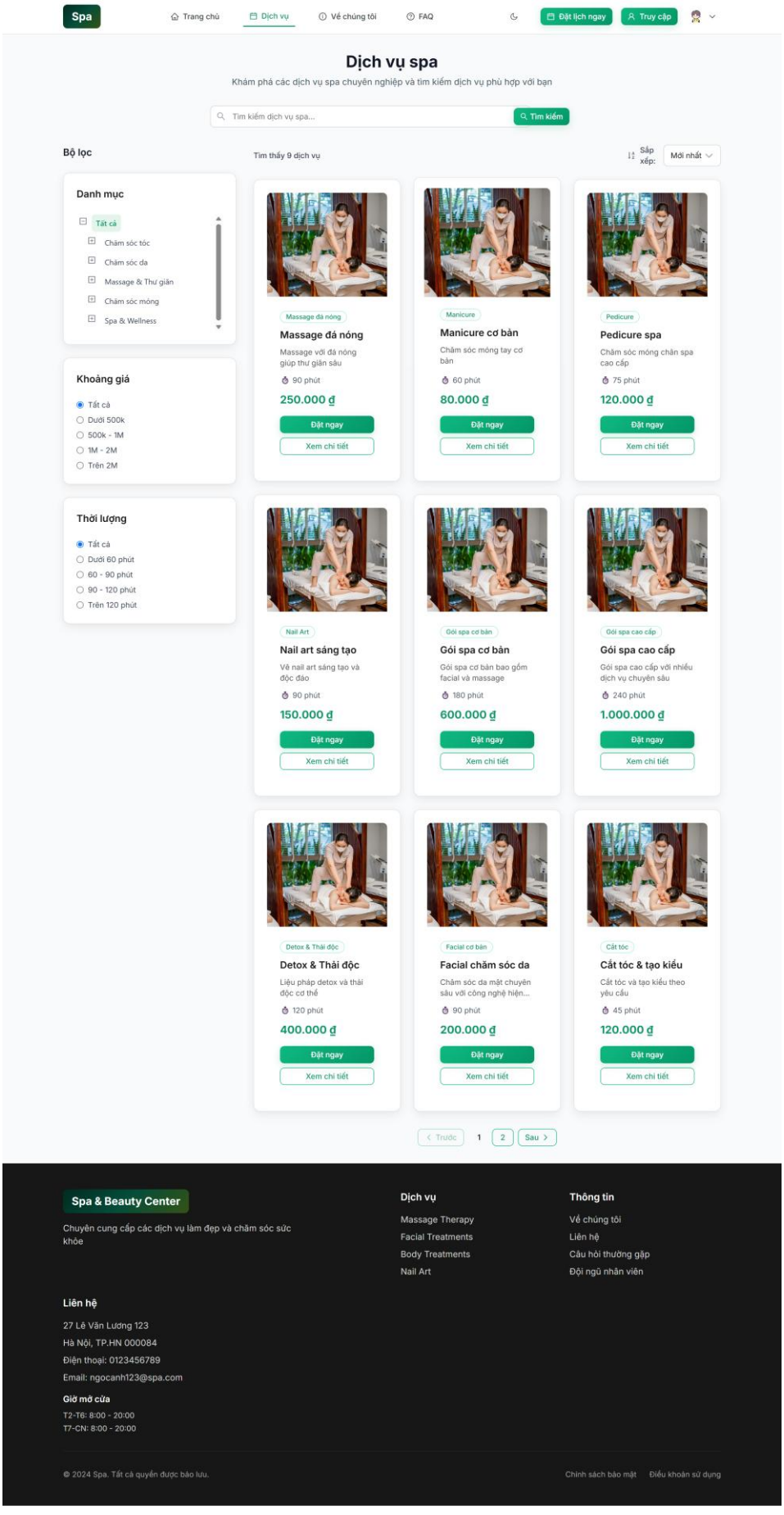
4.4.5.1. Giao diện khách hàng (Public Pages)

Trang	Chức năng	Thành phần chính
Trang chủ (Home)	Hiển thị giới thiệu Spa, dịch vụ nổi bật, đánh giá khách hàng	Banner, ServiceCarousel, Testimonials, Footer
Dịch vụ (Services)	Danh sách dịch vụ, bộ lọc theo danh mục, tìm kiếm	FilterBar, ServiceGrid, Pagination
Chi tiết dịch vụ (Service Detail)	Hiển thị thông tin dịch vụ, thời lượng, giá, hình ảnh, form đặt lịch	ServiceInfoCard, BookingForm
Liên hệ (Contact)	Gửi thông tin phản hồi, hiển thị địa chỉ, bản đồ	ContactForm, MapEmbed

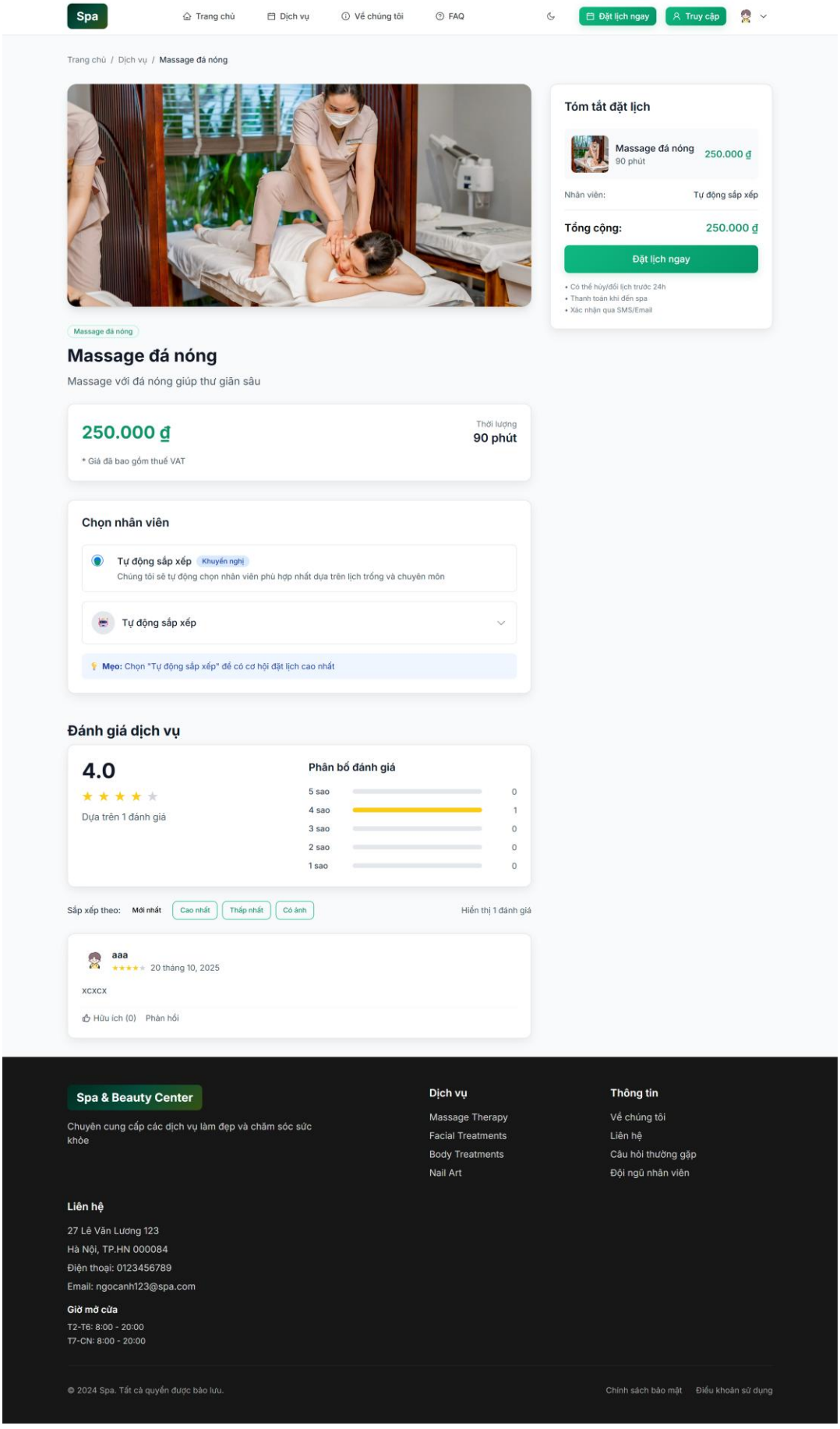
Bảng 4.22. Giao diện khách hàng (Public Pages)



Hình 4.4. Trang chủ



Hình 4.5. Trang Dịch vụ

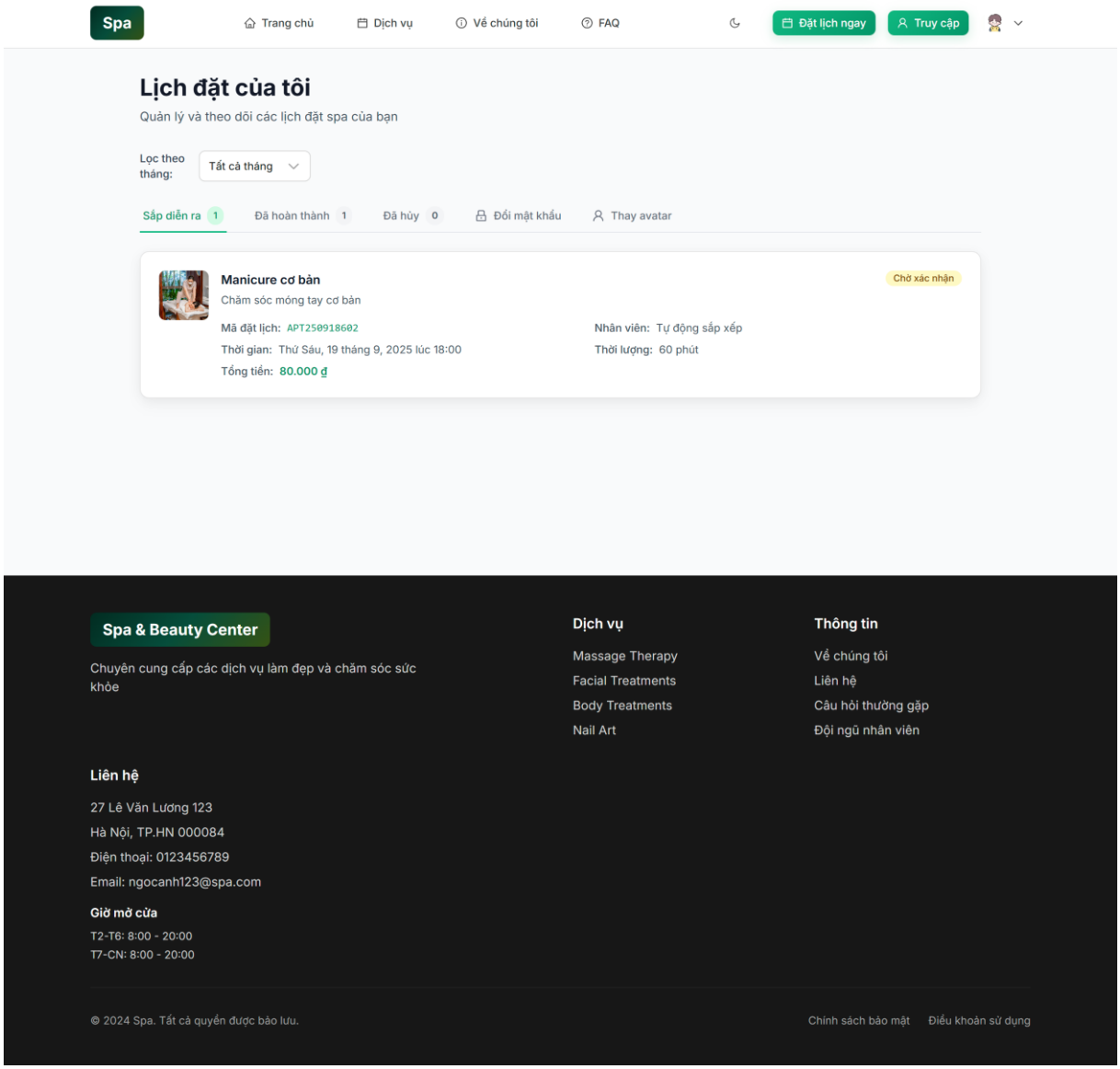


Hình 4.6. Trang Chi Tiết dịch vụ

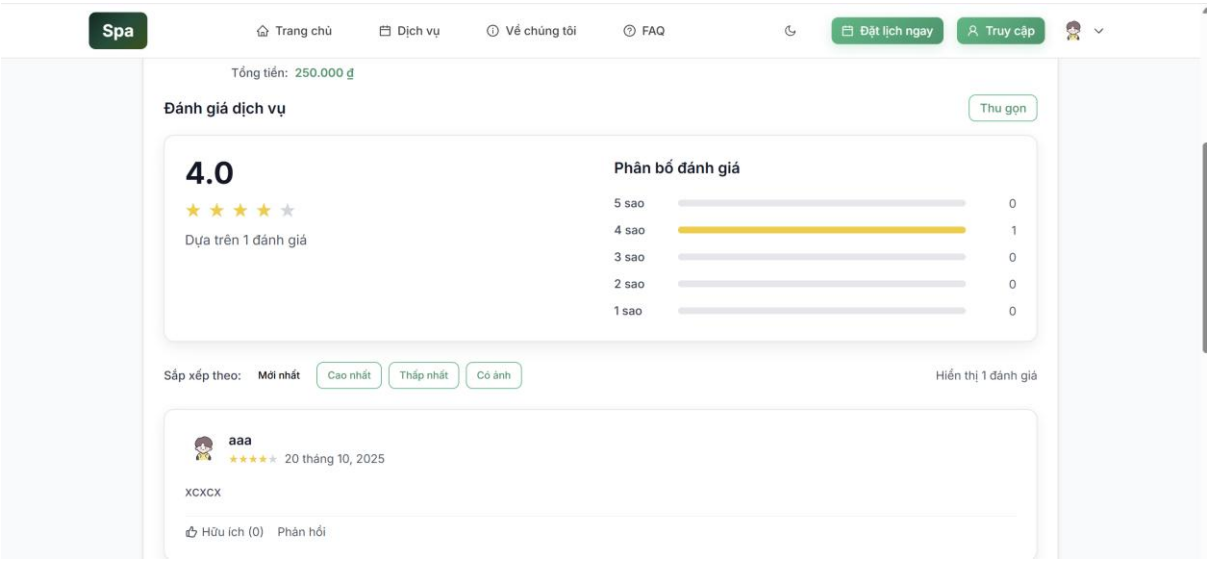
4.4.5.2. Giao diện người dùng (User Pages)

Trang	Chức năng	Thành phần chính
Đặt lịch (My Bookings)	Quản lý các lịch hẹn đã đặt, hủy, đang chờ	BookingList, BookingStatusTabs
Hồ sơ cá nhân (Profile)	Xem, cập nhật thông tin cá nhân, đổi mật khẩu, ảnh đại diện	ProfileCard, EditProfileForm
Đánh giá dịch vụ (Rating)	Gửi đánh giá, nhận xét dịch vụ đã sử dụng	RatingModal, ReviewHistory

Bảng 4.23. Giao diện người dùng (User Pages)



Hình 4.7. Trang My Bookings bao gồm cả Hồ sơ cá nhân

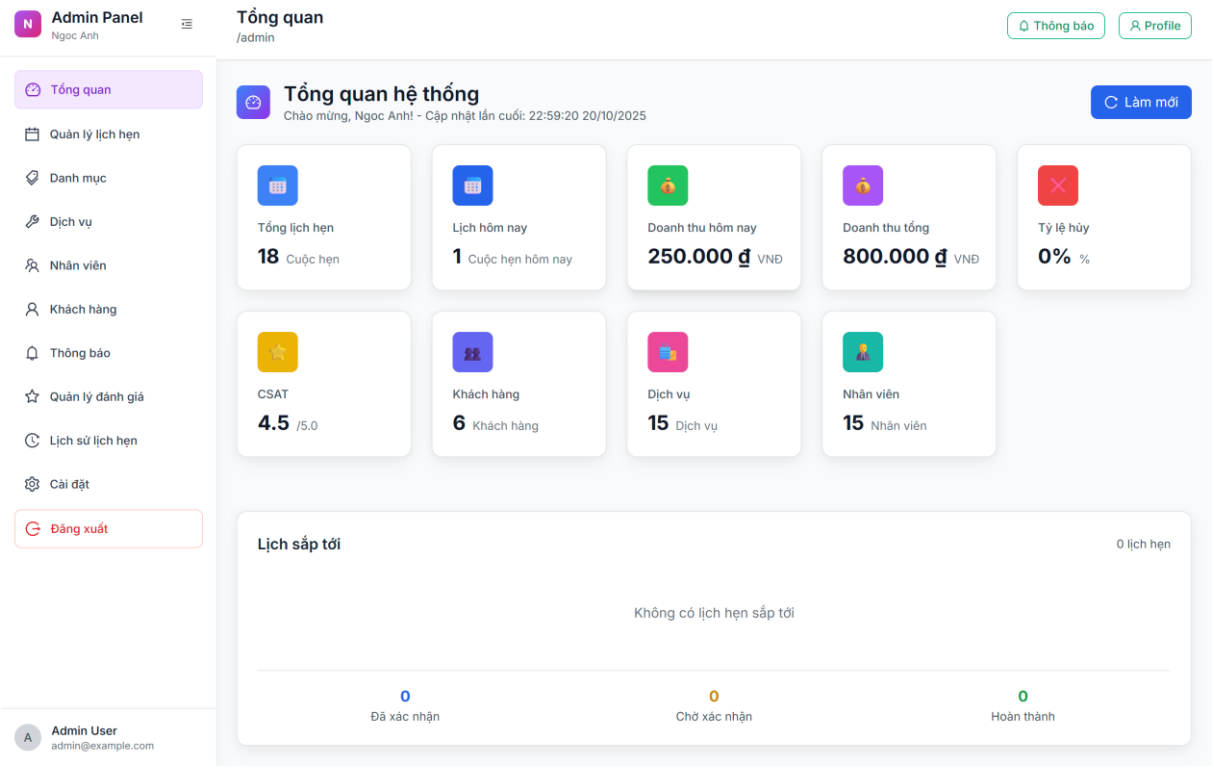


Hình 4.8. Trang đánh giá dịch vụ

4.4.5.3. Giao diện quản trị (Admin Pages)

Trang	Mô tả	Thành phần chính
Dashboard	Thống kê doanh thu, số lượt hẹn, dịch vụ phổ biến	KPI_Cards, RevenueChart, AppointmentTable
Quản lý dịch vụ (Services)	CRUD dịch vụ và danh mục	ServiceTable, ServiceFormModal
Quản lý nhân viên (Staff)	Danh sách, phân công ca làm	StaffTable, ScheduleModal
Quản lý lịch hẹn (Appointments)	Lịch hiển thị dạng bảng và lịch	AppointmentCalendar, AppointmentForm
Quản lý người dùng (Users)	Xem danh sách, thông tin khách hàng	UserTable, UserDetailModal
Cấu hình thông báo (Notifications)	Mẫu email, SMS tự động	TemplateForm, LogViewer

Bảng 4.24. Giao diện quản trị (Admin Pages)



Hình 4.9. Trang admin dashbroad

4.4.6. Nguyên tắc thiết kế Responsive

Để giao diện hoạt động tốt trên mọi thiết bị, hệ thống sử dụng phương pháp **mobile-first**, tức là thiết kế ưu tiên cho màn hình nhỏ trước, sau đó mở rộng cho tablet và desktop.

Kích thước màn hình	**Phân loại**	**Breakpoint (TailwindCSS)**	**Ứng dụng**
< 480px	Điện thoại nhỏ	xs	Ẩn sidebar, hiển thị menu dạng icon
576–768px	Điện thoại lớn	sm	Layout dọc, tối ưu font-size
768–992px	Tablet	md	Chuyển sidebar sang dạng thu gọn
992–1200px	Laptop	lg	Layout đầy đủ, hai cột
>1200px	Màn hình lớn	xl	Dashboard hiển thị biểu đồ và bảng cùng lúc

Bảng 4.25. Nguyên tắc thiết kế Responsive

4.4.7. Cấu trúc trạng thái (State Management)

Hệ thống quản lý trạng thái theo mô hình kết hợp giữa **Local State (useState)** và **Global Context (useContext)**.

- **Local State:** Dùng cho các biến cục bộ như form input, modal visibility, pagination.
- **Global Context:** Dùng cho trạng thái xác thực (AuthContext), giỏ đặt lịch (BookingContext), cấu hình hệ thống (ConfigContext).

4.4.8. Trải nghiệm người dùng (UX Flow)

Luồng đặt lịch (Booking Flow)

1. Khách hàng chọn dịch vụ →
2. Chọn nhân viên (tùy chọn) →
3. Chọn ngày/giờ →
4. Hệ thống kiểm tra xung đột →
5. Nhập thông tin cá nhân →
6. Xác nhận và tạo lịch →
7. Nhận email thông báo tự động.

Luồng này được thiết kế theo nguyên tắc "**3 bước – không vượt quá 1 phút thao tác**" để giảm tỷ lệ bỏ dở khi đặt lịch.

Luồng quản trị viên

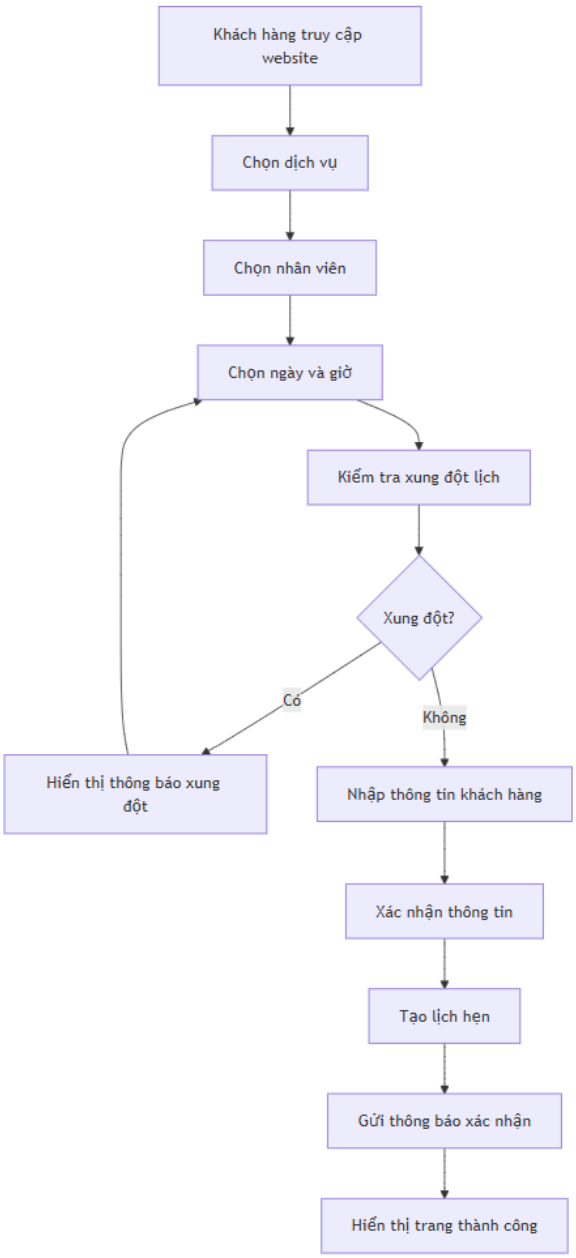
- Quản trị viên đăng nhập → truy cập Dashboard → theo dõi KPI → chuyển đến từng module quản lý (dịch vụ, nhân viên, lịch hẹn).
- Mọi thao tác đều có phản hồi thời gian thực (toast notification) để tăng tính tương tác.

4.5. Sơ đồ luồng nghiệp vụ

4.5.1. Luồng đặt lịch hẹn

Luồng đặt lịch hẹn mô tả trải nghiệm tiêu chuẩn của khách hàng khi tương tác với hệ thống. Quy trình bắt đầu từ việc người dùng truy cập giao diện công khai, chọn dịch vụ mong muốn và, nếu cần, lựa chọn nhân viên ưu tiên. Sau đó, khách hàng lựa chọn ngày và giờ tương thích với lịch hoạt động. Ở bước xác nhận thời gian, hệ thống tiến hành kiểm tra xung đột dựa trên dữ liệu lịch hẹn đã được xác nhận của nhân viên tương ứng; nếu phát hiện trùng, người dùng được hướng dẫn điều chỉnh lại thời gian. Khi thời gian hợp lệ, khách hàng cung cấp thông tin cần thiết, xác nhận nội dung đặt

chỗ và hệ thống khởi tạo lịch hẹn, đồng thời phát hành thông báo xác nhận qua email hoặc SMS. Quy trình kết thúc bằng trang xác nhận thành công, giúp khách hàng nắm được mã lịch hẹn và thông tin chi tiết.

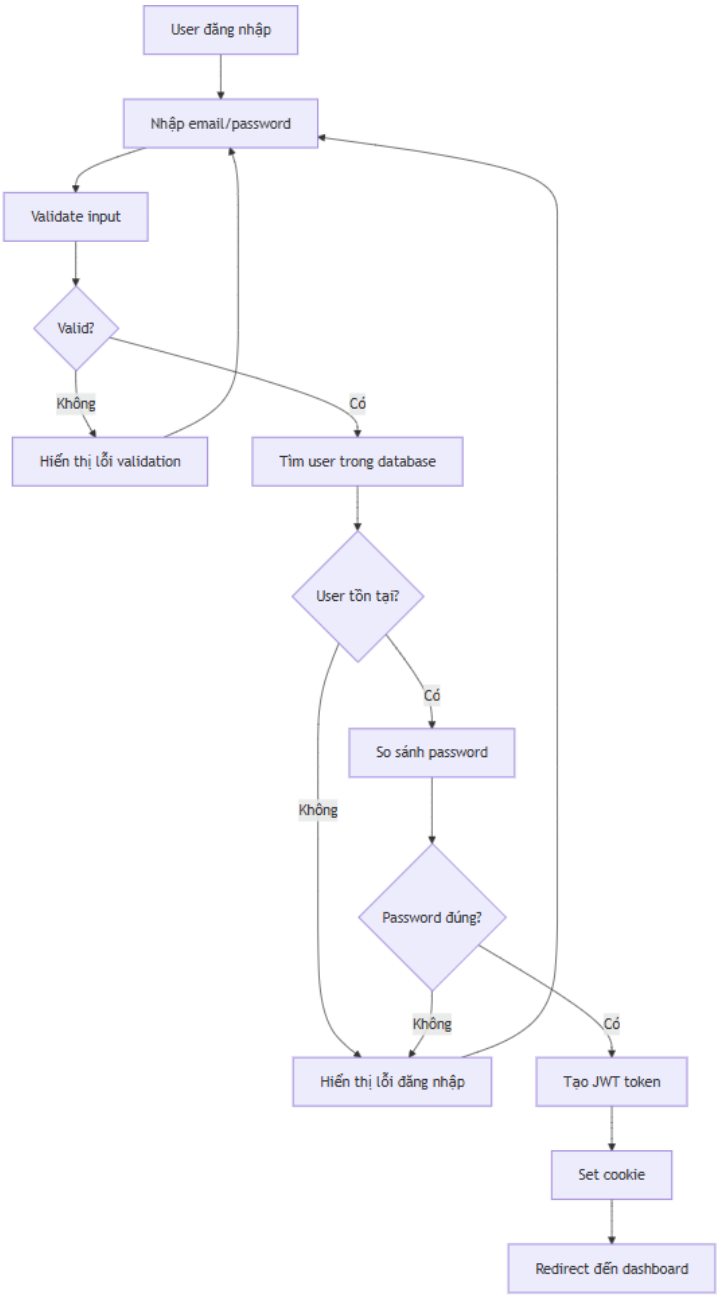


Hình 4.10. Luồng đặt lịch hẹn

4.5.2. Luồng xác thực người dùng

Luồng xác thực đảm bảo chỉ người dùng hợp lệ được phép truy cập các tính năng riêng tư. Người dùng cung cấp email và mật khẩu, hệ thống kiểm tra tính hợp lệ đầu vào, truy vấn bản ghi người dùng, so sánh mật khẩu sau khi băm (**bcrypt**) và, nếu thành công, phát hành **JWT** lưu trong **HTTP-only cookie** để giảm rủi ro XSS. Bất kỳ sai lệch nào trong dữ liệu hoặc thông tin xác thực đều được phản hồi tức thời bằng các thông báo thân thiện, nhằm giảm thiểu lỗi thao tác và hỗ trợ người dùng hoàn thành

đăng nhập thuận lợi. Kết thúc, người dùng được điều hướng đến dashboard phù hợp với vai trò.

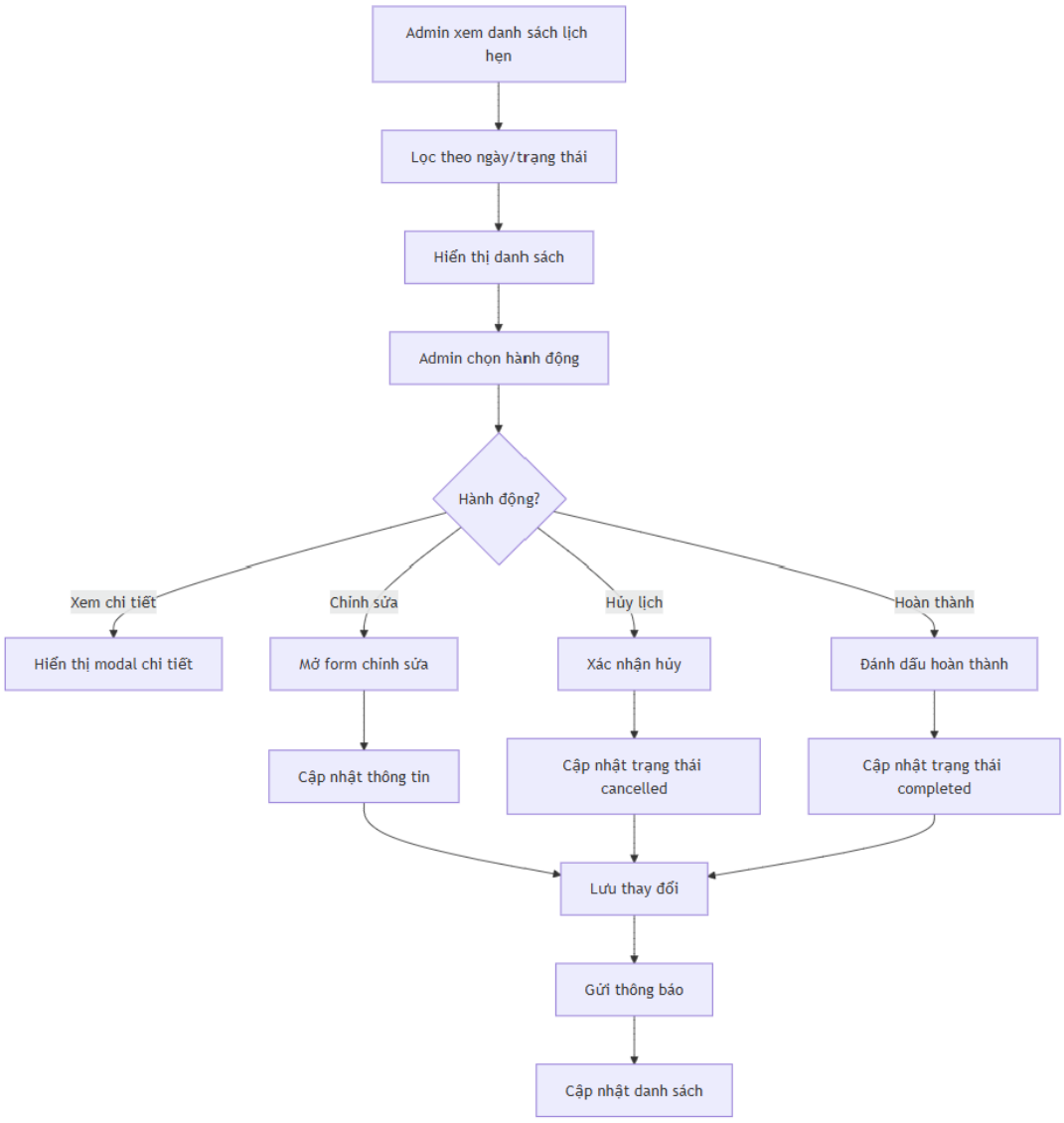


Hình 4.11. Luồng xác thực người dùng

4.5.3. Luồng quản lý lịch hẹn

Luồng quản trị lịch hẹn phản ánh quy trình vận hành của bộ phận điều phối tại Spa. Quản trị viên truy cập danh sách, sử dụng các tiêu chí lọc theo ngày, trạng thái hoặc nhân viên để xác định các phiên hẹn trọng tâm. Từ danh sách, quản trị viên có thể xem chi tiết, chỉnh sửa, hủy hoặc đánh dấu hoàn thành. Mọi thay đổi đều được hệ thống kiểm tra ràng buộc (xung đột thời gian, trạng thái hợp lệ), lưu lại nhật ký và phát hành

thông báo thích hợp đến khách hàng. Danh sách hiển thị được làm tươi ngay sau khi thao tác hoàn tất để đảm bảo tính nhất quán.

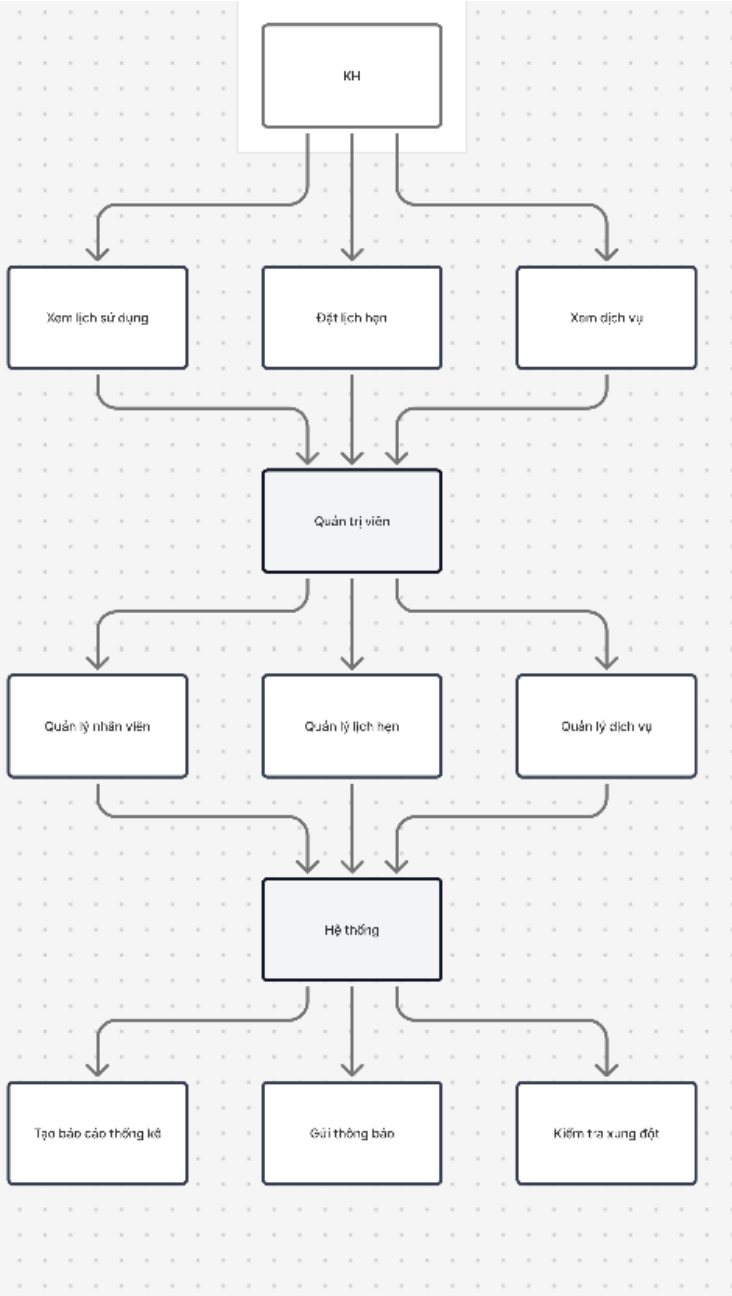


Hình 4.12. Luồng quản lý lịch hẹn

4.6. Sơ đồ use case

4.6.1. Use Case Diagram – Tổng quan

Sơ đồ use case tổng quan thể hiện mối quan hệ giữa các tác nhân chính (khách hàng, quản trị viên) và các chức năng cốt lõi của hệ thống. Khách hàng có khả năng duyệt thông tin dịch vụ, đặt lịch và theo dõi lịch sử sử dụng; quản trị viên chịu trách nhiệm điều hành dịch vụ, lịch hẹn và nhân sự. Bên cạnh đó, các tiến trình tự động của hệ thống như gửi thông báo, kiểm tra xung đột và tổng hợp báo cáo vận hành diễn ra ở tầng nền, đảm bảo sự thông suốt của hoạt động.



Hình 4.13. Use Case Diagram – Tổng quan

4.6.2. Use Case – Đặt lịch hẹn

Use case "Đặt lịch hẹn" cho phép khách hàng tạo phiên hẹn với dịch vụ và nhân viên mong muốn. Trình tự bắt đầu từ việc lựa chọn dịch vụ, tùy chọn chọn nhân viên, xác định thời gian, kiểm tra xung đột, nhập thông tin cá nhân và xác nhận. Hệ thống chỉ khởi tạo lịch hẹn khi dữ liệu hợp lệ và thời gian không trùng với bất kỳ phiên hẹn đã xác nhận của nhân viên. Sau khi tạo, khách hàng nhận thông báo xác nhận kèm chi tiết, đồng thời có thể theo dõi lịch sử trên giao diện người dùng.

Actor: Khách hàng

Precondition: Khách hàng có thể đang đăng nhập hoặc thao tác ở chế độ khách; các dịch vụ/nhân viên được chọn đang hoạt động.

Main flow: (1) Chọn dịch vụ → (2) Chọn nhân viên (tùy chọn) → (3) Chọn ngày/giờ → (4) Kiểm tra xung đột → (5) Nhập thông tin cá nhân → (6) Xác nhận → (7) Tạo lịch hẹn → (8) Gửi thông báo xác nhận → (9) Hiển thị thành công.

Alternative: Nếu phát hiện xung đột ở bước (4), hệ thống yêu cầu người dùng chọn lại thời gian; nếu dữ liệu không hợp lệ ở bước (6), các lỗi được hiển thị theo trường.

Postcondition: Lịch hẹn được tạo và lưu bền vững, trạng thái khởi tạo hợp lệ (ví dụ pending/confirmed theo chính sách), thông báo đã phát hành, giao diện phản ánh dữ liệu mới.

4.6.3. Use Case – Quản lý dịch vụ

Use case "Quản lý dịch vụ" giúp quản trị viên thêm mới, chỉnh sửa hoặc xóa dịch vụ, đảm bảo dữ liệu tuân thủ ràng buộc nghiệp vụ. Khi thêm hoặc sửa, hệ thống xác thực các trường bắt buộc, lưu thay đổi kèm audit log và cập nhật lập chỉ mục tìm kiếm. Khi xóa, hệ thống ưu tiên xóa mềm để bảo toàn lịch sử; xóa cứng chỉ được phép nếu không tồn tại phụ thuộc (ví dụ lịch hẹn liên quan).

Actor: Quản trị viên

Precondition: Đăng nhập với vai trò admin; danh mục liên quan đang hoạt động.

Main flow: Truy cập trang quản lý → xem danh sách → chọn hành động (thêm/sửa/xóa) → nhập dữ liệu → hệ thống validate → lưu thay đổi → giao diện cập nhật danh sách.

Alternative: Nếu dữ liệu không hợp lệ, hệ thống hiển thị lỗi chi tiết theo trường; nếu lỗi lưu dữ liệu, hệ thống giữ nguyên trạng thái cũ và thông báo nguyên nhân.

Postcondition: Bản ghi dịch vụ đã được thêm/cập nhật/xóa mềm một cách nhất quán, có log đầy đủ, hiển thị chính xác ở giao diện admin và public tùy theo trạng thái isActive.

4.7. Kết luận chương

Chương 4 đã hoàn thiện bức tranh thiết kế tổng thể của hệ thống quản lý Spa trực tuyến. Về kiến trúc, hệ thống tuân theo mô hình **Client–Server 3-tier** với **React** ở tầng trình bày, **Node.js/Express** ở tầng nghiệp vụ và **MongoDB** ở tầng dữ liệu, bảo đảm tách biệt rõ ràng các mối quan tâm và thuận tiện cho việc mở rộng. Về dữ liệu, thiết kế cơ sở dữ liệu theo hướng **document-oriented** kết hợp **embed/reference** giúp

tối ưu truy vấn nghiệp vụ đồng thời bảo toàn lịch sử giao dịch; chiến lược **indexing** bám sát mẫu truy vấn thực tế hỗ trợ hiệu năng ở quy mô vận hành. Về tích hợp, thiết kế **RESTful API** nhất quán với chuẩn phản hồi thống nhất và phân nhóm rõ ràng cho phép frontend tích hợp dễ dàng, đảm bảo khả năng bảo trì. Ở lớp giao diện, kiến trúc **component-based** với **Ant Design + Tailwind CSS** mang lại trải nghiệm trực quan, nhất quán và đáp ứng (responsive) trên nhiều thiết bị.

Các sơ đồ luồng nghiệp vụ và use case được trình bày đã mô hình hóa đầy đủ các tình huống sử dụng quan trọng như đặt lịch hẹn, xác thực người dùng và quản lý dịch vụ/lịch hẹn. Điều này đảm bảo chuỗi thao tác của cả khách hàng và quản trị viên đều rõ ràng, có kiểm soát và có thể kiểm thử. Bên cạnh đó, các quyết định thiết kế liên quan đến **bảo mật**, **nhật ký thao tác**, **kiểm tra dữ liệu** và **thông báo tự động** đã được đan cài xuyên suốt các luồng, góp phần bảo đảm tính an toàn, minh bạch và tin cậy của hệ thống.

Tổng thể, chương này đặt nền tảng vững chắc để chuyển sang giai đoạn **phân tích và kiểm thử** trong các chương tiếp theo, nơi từng module sẽ được xác nhận bằng các tiêu chí chất lượng, bộ ca kiểm thử và chỉ số hiệu năng, từ đó bảo đảm hệ thống sẵn sàng triển khai trong môi trường thực tế.

KẾT LUẬN

1. Kết luận về toàn bộ nghiên cứu của đồ án

Đề tài "Trang web quản lý dịch vụ Spa tích hợp đặt lịch làm đẹp trực tuyến" đã hoàn thiện từ khâu phân tích, thiết kế đến xây dựng và kiểm thử hệ thống. Hệ thống đáp ứng tốt nhu cầu quản lý khách hàng, nhân viên, dịch vụ và lịch hẹn, góp phần nâng cao hiệu quả vận hành cho các cơ sở Spa.

2. Các đề nghị rút ra từ kết quả nghiên cứu

Kết quả thực nghiệm cho thấy việc ứng dụng công nghệ web hiện đại giúp tối ưu quy trình quản lý. Tuy nhiên, hệ thống vẫn cần được mở rộng thêm các tính năng như thanh toán trực tuyến, quản lý đa chi nhánh và tối ưu bảo mật để phục vụ thực tế tốt hơn.

3. Hướng phát triển của đề tài

Trong tương lai, hệ thống sẽ được phát triển thêm ứng dụng di động, tích hợp AI gợi ý dịch vụ, chatbot hỗ trợ khách hàng, và kết nối với phần mềm CRM nhằm hoàn thiện giải pháp quản lý Spa toàn diện và hiện đại hơn.

TÀI LIỆU THAM KHẢO

[1] W3Schools – *JavaScript, Node.js, React, Express, MongoDB Tutorials* – https://www.w3schools.com

[2] Mozilla Developer Network (MDN) – *Web Technologies Documentation* https://developer.mozilla.org

[3] Express.js Official Documentation – https://expressjs.com

[4] ReactJS Official Documentation – https://react.dev

[5] MongoDB Manual – https://www.mongodb.com/docs/manual

[6] Tailwind CSS Documentation – https://tailwindcss.com/docs

[7] Ant Design Documentation – https://ant.design/docs/react/introduce

[8] JWT.io – *Introduction to JSON Web Tokens* – https://jwt.io/introduction

[9] bcryptjs GitHub Repository – https://github.com/dcodeIO/bcrypt.js