

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

**Hệ thống cộng đồng chia sẻ bài dịch văn bản tiếng
Nhật**

Đỗ Quang Nam
nam.dq176828@sis.hust.edu.vn

Ngành Công nghệ thông tin

Giảng viên hướng dẫn: TS.Đỗ Quốc Huy

Chữ kí GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 06/2022

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn sâu sắc đến thầy giáo TS.Đỗ Quốc Huy đã trực tiếp hướng dẫn và tạo mọi điều kiện thuận lợi cho em trong quá trình thực hiện đồ án tốt nghiệp.

Em cũng xin gửi lời cảm ơn tới các thầy cô trong trường đại học Bách Khoa Hà Nội, đặc biệt là các thầy cô thuộc Viện công nghệ thông tin và truyền thông đã truyền đạt cho em những kiến thức và kinh nghiệm làm nền tảng cho việc thực hiện đồ án tốt nghiệp cũng như trong quá trình công tác sau này.

Cuối cùng, em xin cảm ơn đến gia đình, bạn bè đã giúp đỡ, động viên và đóng góp ý kiến để em có thể hoàn thành đồ án tốt nghiệp này. Em xin chân thành cảm ơn!

TÓM TẮT NỘI DUNG ĐỒ ÁN

Thời gian gần đây, số lượng người học tiếng Nhật tại Việt Nam đang tăng lên hàng năm, đặc biệt theo kết quả khảo sát của Trung tâm giao lưu văn hóa Nhật Bản tại Việt Nam, trong 3 năm từ 2015-2018 số lượng người học tăng khoảng 110.000 người, tăng nhiều nhất trên thế giới. Chỉ sau 20 năm số cơ sở đào tạo tiếng Nhật tăng hơn 26 lần và số người học tăng lên hơn 17 lần [1][2]. Vì vậy cùng với nhu cầu học và ôn thi tiếng Nhật, nhu cầu tìm kiếm và chia sẻ tài liệu tiếng Nhật ngày một tăng lên. Trong số đó không thể thiếu là các tài liệu dịch qua từ tiếng Nhật qua tiếng Việt ví dụ như: tài liệu dịch các bài báo Nhật, tài liệu dịch các bài đọc và sách ôn thi,... Các tài liệu này đóng góp phần lớn vào việc làm giảm sự khó khăn trong việc học một ngôn ngữ mới, giúp định hướng tư duy người học và làm giảm đa số thời gian khi bắt đầu quá trình học và ôn thi.

Tuy nhiên, là một sinh viên chuyên ngành Việt Nhật, trong quá trình học và ôn thi chứng chỉ tiếng Nhật, bản thân em hiểu rõ và sâu sắc khó khăn trong việc thiếu tài liệu dịch và việc bị “lạc” trong tràn lan các tài liệu dịch khác trên internet, không biết được đâu là tài liệu chính xác. Hơn nữa, đa phần các tài liệu dịch hiện nay đều do các cá nhân tự đăng trên các mạng xã hội như Facebook, Zalo,... hay rải rác trên các website nên việc tìm tài liệu dịch đối với một người mới học sẽ trở nên cực kỳ khó khăn vì không quen thuộc với việc tìm tài liệu. Vì vậy, nhận thực được thực trạng chia sẻ tài liệu rải rác như hiện nay, em mong muốn phát huy kiến thức về CNTT của mình để đưa đề tài Xây dựng hệ thống cộng đồng chia sẻ bài dịch tiếng Nhật vào làm ĐATN và hy vọng sẽ đóng góp được một phần giúp cải thiện được thực tế nêu trên.

Sử dụng hệ thống, người dùng có thể trích dẫn các bài viết, bài báo mà bản thân đang khó khăn trong việc hiểu nội dung lên để nhờ những người dùng khác có kinh nghiệm hơn chia sẻ ý tưởng/ý hiểu của mình, giúp định hình cách đọc hiểu tài liệu cho người hỏi và cải thiện tư duy của những người chia sẻ. Hệ thống được xây dựng với mức độ phân quyền chặt chẽ giúp quản lý các tác nhân và chất lượng bài dịch/bài viết một cách đảm bảo. Hệ thống có tính ứng dụng thực tế cao, mang tính năng đa ngôn ngữ, hướng tới xây dựng một cộng đồng học và chia sẻ chất lượng cho những người đã, đang và sẽ học tiếng Nhật.

Hệ thống được xây dựng dựa trên bộ công cụ lập trình mã nguồn mở dựa trên Javascript là MERN Stack bao gồm ReactJS [3], ExpressJS [4], NodeJS [5] và CSDL MongoDB [6], được đóng gói sử dụng Docker [7] và triển khai trên nền

tảng web với dịch vụ của Digital Ocean [8].

Thông qua quá trình khảo sát thực trạng hiện nay, phân tích các hệ thống đã có, đồ án đưa ra các mục tiêu cần phát triển cho hệ thống, qua đó thực hiện phân tích yêu cầu, thiết kế hệ thống, phát triển và triển khai hệ thống. Đồng thời, đồ án cũng đưa ra các giải pháp và đóng góp đã được xây dựng và triển khai trong quá trình phát triển hệ thống liên quan tới các giải pháp tối ưu trải nghiệm người dùng và tối ưu quy trình phát triển hệ thống. Kết quả của quá trình phân tích, phát triển và triển khai nêu trên là thành công trong việc xây dựng nền hệ thống cộng đồng chia sẻ bài dịch tài liệu tiếng Nhật mang tên SunShare.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu và phạm vi đề tài.....	1
1.3 Định hướng giải pháp.....	2
1.4 Bố cục đồ án	2
CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU.....	5
2.1 Khảo sát hiện trạng	5
2.2 Tổng quan chức năng	6
2.2.1 Biểu đồ use case tổng quan	6
2.2.2 Biểu đồ use case phân rã Xem chi tiết bài dịch.....	9
2.2.3 Biểu đồ use case phân rã: "Quản lý bộ sưu tập"	11
2.2.4 Biểu đồ usecase phân rã “Duyệt bài viết”	12
2.2.5 Quy trình nghiệp vụ	13
2.3 Đặc tả chức năng	15
2.3.1 Đặc tả usecase “Lựa chọn ngôn ngữ hiển thị”	15
2.3.2 Đặc tả usecase “Tạo bài viết mới”.....	16
2.3.3 Đặc tả usecase “Tạo bài dịch mới”	17
2.3.4 Đặc tả usecase “Xem chi tiết bài viết	18
2.3.5 Đặc tả usecase “Xem chi tiết bài dịch.....	19
2.3.6 Đặc tả usecase “Quản trị viên duyệt danh sách bản ghi”	20
2.4 Yêu cầu phi chức năng	22
CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG.....	25
3.1 Giới thiệu về MERN Stack	25
3.1.1 Khái niệm	25

3.1.2 Lý do lựa chọn	25
3.2 Môi trường runtime Javascript: NodeJs	26
3.2.1 Khái niệm	26
3.2.2 Lý do lựa chọn	26
3.3 Framework ExpressJs	27
3.3.1 Khái niệm	27
3.3.2 Lý do lựa chọn	27
3.4 Hệ quản trị cơ sở dữ liệu NoSQL MongoDB	27
3.4.1 Khái niệm	27
3.4.2 Lý do lựa chọn	27
3.5 Thư viện ReactJS	28
3.5.1 Khái niệm	28
3.5.2 Lý do lựa chọn	28
3.6 Docker	29
3.6.1 Khái niệm	29
3.6.2 Lý do lựa chọn	29
3.7 Dịch vụ Cloud Server DigitalOcean	29
3.7.1 Khái niệm	29
3.7.2 Lý do lựa chọn	29
3.8 Cân bằng tải (Load Balancing) với Nginx Server	30
3.8.1 Khái niệm	30
3.8.2 Lý do lựa chọn	30
CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ	31
4.1 Thiết kế kiến trúc	31
4.1.1 Lựa chọn kiến trúc phần mềm	31
4.1.2 Thiết kế tổng quan	32

4.1.3 Thiết kế chi tiết gói	33
4.2 Thiết kế chi tiết.....	34
4.2.1 Thiết kế giao diện	34
4.2.2 Thiết kế lớp	40
4.2.3 Thiết kế cơ sở dữ liệu	44
4.3 Xây dựng ứng dụng.....	46
4.3.1 Thư viện và công cụ sử dụng.....	46
4.3.2 Kết quả đạt được	48
4.3.3 Minh họa các chức năng chính	48
4.4 Kiểm thử.....	53
4.5 Triển khai	57
CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT	61
5.1 Cải thiện trải nghiệm người dùng với thiết kế giao diện responsive	61
5.1.1 Đặt vấn đề	61
5.1.2 Giải pháp	61
5.1.3 Kết quả đạt được	62
5.2 Tối ưu hóa hệ thống với Nginx Server.....	64
5.2.1 Đặt vấn đề	64
5.2.2 Giải pháp	65
5.2.3 Kết quả đạt được	66
5.3 Tối ưu hóa quy trình phát triển và triển khai hệ thống.....	66
5.3.1 Đặt vấn đề	66
5.3.2 Giải pháp	67
5.3.3 Kết quả đạt được	68
5.4 Thiết kế hệ thống đa ngôn ngữ	68
5.4.1 Đặt vấn đề	68

5.4.2 Giải pháp	69
5.4.3 Kết quả đạt được	70
CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	73
6.1 Kết luận	73
6.2 Hướng phát triển.....	74
TÀI LIỆU THAM KHẢO.....	76

DANH MỤC HÌNH VẼ

Hình 2.1	Biểu đồ use case tổng quan	6
Hình 2.2	Biểu đồ phân rã use case Xem chi tiết bài dịch	9
Hình 2.3	Biểu đồ use case phân rã: Quản lý bộ sưu tập	11
Hình 2.4	Biểu đồ use case phân rã: Duyệt bài viết	12
Hình 2.5	Quy trình nghiệp vụ: Xem và quản lý chi tiết bài viết/bài dịch	14
Hình 3.1	Mô hình kiến trúc của MERN Stack	25
Hình 4.1	Kiến trúc phân lớp Layered Pattern	31
Hình 4.2	Cấu trúc phân lớp 3 tầng	32
Hình 4.3	Biểu đồ UML tổng quan hệ thống	33
Hình 4.4	Thiết kế chi tiết gói hệ thống	34
Hình 4.5	Nút mặc định chính	35
Hình 4.6	Nút mặc định phụ	36
Hình 4.7	Nút hủy bỏ	36
Hình 4.8	Nút biểu tượng	36
Hình 4.9	Thông điệp phản hồi khi tác vụ thành công	36
Hình 4.10	Thông điệp phản hồi khi tác vụ thất bại	36
Hình 4.11	Thiết kế giao diện: Giao diện chung	37
Hình 4.12	Thiết kế giao diện: Màn danh sách bài viết	38
Hình 4.13	Thiết kế giao diện: Màn chi tiết bài viết	38
Hình 4.14	Thiết kế giao diện: Màn chi tiết bộ sưu tập	39
Hình 4.15	Thiết kế lớp cho chức năng liên quan tới bài viết	40
Hình 4.16	Biểu đồ trình tự thiết kế lớp: Tạo bài viết	43
Hình 4.17	Biểu đồ trình tự thiết kế lớp: Tạo bài dịch	43
Hình 4.18	Giao diện danh sách bài viết	50
Hình 4.19	Giao diện chi tiết bài viết (1)	50
Hình 4.20	Giao diện chi tiết bài viết (2)	51
Hình 4.21	Giao diện tạo bài viết mới	51
Hình 4.22	Giao diện chỉnh sửa thông tin cá nhân	52
Hình 4.23	Giao diện cập nhật mật khẩu	52
Hình 4.24	Quy trình khởi chạy hệ thống Front-end	58
Hình 4.25	Quy trình khởi chạy hệ thống Back-end	58
Hình 5.1	Giao diện responsive: Trang chủ	63
Hình 5.2	Giao diện responsive: Chi tiết bài viết và Thông tin cá nhân	63

Hình 5.3	Cấu trúc hệ thống Front-end sử dụng React Development Server	64
Hình 5.4	Ý tưởng cấu trúc cân bằng tải hệ thống Back-end	65
Hình 5.5	Cấu trúc hệ thống Front-end sử dụng Nginx Server	65
Hình 5.6	Ý tưởng cấu trúc cân bằng tải hệ thống Back-end với Nginx Server	66
Hình 5.7	Quy trình phát triển và triển khai hệ thống ban đầu	67
Hình 5.8	Quy trình phát triển và triển khai hệ thống đã tối ưu hóa	68
Hình 5.9	Sơ đồ luồng hoạt động xử lý đa ngôn ngữ của FormatJS	69
Hình 5.10	Giao diện đa ngôn ngữ: Tiếng Anh	70
Hình 5.11	Giao diện đa ngôn ngữ: Tiếng Nhật	70

DANH MỤC BẢNG BIỂU

Bảng 2.1	Kết quả khảo sát so sánh hệ thống hiện có	5
Bảng 2.2	Mô tả use case tổng quan với tác nhân người dùng khách	7
Bảng 2.3	Mô tả use case tổng quan với tác nhân người dùnggg	8
Bảng 2.4	Mô tả use case tổng quan với tác nhân quản trị viên hệ thống	8
Bảng 2.5	Mô tả use case phân rã: Xem chi tiết bài viết/bài dịch	10
Bảng 2.6	Mô tả use case phân rã: Quản lý bộ sưu tập	11
Bảng 2.7	Mô tả use case phân rã: Duyệt bài viết	13
Bảng 2.8	Mô tả luồng sự kiện usecase: Lựa chọn ngôn ngữ hiển thị	15
Bảng 2.9	Mô tả luồng sự kiện usecase: Tạo bài viết mới	16
Bảng 2.10	Mô tả luồng sự kiện usecase: Tạo bài dịch mới	17
Bảng 2.11	Mô tả luồng sự kiện usecase: Xem chi tiết bài viết	19
Bảng 2.12	Mô tả luồng sự kiện usecase: Xem chi tiết bài dịch	20
Bảng 2.13	Mô tả luồng sự kiện usecase: Quản trị viên duyệt danh sách bản ghi	21
Bảng 4.1	Đặc tả thiết kế giao diện	35
Bảng 4.2	Đặc tả lớp ArticleModel	40
Bảng 4.3	Đặc tả lớp ArticleController	41
Bảng 4.4	Đặc tả lớp ArticleRepository	42
Bảng 4.5	Thiết kế database:Cấu trúc user model	44
Bảng 4.6	Thiết kế database: Cấu trúc article model	44
Bảng 4.7	Thiết kế database: Cấu trúc category model	45
Bảng 4.8	Thiết kế database: Cấu trúc series model	46
Bảng 4.9	Thiết kế database: Cấu trúc các models còn lại trong database	46
Bảng 4.10	Danh sách công cụ và thư viện sử dụng	47
Bảng 4.11	Thông kê thông tin tổng quan hệ thống	48
Bảng 4.12	Kịch bản kiểm thử hệ thống	53
Bảng 4.13	Kết quả kiểm thử trên trình duyệt máy tính	56
Bảng 5.1	Quy chuẩn thiết kế giao diện responsive	62

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ và từ viết tắt	Ý nghĩa
Asynchronous	Xử lý bất đồng bộ
Build	Tối ưu mã nguồn cho trang web
Container	Môi trường chạy dịch vụ phần mềm của Docker
Development Environment	Môi trường phát triển hệ thống
Event Driven	Kiến trúc hướng sự kiện
Image	Đơn vị đóng gói ứng dụng trong Docker
Load Balancing	Cân bằng tải hệ thống
Port	Cổng tiếp nhận yêu cầu của hệ thống
Production Environment	Môi trường triển khai hệ thống
Responsive Design	Thiết kế web đáp ứng
Server	Máy chủ phát triển hệ thống
Synchronous	Xử lý đồng bộ
Tablet portrait	Giao diện máy tính bảng nằm dọc
Tablet landscape	Giao diện máy tính bảng nằm ngang

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Trong 5 năm trở lại đây, Nhật Bản – đất nước đang trong giai đoạn già hóa dân số đang tích cực đưa ra các chính sách nhằm thu hút nhân lực lao động từ nước ngoài đến hỗ trợ phát triển trong mọi lĩnh vực: CNTT, dệt may, xây dựng, điều dưỡng,... Thực tế hiện nay có thể thấy: nhu cầu học và ôn thi tiếng Nhật tại Việt Nam đang là rất lớn, số lượng người học và số lượng cơ sở đào tạo tiếng Nhật đang ngày một tăng lên. Chỉ sau 20 năm số cơ sở đào tạo tiếng Nhật tăng hơn 26 lần và số lượng người học tăng lên hơn 17 lần. Đây chính là nguồn cơn thổi bùng các mô hình, các hệ thống chia sẻ kiến thức và tài liệu học tiếng Nhật ra đời và phát triển với tốc độ cực kỳ nhanh chóng nhằm đáp ứng nhu cầu của thị trường.[1] [2] [9]

Đối với những người mới học tiếng Nhật, không thể tránh khỏi việc gặp khó khăn trong việc đọc hiểu các tài liệu thuần Nhật. Từ đó phát sinh ra nhu cầu cần tìm kiếm các tài liệu dịch sang tiếng Việt để hỗ trợ người học dễ dàng hiểu và rút gọn thời gian học hơn. Tuy nhiên bản thân em hiểu rõ và sâu sắc khó khăn trong việc thiếu tài liệu dịch và việc bị “lạc” trong tràn lan các tài liệu dịch khác trên internet, không biết được đâu là tài liệu chính xác. Hơn nữa, đa phần các tài liệu dịch hiện nay đều do các cá nhân tự đăng trên các mạng xã hội như Facebook, Zalo,... hay rải rác trên các website nên việc tìm tài liệu dịch đối với một người mới học sẽ trở nên cực kỳ khó khăn vì không quen thuộc với việc tìm tài liệu.

Vì vậy, em lựa chọn nghiên cứu và phát triển đề tài Xây dựng hệ thống cộng đồng chia sẻ tài liệu dịch Tiếng Nhật với mục đích tạo dựng một hệ thống cộng đồng chung hỗ trợ người đang học tiếng Nhật có thể hỏi/đáp và tìm kiếm các giải pháp trong việc hiểu tài liệu tiếng Nhật trong quá trình học. Trong một hệ thống cộng đồng như vậy, việc đảm bảo số lượng và chất lượng các bài viết/bài dịch thiết nghĩ là một vấn đề chính cần được chú trọng và quan tâm xây dựng.

1.2 Mục tiêu và phạm vi đề tài

Hiện nay, đáp ứng xu hướng toàn cầu hóa, với nhu cầu ngày càng lớn của việc học ngôn ngữ mới, các hệ thống cộng đồng chia sẻ tài liệu và kiến thức của những người học ngôn ngữ đang ngày càng xuất hiện nhiều và đa dạng. Người dùng internet có thể chia sẻ kiến thức và tài liệu ở mọi nền tảng bao gồm: mạng xã hội như Facebook (facebook.com), Twitter (twitter.com),... (2) các website blogs và các website cộng đồng như: Viblo (viblo.asia), NihonBlog (nihonblog.com),.... Ngoài ra, các hệ thống dịch tài liệu cũng phát triển mạnh mẽ điển hình như: Google Translate (translate.google.com), DeepL (deepl.com),... Tuy nhiên để đáp ứng các vấn đề đã

nêu ở mục 1.1, các hệ thống trên vẫn còn tồn tại một số vấn đề như (i) không tập trung chuyên về chức năng chia sẻ tài liệu về ngôn ngữ, (ii) nội dung chia sẻ tràn lan và không cụ thể chức năng, (iii) thiếu sự tương tác giữa người với người – yếu tố cần thiết trong việc học ngôn ngữ.

Nhận thức được các vấn đề trên, đồ án đưa ra 2 mục đích chính của hệ thống: (i) là một hệ thống cộng đồng chia sẻ tài liệu và kiến thức, cụ thể là về ngôn ngữ Nhật, (ii) là một hệ thống giúp người dùng có thể tìm kiếm hoặc ngược lại là cung cấp các bài dịch từ tiếng Nhật sang tiếng Việt.

Hệ thống được xây dựng gồm ba đối tượng chính tham gia:

1. Người dùng khách: tác nhân không có tài khoản trong hệ thống, có thể sử dụng một số chức năng chính cơ bản của hệ thống.
2. Người dùng đã đăng ký (Người dùng): tác nhân đã tạo và sử dụng tài khoản hệ thống, có thể tạo yêu cầu bài viết, tạo bài dịch hay quản lý thông tin tài khoản và lưu trữ bài viết/bài dịch theo bộ sưu tập.
3. Quản trị viên: tác nhân phụ trách quản lý các thông tin về bài viết, người dùng, danh mục, ... có trong hệ thống.

1.3 Định hướng giải pháp

Hệ thống được xây dựng theo bộ công cụ lập trình mã nguồn mở liên quan tới Javascript là MERN Stack bao gồm M (MongoDB), E (ExpressJs), R (ReactJs) và N (NodeJs). Hệ thống gồm hai phần chính:

- Backend: là phần xử lý của hệ thống được viết bằng ngôn ngữ NodeJs, sử dụng framework là ExpressJs. ExpressJs là một framework được xây dựng trên nền tảng NodeJs, cung cấp các thành phần định tuyến và phần mềm trung gian (middleware) để hỗ trợ giúp phát triển ứng dụng web một cách dễ dàng hơn.
- Frontend: được phát triển bằng thư viện ReactJs – một thư viện mã nguồn mở Javascript hiệu quả được phát triển bởi Facebook, được sử dụng phổ biến để phát triển xây dựng giao diện người dùng hay UI. ReactJs cung cấp khả năng tái sử dụng code (Reusable Code) giúp tăng tốc quá trình phát triển và giảm thiểu những rủi ro có thể xảy ra trong quá trình phát triển.

1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 mô tả kết quả khảo sát một số trang web cộng đồng chia sẻ tài liệu có cùng ý tưởng đã tồn tại được đánh giá cao và nhiều người sử dụng. Trên cơ sở đó, đồ án đưa ra các chức năng cần thiết và biểu đồ use case để thấy rõ tương tác của

ba tác nhân người dùng khách, người dùng đã đăng ký và quản trị viên đối với hệ thống và đồng thời đưa ra một số điều kiện của hệ thống đáp ứng các yêu cầu phi chức năng.

Chương 3 giới thiệu về các công nghệ được sử dụng để xây dựng hệ thống như: mô hình Client Server, thư viện lập trình, công cụ triển khai, ...và các công nghệ được sử dụng để thiết kế hệ thống. Việc hiểu rõ trước mục đích sử dụng và nội dung của các hệ thống này là tiền đề để đi tới bước tiếp theo ở Chương 4: Thiết kế và triển khai.

Chương 4 cũng là chương quan trọng nhất, đi sâu vào chi tiết thiết kế hệ thống, cách thức hệ thống được triển khai và sự áp dụng các công nghệ đã nêu lên ở Chương 3 vào việc thiết kế hệ thống này.

Chương 5 đóng vai trò tổng kết, trình bày về các đóng góp nổi bật của cả hệ thống. Đó là giải pháp giải quyết vấn đề người dùng trong hệ thống, các vấn đề hệ thống đã giải quyết được và đồng thời cũng đưa ra những tâm đắc của em khi phát triển hệ thống này.

Chương 6, chương kết của báo cáo đồ án, để thông qua đó em nhìn lại kết quả mình đã đạt được, tổng kết kiến thức đã học và các kinh nghiệm rút ra.

Như vậy, trong chương 1 đã thể hiện rõ được các vấn đề xung quanh đề tài, hướng đi của đề tài và bối cảnh đề tài. Tiếp nối định hướng trong chương 1, ở chương 2 đồ án sẽ trình bày về kết quả khảo sát hiện trạng và phân tích chi tiết từng chức năng của hệ thống.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

Ở Chương 1, đồ án đã giới thiệu về mục tiêu và các chức năng chính của hệ thống. Tại Chương 2, đồ án sẽ trình bày kết quả khảo sát người dùng, khảo sát hiện trạng về các hệ thống tương tự trên thị trường và phân tích thiết kế hệ thống.

2.1 Khảo sát hiện trạng

Thực tế, với vai trò là một người đang và học tiếng Nhật, tôi nhận thấy rằng các công cụ hỗ trợ học tiếng Nhật hiện nay vẫn còn rất hạn chế. Đặc biệt là các công cụ và trang web hỗ trợ chia sẻ tài liệu tiếng Nhật. Thông qua khảo sát những người cùng học tiếng Nhật, tôi nhận thấy một thực trạng hiện nay rằng khi gặp một từ vựng hoặc đoạn văn khó trong quá trình học, đa phần mọi người sẽ sử dụng 2 công cụ chính là từ điển online và Google Translate (công cụ dịch văn bản của Google).

Ngoài ra, tuy trong các hiệu sách, đối với một vài đầu sách cũng có bán các tài liệu đã được dịch sang bản tiếng Việt nhưng đó chỉ là các tài liệu hỗ trợ học, còn các tài liệu khác như sách báo hay tài liệu trên mạng thì không có bản dịch chính thức. Trên các trang mạng xã hội như Facebook cũng tồn tại các nhóm và page chia sẻ tài liệu dịch tiếng Nhật, nhưng vì đều là các nhóm nhỏ lẻ không chính thống, hoặc do quá nhiều người chia sẻ mà không có hệ thống tổng hợp dẫn đến các tài liệu chia sẻ trên mạng xã hội và các trang web bị rải rác và khó tìm thấy, gây khó khăn cho người mới học và những người không quen thao tác tìm kiếm trên internet.

Thông qua quá trình khảo sát thực tế các vấn đề nêu trên và khảo sát các hệ thống tương tự đã tồn tại bao gồm Google Translate [10] và Viblo [11], tôi đưa ra bảng so sánh tính năng và hạn chế của các hệ thống đó. Chi tiết so sánh của các hệ thống được mô tả trong Bảng 2.1

Bảng 2.1: Kết quả khảo sát so sánh hệ thống hiện có

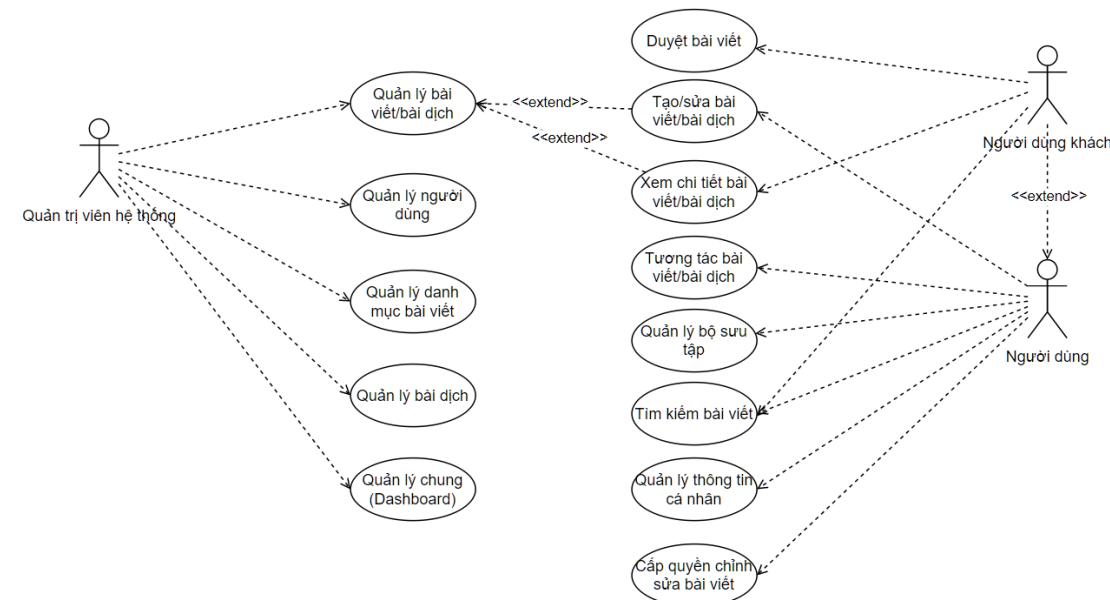
Hệ thống	Tính năng	Hạn chế
Google Translate	Hỗ trợ dịch đoạn văn và tài liệu trực tiếp Tính năng real-time, cung cấp bài dịch ngay lập tức Hỗ trợ dịch đa ngôn ngữ	Thiếu sự tương tác giữa người với người. Kết quả dịch nhiều trường hợp khó hiểu và lộn xộn do không phải dịch bởi người hoàn toàn.

Viblo	Cộng đồng chia sẻ bài viết đa chủ đề (công nghệ, tiếng Nhật,...) Chức năng lưu lại các bài viết yêu thích hoặc đã bình luận. Có sự tương tác giữa người với người thông qua bình luận, chia sẻ,...	Là một hệ thống chia sẻ đa chủ đề không chuyên về tiếng Nhật
-------	--	--

2.2 Tổng quan chức năng

Qua quá trình khảo sát thực tế và phân tích hệ thống hiện có, các chức năng của hệ thống được thể hiện qua biểu đồ use case tổng quan ở Hình 2.1.

2.2.1 Biểu đồ use case tổng quan



Hình 2.1: Biểu đồ use case tổng quan

Mô tả biểu đồ usecase tổng quan:

- Tác nhân: Quản trị viên hệ thống và người dùng khách và người dùng.
- Điều kiện: Đối với người dùng và quản trị viên hệ thống, tất cả các usecase được thực hiện đều bắt buộc phải đăng nhập.
- Vai trò:

- **Người dùng khách** không cần thực hiện đăng nhập để sử dụng hệ thống. Có thể sử dụng một vài chức năng cơ bản như duyệt bài viết, xem chi tiết bài viết,... nhưng không thể thực hiện các chức năng chính khác như xem chi tiết bài dịch, bình luận, lưu trữ bài viết,...
- **Người dùng** là người sử dụng đã đăng ký tài khoản trong hệ thống. Có thể sử dụng đầy đủ các chức năng của hệ thống dành cho người dùng như duyệt bài viết, xem chi tiết bài viết và bài dịch, đóng góp bài dịch, quản lý tài khoản cá nhân,...
- **Quản trị viên hệ thống** có vai trò quản lý các thông tin trên hệ thống bao gồm thông tin các bài viết và bài dịch trong hệ thống, thông tin người dùng,...

Các use case chính với tác nhân là người dùng khách được mô tả trong Bảng 2.2.

Bảng 2.2: Mô tả use case tổng quan với tác nhân người dùng khách

Use case	Mô tả tóm tắt
Duyệt danh sách bài viết	Duyệt danh sách bài viết trên trang chủ theo chủ đề (category).
Tìm kiếm bài viết	Tìm kiếm bài viết trên hệ thống theo từ khóa hoặc theo category. Sau khi thực hiện tìm kiếm, hệ thống hiển thị danh sách các bài viết thỏa mãn yêu cầu của người dùng khách.
Xem chi tiết bài viết/bài dịch	Người dùng khách có thể xem chi tiết nội dung bài viết và bài dịch của bài viết đó. Nhưng không thể tương tác với bài viết ví dụ như bình luận hay yêu thích,...

Tác nhân người dùng có đầy đủ các usecase mà tác nhân người dùng khách có. Ngoài ra, các use case chính khác với tác nhân là người dùng được mô tả trong Bảng 2.3.

Bảng 2.3: Mô tả use case tổng quan với tác nhân người dùng

Use case	Mô tả tóm tắt
Tạo/sửa/xóa bài viết hoặc bài dịch	Người dùng có thể tạo mới một bài viết và chỉnh sửa bài viết mình đã tạo, thêm/sửa một bài dịch mới hoặc xóa bài viết hoặc bài dịch mình đã tạo
Xem chi tiết bài viết hoặc bài dịch	Xem nội dung chi tiết một bài viết và các bài dịch của bài viết đó Khác với người dùng khách, người dùng có thể tương tác với bài viết/bài dịch bao gồm: bình luận, yêu thích bài viết/bài dịch,...
Quản lý bộ sưu tập	Người dùng có thể tạo bộ sưu tập để lưu trữ các bài viết mình mong muốn. Đồng thời có thể sửa/xóa bộ sưu tập mình đã tạo
Quản lý thông tin cá nhân	Người dùng có thể xem và cập nhật thông tin tài khoản và thông tin cá nhân của bản thân

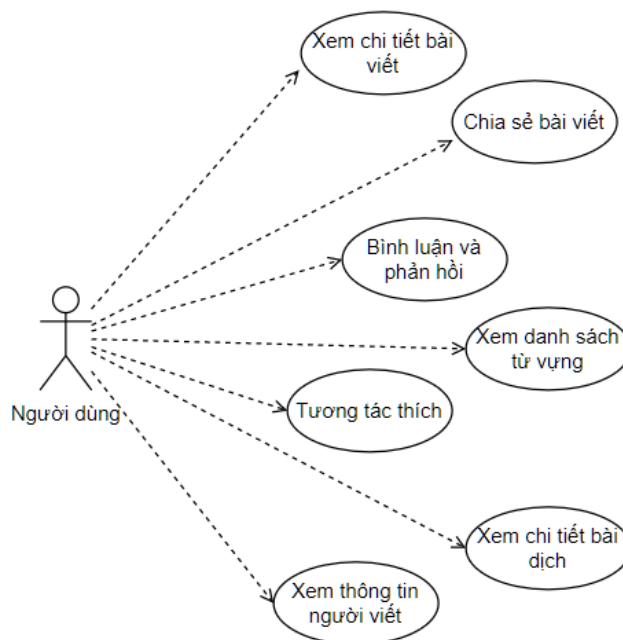
Các use case với tác nhân quản trị viên hệ thống được mô tả trong Bảng 2.4.

Bảng 2.4: Mô tả use case tổng quan với tác nhân quản trị viên hệ thống

Use case	Mô tả tóm tắt
Quản lý danh sách bài viết	Quản trị viên hệ thống có thể xem danh sách các bài viết có trong hệ thống Có thể xem chi tiết thông tin bài viết và thực hiện thao tác duyệt, từ chối hoặc xóa bài viết
Quản lý danh sách bài dịch	Quản trị viên hệ thống có thể xem danh sách các bài dịch có trong hệ thống Có thể xem chi tiết thông tin bài dịch và bài viết gốc và thực hiện thao tác duyệt, từ chối hoặc xóa bài viết

Quản lý danh mục bài viết	Mỗi bài viết sẽ thuộc một danh mục nào đó. Quản trị viên có thể xem danh sách các danh mục bài viết có trong hệ thống Các danh mục bài viết có trong hệ thống sẽ được người dùng lựa chọn khi tạo một bài viết mới
Quản lý người dùng	Quản trị viên có thể xem danh sách người dùng có trong hệ thống Có thể xem chi tiết thông tin người dùng và thực hiện thao tác xóa, khóa hoặc mở lại tài khoản người dùng
Quản lý chung	Quản trị viên có thể quản lý các thông tin chung ví dụ như: tổng số người dùng, tổng số bài viết, tổng số bài dịch,... có trong hệ thống .

2.2.2 Biểu đồ use case phân rã Xem chi tiết bài dịch



Hình 2.2: Biểu đồ phân rã use case Xem chi tiết bài dịch

Tác nhân: Người dùng

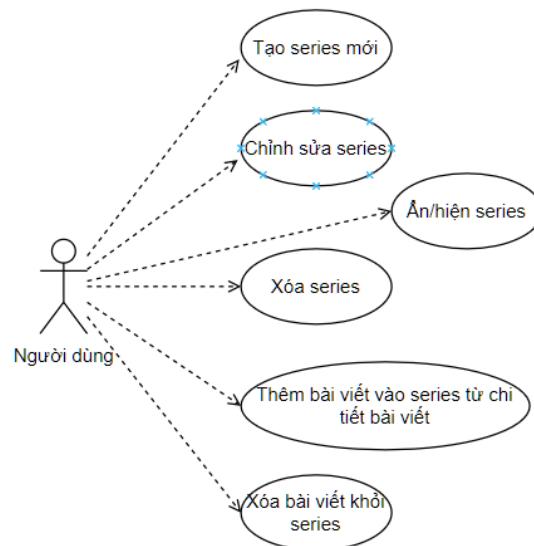
Các use case với tác nhân quản trị viên hệ thống được mô tả trong Bảng 2.5.

Bảng 2.5: Mô tả use case phân rã: Xem chi tiết bài viết/bài dịch

Use case	Mô tả tóm tắt
Xem chi tiết bài viết	Người dùng có thể xem chi tiết thông tin bài viết bao gồm: nội dung gốc của bài viết, nội dung các bài dịch tương ứng với bài viết, ngày tạo bài viết, tiêu đề,...
Chia sẻ quyền chỉnh sửa bài viết	Người dùng có thể cấp quyền cho người dùng khác cùng chỉnh sửa bài viết.
Bình luận và phản hồi	Người dùng có thể chia sẻ ý kiến của bản thân về bài viết/bài dịch và phản hồi lại bình luận của người dùng khác.
Xem danh sách từ vựng	Mỗi bài viết đi kèm với một danh sách từ vựng tương ứng với bài viết đó do người tạo bài viết lựa chọn. Người dùng có thể tham khảo danh sách từ vựng trong bài viết giúp cho việc đọc bài viết được dễ dàng hơn.
Tương tác bài viết	Đối với bài viết mà người dùng cảm thấy hay, người dùng có thể tương tác “Thích” với bài viết. Bài viết được tương tác thích sẽ được tự động lưu vào danh sách các bài viết đã thích của người dùng.
Xem chi tiết bài dịch	Ứng với mỗi bài viết có thể chưa có bài dịch hoặc có nhiều bài dịch tương ứng với bài viết đó. Các bài dịch được sắp xếp theo số lượng yêu thích đối với bài viết đó. Một bài viết có nhiều lượt yêu thích sẽ được hiển thị lên đầu tiên. Người dùng có thể lựa chọn xem và tham khảo chi tiết bài dịch tùy ý.
Tạo bài dịch	Người dùng có thể đóng góp bài dịch cho bài viết hiện tại.

Xem thông tin người viết	Thông tin người tạo bài viết và người viết bài dịch sẽ được hiển thị tương ứng. Người dùng có thể xem chi tiết thông tin của người tạo bài viết/bài dịch.
--------------------------	---

2.2.3 Biểu đồ use case phân rã: "Quản lý bộ sưu tập"



Hình 2.3: Biểu đồ use case phân rã: Quản lý bộ sưu tập

Tác nhân: Người dùng

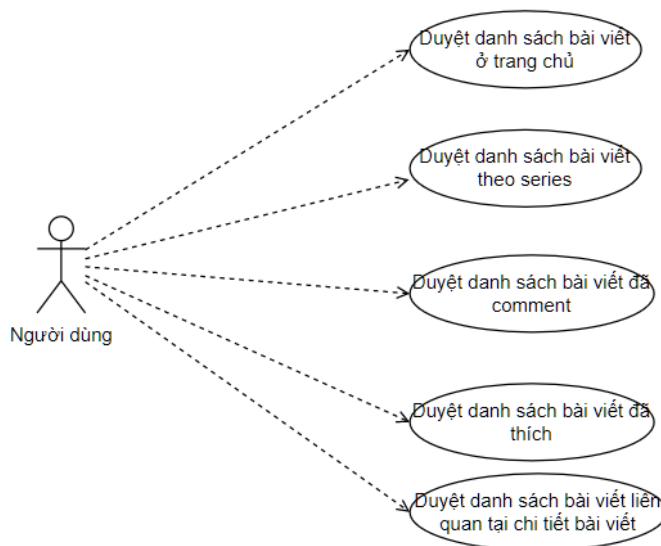
Các use case với tác nhân quản trị viên hệ thống được mô tả trong Bảng 2.6.

Bảng 2.6: Mô tả use case phân rã: Quản lý bộ sưu tập

Use case	Mô tả tóm tắt
Tạo bộ sưu tập mới	Người dùng có thể tạo bộ sưu tập để lưu lại nhiều bài viết theo chủ đề mong muốn. Một bộ sưu tập mới được tạo mặc định sẽ ở trạng thái ẩn với người khác (chỉ chủ sở hữu có thể xem nội dung)

Chỉnh sửa thông tin bộ sưu tập	Người dùng có thể chỉnh sửa thông tin bộ sưu tập đã tạo
Xóa bộ sưu tập	Người dùng có thể xóa bộ sưu tập đã tạo
Thêm bài viết vào bộ sưu tập	Người dùng có thể thêm bài viết mới vào một hoặc nhiều bộ sưu tập đã tạo
Xóa bài viết khỏi bộ sưu tập	Người dùng có thể xóa bài viết đã thêm khỏi bộ sưu tập của mình
Ẩn/hiện bộ sưu tập	Một bộ sưu tập có 2 trạng thái là hiện (mọi người có thể xem) và ẩn (chỉ chủ sở hữu có thể xem). Người dùng có thể thay đổi trạng thái ẩn hoặc hiện của bộ sưu tập Các bộ sưu tập có trạng thái hiện sẽ được hiển thị tại giao diện danh sách bộ sưu tập và bất cứ ai cũng có thể xem.

2.2.4 Biểu đồ usecase phân rã “Duyệt bài viết”



Hình 2.4: Biểu đồ use case phân rã: Duyệt bài viết

Tác nhân: Người dùng, Người dùng khách

Chi tiết phân rã use case Duyệt bài viết được mô tả trong Bảng 2.7.

Bảng 2.7: Mô tả use case phân rã: Duyệt bài viết

Use case	Mô tả tóm tắt
Duyệt danh sách bài viết ở trang chủ	Danh sách các bài viết được hiển thị theo từng danh mục tại trang chủ
Duyệt danh sách bài viết theo bộ sưu tập	Tác nhân có thể xem lại danh sách bài viết đã lưu vào bộ sưu tập của mình
Duyệt danh sách bài viết được cấp quyền chia sẻ	Khi tác nhân được người dùng khác cấp quyền chia sẻ chỉnh sửa một bài viết, bài viết đó sẽ được lưu vào danh sách bài viết được chia sẻ của tác nhân
Duyệt danh sách bài viết đã yêu thích	Khi tác nhân tương tác thích vào một bài viết bất kỳ, bài viết đó sẽ lưu vào danh sách bài viết yêu thích của tác nhân
Duyệt danh sách bài viết liên quan	Khi xem chi tiết nội dung của một bài viết, tác nhân có thể duyệt danh sách các bài viết có liên quan tới bài viết đang xem

2.2.5 Quy trình nghiệp vụ

Dưới đây đồ án trình bày về quy trình nghiệp vụ chính quan trọng của hệ thống là quy trình xem và quản lý chi tiết bài viết/bài dịch. Chi tiết nghiệp vụ của quy trình trên được mô tả bằng biểu đồ UML như Hình 2.5

Người dùng sau khi truy cập vào trang web, hệ thống sẽ hiển thị danh sách các bài viết được phân chia theo các danh mục. Sau khi lựa chọn xem chi tiết một bài viết trong danh sách bài viết, bất kể người dùng đã đăng nhập hay chưa đều có thể xem thông tin nội dung bài viết, danh sách bài dịch, thông tin tác giả và thông tin bình luận. Đối với trường hợp đã đăng nhập vào hệ thống, người dùng có thể tương tác nhiều hơn với bài viết thông qua các chức năng: tương tác thích, tạo bài dịch mới, bình luận và thêm bài viết vào bộ sưu tập.

**Hình 2.5:** Quy trình nghiệp vụ: Xem và quản lý chi tiết bài viết/bài dịch

2.3 Đặc tả chức năng

2.3.1 Đặc tả usecase “Lựa chọn ngôn ngữ hiển thị”

- Mô tả ngắn gọn:** Use case này mô tả cách thức người dùng đổi ngôn ngữ hiển thị của hệ thống, lựa chọn ngôn ngữ hiển thị trên giao diện người dùng.
- Luồng sự kiện chính:** Đặc tả luồng sự kiện của use case Lựa chọn ngôn ngữ hiển thị được mô tả trong Bảng 2.8.

Bảng 2.8: Mô tả luồng sự kiện use case: Lựa chọn ngôn ngữ hiển thị

ID	Tác nhân	Hệ thống
BF-0	Tại thanh navbar, lựa chọn chức năng "Đổi ngôn ngữ"	
BF-1		Hiển thị giao diện dropdown list chứa danh sách các ngôn ngữ có thể lựa chọn
BF-2	Lựa chọn ngôn ngữ mong muốn từ danh sách	
BF-3	Tại thanh navbar, lựa chọn chức năng "Đổi ngôn ngữ"	Hệ thống cập nhật lại ngôn ngữ và hiển thị lại giao diện theo ngôn ngữ đã lựa chọn AF-1: Hệ thống lỗi khi cập nhật lại ngôn ngữ
BF-4		Hệ thống lưu lại ngôn ngữ người dùng đã lựa chọn cho các lần đăng nhập sau
BF-5		Thông báo thành công

- Luồng sự kiện ngoại lệ:**

- AF-1: Trường hợp hệ thống lỗi khi đổi ngôn ngữ hiển thị trên giao diện, hiển thị thông báo lỗi và hiển thị ngôn ngữ mặc định của hệ thống là Tiếng Anh (English)

- Tiền điều kiện:** Không có

- Hậu điều kiện:**

- Nếu thành công, giao diện được hiển thị bằng ngôn ngữ người dùng đã lựa chọn
- Nếu thất bại, hiển thị thông báo lý do

2.3.2 Đặc tả usecase “Tạo bài viết mới”

- **Mô tả ngắn gọn:** Usecase này mô tả cách thức người dùng tạo một bài viết mới.
- **Luồng sự kiện chính:** Đặc tả luồng sự kiện của usecase Tạo bài viết mới được mô tả trong Bảng 2.9.

Bảng 2.9: Mô tả luồng sự kiện usecase: Tạo bài viết mới

ID	Tác nhân	Hệ thống
BF-0	Click button "Tạo bài viết" tại thanh navbar	
BF-1		Hiển thị giao diện tạo bài viết
BF-2	Người dùng nhập đủ các thông tin bắt buộc của bài viết (tiêu đề, nội dung,...) Người dùng có thể upload ảnh thumbnail của bài viết, trường hợp không lựa chọn, thumbnail là ảnh avatar của người dùng	
BF-3	Người dùng click button "Tạo bài viết"	
BF-4		Hệ thống kiểm tra các trường thông tin đã nhập: AF-1: Các trường thông tin không hợp lệ: AF-2: Token đăng nhập của người dùng đã hết hạn
BF-5		Hệ thống tạo bài viết mới thành công, thông báo thành công cho người dùng

- **Luồng sự kiện ngoại lệ:**

- AF-1: Trường hợp các trường thông tin người dùng nhập không hợp lệ, hiển thị thông báo lỗi
- AF-2: Trường hợp token đăng nhập của người dùng đã hết hạn, thông báo lỗi và điều hướng người dùng về giao diện Đăng nhập

- **Tiền điều kiện:** Người dùng đã đăng nhập thành công vào hệ thống

- **Hậu điều kiện:**

- Nếu thành công, hiển thị thông báo tạo bài viết mới thành công
- Nếu thất bại, hiển thị thông báo lý do

2.3.3 Đặc tả usecase “Tạo bài dịch mới”

- **Mô tả ngắn gọn:** Usecase này mô tả cách thức người dùng tạo một bài dịch mới cho một bài viết.
- **Luồng sự kiện chính:** Đặc tả luồng sự kiện của usecase Tạo bài dịch mới được mô tả trong Bảng 2.10.

Bảng 2.10: Mô tả luồng sự kiện usecase: Tạo bài dịch mới

ID	Tác nhân	Hệ thống
BF-0	Người dùng truy cập bài viết muốn tạo thêm bài dịch	
BF-1		Hiển thị giao diện chi tiết bài viết và danh sách các bài dịch tương ứng với bài viết
BF-2	Người dùng lựa chọn chức năng tạo thêm bài dịch	
BF-3		Hiển thị giao diện nhập nội dung bài dịch
BF-4	Người dùng nhập thông tin nội dung bài dịch và click button “Tạo bài dịch”	

BF-5		Hệ thống kiểm tra tính hợp lệ của nội dung đã nhập: AF-1: Nội dung đã nhập không hợp lệ AF-2: Token đăng nhập của người dùng đã hết hạn
BF-6		Hệ thống cập nhật bài dịch cho bài viết và hiển thị thông báo thành công

- **Luồng sự kiện ngoại lệ:**

- AF-1: Trường hợp các trường thông tin người dùng nhập không hợp lệ, hiển thị thông báo lỗi
- AF-2: Trường hợp token đăng nhập của người dùng đã hết hạn, thông báo lỗi và điều hướng người dùng về giao diện Đăng nhập

- **Tiền điều kiện:** Người dùng đã đăng nhập thành công vào hệ thống

- **Hậu điều kiện:**

- Nếu thành công, hiển thị thông báo tạo bài dịch thành công
- Nếu thất bại, hiển thị thông báo lý do

2.3.4 Đặc tả usecase “Xem chi tiết bài viết”

- **Mô tả ngắn gọn:** Usecase này mô tả cách thức người dùng xem chi tiết một bài viết
- **Luồng sự kiện chính:** Đặc tả luồng sự kiện của usecase Xem chi tiết bài viết được mô tả trong Bảng 2.11.

Bảng 2.11: Mô tả luồng sự kiện usecase: Xem chi tiết bài viết

ID	Tác nhân	Hệ thống
BF-0	Click vào bài viết bất kỳ tại giao diện “Danh sách bài viết” Click và bài viết bất kỳ tại giao diện Series-Bài viết Click vào bài viết bất kỳ tại giao diện Profile-Danh sách bài viết	
BF-1		Hiển thị giao diện bài viết tương ứng AF-1: Token truy cập của người dùng đã hết hạn hoặc người dùng chưa đăng nhập AF-2: Bài viết không tồn tại trong hệ thống
BF-2	Thông báo hiển thị giao diện bài viết thành công	

- **Luồng sự kiện ngoại lệ:**

- AF-1: Trường hợp token truy cập của người dùng đã hết hạn hoặc người dùng chưa đăng nhập, người dùng chỉ có thể đọc và không thể thao tác khác (bình luận, thích, tạo bài dịch,...)
- AF-2: Trường hợp bài viết không tồn tại trong hệ thống, thông báo lỗi

- **Tiền điều kiện:** Không có

- **Hậu điều kiện:**

- Nếu thành công, hiển thị thông tin bài viết lên giao diện
- Nếu thất bại, hiển thị thông báo lý do

2.3.5 Đặc tả usecase “Xem chi tiết bài dịch

- **Mô tả ngắn gọn:** Usecase này mô tả cách thức người dùng xem chi tiết các bài dịch của một bài viết
- **Luồng sự kiện chính:** Đặc tả luồng sự kiện của usecase Xem chi tiết bài dịch được mô tả trong Bảng 2.12.

Bảng 2.12: Mô tả luồng sự kiện usecase: Xem chi tiết bài dịch

ID	Tác nhân	Hệ thống
BF-0	Người dùng truy cập bài viết	
BF-1		Hiển thị giao diện chi tiết bài viết và danh sách các bài dịch tương ứng với bài viết
BF-2	Lựa chọn bài dịch mong muốn xem từ danh sách bài dịch của bài viết	
BF-3		Hiển thị nội dung bài dịch đã chọn AF-1: Bài dịch không tồn tại trong hệ thống

- **Luồng sự kiện ngoại lệ:**

- AF-1: Trường hợp không lấy được bài dịch đã lựa chọn, hiển thị thông báo lỗi

- **Tiền điều kiện:** Không có

- **Hậu điều kiện:**

- Nếu thành công, hiển thị nội dung bài dịch đã chọn của bài viết
- Nếu thất bại, hiển thị thông báo lý do

2.3.6 Đặc tả usecase “Quản trị viên duyệt danh sách bản ghi”

- **Mô tả ngắn gọn:**

- Usecase này mô tả cách thức quản trị viên duyệt danh sách các bản ghi có trong hệ thống
- Mục đích giúp cho quản trị viên có thể quản lý, theo dõi và tương tác với dữ liệu trong hệ thống
- “Bản ghi” ở đây được hiểu là các mục dữ liệu mà quản trị viên có thể quản lý bao gồm: dữ liệu tài khoản người dùng, dữ liệu bài viết, dữ liệu bài dịch, dữ liệu danh mục bài viết, ...

- **Luồng sự kiện chính:** Đặc tả luồng sự kiện của usecase Quản trị viên duyệt

danh sách bản ghi được mô tả trong Bảng 2.13.

Bảng 2.13: Mô tả luồng sự kiện usecase: Quản trị viên duyệt danh sách bản ghi

ID	Tác nhân	Hệ thống
BF-0	Tại giao diện web dành cho quản trị viên, lựa chọn mục "Quản lý" (trong đó là một trong số các mục dữ liệu đã nêu trên)	
BF-1		<p>Hiển thị danh sách bản ghi của có trong hệ thống dưới dạng bảng (table).</p> <p>Trường hợp có nhiều bản ghi, danh sách được hiển thị theo nhiều trang, mỗi trang hiển thị 10-15 bản ghi</p> <p>AF-1: Không có bản ghi nào trong hệ thống</p> <p>AF-2: Lỗi hiển thị danh sách bản ghi trong hệ thống</p>
BF-2	Quản trị viên lựa chọn hiển thị 1 trang cụ thể trong danh sách	
BF-3		Cập nhật lại danh sách bản ghi, hiển thị các bản ghi ở trang tương ứng.
BF-4	Quản trị viên tìm kiếm bản ghi cụ thể theo từ khóa	
BF-5		Cập nhật lại danh sách bản ghi thỏa mãn điều kiện tìm kiếm đã được nhập
BF-6	Quản trị viên sử dụng chức năng filter hiển thị danh sách bản ghi theo điều kiện	

BF-7		Cập nhật lại danh sách bản ghi thỏa mãn điều kiện được lựa chọn
------	--	---

- **Luồng sự kiện ngoại lệ:**

- AF-1: Trường hợp không có bản ghi nào thuộc mục dữ liệu đã chọn trong hệ thống, không hiển thị bảng (table), thay vào đó hiển thị mô tả không có dữ liệu về mục dữ liệu đã chọn trong hệ thống
- AF-2: Trường hợp lỗi hiển thị danh sách bản ghi, hiển thị thông báo lỗi và hiển thị bảng (table) không có nội dung (chỉ có tên cột)

- **Tiền điều kiện:** Quản trị viên đã đăng nhập thành công vào hệ thống bằng tài khoản quản trị viên

- **Hậu điều kiện:**

- Nếu thành công, hiển thị giao diện danh sách bản ghi của mục dữ liệu đã chọn trong hệ thống theo dạng bảng (table)
- Nếu thất bại, hiển thị thông báo lý do

2.4 Yêu cầu phi chức năng

Bên cạnh các yêu cầu về chức năng, hệ thống còn đáp ứng các yêu cầu phi chức năng như sau:

- **Tính dễ dùng:**

- Hệ thống cung cấp trang web có giao diện thân thiện với người dùng, thống nhất về màu sắc, vị trí.
- Sự phân chia quyền chức năng và giao diện giữa quản trị viên và người dùng mạch lạc và chặt chẽ giúp quản lý hệ thống dễ dàng.

- **Yêu cầu linh động:**

- Trang web dự kiến sẽ được đẩy lên server, có tên miền cụ thể nên người dùng có thể truy cập mọi lúc, mọi nơi.
- Việc phát triển riêng biệt Front-end và Back-end giúp cho hệ thống chạy ổn định, tăng khả năng trải nghiệm của người dùng

- **Độ tin cậy:**

- Qua quá trình kiểm thử, các lỗi phát sinh cơ bản của hệ thống đã được sửa toàn bộ giúp hệ thống có độ tin cậy cao.
- Yêu cầu tương thích:
 - Hệ thống cung cấp giao diện web tương thích hết với tất cả các trình duyệt hiện nay như Chrome, Microsoft Edge.
 - Hệ thống cung cấp giao diện responsive tương thích với mọi màn hình có độ rộng khác nhau của mọi thiết bị

Trong chương này, đồ án đã trình bày về quá trình tìm hiểu và rút ra các yêu cầu cần thiết cho hệ thống. Các chức năng của trang web cũng được biểu diễn qua biểu đồ use case với các tác nhân.

Tiếp theo, ở chương 3 đồ án sẽ trình bày về các công nghệ sử dụng cho để phát triển và triển khai hệ thống.

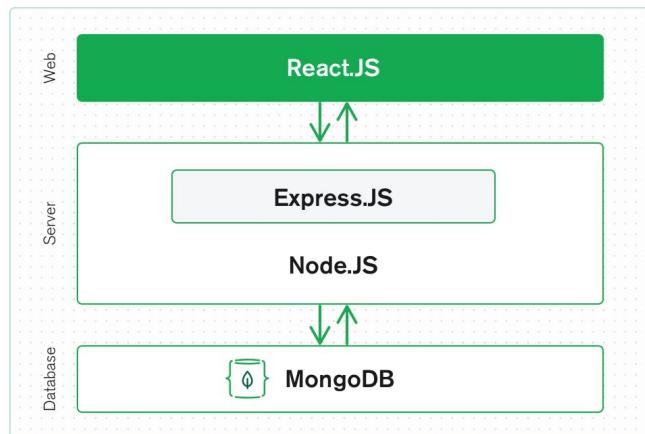
CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

Từ việc khảo sát và phân tích yêu cầu , tiếp theo em sẽ giới thiệu về một số lý thuyết và công nghệ sử dụng. Đó là công nghệ lập trình Web, em xin trình bày dưới đây.

3.1 Giới thiệu về MERN Stack

3.1.1 Khái niệm

MERN là một thuật ngữ rút gọn của MongoDB (M) [6], ExpressJS (E) [4], ReactJs (R) **react-js** và NodeJs (N) [5]. MERN Stack là một stack Javascript được thiết kế để giúp phát triển ứng dụng web toàn ngăn xếp dễ dàng hơn và nhanh hơn. Tất cả bốn công nghệ này cung cấp một khuôn khổ hoàn chỉnh cho các nhà phát triển để tạo ra bất kỳ ứng dụng web nào. MERN đang tuân theo kiến trúc 3 tầng truyền thống, bao gồm tầng hiển thị front-end (ReactJs), tầng ứng dụng (ExpressJs và NodeJs) và tầng cơ sở dữ liệu (MongoDB). Kiến trúc mô hình của MERN Stack được mô tả trong Hình 3.1



Hình 3.1: Mô hình kiến trúc của MERN Stack

3.1.2 Lý do lựa chọn

MERN Stack cho phép nhà phát triển dễ dàng phát triển hệ thống web với kiến trúc 3 tầng bao gồm front-end, back-end và cơ sở dữ liệu bằng cách sử dụng JavaScript và JSON. Trong đó:

- MongoDB, là cơ sở của ngăn xếp MERN, được thiết kế để lưu trữ dữ liệu JSON nguyên bản. Mọi thứ trong đó, bao gồm CLI và ngôn ngữ truy vấn, được xây dựng bằng JSON và JS. Hệ quản trị cơ sở dữ liệu NoSQL hoạt động

tốt với NodeJS và do đó, cho phép thao tác, biểu diễn và lưu trữ dữ liệu JSON ở mọi cấp của ứng dụng.

- ExpressJs là một web application framework cho NodeJs, cung cấp các tính năng mạnh mẽ cho việc xây dựng một ứng dụng web đúng nghĩa hoặc ngược lại. ExpressJs cũng có thể sử dụng để xây dựng bộ APIs mạnh mẽ và thân thiện với người dùng, vì nó cung cấp rất nhiều tiện ích HTTP và middleware cho việc kết nối. Điều này bổ sung hoàn hảo cho khung ReactJS, một khung JS front-end để phát triển giao diện người dùng tương tác trong HTML trong khi giao tiếp với máy chủ.

Khi hai công nghệ này hoạt động với JSON, dữ liệu sẽ được lưu chuyển liên tục, giúp hệ thống có thể phát triển nhanh chóng và dễ dàng. Hơn nữa, để hiểu toàn bộ hệ thống, thay vì phải tìm hiểu và phát triển bằng nhiều ngôn ngữ, nhiều công nghệ và thư viện, với MERN Stack ta chỉ cần hiểu một ngôn ngữ, tức là JavaScript và cấu trúc tài liệu JSON. Điều này giúp giảm đáng kể khó khăn trong việc phát triển hệ thống đối với lập trình viên và cũng nhờ vậy MERN Stack đang ngày càng được sử dụng rộng rãi.

3.2 Môi trường runtime Javascript: NodeJs

3.2.1 Khái niệm

NodeJS là một môi trường runtime chạy JavaScript đa nền tảng và có mã nguồn mở, được sử dụng để chạy các ứng dụng web bên ngoài trình duyệt của client. Nền tảng này được phát triển bởi Ryan Dahl vào năm 2009, được xem là một giải pháp hoàn hảo cho các ứng dụng sử dụng nhiều dữ liệu nhờ vào mô hình hướng sự kiện (event-driven) không đồng bộ.

3.2.2 Lý do lựa chọn

Nodejs được xây dựng trên V8 JavaScript Engine – trình thông dịch thực thi mã JavaScript giúp chúng ta có thể xây dựng được các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp một cách nhanh chóng và dễ dàng mở rộng.

NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất, vì vậy NodeJs được ưa chuộng và ngày càng được sử dụng rộng rãi nhờ các đặc điểm (i) IO hướng sự kiện không đồng bộ, cho phép xử lý nhiều yêu cầu đồng thời, (ii) sử dụng Javascript – một ngôn ngữ lập trình dễ học và (iii) NPM (Node Package Manager) và module Node đang ngày càng phát triển mạnh mẽ.

Từ các đặc điểm trên, tôi đã chọn NodeJs là công cụ phát triển môi trường

backend cho hệ thống.

3.3 Framework ExpressJs

3.3.1 Khái niệm

Expressjs là một framework được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. Expressjs hỗ trợ các method HTTP và middleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.

3.3.2 Lý do lựa chọn

Một số chức năng chính của Express bao gồm:

- Thiết lập các lớp trung gian để trả về các HTTP request.
- Định nghĩa router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL.
- Cho phép trả về các trang HTML dựa vào các tham số.

Vì ExpressJs chỉ yêu cầu ngôn ngữ lập trình Javascript nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn với các lập trình viên và nhà phát triển. Expressjs cũng là một khuôn khổ của Node.js do đó hầu hết các mã code đã được viết sẵn cho các lập trình viên có thể làm việc. Nhờ vậy, nhà phát triển có thể dễ dàng tạo các ứng dụng 1 web, nhiều web hoặc kết hợp. Do có dung lượng khá nhẹ, Expressjs giúp cho việc tổ chức các ứng dụng web thành một kiến trúc MVC có tổ chức hơn. Từ các ưu điểm và tính năng trên, tôi đã lựa chọn ExpressJs là framework hỗ trợ phát triển môi trường backend cho hệ thống.

3.4 Hệ quản trị cơ sở dữ liệu NoSQL MongoDB

3.4.1 Khái niệm

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

3.4.2 Lý do lựa chọn

MongoDB là một NoSQL, mang một số lợi thế hơn so với các cơ sở dữ liệu dạng quan hệ SQL (RDBMS) bao gồm:

- Ít Schema hơn: MongoDB là một cơ sở dữ liệu dựa trên Document, trong đó một Collection giữ các Document khác nhau. Số trường, nội dung và kích cỡ của Document này có thể khác với Document khác.

- Cấu trúc của một đối tượng là rõ ràng.
- Không có các Join phức tạp.
- Khả năng truy vấn sâu hơn. MongoDB hỗ trợ các truy vấn động trên các Document bởi sử dụng một ngôn ngữ truy vấn dựa trên Document mà mạnh mẽ như SQL.
- Sử dụng bộ nhớ nội tại để lưu giữ phần công việc, giúp truy cập dữ liệu nhanh hơn.

Qua các lợi thế nêu trên có thể nhận thấy đặc điểm của MongoDB là có tốc độ truy xuất dữ liệu nhanh, phù hợp cho các ứng dụng cần tốc độ phản hồi nhanh và realtime (Facebook, Blogs,...). Hơn nữa, xét thấy đặc điểm của hệ thống này không cần quá chú trọng vào tính toàn vẹn của dữ liệu như các hệ thống tính toán (ví dụ banking, ...) nên tôi đã lựa chọn MongoDB là cơ sở dữ liệu của hệ thống.

3.5 Thư viện ReactJS

3.5.1 Khái niệm

ReactJS là một opensource được phát triển bởi Facebook, ra mắt vào năm 2013, bản thân nó là một thư viện Javascript được dùng để xây dựng các tương tác với các thành phần trên website. Một trong những điểm nổi bật nhất của ReactJS đó là việc render dữ liệu không chỉ thực hiện được trên tầng Server mà còn ở dưới Client nữa.

3.5.2 Lý do lựa chọn

ReactJS ra đời với mục đích chính là cải thiện quá trình phát triển UI. Để tăng tốc quá trình phát triển và giảm thiểu những rủi ro có thể xảy ra trong khi coding, React cung cấp cho lập trình viên khả năng Reusable Code (tái sử dụng code) bằng cách đưa ra 2 khái niệm quan trọng bao gồm: JSX và Virtual DOM. Nhờ vậy, React được đánh giá cao trong việc hỗ trợ xây dựng giao diện nhanh, hạn chế lỗi trong quá trình code và cải thiện hiệu năng (performance) của website. Ngoài ra một số tính năng đặc biệt khác của ReactJS cũng là lý do tôi lựa chọn thư viện này để phát triển Front-end cho hệ thống:

- Phù hợp với đa dạng thể loại website: ReactJS cung cấp đủ các công cụ để dễ dàng khởi tạo phát triển một website thay vì phải code nhiều như khi tạo website thuần với HTML và Javascript
- Thân thiện với SEO: Bản chất ReactJS là một thư viện JavaScript, Google Search Engine hiện nay đã crawl và index được code JavaScript
- Debug dễ dàng

3.6 Docker

3.6.1 Khái niệm

Docker là một nền tảng cho lập trình viên và quản trị viên hệ thống phát triển và triển khai hệ thống. Nó cho phép tạo các môi trường độc lập và tách biệt để khởi chạy và phát triển ứng dụng và môi trường này được gọi là container. Khi cần triển khai hệ thống lên bất kỳ máy chủ (server) nào chỉ cần khởi chạy container của Docker thì hệ thống ứng dụng sẽ được khởi chạy ngay lập tức.

3.6.2 Lý do lựa chọn

Việc thiết lập và triển khai hệ thống ứng dụng lên một hoặc nhiều server rất vất vả từ việc phải cài đặt các công cụ, môi trường cần cho hệ thống, đến việc chạy được ứng dụng. Chưa kể việc không đồng nhất giữa các môi trường trên nhiều server khác nhau. Chính vì lý do đó Docker được ra đời để giải quyết vấn đề này. Hơn nữa, nhờ sử dụng Docker, việc thiết lập môi trường phát triển sẽ trở nên dễ dàng. Chỉ cần thiết lập một lần duy nhất và không bao giờ phải cài đặt lại các thư viện môi trường lại. Vì vậy, tôi đã lựa chọn Docker là công cụ chính để thiết lập và triển khai hệ thống lên server thực tế.

3.7 Dịch vụ Cloud Server DigitalOcean

3.7.1 Khái niệm

DigitalOcean VPS là một nhà cung cấp dịch vụ máy chủ ảo, tương tự như các nhà cung cấp dịch vụ máy chủ ảo khác như Microsoft Azure hay Amazon VPS, DigitalOcean VPS cung cấp các dịch vụ Cloud Server VPS của họ dựa trên nền tảng điện toán đám mây giúp triển khai và mở rộng ứng dụng chạy đồng thời trên nhiều máy tính với các tính năng tối ưu từ đám mây (Cloud).

3.7.2 Lý do lựa chọn

Nói tới điện toán đám mây thì ta có thể hình dung rằng mọi ứng dụng được cung cấp dưới dạng dịch vụ, lập trình viên sẽ tiết kiệm chi phí, thay vì cần có một máy chủ thật, ta sẽ không cần phải quản lý máy chủ vật lý, mọi việc được DigitalOcean đơn giản hóa, ta có thể mở rộng server, nâng cấp cấu hình,... dễ dàng nhờ việc sử dụng các dịch vụ được cung cấp bởi các nhà cung cấp dịch vụ máy chủ ảo. Việc này, với máy chủ thật ta sẽ mất hàng giờ, thậm chí hàng ngày để thực hiện.

Hơn nữa, đặc điểm của OceanDigital VPS là cung cấp các dịch vụ phù hợp với các doanh nghiệp hay cá nhân sở hữu website với mức lưu lượng truy cập ổn định mà không cần mở rộng quy mô động, có thể dễ dàng nâng cấp VPS từ bảng điều khiển quản trị với giao diện thân thiện tương thích mọi thiết bị. Vì vậy tôi đã lựa chọn sử dụng dịch vụ máy chủ ảo của OceanDigital VPS để thiết lập và triển khai

hệ thống.

3.8 Cân bằng tải (Load Balancing) với Nginx Server

3.8.1 Khái niệm

NGINX là một phần mềm web server mã nguồn mở, sử dụng kiến trúc hướng sự kiện (event-driven) không đồng bộ (asynchronous). Mục tiêu ban đầu để phục vụ HTTP cache nhưng sau được áp dụng vào reverse proxy, HTTP load balancer và các giao thức truyền mail như IMAP4, POP3, và SMTP. Tôi sử dụng Nginx Server trong hệ thống này với mục đích thiết kế cân bằng tải HTTP load balancer cho hệ thống.

3.8.2 Lý do lựa chọn

Loadbalancing hay còn được biết đến với tên Cân bằng tải là một kĩ thuật được sử dụng cực kì phổ biến của các web developer nhằm mục đích tối ưu hóa việc sử dụng tài nguyên , băng thông, giảm độ trễ cũng như khả năng xử lý lỗi của của 1 website. Thay vì sử dụng một server cho một website, ta sẽ sử dụng nhiều server để làm việc đó. Cùng với sự gia tăng về web server ta sẽ cần có 1 máy chủ để phân phối các lưu lượng truy cập cho các server một cách hợp lý . Máy chủ này gọi là Loadbalancer. Vì vậy để phục vụ thiết kế hệ thống hiện tại với số lượng người lớn và để tăng tính ổn định của hệ thống, tôi đã quyết định sử dụng Nginx Server làm load balancer cho hệ thống.

Trong chương này tôi đã trình bày về khái niệm và lý do lựa chọn của những công nghệ chính được sử dụng trong hệ thống. Tiếp theo trong Chương tới, tôi sẽ trình bày thiết kế chi tiết và quá trình triển khai hệ thống.

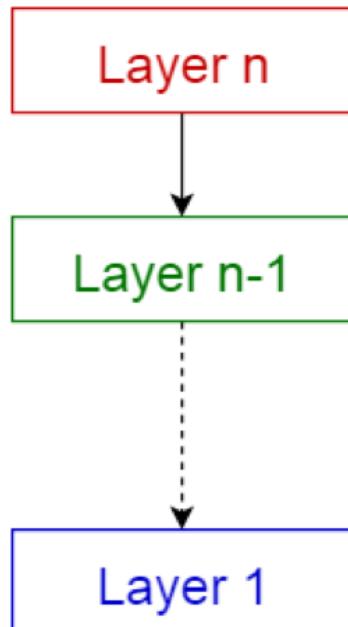
CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ

Trong Chương 3, tôi đã giới thiệu các công nghệ được sử dụng để phát triển hệ thống. Ở Chương này, tôi sẽ trình bày về thiết kế chi tiết hệ thống, quá trình phát triển và triển khai hệ thống.

4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm

Hệ thống được thiết kế theo kiến trúc phân lớp (Layered Pattern). Đây là một hệ thống mang tính phân lớp, mỗi lớp sẽ phụ thuộc vào các lớp bên dưới nó nhưng không biết và không phụ thuộc vào các lớp bên trên sử dụng nó. Kiến trúc này được mô tả trong hình 4.1



Hình 4.1: Kiến trúc phân lớp Layered Pattern

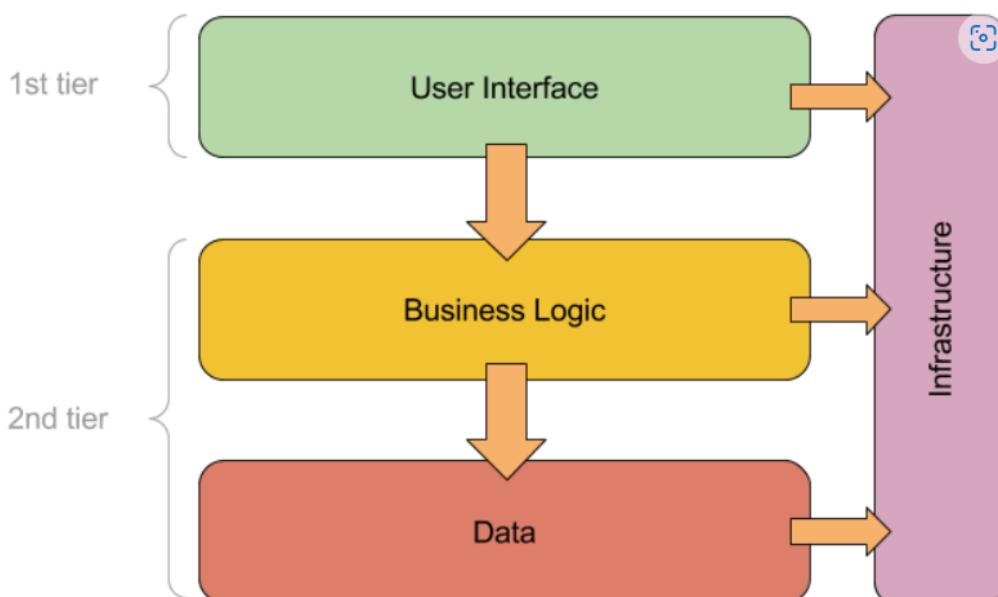
Mô hình này thể hiện tính chuyên nghiệp trong lập trình và phân tích thiết kế. Do hệ thống được chia thành các lớp nhỏ nên giúp phát triển ứng dụng nhanh, đơn giản, sản phẩm dễ nâng cấp, bảo trì. Hệ thống em xây dựng tuân thủ theo đúng kiến trúc phân lớp Layered Pattern. Kiến trúc này được chia làm 3 lớp chính bao gồm:

- **User Interface Layer:** đóng vai trò là Client, là những chương trình chạy trên desktop / CLI hoặc các web page. Các chương trình client lúc này thường là các rich client (nơi luồng công việc và kiểm tra input đầu vào đặt ở client). *Client ở đây là hệ thống Front-end application chạy trên Web Browser, làm nhiệm vụ*

render giao diện và gửi / nhận request tới Server.

- **Business Logic Layer:** đóng vai trò là Backend Server, là nơi đặt các logic về nghiệp vụ chính của hệ thống. *Server ở đây là hệ thống Back-end application, làm nhiệm vụ tiếp nhận các request từ phía Client, xử lý và lưu trữ data thông qua Data source layer.*
- **Data Source Layer:** đóng vai trò là Database, được tối ưu cho mục đích lưu trữ dữ liệu của hệ thống. *Nhiệm vụ chính là nhận yêu cầu thao tác với dữ liệu từ Backend Server và thực thi các tác vụ lưu trữ dữ liệu..*

Cấu trúc kiến trúc ba tầng sử dụng trong hệ thống được mô tả trong Hình 4.2



Hình 4.2: Cấu trúc phân lớp 3 tầng

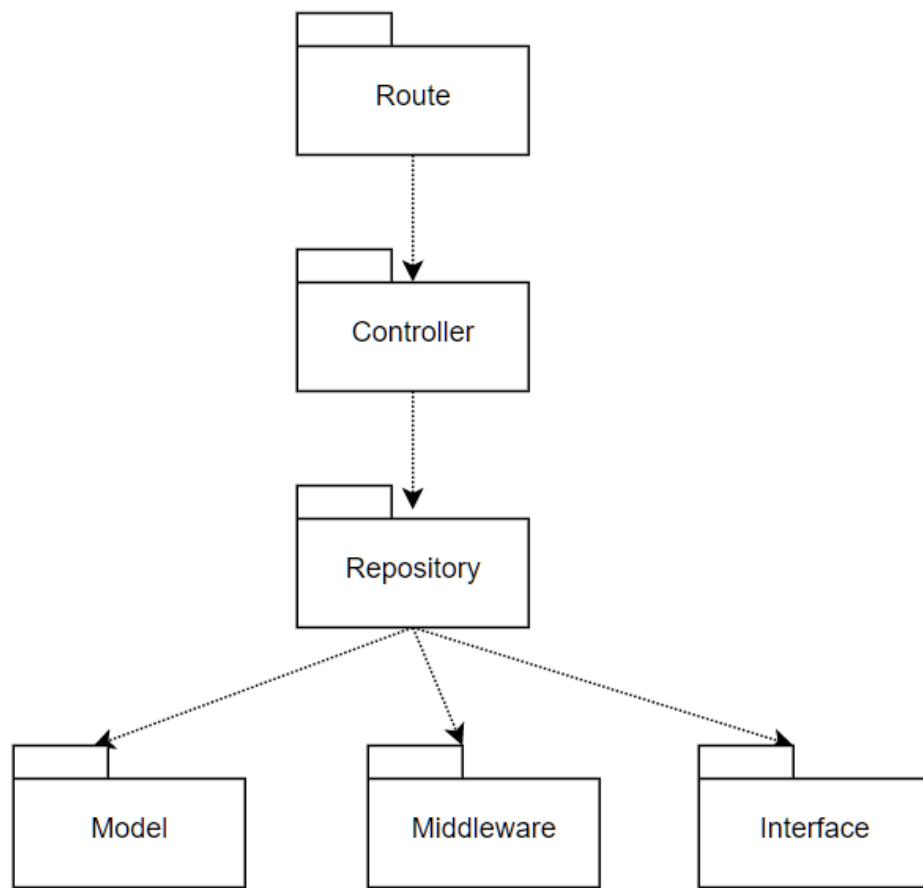
4.1.2 Thiết kế tổng quan

Thiết kế tổng quan của hệ thống được biểu diễn qua biểu đồ UML (UML package diagram) biểu thị sự phụ thuộc giữa các gói (package) như Hình 4.3

Hệ thống bao gồm 6 gói (package) không phụ thuộc lẫn nhau: Route, Controller, Repository, Model, Middleware và Interface. Nhiệm vụ chính và mục đích của từng gói được mô tả như dưới đây:

- Gói Route: định nghĩa các API endpoint – làm nhiệm vụ điều hướng các request đến Controller tương ứng.
- Gói Controller: định nghĩa các xử lý nhận request và trả response – làm nhiệm vụ xử lý nhận request, điều hướng tới xử lý ở gói Repository, nhận kết quả và xử lý trả response

- Gói Repository: định nghĩa các xử lý chính liên quan tới dữ liệu – làm nhiệm vụ thao tác với các gói khác và CSDL, trả về kết quả cho các gói tầng trên.
- Gói Model: định nghĩa thuộc tính và thể hiện mỗi liên hệ của các thực thể trong CSDL.
- Gói Middleware: định nghĩa các middleware chung thực hiện xử lý trung gian giữa request và response.
- Gói Interface: khai báo các phương thức cho gói Repository



Hình 4.3: Biểu đồ UML tổng quan hệ thống

4.1.3 Thiết kế chi tiết gói

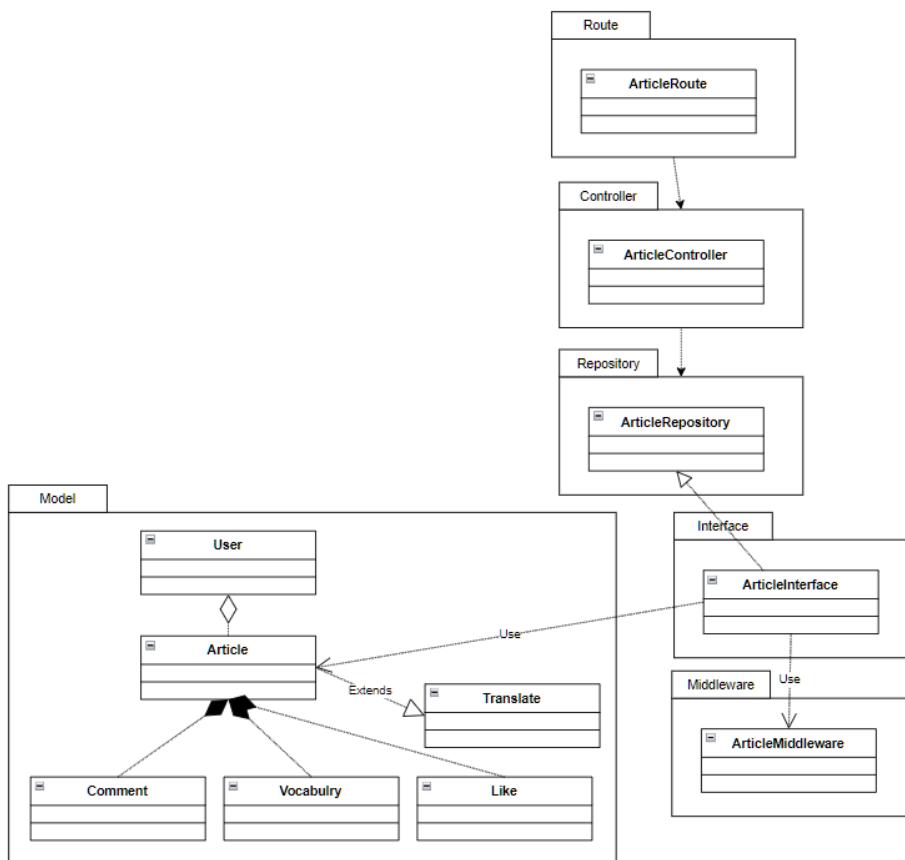
Hình 4.4 mô tả thiết kế chi tiết gói và mối quan hệ giữa các lớp trong gói tham gia nhóm chức năng liên quan tới bài viết/bài dịch.

Cụ thể, mối quan hệ giữa các gói và các lớp trong gói được biểu diễn trong hình trên như sau:

- Tại gói Route: Khi một request yêu cầu dữ liệu được gửi tới hệ thống, hệ thống sẽ kiểm tra loại dữ liệu mà request yêu cầu và điều hướng request tới xử lý ở

gói Route tương ứng (ở đây dữ liệu yêu cầu là article nên request sẽ được điều hướng tới ArticleRoute)

- Tại gói Controller: dựa vào thông tin request, điều hướng request tới xử lý được định nghĩa ở gói Repository
- Tại gói Repository: thao tác với các gói Model, Interface và Middleware - thực hiện xử lý request và trả về response
- Tại gói Model: Class Article có quan hệ hợp thành từ các class Comment, Vocabulary, Like và User. Mỗi bài viết được tạo bởi một user và mỗi bài viết chứa các thông tin về comment, vocabulary và like,... Ngoài ra, class Translate định nghĩa các thuộc tính liên quan tới bài dịch, là một class kế thừa của class Article.



Hình 4.4: Thiết kế chi tiết gói hệ thống

4.2 Thiết kế chi tiết

4.2.1 Thiết kế giao diện

Đặc tả thông tin về màn hình mà hệ thống đang hướng tới được thiết kế theo như mô tả trong Bảng 4.1:

Bảng 4.1: Đặc tả thiết kế giao diện

Thông tin	Thông số
Độ phân giải màn hình	HD (720p), Full HD (1080p)
Hỗ trợ responsive	Có
Độ rộng màn hình responsive	Màn hình lớn: 1200px trở lên Màn hình tiêu chuẩn: 992px-1200px Màn hình tablet: 768px-992px Màn hình điện thoại: 576px-768px
Kích thước màn hình tiêu chuẩn	1200x768px
Màu sắc chủ đạo	Xanh lá (55C57A) Xanh sáng (7ED56F) Xanh đậm (28B485) Xám nhạt (F7F7F7) Xám đậm (777)
Văn bản	Màu chữ: đen (000) Font chữ: Google Lato Font Cỡ chữ: 16px (Tiêu chuẩn), 18px (Tiêu đề)

Nút được thiết kế gồm 4 loại chính bao gồm:

- Nút mặc định chính: chữ trắng nền xanh lá như trong Hình 4.5. Được sử dụng trong các trường hợp thông thường như thực hiện chức năng Tìm kiếm, Tạo bài viết,... hoặc sử dụng trong các trường hợp xác nhận thông tin.

**Hình 4.5:** Nút mặc định chính

- Nút mặc định phụ: chữ đen, nền trắng như trong Hình 4.6. Được sử dụng trong các trường hợp chức năng thông thường như nút mặc định, nhưng mang ý nghĩa chức năng phụ, ít quan trọng hơn các chức năng sử dụng nút mặc định.



+ Button

Hình 4.6: Nút mặc định phụ

- Nút hủy bỏ: chữ đỏ nền trắng như trong Hình 4.7 . Được sử dụng trong trường hợp hủy thao tác.



Button

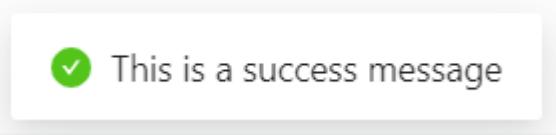
Hình 4.7: Nút hủy bỏ

- Nút biểu tượng: Không sử dụng text, chỉ sử dụng biểu tượng để thể hiện ý nghĩa của nút như trong Hình 4.8. Chức năng giống các nút ở trên, được sử dụng thay cho các nút ở trên tránh giao diện bị trùng lặp và căn chỉnh tốt hơn khi thiết kế giao diện màn hình nhỏ.



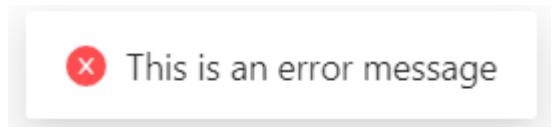
Hình 4.8: Nút biểu tượng

Thông điệp phản hồi được sử dụng để hiển thị thông báo trạng thái thành công hoặc lỗi của một tác vụ. Được hiển thị ở chính giữa trên cùng màn hình, bao gồm icon trạng thái và nội dung thông điệp. Hình 4.9 là ví dụ mô tả thông điệp khi tác vụ thành công. Hình 4.10 là ví dụ mô tả thông điệp khi tác vụ thất bại.



✓ This is a success message

Hình 4.9: Thông điệp phản hồi khi tác vụ thành công



✗ This is an error message

Hình 4.10: Thông điệp phản hồi khi tác vụ thất bại

Giao diện chung của hệ thống được thiết kế như Hình 4.11. Giao diện bao gồm Header nằm trên cùng và nội dung chính của giao diện ở giữa màn hình dưới

Header. Trong thiết kế Header, trên cùng bên trái là Logo của hệ thống cùng thanh tìm kiếm, trên cùng bên phải là nút đăng nhập và nút đăng ký (trường hợp chưa đăng nhập) hoặc danh mục lựa chọn chức năng người dùng và hiển thị danh sách thông báo (trường hợp đã đăng nhập). Bên dưới là thanh Navbar nằm ngang chứa các nút điều hướng, khi ấn vào sẽ điều hướng người dùng tới các giao diện khác của hệ thống.

LOGO	SEARCH BAR			FILTER		NOTIFY ICON	USER ICON	
	NAVBAR-1	NAVBAR-2	NAVBAR-3					
				MAIN CONTENT				

Hình 4.11: Thiết kế giao diện: Giao diện chung

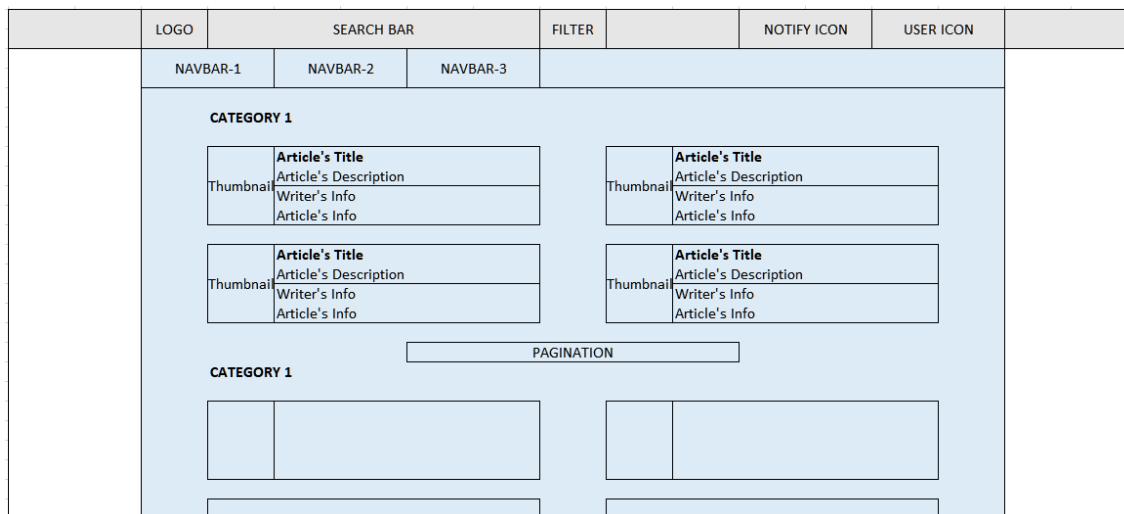
Dưới đây tôi xin đưa ra thiết kế giao diện của một số chức năng chính quan trọng của hệ thống:

Giao diện màn danh sách bài viết được thiết kế như Hình 4.12 . Giao diện hiển thị danh sách các bài viết được phân chia theo nhiều danh mục. Mỗi danh mục hiển thị tối đa 8 bài viết được chia làm 2 cột và 4 hàng, trong trường hợp danh mục đó có nhiều bài viết, người dùng có thể điều hướng thông qua mục Phân trang (PAGINATION). Mỗi bài viết sẽ hiển thị các thông tin cơ bản bao gồm ảnh thumbnail, tiêu đề, mô tả, thông tin người viết, danh mục, ..

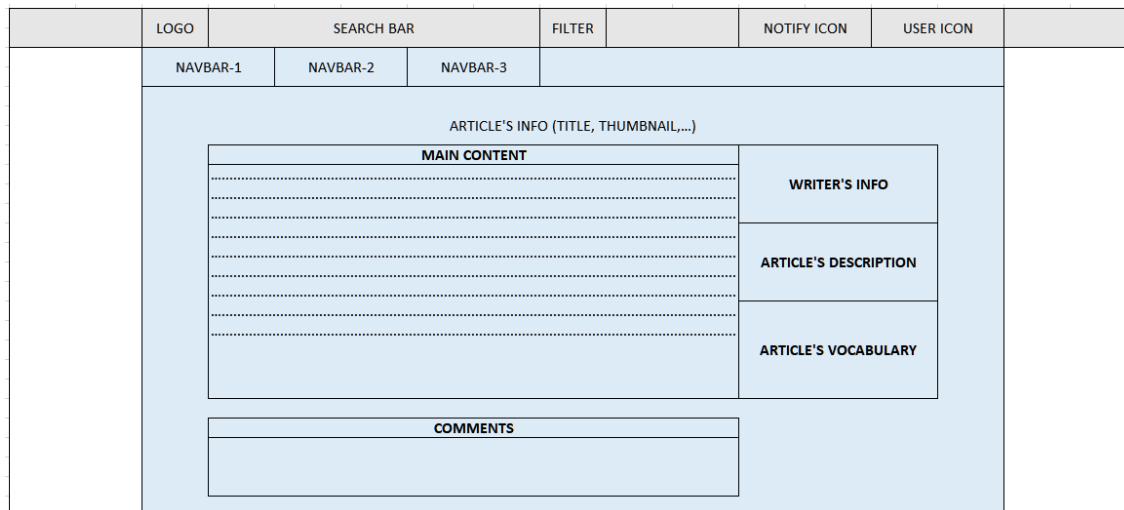
Giao diện màn Chi tiết bài viết được thiết kế như Hình 4.13. Trên cùng là thông tin về tiêu đề và thumbnail của bài viết. Tiếp đến giao diện được hiển thị thành 2 mục: bên trái chiếm 2/3 màn hình là nội dung chính của bài viết, bên phải chiếm 1/3 màn hình là các thông tin khác của bài viết bao gồm thông tin người viết, mô tả bài viết và danh sách từ vựng của bài viết. Cuối cùng là mục danh sách các bình luận của người dùng về bài viết. Giao diện Chi tiết bài viết được hiển thị thành các phần mục rõ ràng, không quá nhiều thông tin khiến cho người dùng dễ nắm bắt được các thông tin chính của bài viết.

Giao diện màn danh sách và chi tiết bộ sưu tập được hiển thị như Hình 4.14. Giao diện được chia ra làm 2 phần rõ ràng gồm: bên trái là danh sách các bộ sưu

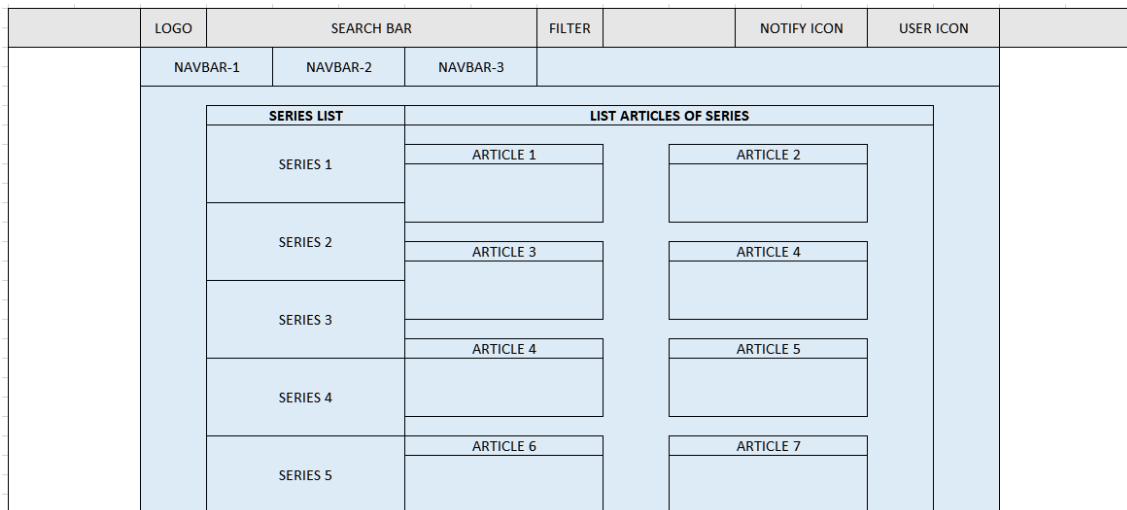
tập người dùng đã tạo, bên phải hiển thị danh sách các bài viết của một bộ sưu tập. Khi người dùng lựa chọn một bộ sưu tập từ danh sách bộ sưu tập, mục bên phải giao diện sẽ cập nhật và hiển thị danh sách các bài viết của bộ sưu tập đó. Thiết kế được chia phần rõ ràng, đồng thời khi muốn xem chi tiết một bộ sưu tập người dùng không cần phải điều hướng sang một giao diện khác là một điểm nổi bật của thiết kế này.



Hình 4.12: Thiết kế giao diện: Màn danh sách bài viết



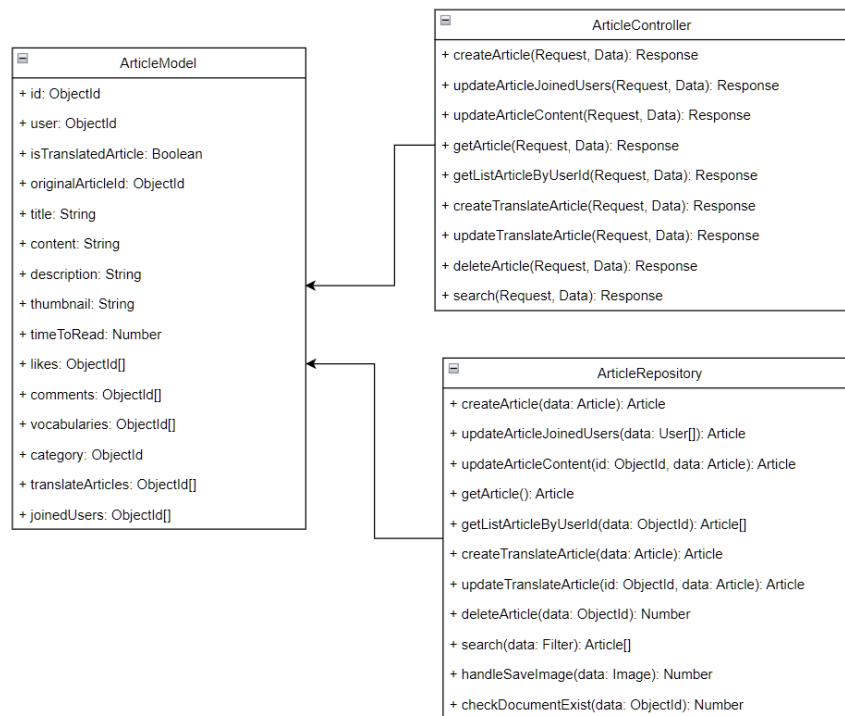
Hình 4.13: Thiết kế giao diện: Màn chi tiết bài viết



Hình 4.14: Thiết kế giao diện: Màn chi tiết bộ sưu tập

4.2.2 Thiết kế lớp

Hình 4.15 mô tả mô tả các lớp chủ đạo cho các chức năng liên quan tới quản lý bài viết/bài dịch của người dùng:



Hình 4.15: Thiết kế lớp cho chức năng liên quan tới bài viết

Chi tiết về thuộc tính và phương thức của các lớp được mô tả từ Bảng 4.2 đến Bảng 4.4:

Bảng 4.2: Đặc tả lớp ArticleModel

Tên trường	Kiểu dữ liệu	Mô tả
user	ObjectId	Người tạo
rootArticleId	ObjectId	Id bài viết gốc
isTranslated	Boolean	Xác định bài viết hay bài dịch
title	String	Tiêu đề
content	String	Nội dung
description	String	Mô tả
category	String	Danh mục

thumbnail	String	Ảnh thumbnail bài viết
timeToRead	Number	Thời gian đọc
likes	ObjectId[]	Danh sách lượt thích
comments	ObjectId[]	Danh sách bình luận
vocabularies	ObjectId[]	Danh sách từ vựng
createdAt	Timestamp	Thời gian tạo
updatedAt	Timestamp	Thời gian cập nhật

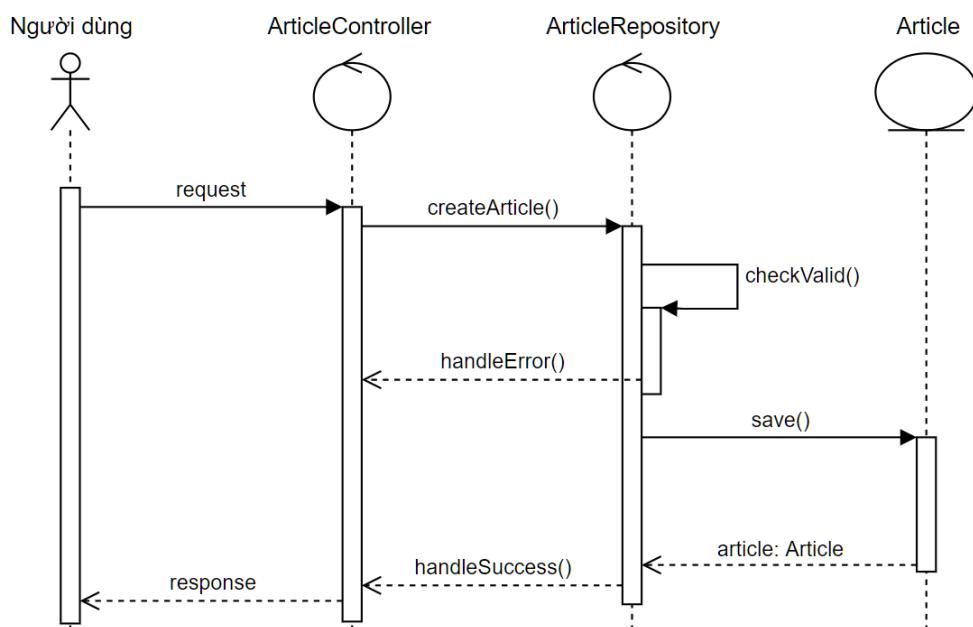
Bảng 4.3: Đặc tả lớp ArticleController

Tên phương thức	Tham số	Ý nghĩa
createArticle	Request, Data	Xử lý yêu cầu tạo bài viết
updateArticleContent	Request, Data	Xử lý yêu cầu cập nhật bài viết
updateArticleJoinedUsers	Request, Data	Xử lý yêu cầu cập nhật người tham gia bài viết
getArticle	Request, Data	Xử lý yêu cầu lấy chi tiết bài viết
getListArticleByUserId	Request, Data	Xử lý yêu cầu lấy danh sách bài viết của người dùng
createTranslateArticle	Request, Data	Xử lý yêu cầu tạo bài dịch
updateTranslateArticle	Request, Data	Xử lý yêu cầu cập nhật bài dịch
deleteArticle	Request, Data	Xử lý yêu cầu xóa bài viết/bài dịch
search	Request, Data	Xử lý yêu cầu tìm kiếm bài viết/bài dịch

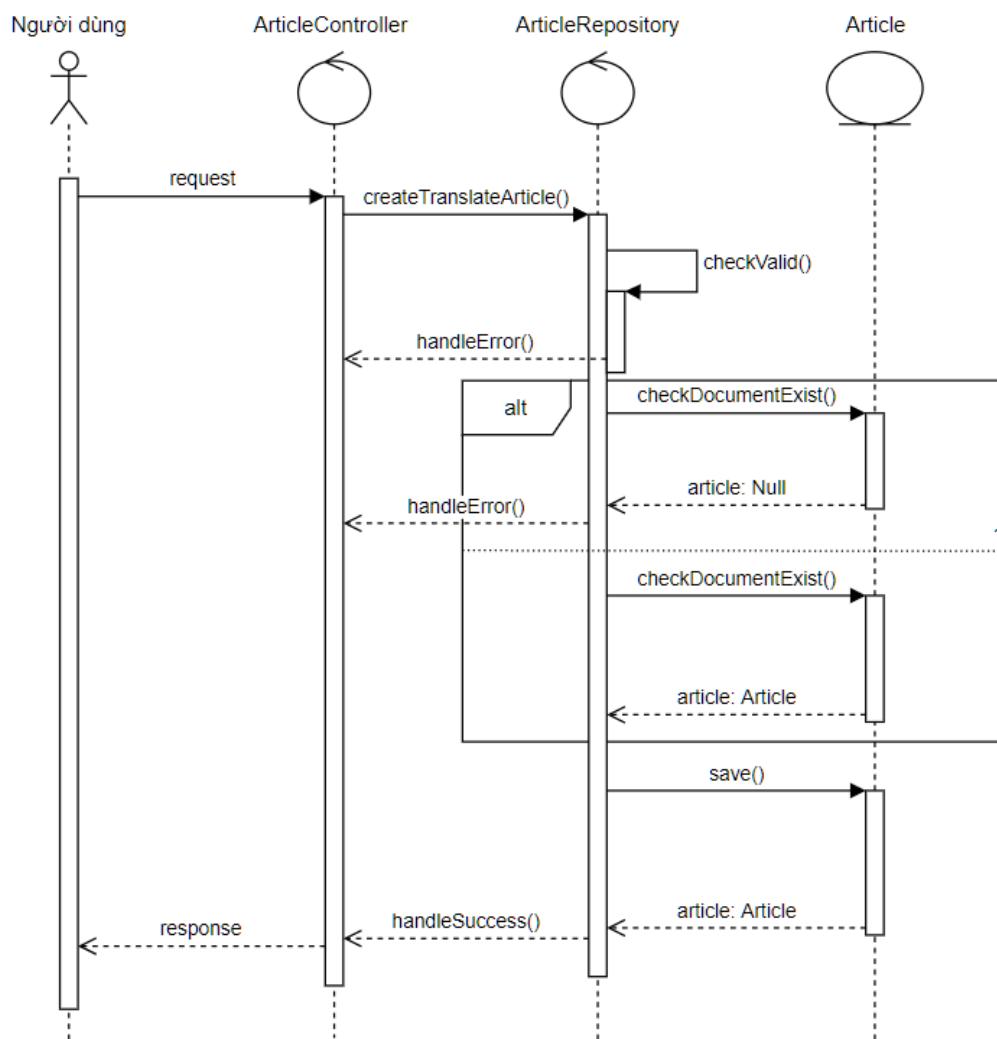
Bảng 4.4: Đặc tả lớp ArticleRepository

Tên phương thức	Tham số	Ý nghĩa
createArticle	data: Article	Tạo bài viết
updateArticleContent	data: User[]	Cập nhật bài viết
updateArticleJoinedUsers	id: ObjectId, data: Article	Cập nhật người tham gia bài viết
getArticle	id: ObjectId	Lấy chi tiết bài viết
getListArticleByUserId	data: ObjectId	Lấy danh sách bài viết của người dùng
createTranslateArticle	data: Article	Tạo bài dịch
updateTranslateArticle	id: ObjectId, data: Article	Cập nhật bài dịch
deleteArticle	id: ObejctId	Xóa bài viết/bài dịch
search	filter: Filter	Tìm kiếm bài viết/bài dịch
handleSaveImage	image:Image	Lưu dữ liệu ảnh
checkDocumentExist	id: ObjectId	Kiểm tra một document với id cụ thể đã tồn tại chưa

Để minh họa cho các thiết kế lớp đã trình bày ở trên, dưới đây tôi xin đưa ra biểu đồ trình tự trình bày luồng truyền thông điệp giữa các lớp. Biểu đồ trình tự cho 2 chức năng tạo bài viết và tạo bài dịch được trình bày trong Hình 4.16 và Hình 4.17.



Hình 4.16: Biểu đồ trình tự thiết kế lớp: Tạo bài viết



Hình 4.17: Biểu đồ trình tự thiết kế lớp: Tạo bài dịch

4.2.3 Thiết kế cơ sở dữ liệu

Hệ thống cơ sở dữ liệu của hệ thống được thiết kế gồm những models sau:

- **User Model:** Model lưu thông tin người dùng

Bảng 4.5: Thiết kế database:Cấu trúc user model

Tên trường	Kiểu dữ liệu	Mô tả	Required
name	String	Tên người dùng	Yes
account	String	Tên tài khoản người dùng	Yes
password	String	Mật khẩu đăng nhập	Yes
fullName	String	Tên đầy đủ	No
phoneNumber	String	Số điện thoại	No
birthday	String	Ngày tháng năm sinh	No
gender	String	Giới tính	No
email	String	Địa chỉ email	No
avatar	String	Avatar	No
role	String	Vai trò	Yes
isDelete	Number	Tài khoản đã xóa	Yes
rfToken	String	Refresh Token	No
updatedAt	Timestamp	Thời gian update gần nhất	Yes

- **Article Model:** Model lưu thông tin bài viết/bài dịch

Bảng 4.6: Thiết kế database: Cấu trúc article model

Tên trường	Kiểu dữ liệu	Mô tả	Required
user	ObjectId	Người tạo	Yes
rootArticleId	ObjectId	Id bài viết gốc	Yes

isTranslated	Boolean	Xác định bài viết hay bài dịch	Yes
title	String	Tiêu đề	Yes
content	String	Nội dung	Yes
description	String	Mô tả	Yes
category	String	Danh mục	Yes
thumbnail	String	Ảnh thumbnail bài viết	Yes
timeToRead	Number	Thời gian đọc	No
likes	ObjectId[]	Danh sách lượt thích	No
comments	ObjectId[]	Danh sách bình luận	No
vocabularies	ObjectId[]	Danh sách từ vựng	No
createdAt	Timestamp	Thời gian tạo	Yes
updatedAt	Timestamp	Thời gian cập nhật	Yes

- **Category Model:** Model lưu thông tin danh mục bài viết

Bảng 4.7: Thiết kế database: Cấu trúc category model

Tên trường	Kiểu dữ liệu	Mô tả	Required
name	String	Tên danh mục	Yes
slug	String	Tên viết tắt danh mục	Yes
isCanUse	Boolean	Có thể sử dụng khi tạo bài viết	Yes
createdAt	Timestamp	Thời gian tạo	Yes
updatedAt	Timestamp	Thời gian cập nhật	No

- **Series Model:** Model lưu thông tin bộ sưu tập

Bảng 4.8: Thiết kế database: Cấu trúc series model

Tên trường	Kiểu dữ liệu	Mô tả	Required
user	ObjectId	Id người tạo	Yes
seriesName	String	Tên bộ sưu tập	Yes
thumbnail	String	Ảnh thumbnail bộ sưu tập	No
mode	String	Chế độ private hoặc public của bộ sưu tập	Yes
articles	ObjectId[]	Danh sách bài viết	No
createdAt	Timestamp	Thời gian tạo	Yes
updatedAt	Timestamp	Thời gian cập nhật	No

Ngoài các model chính đã mô tả ở trên, chi tiết các model còn lại được mô tả trong Bảng 4.9

Bảng 4.9: Thiết kế database: Cấu trúc các models còn lại trong database

Tên model	Mô tả
Comment	Thông tin bình luận
Like	Thông tin tương tác thích
Dislike	Thông tin tương tác không thích
Vocabulary	Thông tin từ vựng
Notify	Thông tin thông báo

4.3 Xây dựng ứng dụng

4.3.1 Thư viện và công cụ sử dụng

Danh sách các thư viện và công cụ được sử dụng khi xây dựng hệ thống được mô tả trong Bảng 4.10:

Bảng 4.10: Danh sách công cụ và thư viện sử dụng

Mục đích	Công cụ	Thông tin
Trình soạn thảo	Visual Studio Code	https://code.visualstudio.com/ Version: 1.69.1
Thư viện Javascript phát triển Front-end	ReactJS	https://reactjs.org/ Version: 16.14.0
Thư viện phát triển giao diện	Ant Design	https://ant.design/ Version: 4.22.0
Thư viện quản lý trạng thái ứng dụng	Redux	https://react-redux.js.org/ Version: 7.2.6
Thư viện định tuyến trong React	React Router	https://reactrouter.com/ Version: 5.3.3
Thư viện phát triển giao diện	Ant Design	https://ant.design/ Version: 4.22.0
Thư viện chuyển đổi đa ngôn ngữ	i18n	https://www.i18next.com/ Version: 11.18.3
Môi trường phát triển Back-end	NodeJS + Npm	https://nodejs.org/en/ Version: 16.14.3 https://www.npmjs.com/ Version:
Framework phát triển Back-end	ExpressJS	https://expressjs.com/ Version: 5.x
Thư viện Javascript phát triển Front-end	ReactJS	https://reactjs.org/ Version: 16.14.0
Công cụ quản lý và kiểm thử APIs	Postman	https://www.postman.com/ Version: Lastest Version
Thư viện hỗ trợ lập trình với MongoDB	Mongoose	https://mongoosejs.com/ Version: 5.13.14

Dịch vụ quản lý và triển khai cơ sở dữ liệu MongoDB nền tảng điện toán đám mây	MongoDB Atlas	https://cloud.mongodb.com
Công cụ trực quan hóa GUI cho MongoDB	MongoDB Compass	https://mongodb.com/product/compass Version:
Công cụ đóng gói và deploy (triển khai) hệ thống	Docker	https://www.docker.com/ Version:
Môi trường triển khai hệ thống	Ubuntu Server 20.04	https://releases.ubuntu.com/20.04/ Version: 20.04

4.3.2 Kết quả đạt được

Sau quá trình xây dựng và triển khai, hệ thống cộng đồng chia sẻ bản dịch văn bản tiếng Nhật đã được hoàn thành đúng với mục tiêu đề ra. Các thông tin tổng quan về hệ thống được tổng hợp thể hiện trong Bảng 4.11

Bảng 4.11: Thống kê thông tin tổng quan hệ thống

Thông tin	Thông kê
Tên hệ thống	Sunshare
Địa chỉ IP	http://137.184.59.10/
Dung lượng mã nguồn và thư viện	750MB
Số component Front-end	45
Số APIs	52
Số models trong cơ sở dữ liệu	12

4.3.3 Minh họa các chức năng chính

Chức năng xem danh sách bài viết

Hình 4.18 minh họa chức năng xem danh sách bài viết tại Trang chủ. Khi người dùng truy cập màn hình trang chủ, mặc định hệ thống hiển thị tab “List actions”, hiển thị danh sách các bài viết theo từng danh mục (category). Mỗi danh mục hiển thị tối đa 8 bài viết. Trong trường hợp số lượng bài viết của một danh mục lớn hơn 8, người dùng có thể duyệt danh sách bài viết của danh mục đó thông qua chức năng phân trang. Người dùng cũng có thể sử dụng chức năng tìm kiếm theo bộ lọc là danh mục để tìm kiếm bài viết phù hợp.

Chức năng xem chi tiết bài viết

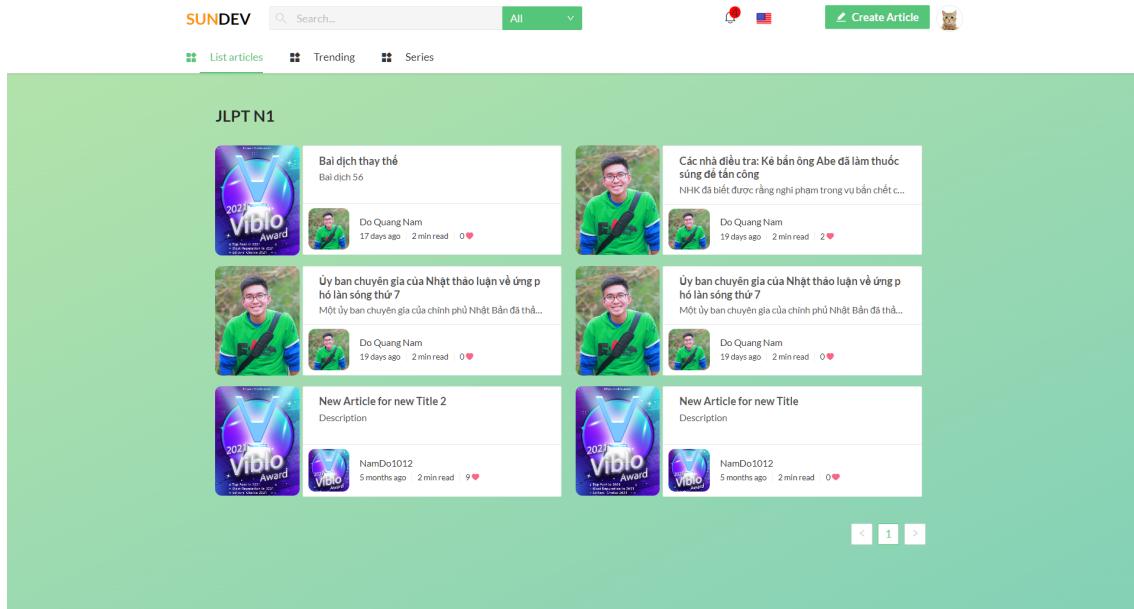
Khi lựa chọn một bài viết từ danh sách bài viết, giao diện chi tiết bài viết được hiển thị như Hình 4.19 và Hình 4.20. Tại đây người dùng có thể duyệt nội dung bài viết, xem các thông tin liên quan như từ vựng, thông tin tác giả, danh sách bài dịch ... Trường hợp đã đăng nhập vào hệ thống, người dùng có thể tương tác với bài viết thông qua các chức năng: tương tác thích, bình luận, thêm bài viết vào bộ sưu tập,...

Chức năng tạo bài viết mới

Người dùng có thể tạo bài viết cho riêng mình thông qua chức năng tạo bài viết mới. Giao diện tạo bài viết mới được hiển thị như Hình 4.21. Sau khi nhập đầy đủ các thông tin về bài viết như tiêu đề, nội dung, danh mục,... người dùng có thể tạo bài viết mới. Ngoài ra, thông qua chức năng mời chỉnh sửa, người dùng có thể cấp quyền chỉnh sửa bài viết cho nhiều người dùng khác trong hệ thống.

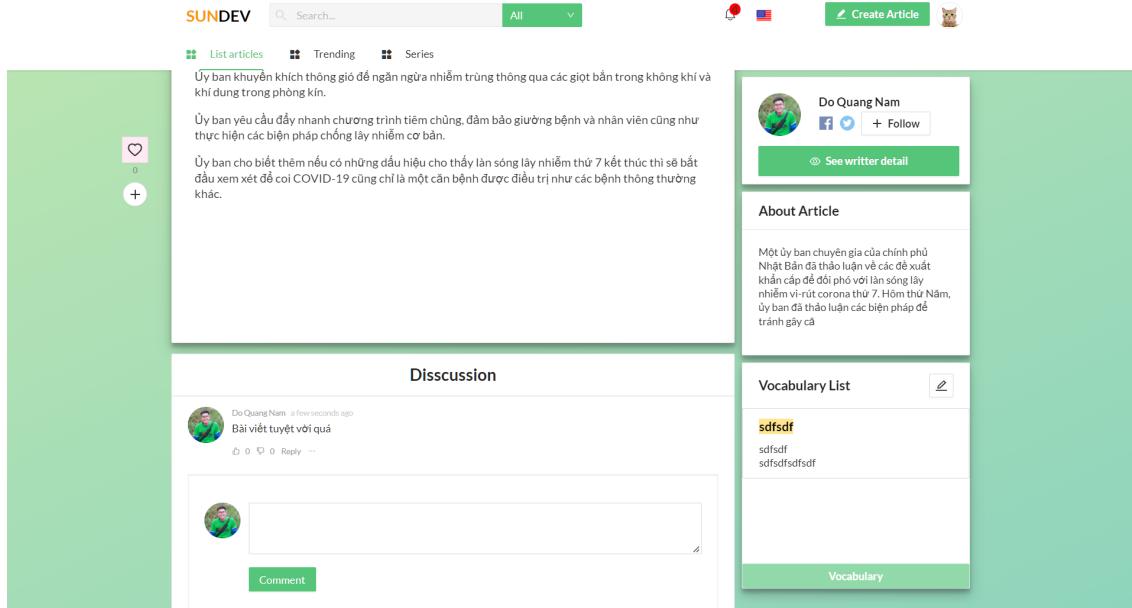
Chức năng quản lý thông tin cá nhân

Người dùng có thể quản lý và cập nhật thông tin cá nhân, thông tin tài khoản khi cần thiết. Giao diện cập nhật thông tin cá nhân và thay đổi mật khẩu được hiển thị như Hình 4.22 và Hình 4.23.

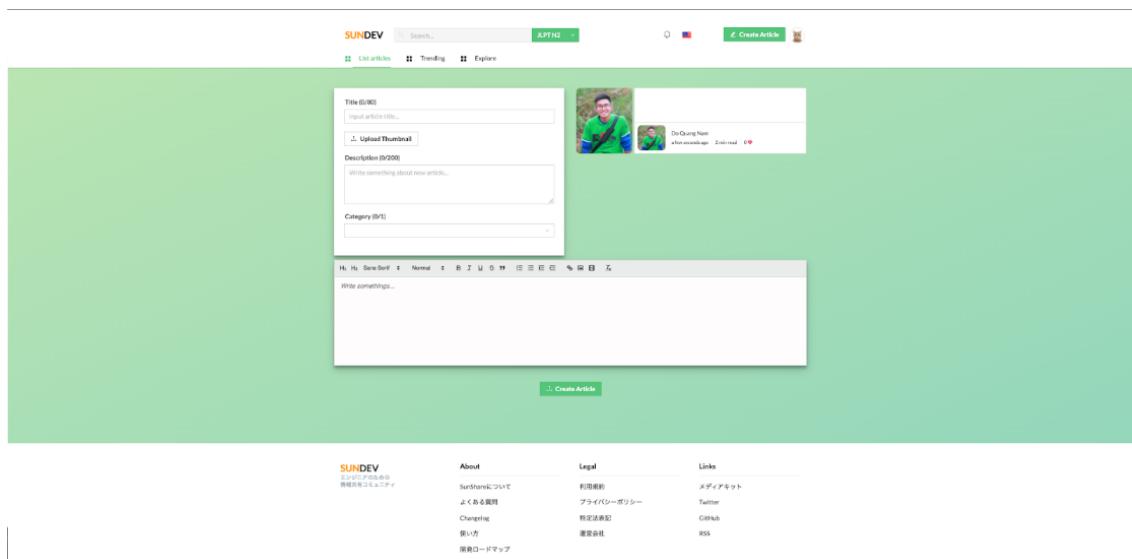


Hình 4.18: Giao diện danh sách bài viết

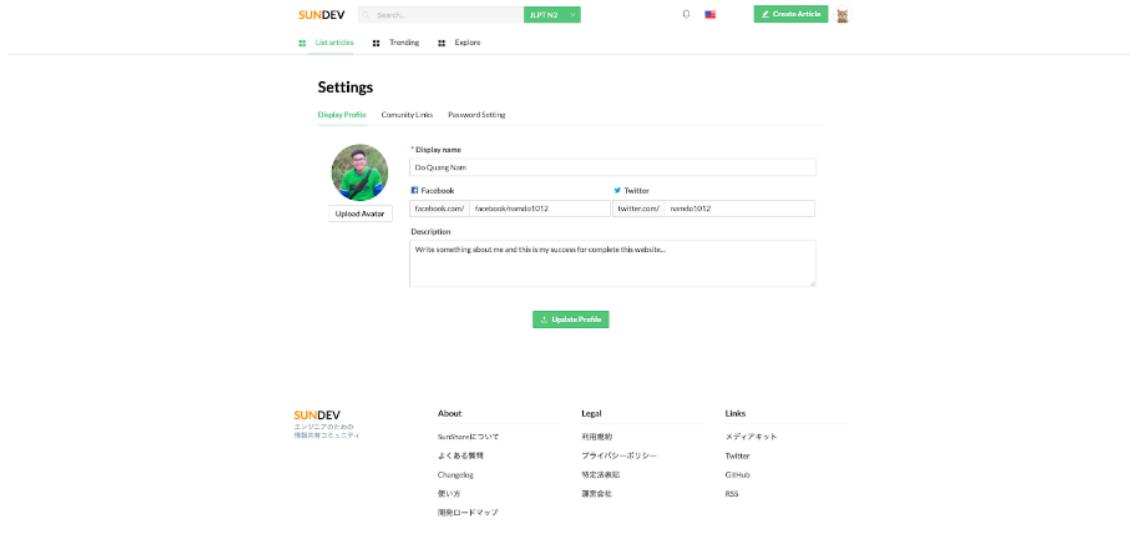
Hình 4.19: Giao diện chi tiết bài viết (1)



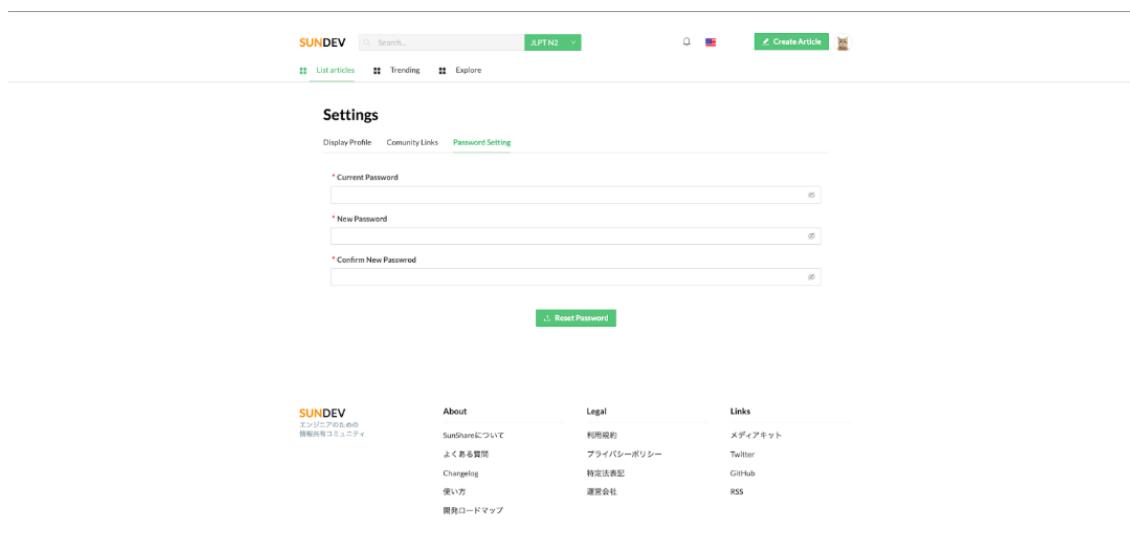
Hình 4.20: Giao diện chi tiết bài viết (2)



Hình 4.21: Giao diện tạo bài viết mới



Hình 4.22: Giao diện chỉnh sửa thông tin cá nhân



Hình 4.23: Giao diện cập nhật mật khẩu

4.4 Kiểm thử

Để thực hiện kiểm thử cho hệ thống, tôi đã sử dụng kỹ thuật kiểm thử hộp đen. Kiểm thử hộp đen là một phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các tester kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp.

Ở đây tôi xin đưa ra các yêu cầu đầu vào (được mô tả trong mục “kịch bản kiểm thử”) sẽ được đưa vào “hộp đen”, và chỉ chú ý tới kết quả trả về có đạt yêu cầu hay chưa (được mô tả trong mục “kết quả mong đợi”).

Kịch bản cho một số tính năng quan trọng của hệ thống được mô tả như trong Bảng 4.12.

Bảng 4.12: Kịch bản kiểm thử hệ thống

STT	Chức năng	Kịch bản kiểm thử	Kết quả mong đợi
1	Xem danh sách bài viết tại trang chủ	1. Người dùng chưa đăng nhập, truy cập vào màn hình trang chủ. 2. Tại danh mục có phân trang, chọn một trang bất kỳ. 3. Ấn vào một bài viết bất kỳ. 4.1. Nhập dưới 2 ký tự vào ô tìm kiếm. 4.2. Nhập trên 2 ký tự vào ô tìm kiếm. 4.3. Lựa chọn bộ lọc tìm kiếm theo danh mục. 4.4. Ấn vào một bài viết trong danh sách kết quả tìm kiếm.	1. Hiển thị giao diện danh sách bài viết tại trang chủ. 2. Hiển thị đủ danh sách bài viết tại trang đã lựa chọn của danh mục đã chọn. 3. Chuyển hướng tới giao diện chi tiết bài viết đã chọn. 4.1. Không thực hiện tìm kiếm. 4.2. Thực hiện tìm kiếm mỗi khi người dùng nhập xong chuỗi ký tự và dừng trong khoảng 0.3s. 4.3. Thực hiện tìm kiếm theo danh mục đã chọn. 4.4. Chuyển hướng tới giao diện chi tiết bài viết đã chọn.

2	Xem chi tiết bài viết/bài dịch (Chung)	<p>1. Truy cập màn hình chi tiết bài viết bất kỳ.</p> <p>2. Ấn vào nút “See writer detail” (xem thông tin người viết).</p> <p>3. Trường hợp bài viết chưa có bình luận nào.</p> <p>4.1. Người đăng nhập là người đã tạo bài viết hoặc có quyền chỉnh sửa bài viết.</p> <p>4.2. Người đăng nhập không phải là người tạo bài viết hoặc không có quyền sửa bài viết.</p> <p>5. Tại tab danh sách bài dịch, lựa chọn một bài dịch bất kỳ.</p>	<p>1. 1. Hiển thị các thông tin chính của bài viết tương tự giao diện thiết kế.</p> <p>2. 2. Chuyển hướng tới giao diện xem thông tin người dùng.</p> <p>3. Hiển thị ảnh mặc định khi không có bình luận.</p> <p>4.1. Hiển thị nút “Edit” (sửa bài viết)</p> <p>4.2. Ấn nút “Edit”.</p> <p>5. Hiển thị nội dung bài dịch.</p>
3	Xem chi tiết bài viết/bài dịch (Chưa đăng nhập)	<p>1. Sử dụng các chức năng yêu thích, bình luận, thêm bài dịch, thêm vào bộ sưu tập...</p>	<p>1. Hiển thị thông báo “Please login to use this feature!” (yêu cầu đăng nhập để sử dụng chức năng).</p>

4	Xem chi tiết bài viết/bài dịch (Đã đăng nhập)	<p>1. Ấn nút có biểu tượng trái tim (yêu thích bài viết).</p> <p>2. Nhập bình luận và ấn nút “Comment”.</p> <p>3. Ấn nút “Add translate”.</p> <p>3.1. Ấn nút hiển thị nội dung bài viết gốc.</p> <p>3.2. Nhập đầy đủ các thông tin của bài dịch và ấn nút “Create”.</p> <p>4. Ấn nút “+” (thêm vào bộ sưu tập).</p> <p>4.1. Chọn một bộ sưu tập lưu bài viết.</p>	<p>1. Hiển thị giao diện người dùng đã yêu thích bài viết và thông báo thành công.</p> <p>2. Cập nhật danh sách bình luận của bài viết và hiển thị thông báo thành công.</p> <p>3. Hiển thị giao diện tạo bài dịch tại tab mới trong danh sách các tab bài dịch của bài viết.</p> <p>3.1. Hiển thị nội dung bài viết gốc.</p> <p>3.2. Lưu bài dịch thành công và hiển thị thông báo.</p> <p>4. Hiển thị danh sách bộ sưu tập của người dùng.</p> <p>4.1. Thêm bài viết vào danh sách bài viết của bộ sưu tập đã chọn thành công và hiển thị thông báo.</p>
---	--	---	--

5	Tạo mới bài viết	1. Nhấn nút “Create article”. (tạo bài viết mới) 2. Nhập đầy đủ thông tin bài viết và nhấn nút “Create article” 2.1.a. Nhấn nút “Invite” (thêm người chỉnh sửa) 2.1.b. Chọn người dùng muốn thêm và nhấn “Done” 2.2. Nhấn nút “Upload thumbnail” (thêm ảnh thumbnail) 2.2.b. Hoàn thành lựa chọn ảnh	1. Hiển thị giao diện tạo bài viết mới. 2. Tạo bài viết thành công và hiển thị thông báo cho người dùng. 2.1.a. Hiển thị danh sách người dùng trong hệ thống có thể thêm. 2.1.b. Hiển thị xem trước danh sách người dùng đã thêm vào bài viết. 2.2.a. Hiển thị cửa sổ cho người dùng lựa chọn ảnh từ máy tính. 2.2.b. Hiển thị xem trước ảnh thumbnail cho bài viết và thông tin ảnh đã upload.
---	------------------	---	--

Quá trình kiểm thử các chức năng chính trên được thực hiện trên 2 trình duyệt được sử dụng nhiều hiện nay là Chrome và Edge. Kết quả kiểm thử hệ thống trên 2 trình duyệt trên được mô tả trong Bảng 4.13:

Bảng 4.13: Kết quả kiểm thử trên trình duyệt máy tính

Chức năng / Nhóm chức năng	Chrome	Microsoft Edge
Xem danh sách bài viết tại trang chủ	Đạt	Đạt
Xem chi tiết bài viết / bài dịch (Chung)	Đạt	Đạt
Xem chi tiết bài viết / bài dịch (Chưa đăng nhập)	Đạt	Đạt
Xem chi tiết bài viết / bài dịch (Đã đăng nhập)	Đạt	Đạt

Tạo mới bài viết	Đạt	Đạt
------------------	-----	-----

4.5 Triển khai

Trang web của hệ thống SunShare đã được triển khai trên server của Digital Ocean. Cụ thể cấu hình của server như dưới đây:

- Địa chỉ IP: http://137.184.59.10/
- CPU: 1 nhân
- RAM: 1GB
- Bộ nhớ: 25GB
- Hệ điều hành: Ubuntu 20.04 LTS

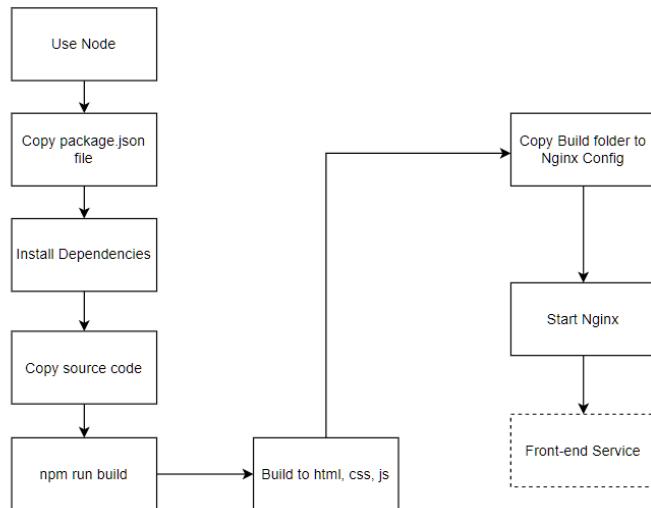
Các bước triển khai hệ thống bao gồm:

- Bước 1: Tải về các images của hệ thống Front-end và Back-end và các dịch vụ liên quan (MongoDB, Nginx, ...) từ DockerHub
- Bước 2: Khởi chạy các images của hệ thống và các dịch vụ liên quan tạo ra các container
- Bước 3: Khởi chạy các container tạo ra các dịch vụ (service). Các dịch vụ giao tiếp với nhau tạo nên hệ thống.

Quy trình khởi chạy hệ thống Front-end: Chi tiết quy trình khởi chạy hệ thống Front-end được mô tả như trong Hình 4.24

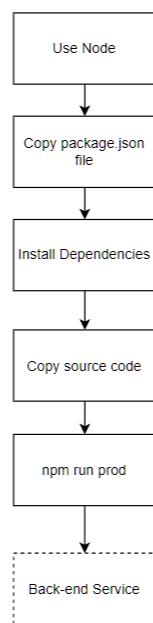
Thiết lập docker để thực hiện tạo một docker image chứa hệ thống frontend sẽ được khởi chạy theo các bước dưới đây:

- Bước 1: Khởi tạo môi trường Node.
- Bước 2: Sao chép tệp package.json từ thư mục dự án sang thư mục của container.
- Bước 3: Container cài đặt các thư viện hệ thống được định nghĩa trong tệp package.json.
- Bước 4: Sao chép mã nguồn của dự án vào container.
- Bước 5: Thực hiện tác vụ build các tệp html, css và js của hệ thống.
- Bước 6: Sao chép các tệp đã build được sang thư mục quản lý thiết lập của container Nginx.

**Hình 4.24:** Quy trình khởi chạy hệ thống Front-end

- Bước 7: Khởi tạo docker container.
- Bước 8: Khởi chạy docker container đã được tạo.

Quy trình khởi chạy hệ thống Back-end: chi tiết quy trình khởi chạy hệ thống Back-end được mô tả như trong Hình 4.25

**Hình 4.25:** Quy trình khởi chạy hệ thống Back-end

Thiết lập docker để thực hiện tạo một docker image chứa hệ thống backend sẽ được khởi chạy theo các bước dưới đây:

- Bước 1: Khởi tạo môi trường Node.
- Bước 2: Sao chép tệp package.json từ thư mục dự án sang thư mục của container.
- Bước 3: Container cài đặt các thư viện hệ thống được định nghĩa trong tệp package.json.
- Bước 4: Sao chép mã nguồn của dự án vào container.
- Bước 5: Khởi tạo docker container.
- Bước 6: Khởi chạy docker container đã được tạo.

CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

Ở chương 4 tôi đã trình bày chi tiết về các phương pháp và công cụ sử dụng khi thiết kế kiến trúc hệ thống, các thức kiểm thử và triển khai hệ thống. Trong chương này tôi sẽ trình bày về những vấn đề đã gặp phải khi xây dựng hệ thống và giải pháp giải quyết. Đồng thời tôi cũng sẽ trình bày về những nội dung chức năng của hệ thống mà tôi nghĩ rằng sẽ là điểm đặc biệt cần được chú ý của hệ thống này.

5.1 Cải thiện trải nghiệm người dùng với thiết kế giao diện responsive

5.1.1 Đặt vấn đề

Theo kết quả khảo sát của dự án 'Khảo sát dành cho các cơ quan đào tạo tiếng Nhật' được tổ chức bởi Japan Foundation - Quỹ Giao lưu Quốc tế Nhật Bản, tính đến năm 2018, Việt Nam đứng thứ 6 trên thế giới về số lượng người học tiếng Nhật với khoảng 174.500 học viên và 818 cơ quan dạy tiếng Nhật. Chiếm số lượng lớn trong số đó là các học viên thuộc độ tuổi từ 16-35 tuổi. Vì vậy, hướng tới một hệ thống có tính ứng dụng cao dành cho những người học tiếng Nhật, tôi mong muốn xây dựng một hệ thống đáp ứng được hành vi của tập người dùng này, tức là những người thuộc độ tuổi từ 16-35 tuổi.[9] [12]

Trong bối cảnh số hóa công nghệ như hiện nay, những người thuộc độ tuổi này thường sở hữu ít nhất là 1 thiết bị số, đa phần là điện thoại thông minh. Ngoài ra còn nhiều thiết bị khác cũng được sử dụng chiếm số lượng nhiều là máy tính, máy tính bảng, ... Với mục tiêu chính nhất của hệ thống là xây dựng một website cung cấp giao diện bài dịch tiếng Nhật cho người học tiếng Nhật, tức là hành vi chính của người dùng đối với hệ thống là "đọc", thì một hệ thống có thể sử dụng được với bất cứ thiết bị nào là phù hợp với tập đối tượng của hệ thống này. Nhờ vậy, người sử dụng có thể học tiếng Nhật ở bất cứ đâu với bất cứ thiết bị nào, và trải nghiệm của người dùng đối với hệ thống sẽ được nâng cao.

5.1.2 Giải pháp

Để đáp ứng được mục tiêu trên, đồ án đã đưa ra thiết kế giao diện hệ thống theo hướng responsive web [13].

Web Responsive là phong cách thiết kế trang web sao cho các nội dung có thể hiển thị tương thích trên nhiều loại thiết bị khác nhau. Nói cách khác, bố cục của trang web sẽ được tự động thay đổi, điều chỉnh để xuất hiện vừa in trên màn hình của máy tính, điện thoại hay bất kỳ thiết bị nào mà bạn sử dụng. Ngày nay, responsive đang dần trở thành một trong những yếu tố chính để đánh giá sự hiệu quả của trang web.

Hệ thống được thiết kế giao diện responsive theo chuẩn Desktop-first [14]: Thiết kế giao diện trang web tương thích với tỉ lệ giao diện máy tính, sau đó sẽ điều chỉnh, thu nhỏ các nội dung sao cho phù hợp với giao diện máy tính bảng, cuối cùng là điện thoại.

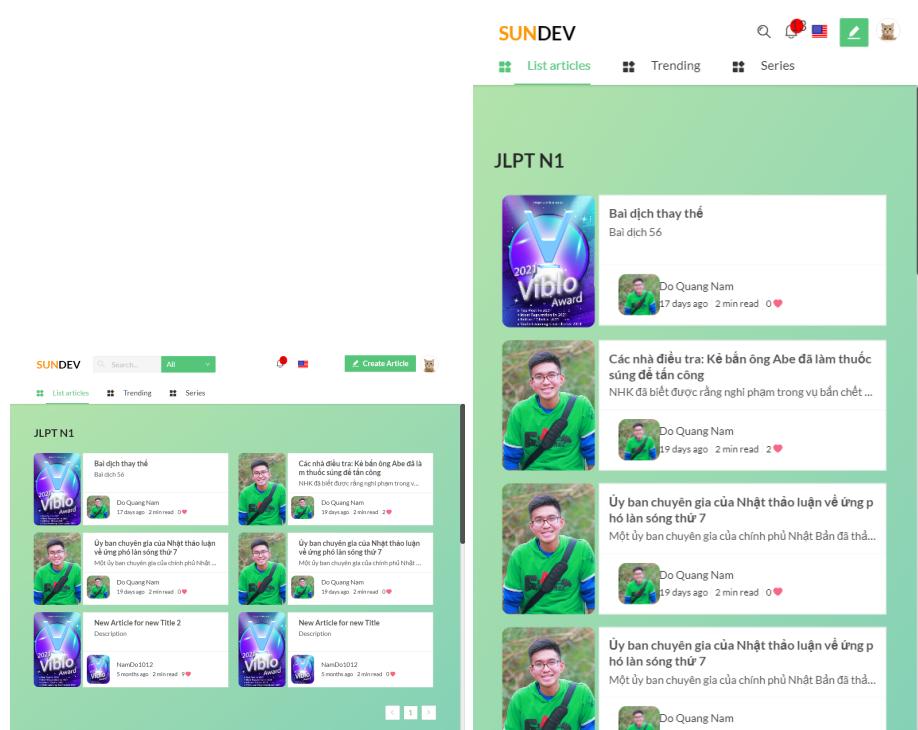
Để thiết kế responsive cho giao diện, tôi đã đưa ra một số quy chuẩn màn hình, được mô tả như trong Bảng 5.1:

Bảng 5.1: Quy chuẩn thiết kế giao diện responsive

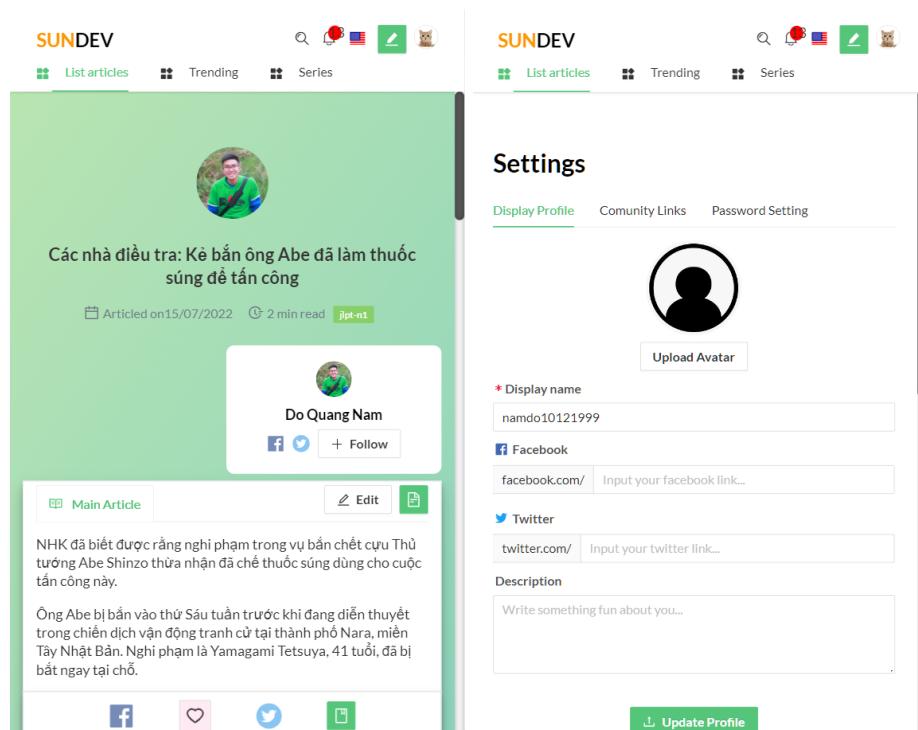
Thiết bị	Ký hiệu	Độ rộng màn hình (px)	Cỡ chữ mặc định
Desktop	@screen-xl	> 1200px	16px
Tablet-landscape	@screen-lg	992px - 1200px	16px
Tablet-portrait	@screen-md	768px - 991px	14px
Small tablet	@screen-sm	576px – 767px	12px
Mobile	@screen-xs	< 576px	12px

5.1.3 Kết quả đạt được

Dựa vào các quy chuẩn đã đưa ra và ứng dụng các quy tắc thiết kế responsive website, hệ thống đã hoàn thành được giao diện đối ứng được với các thiết bị với nhiều quy chuẩn màn hình khác nhau. Kết quả giao diện của hệ thống đáp ứng responsive được thể hiện như Hình 5.1 và Hình 5.2.



Hình 5.1: Giao diện responsive: Trang chủ



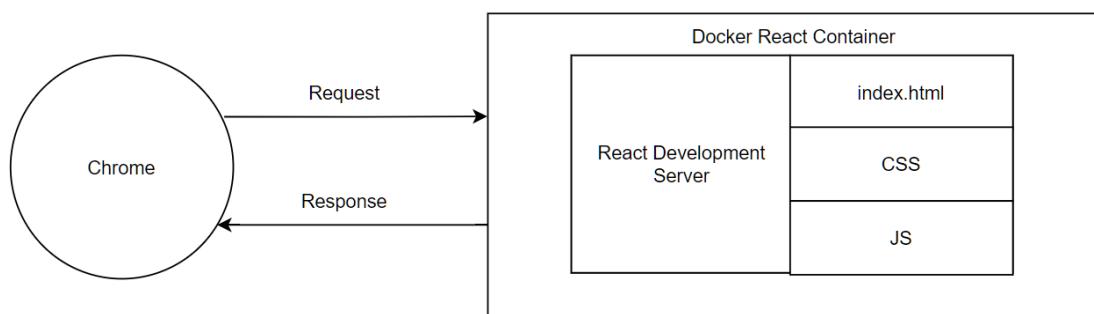
Hình 5.2: Giao diện responsive: Chi tiết bài viết và Thông tin cá nhân

5.2 Tối ưu hóa hệ thống với Nginx Server

5.2.1 Đặt vấn đề

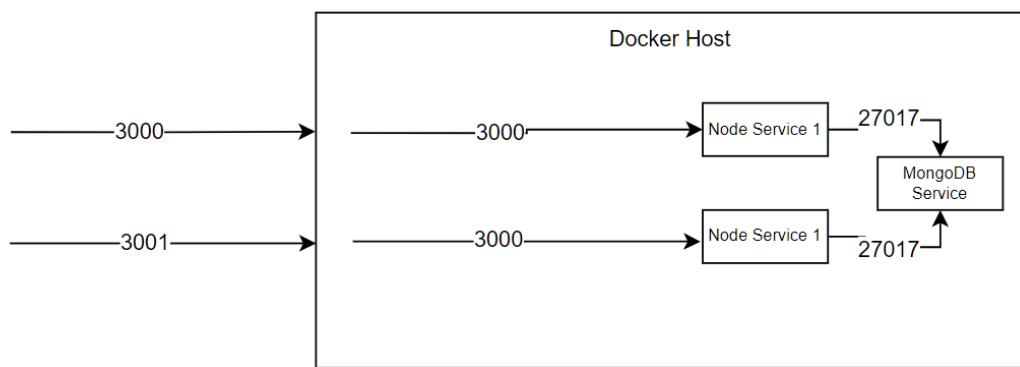
Trong quá trình phát triển, React cung cấp cho lập trình viên câu lệnh khởi chạy hệ thống sử dụng server phát triển của React được gọi là React Development Server. Cấu trúc hệ thống Front-end sử dụng React Development Server được mô tả như Hình 5.3. Tuy nhiên, khi sử dụng server phát triển sẵn có của React trong môi trường production, đối với hệ thống Front-end có 2 vấn đề:

- Thứ nhất là sử dụng React Development Server không phù hợp cho môi trường production, vì đây là máy chủ dành cho phát triển, vì vậy khi số lượng người tăng thì khả năng quá tải của hệ thống sẽ tăng lên.
- Thứ hai là React cung cấp câu lệnh giúp nhà phát triển build mã nguồn hệ thống sang các tệp html, css và js. Tuy nhiên nếu không sử dụng server của React, ta cần sử dụng một server khác để phục vụ các tệp trên khi có yêu cầu phục vụ hệ thống được gửi tới máy chủ production.



Hình 5.3: Cấu trúc hệ thống Front-end sử dụng React Development Server

Đối với Backend, khi số lượng người dùng tăng lên, nếu chỉ có một Back-end service phụ trách xử lý yêu cầu thì sẽ dễ xảy ra tình trạng hệ thống bị quá tải. Vì vậy ta cần khởi chạy thêm back-end service để có thể xử lý trong trường hợp có nhiều yêu cầu được gửi đến máy chủ trong một thời gian ngắn. Cơ bản ý tưởng sẽ được mô tả như Hình 5.4:

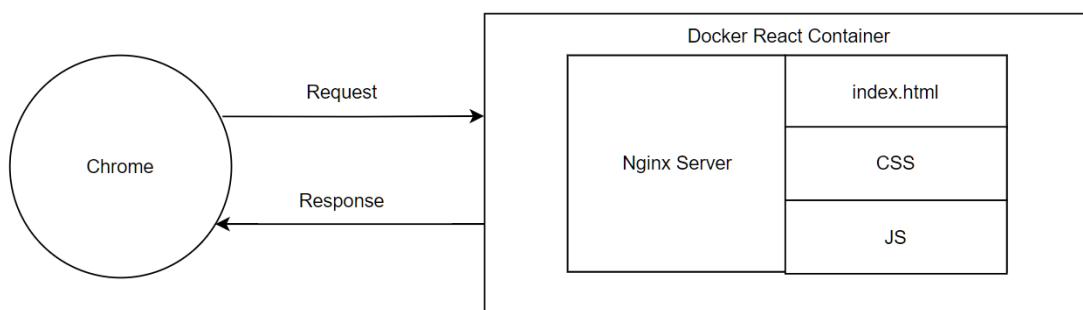
**Hình 5.4:** Ý tưởng cấu trúc cân bằng tải hệ thống Back-end

Ý tưởng trên tồn tại một vấn đề đó là cần mở nhiều port (cổng) trong máy chủ để có thể chạy nhiều dịch vụ back-end. Mỗi dịch vụ yêu cầu mở một port. Điều này là không hợp lý đối với hệ thống vì tồn tại các vấn đề sau:

- Hệ thống Front-end không thể biết được có bao nhiêu dịch vụ back-end đang được khởi chạy và chạy ở những port nào để có thể gửi yêu cầu tới
- Liên quan tới vấn đề bảo mật hệ thống vì nếu ta chuyển xử lý sang hệ thống Front-end, Front-end cần biết về số lượng và định danh của các dịch vụ back-end đang tồn tại.
- Khó quản lý số lượng và định danh của các dịch vụ back-end đang được khởi chạy trong trường hợp có quá nhiều dịch vụ và cần mở nhiều port để khởi chạy số lượng lớn dịch vụ back-end

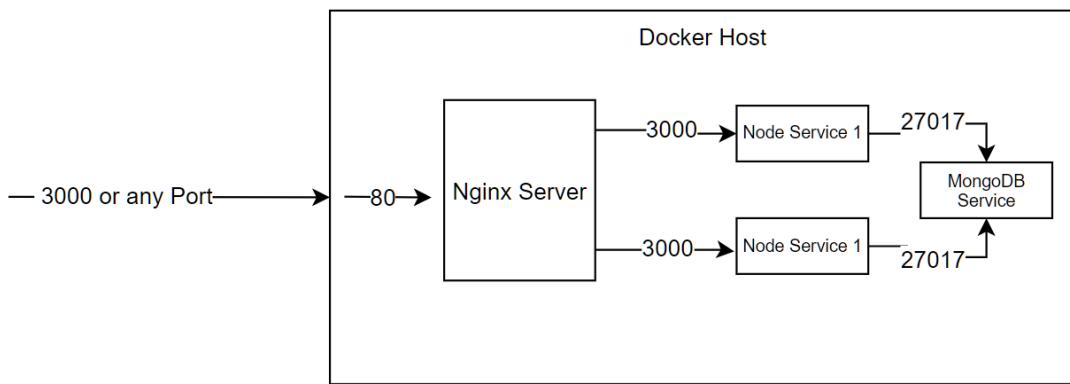
5.2.2 Giải pháp

Đối với hệ thống Front-end, ta thay thế React Development Server với Nginx theo cấu trúc được mô tả như trong Hình 5.5:

**Hình 5.5:** Cấu trúc hệ thống Front-end sử dụng Nginx Server

Ta cần cấu hình Docker để sử dụng dịch vụ máy chủ nginx server trong hệ thống. Máy chủ Nginx mang nhiều ưu điểm thích hợp để sử dụng làm một máy chủ production nên nhờ vậy hệ thống có thể xử lý tốt hơn nhờ có máy chủ Nginx.

Đối với hệ thống Back-end, ta sử dụng Nginx để giải quyết vấn đề cân bằng tải hệ thống đã nêu trên như mô tả trong Hình 5.6:



Hình 5.6: Ý tưởng cấu trúc cân bằng tải hệ thống Back-end với Nginx Server

Ở đây, Nginx đóng vai trò là một máy chủ trung gian có nhiệm vụ nhận yêu cầu từ các máy chủ khác và điều hướng các yêu cầu đó tới với các dịch vụ back-end đang khởi chạy. Nhờ vậy ta đã giải quyết được các vấn đề đã nêu lên ở trên.

5.2.3 Kết quả đạt được

Nhờ sử dụng máy chủ Nginx, ta đã tối ưu hóa được hệ thống trên máy chủ production một cách hiệu quả hơn và đem lại các lợi ích sau:

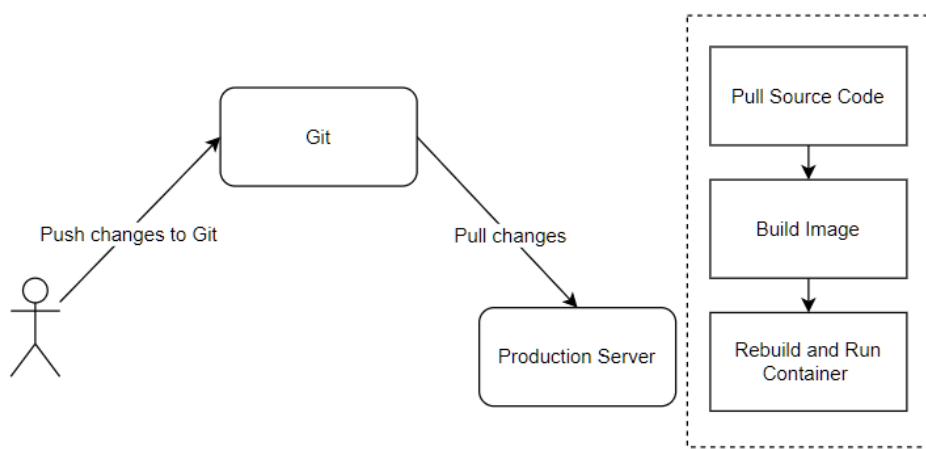
- Các tệp mã nguồn Front-end được phục vụ bởi máy chủ mang tính “production” hơn.
- Nhờ đặc điểm của máy chủ Nginx là một máy chủ trung gian có thể điều phối các yêu cầu, hệ thống Back-end được cân bằng tải tốt hơn.
- Tăng yêu tố bảo mật: Các hệ thống bên ngoài chỉ có thể giao tiếp với hệ thống Back-end qua cổng mặc định của máy chủ Nginx là cổng 80, và không thể biết được số lượng và định danh của các dịch vụ back-end đang được chạy trong hệ thống.

5.3 Tối ưu hóa quy trình phát triển và triển khai hệ thống

5.3.1 Đặt vấn đề

Ngoài việc tối ưu hóa hệ thống như đã nêu ở mục trên, việc tối ưu hóa quy trình phát triển và triển khai hệ thống cũng đóng một vai trò quan trọng giúp giảm đáng

kể công sức bảo trì và sửa chữa hệ thống.



Hình 5.7: Quy trình phát triển và triển khai hệ thống ban đầu

Quy trình phát triển ban đầu của hệ thống được mô tả như Hình 5.7. Cụ thể, khi một thay đổi được tạo ra trên mã nguồn hệ thống, ta cần push thay đổi đó lên git. Sau đó, ta cần đăng nhập vào máy chủ production, thực hiện thao tác pull các thay đổi trong mã nguồn về và bắt đầu thực hiện build docker images và khởi chạy lại docker container để khởi chạy lại hệ thống. Tuy nhiên, quy trình này tồn tại một số vấn đề liên quan tới hiệu năng của máy chủ production. Vấn đề chính là tác vụ build docker image được thực hiện ngay tại máy chủ production. Điều này làm giảm hiệu năng của máy chủ vì để build một image máy chủ cần tiêu tốn các tài nguyên hệ thống, bộ nhớ và cpu. Dẫn đến tài nguyên của máy chủ dùng cho việc xử lý các tác vụ khác như xử lý yêu cầu từ các máy chủ khác sẽ bị giảm, dẫn đến giảm hiệu năng của máy chủ production.

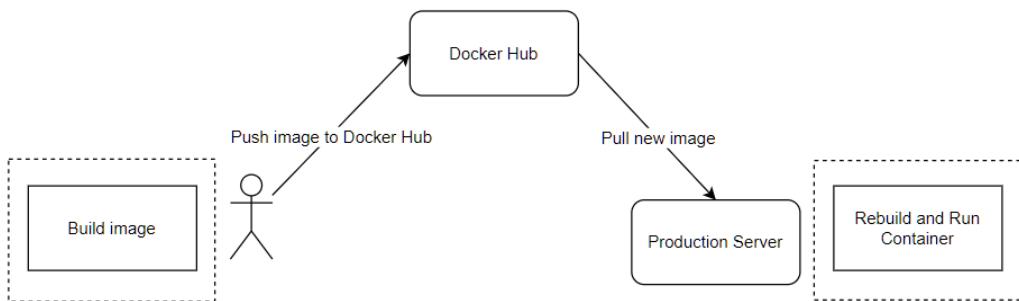
5.3.2 Giải pháp

Để giải quyết vấn đề nêu trên, hệ thống cần đưa ra một mục tiêu chính là chuyên môn hóa hệ thống. Tất cả tài nguyên của máy chủ cần được tập trung sử dụng cho việc xử lý các yêu cầu gửi tới máy chủ hay dành cho việc khởi chạy hệ thống. Phương pháp giải quyết tối đã áp dụng là chuyển tác vụ build docker image từ máy chủ production sang một máy chủ khác (máy tính local hoặc máy chủ development, ...).

Quy trình phát triển mới đã cải tiến của hệ thống được mô tả như Hình 5.8:

Quy trình trên gồm các bước như sau:

- Bước 1: Thực hiện tác vụ build image trên máy chủ development.
- Bước 2: Lưu trữ image trên dockerHub – kho lưu trữ images của Docker



Hình 5.8: Quy trình phát triển và triển khai hệ thống đã tối ưu hóa

- Bước 3: Tại máy chủ production, tải image mới về và khởi chạy để rebuild lại container.

5.3.3 Kết quả đạt được

Sau khi áp dụng quy trình phát triển trên, quy trình phát triển và triển khai hệ thống đã trở nên hiệu quả, gọn và dễ hiểu hơn. Sử dụng quy trình phát triển mới đem lại một số lợi ích sau:

- Tăng hiệu năng của máy chủ production: Chuyển công việc tiêu tốn nhiều tài nguyên máy chủ là build image sang máy chủ khác, dành tài nguyên máy chủ cho các tác vụ khác.
- Chuyên môn hóa máy chủ production: Máy chủ production chỉ có một nhiệm vụ chính là thực hiện khởi chạy hệ thống và không cần thực hiện các tác vụ xây dựng hệ thống nào.
- Tận dụng được ưu điểm của việc Docker hóa hệ thống: Hệ thống được đóng gói và lưu trữ dưới dạng image sử dụng kho lưu trữ DockerHub giúp chuyên môn hóa quá trình phát triển và quá trình triển khai.

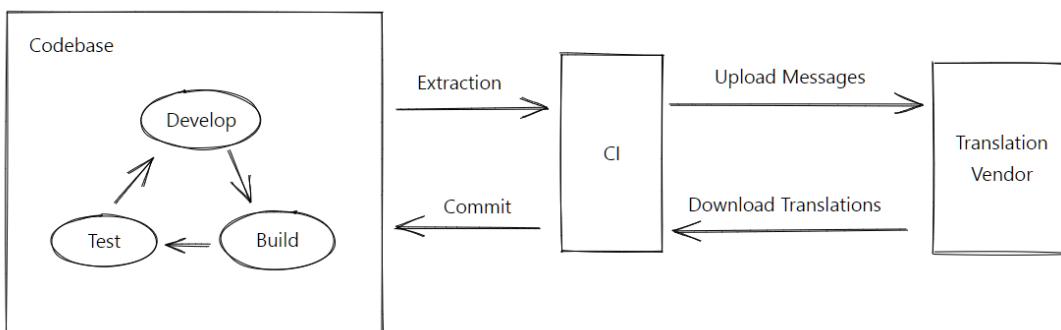
5.4 Thiết kế hệ thống đa ngôn ngữ

5.4.1 Đặt vấn đề

Như đã trình bày ở trên, hệ thống được xây dựng hướng tới đối tượng là những người học tiếng Nhật, với mục đích chia sẻ các bài dịch văn bản tiếng Nhật sang ngôn ngữ khác. Vì vậy, hiển nhiên nếu có thể thiết kế hệ thống được hiển thị với giao diện là ngôn ngữ tiếng Nhật thì trải nghiệm của người dùng sẽ mong đợi tăng lên đáng kể. Vì vậy hệ thống đã được thiết kế sử dụng 2 ngôn ngữ chính là tiếng Nhật và Tiếng Anh với mục đích: sử dụng tiếng Nhật với người dùng muốn chuyên môn hóa trải nghiệm tiếng Nhật và sử dụng tiếng Anh với người dùng chưa có nhiều vốn từ vựng tiếng Nhật và hướng tới việc quốc tế hóa trang web.

5.4.2 Giải pháp

Sau thời gian tìm hiểu, tôi đã lựa chọn sử dụng thư viện React-intl – thư viện hỗ trợ đa ngôn ngữ hóa website sử dụng thư viện ReactJS. Thư viện React-intl cung cấp các component và các APIs để định dạng ngày, tháng, số và chuỗi, bao gồm cả xử lý chuyển đổi, xử lý văn bản một cách đa dạng. Thư viện được thiết kế dựa trên thư viện FormatJS – thư viện hỗ trợ đa ngôn ngữ hóa hệ thống sử dụng Javascript. Mô tả ý tưởng về cách thức hoạt động của việc đa ngôn ngữ hóa được FormatJS định nghĩa như Hình 5.9. Chi tiết ý tưởng được trình bày rõ hơn trong tài liệu thiết kế của FormatJS [15].



Hình 5.9: Sơ đồ luồng hoạt động xử lý đa ngôn ngữ của FormatJS

Ý nghĩa của sơ đồ trên được giải thích như sau: Khi xây dựng hệ thống, những chuỗi ký tự cần được đa ngôn ngữ hóa sẽ được định nghĩa dưới dạng chuỗi ký tự mặc định, được gọi là defaultMessage. Những chuỗi ký tự này được sử dụng như ngôn ngữ mặc định của hệ thống. Khi một yêu cầu chuyển đổi ngôn ngữ được đưa ra, các bước sau sẽ được thực hiện:

- Bước 1: Extraction (trích xuất): tổng hợp tất cả các chuỗi ký tự mặc định trong hệ thống thành một tệp JSON, sẵn sàng chờ được xử lý dịch
- Bước 2: Upload Messages (Gửi lên thông điệp): gửi tệp JSON lên bộ phận xử lý dịch
- Bước 3: Download Translations (Tải về bản dịch): kết nối với bộ phận xử lý dịch lấy về tệp JSON chứa các chuỗi ký tự đã được xử lý dịch
- Bước 4: Commit: Trả về kết quả đã dịch cho hệ thống.

Dựa trên ý tưởng như trên, sử dụng thư viện React-intl ta có thể thiết kế đa ngôn ngữ hóa cho hệ thống. Về cơ bản, khi sử dụng các phương thức được cung cấp bởi React-intl, công việc ta cần làm là:

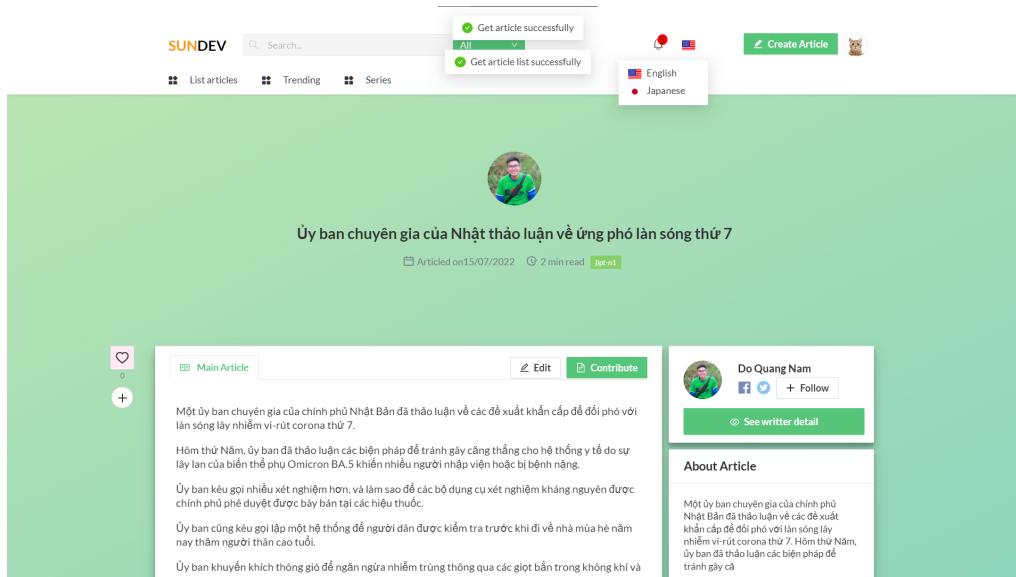
- Định nghĩa các chuỗi ký tự mặc định trong code base: hỗ trợ thư viện trích

xuất chuỗi ký tự mặc định từ hệ thống.

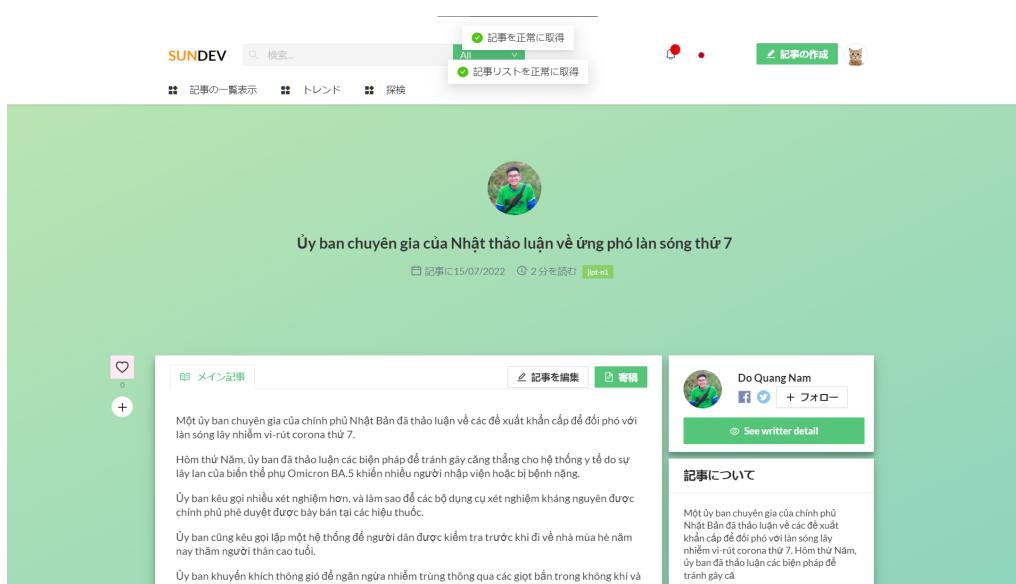
- Cung cấp tệp JSON định nghĩa kết quả dịch của chuỗi ký tự mặc định: thay cho tác vụ kết nối với bộ phận xử lý dịch để nhận về tệp JSON chứa kết quả đã dịch.

5.4.3 Kết quả đạt được

Sau khi áp dụng thiết kế đa ngôn ngữ cho hệ thống, hiện nay hệ thống đã có thể sử dụng 2 ngôn ngữ là Tiếng Anh (English) và Tiếng Nhật (Japanese). Việc chuyển đổi giữa 2 ngôn ngữ cũng rất dễ dàng làm tăng trải nghiệm của người dùng đối với hệ thống. Giao diện hệ thống khi hiện thị dưới 2 ngôn ngữ, được mô tả qua Hình 5.10 và Hình 5.11.



Hình 5.10: Giao diện đa ngôn ngữ: Tiếng Anh



Hình 5.11: Giao diện đa ngôn ngữ: Tiếng Nhật

Như vậy, Chương 5 đã trình bày về các vấn đề, giải pháp, các đóng góp và ưu điểm nổi bật của hệ thống. Các vấn đề và giải pháp được đưa ra đều gắn liền với thực tế và nhằm mục đích cải thiện chất lượng của hệ thống, nâng cao trải nghiệm của người dùng, nâng cao hiệu suất phát triển và bảo trì hệ thống. Đây đều là những yêu cầu thiết yếu trong quá trình phát triển phần mềm. Trong Chương 6, tôi xin tổng kết lại và đưa ra hướng phát triển tiếp theo của đồ án.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Nhờ sự giúp đỡ tận tình của TS. Đỗ Quốc Huy trong suốt quá trình làm đồ án tốt nghiệp, tôi đã hoàn thành hệ thống cộng đồng chia sẻ bài dịch tiếng Nhật SunShare với đầy đủ tính năng cần thiết và đáp ứng các yêu cầu đề ra ban đầu mang các đặc điểm:

- Là một hệ thống cộng đồng chia sẻ về tài liệu và kiến thức về ngôn ngữ - cụ thể là tiếng Nhật.
- Là một hệ thống giúp người dùng có thể tìm kiếm và khảo bài dịch của người khác hoặc ngược lại có thể cung cấp các bài dịch từ tiếng Nhật sang tiếng Việt.

Đồng thời hệ thống đã giải quyết được các vấn đề tồn đọng của các hệ thống sẵn có và cung cấp thêm nhiều tính năng nổi bật: Là một hệ thống có sự phân quyền giúp ngăn chặn các nội dung rác, nội dung không lành mạnh đồng thời đảm bảo thông tin người dùng được quản lý tốt. Là một hệ thống đa ngôn ngữ phù hợp với đối tượng người dùng hệ thống đang hướng tới là những người đang học tiếng Nhật. Là một hệ thống có sự tương tác giữa người với người, không đơn thuần chỉ là một hệ thống dịch tài liệu thuần. Mang tính năng responsive, có thể sử dụng được ở nhiều thiết bị với độ rộng màn hình khác nhau, giúp tăng trải nghiệm của người dùng

Thông qua quá trình xây dựng và phát triển đồ án, tôi đã học hỏi thêm được rất nhiều điều và rút ra được cho mình thêm nhiều bài học kinh nghiệm:

- Thiết kế giao diện cần phù hợp với yêu cầu người dùng, thiết kế đẹp là chưa đủ, cần đáp ứng hành vi của người dùng.
- Cần đặt mục tiêu đáp ứng yêu cầu và trải nghiệm của người dùng lên là yếu tố quan trọng nhất chứ không phải số lượng tính năng. Dù hệ thống có nhiều tính năng nhưng nếu không đem lại trải nghiệm tốt cho người dùng thì hệ thống cũng không được sử dụng.
- Quá trình xây dựng và phát triển một hệ thống cần lên kế hoạch và bám sát theo một quy trình để đảm bảo chất lượng sản phẩm.
- Việc phát triển sản phẩm là quan trọng nhưng việc bảo trì hệ thống trong tương lai cũng là một yếu tố không thể bỏ qua.

6.2 Hướng phát triển

Bên cạnh những tính năng nổi bật đã được phát triển như trên, thông qua tham khảo ý kiến của TS. Đỗ Quốc Huy và của những người dùng thử hệ thống khác, có thể thấy hệ thống vẫn còn tồn tại nhiều hạn chế và cần tiếp tục cải thiện trong tương lai:

- Chức năng của tác nhân quản trị viên vẫn còn hạn chế, không đóng nhiều vai trò trong hệ thống
- Cần có chức năng phê duyệt mỗi khi một bài viết hoặc bài dịch được người dùng tạo mới nhằm mục đích hạn chế các nội dung không phù hợp với yêu cầu của hệ thống
- Tài liệu có thể dịch của hệ thống hiện tại chỉ là văn bản, cần thêm các chức năng hỗ trợ dịch các dữ liệu khác như dữ liệu âm thanh, dữ liệu video,

Các góp ý và phản hồi trên đều là những đánh giá thực tế và hợp lý của mọi người dưới vai trò là những người trải nghiệm sản phẩm hệ thống. Vì vậy, để tiếp tục phát triển nhằm mục đích cải thiện hệ thống phù hợp hơn với yêu cầu của người dùng và cung cấp trải nghiệm người dùng tốt nhất, việc lên kế hoạch, thiết kế quy trình và phát triển cải thiện các phản hồi trên sẽ là hướng phát triển tiếp theo của đồ án. Bên cạnh việc cải thiện trải nghiệm người dùng, việc cải thiện quy trình phát triển để bảo trì và phát triển hệ thống trong tương lai hiệu quả cũng là một điều thiết yếu cần lên kế hoạch của đồ án trong tương lai. Trước mắt các điểm cần cải thiện bao gồm:

- Hệ thống chưa tự động triển khai lại sau khi có thay đổi mới trong mã nguồn mà cần khởi động lại một cách thủ công.
- Cơ sở dữ liệu MongoDB của hệ thống đang chưa được triển khai trên máy chủ.

Trên đây là tổng kết lại những điểm đã hoàn thành và kế hoạch phát triển trong tương lai của đồ án. Với các tính năng mà hệ thống đem lại, mong rằng hệ thống SunShare sẽ phát triển thành một cộng đồng mang nhiều lợi ích cho những người học tiếng Nhật. Hệ thống sẽ tiếp tục được duy trì và cải thiện trong tương lai để tiếp tục đem lại trải nghiệm tốt nhất cho người dùng và đem lại những tính năng tốt đáp ứng nhu cầu của người dùng.

TÀI LIỆU THAM KHẢO

- [1] J. Foundation, *Thông tin chung của giáo dục tiếng nhật*. [Online]. Available: <https://jpfo.org/thong-tin-chung-cua-giao-duc-tieng-nhat> (visited on 03/08/2022).
- [2] VOV2, *Việt nam đứng thứ 6 thế giới về số người học tiếng nhật*. [Online]. Available: <https://vov2.vov.vn/giao-duc-dao-tao/viet-nam-dung-thu-6-the-gioi-ve-so-nguoи-hoc-tieng-nhat-31029.vov2> (visited on 07/28/2022).
- [3] M. O. Source, *Tutorial: Intro to react*. [Online]. Available: <https://reactjs.org/tutorial/tutorial.html#what-is-react> (visited on 06/25/2022).
- [4] StrongLoop, *Fast, unopinionated, minimalist web framework for node.js*. [Online]. Available: <https://expressjs.com> (visited on 06/25/2022).
- [5] O. Foundation, *Node.js® is a javascript runtime built on chrome's v8 javascript engine*. [Online]. Available: <https://nodejs.org/en/> (visited on 06/25/2022).
- [6] MongoDB, *Mongodb: The developer data platform*. [Online]. Available: <https://www.mongodb.com/> (visited on 06/25/2022).
- [7] D. Inc., *Developers love docker. businesses trust it*. [Online]. Available: <https://docker.com> (visited on 06/30/2022).
- [8] D. Inc., *Simpler cloud. happier devs. better results*. [Online]. Available: <https://www.digitalocean.com/> (visited on 06/30/2022).
- [9] Án ngoại ngữ Quốc gia, *Tốc độ tăng số người học tiếng nhật của việt nam đứng đầu thế giới*. [Online]. Available: <http://ngoainguquocgia.moeit.gov.vn/toc-do-tang-so-nguoи-hoc-tieng-nhat-cua-viet-nam-dung-dau-the-gioi-1704905.html> (visited on 07/28/2022).
- [10] Google, *Google translate*. [Online]. Available: <https://translate.google.com/> (visited on 08/08/2022).
- [11] S. Asterisk, *Viblo asia*. [Online]. Available: <https://viblo.asia> (visited on 08/06/2022).
- [12] Kilala, *Khảo sát dành cho các cơ quan đào tạo tiếng nhật năm 2021 tại việt nam*. [Online]. Available: <https://kilala.vn/tieng-nhat/khao-sat-danh-cho-cac-co-quan-dao-tao-tieng-nhat-nam-2021-tai-viet-nam.html> (visited on 08/03/2022).

- [13] W3School, *Html responsive web design*. [Online]. Available: https://www.w3schools.com/html/html_responsive.asp (visited on 08/03/2022).
- [14] Toomba, *Mobile or desktop first design?* [Online]. Available: <https://toomba.com/en/blogs/mobile-or-desktop-first-design/> (visited on 08/03/2022).
- [15] FormatJS, *Application workflow*. [Online]. Available: <https://formatjs.io/docs/getting-started/application-workflow/> (visited on 08/03/2022).