

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN

Đàm Thị Xuân Ý - Võ Hoàng Vũ

BUILDFIFY: HỆ THỐNG WEBSITE  
BUILDER HỖ TRỢ PHÁT SINH  
MÃ NGUỒN FRONT-END

KHÓA LUẬN TỐT NGHIỆP CỦ NHÂN  
CHƯƠNG TRÌNH CỦ NHÂN TÀI NĂNG

Tp. Hồ Chí Minh, tháng 07/2023

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN

Đàm Thị Xuân Ý - 19120160  
Võ Hoàng Vũ - 19120727

**BUILDFIFY: HỆ THỐNG WEBSITE  
BUILDER HỖ TRỢ PHÁT SINH  
MÃ NGUỒN FRONT-END**

KHÓA LUẬN TỐT NGHIỆP CỦ NHÂN  
CHƯƠNG TRÌNH CỦ NHÂN TÀI NĂNG

**GIÁO VIÊN HƯỚNG DẪN**  
ThS. Lương Vĩ Minh

Tp. Hồ Chí Minh, tháng 07/2023

## NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Tp. Hồ Chí Minh, ngày ... tháng ... năm .....

## Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

## NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

Tp. Hồ Chí Minh, ngày ... tháng ... năm .....  
Giảng viên phản biện  
(Ký và ghi rõ họ tên)

# Lời cảm ơn

Sau khi hoàn thành khóa luận, nhóm chúng em xin dành lời cảm ơn chân thành và sâu sắc nhất:

Ban giám hiệu trường Đại học Khoa Học Tự Nhiên - Đại học Quốc gia thành phố Hồ Chí Minh, và ban chủ nhiệm khoa Công Nghệ Thông Tin, đã tổ chức và hỗ trợ chúng em trong quá trình triển khai và bảo vệ khóa luận. Nhờ sự tận tâm và hướng dẫn của các thầy cô, chúng em đã có một môi trường nghiên cứu thuận lợi và những điều kiện tốt nhất để hoàn thiện khóa luận.

Chúng em xin chân thành cảm ơn thầy Nguyễn Đức Huy, thầy Lương Vĩ Minh và thầy Trần Minh Triết vì đã dành thời gian và kiến thức chuyên môn của mình để theo dõi và đánh giá đề tài khóa luận của chúng em. Chúng em cũng muốn gửi lời cảm ơn đến các anh chị khóa trên cùng các bạn sinh viên trường Đại học Khoa Học Tự Nhiên, đã giúp đỡ nhóm rất nhiều trong quá trình tìm kiếm tài liệu, thu thập thông tin và thực hiện thử nghiệm.

Trong quá trình thực hiện đề tài, chúng em nhận thấy rằng còn nhiều hạn chế và thiếu sót, tuy nhiên tất cả đều đã được cải thiện đáng kể nhờ những lời dạy bảo của các thầy. Chúng em rất mong được các thầy cô chỉ bảo và đóng góp thêm để chúng em có thể chỉnh sửa, bổ sung và hoàn thiện khóa luận của nhóm.

Một lần nữa, chúng em xin chân thành cảm ơn các thầy cô đã dành thời gian và sự đồng hành để nhóm chúng em vượt qua những khó khăn và hoàn thiện khóa luận một cách tốt nhất.



**fit@hcmus**

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN

## ĐỀ CƯƠNG KHOÁ LUẬN TỐT NGHIỆP

# Buildify: Hệ thống website builder hỗ trợ phát sinh mã nguồn front-end

*Buildify: A Website Builder Approach to Front-end Source Code Generation*

## 1 THÔNG TIN CHUNG

**Người hướng dẫn:**

- ThS. Lương Vĩ Minh (Khoa Công nghệ Thông tin)

**[Nhóm] Sinh viên thực hiện:**

1. Đàm Thị Xuân Ý (MSSV: 19120160)
2. Võ Hoàng Vũ (MSSV: 19120727)

**Loại đề tài:** Nghiên cứu

**Thời gian thực hiện:** Từ 02/2023 đến 07/2023

## **2 NỘI DUNG THỰC HIỆN**

### **2.1 Giới thiệu về đề tài**

Trong thời đại số hóa hiện nay, việc sở hữu một trang web chuyên nghiệp đã trở thành một yếu tố vô cùng quan trọng trong việc quảng bá sản phẩm, dịch vụ hay thương hiệu của một doanh nghiệp hay cá nhân. Tuy nhiên, để thiết kế và xây dựng một trang web chuyên nghiệp đòi hỏi kiến thức chuyên môn về lập trình và thiết kế, gây khó khăn cho những người không có kinh nghiệm hoặc muốn tạo ra trang web nhanh chóng trong thời gian ngắn. Website builder ra đời và được đề xuất như là một giải pháp hữu hiệu và phổ biến để tạo nhanh chóng các trang web đẹp mắt mà không cần quá nhiều thời gian và chi phí. Tuy nhiên, một điểm yếu của các website builder hiện nay đó là hạn chế sở hữu mã nguồn khiến cho người dùng không thể phát triển trang web của mình một cách linh hoạt. Hơn nữa, mã nguồn phát sinh từ các website builder hiện nay thường không được tổ chức hợp lý, gây khó khăn cho việc đọc và hiểu mã nguồn. Việc tìm kiếm một giải pháp để sở hữu mã nguồn của trang web và tiếp tục phát triển trong tương lai vẫn đang là một thách thức đối với người dùng.

Để giải quyết vấn đề này, đề tài "Buildify: Hệ thống website builder hỗ trợ phát sinh mã nguồn front-end" ra đời. Với giao diện đơn giản, dễ sử dụng và thân thiện với người dùng cùng với các mẫu thiết kế website đa dạng và chất lượng, Buildify cho phép người dùng xây dựng trang web nhanh chóng và dễ dàng chỉ bằng việc kéo thả. Đặc biệt, khả năng phát sinh mã nguồn cho các trang web này mang lại cho người dùng khả năng tùy chỉnh và tái sử dụng cao. Người dùng có thể kết hợp nó với chuyên môn và kinh nghiệm của những lập trình viên để phát triển các tính năng nâng cao và phức tạp hơn cho trang web của mình.

Đề tài không chỉ giúp cho nhiều người dùng, đặc biệt là các doanh nghiệp và cá nhân có nhu cầu xây dựng trang web, tiết kiệm được thời gian và chi phí đáng kể trong việc tạo ra các trang web chất lượng cao, mà còn có tiềm năng phát triển

và ứng dụng rộng rãi trong lĩnh vực phát triển web.

## 2.2 Mục tiêu đề tài

Sau thời gian tìm hiểu và nghiên cứu lĩnh vực website builder, nhóm đã nhận thấy rằng nhiều website builder hiện tại thường tạo ra mã nguồn còn khá thô, đơn giản hoặc không thân thiện với lập trình viên khi cần phát triển tiếp trong tương lai. Ngoài ra, nhóm nhận thấy có khá ít nghiên cứu được công bố (có giải thích chi tiết) về phương pháp xây dựng một website builder hỗ trợ với việc phát sinh mã nguồn giao diện front-end dạng framework. Với mục tiêu giải quyết các vấn đề này, nhóm quyết định đề xuất đề tài nghiên cứu và phát triển hệ thống website builder hỗ trợ phát sinh mã nguồn front-end - Buildify với các mục tiêu sau:

- Mục tiêu đầu tiên của đề tài là nghiên cứu, tổng hợp và đề xuất một phương pháp xây dựng website builder hoàn chỉnh hướng đến việc phát sinh mã nguồn front-end. Nhóm sẽ đóng góp một mã nguồn mở cho phương pháp này, kèm theo những giải thích về kiến trúc và các luồng xử lý chính.
- Mục tiêu thứ hai của đề tài là xây dựng hệ thống website builder hoàn thiện với bộ tính năng đầy đủ, giao diện thân thiện và dễ sử dụng. Buildify sẽ giúp người dùng có thể nhanh chóng tạo ra trang web chỉ bằng cách kéo và thả các thành phần đã được thiết kế sẵn và chỉnh sửa các thông số thuộc tính theo mong muốn. Không chỉ hỗ trợ thiết kế về mặt giao diện, Buildify còn cho phép người dùng gắn một số sự kiện tương tác cho trang web, đồng thời hỗ trợ thêm việc triển khai các khía cạnh khác như: cơ sở dữ liệu, bộ chủ đề, phân trang và chuyển trang.
- Mục tiêu quan trọng tiếp theo của đề tài là phát triển tính năng phát sinh mã nguồn front-end với framework ReactJS từ giao diện thiết kế. Mã nguồn cần có cấu trúc tổ chức tốt, dễ đọc hiểu và cho phép tiếp tục phát triển. Việc phát sinh mã nguồn là một phần quan trọng của website builder, cho phép

người dùng xem và sửa đổi mã nguồn của trang web mà họ đã tạo. Hơn nữa, nó cho phép người dùng sở hữu mã nguồn của trang web mà họ đã thiết kế, giúp họ có thể tự tin và dễ dàng triển khai trang web trên các nền tảng khác nhau mà không phụ thuộc vào công cụ hay dịch vụ nào khác.

Nhóm sẽ áp dụng các kỹ thuật phát triển phần mềm và ứng dụng công nghệ hiện đại để đảm bảo rằng mã nguồn được phát sinh ra là thân thiện, tổ chức tốt, dễ đọc hiểu và mở rộng, từ đó giúp người dùng dễ dàng sửa đổi và phát triển trang web theo ý muốn, tiết kiệm thời gian, tăng năng suất và giảm chi phí cho người dùng trong quá trình phát triển trang web. Ngoài ra, đây sẽ là một nguồn tham khảo chất lượng cho các lập trình viên về phương pháp cài đặt và xây dựng một website builder hoàn thiện hướng đến việc phát sinh mã nguồn front-end. Từ đó, lập trình viên có thể chọn 1 trong 2 cách: tiếp tục phát triển thêm tính năng dựa trên mã nguồn mở hiện tại hoặc tham khảo về ý tưởng cài đặt để tự tổng hợp được phương pháp riêng, nhằm tạo ra một sản phẩm website builder cho mình.

Về mặt học thuật, đề tài tạo cơ hội để nhóm nghiên cứu sâu hơn về lĩnh vực website builder và phát sinh mã nguồn. Từ đó, nhóm tiến đến xây dựng một sản phẩm hoàn chỉnh, đồng thời rèn luyện và hoàn thiện các kỹ năng chuyên môn của mình trong lĩnh vực kỹ thuật phần mềm.

### 2.3 Phạm vi của đề tài

- Nội dung nghiên cứu của đề tài:
  - Công nghệ:
    - \* Front-end: NextJS, ReactJS, TypeScript, HTML, CSS, JavaScript, Material-UI, Styled-components, Ant Design, Tailwind CSS.
    - \* Back-end: Go, Bazel, gRPC, MongoDB.
    - \* DevOps: Docker, Kubernetes phục vụ việc đóng gói ứng dụng.
  - Website builder: toàn bộ lý thuyết và kiến trúc xây dựng các thành phần

của website builder, bao gồm: editor, node, element, layer, history...

- Nghiên cứu và áp dụng hệ thống phân tán:
  - \* Microservices: các services ở back-end giao tiếp với nhau bằng gRPC.
  - \* Git submodules.

- Đối tượng hướng đến:

- Bất kỳ đối tượng nào (cá nhân, doanh nghiệp vừa và nhỏ...) có nhu cầu thiết kế nhanh chóng website cho mục đích riêng với chi phí thấp.
- Những lập trình viên muốn thiết kế nhanh chóng giao diện người dùng, sau đó tiếp tục phát triển các xử lý logic phức tạp hơn dựa trên mã nguồn được sinh ra.
- Không gian: website - thiết kế trực tiếp trên giao diện desktop.
- Thời gian: 01/02/2023 - 14/07/2023

## 2.4 Cách tiếp cận dự kiến

### 2.4.1 Các phương pháp phát sinh mã nguồn front-end hiện nay

Phát sinh mã nguồn front-end hiện nay có những cách tiếp cận phổ biến sau:

- Phát sinh mã nguồn từ hình ảnh: trích xuất thông tin từ hình ảnh màn hình và phát sinh mã nguồn tương ứng.
- Phát sinh mã nguồn từ bản vẽ phác thảo: trích xuất thông tin từ các bản vẽ phác thảo để phát sinh mã nguồn.
- Phát sinh mã nguồn từ công cụ thiết kế - nổi bật với Figma: sử dụng công cụ thiết kế đồ họa Figma để thiết kế giao diện người dùng, sau đó sử dụng plugin để sinh mã nguồn.
- Phát sinh mã nguồn từ website builder: sử dụng các công cụ thiết kế trang web để tạo ra giao diện, sau đó sinh mã nguồn tương ứng.

#### **2.4.2 Ứng dụng học máy để phát sinh mã nguồn**

Nhóm đã tiến hành khảo sát các bài báo nghiên cứu khoa học về lĩnh vực phát sinh mã nguồn hiện nay, một vài bài báo có thể kể đến như: pix2code: Generating Code from a Graphical User Interface Screenshot [1], Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps [2], Generating webpages from screenshots [3].

Tuy nhiên đặc điểm chung của các cách tiếp cận trên là cần lượng data training đáng kể và chỉ đạt hiệu quả cao ở trường hợp website đơn giản, cấu trúc phổ biến. Hơn nữa, mã nguồn phát sinh có thể phức tạp, khó hiểu và thiếu tổ chức, điều này gây khó khăn cho việc sửa đổi, tùy chỉnh giao diện kết quả và phát triển thêm sau này. Vấn đề quan trọng nhất là tính cá nhân hoá của người dùng (theme, primary-color, font-family...) gần như không có.

#### **2.4.3 Sử dụng plugin để phát sinh mã nguồn từ bản thiết kế Figma**

Đối với việc phát sinh mã nguồn từ Figma, hầu hết các công cụ được phát triển là các plugin đi kèm theo Figma, một vài cái tên nổi bật [4] như Anima, TeleportHQ, Locofy, pxCode. Figma không dễ sử dụng thành thạo và để phát sinh được mã nguồn đủ tốt từ các plugins này, người dùng phải cực kỳ tỉ mỉ và tuân thủ các quy tắc (như đặt tên, nhóm các elements, phân chia layout...) và hiểu cơ chế phát sinh mã nguồn của plugin. Việc này rất tốn thời gian và không dễ để thực hiện được.

#### **2.4.4 Phát sinh mã nguồn với sự hỗ trợ từ website builder**

Nhận thấy vấn đề từ các hướng đi trên, nhóm quyết định sử dụng website builder để hỗ trợ việc phát sinh mã nguồn được chính xác, hiệu quả và đáp ứng nhu cầu dễ sử dụng, dễ tùy chỉnh của người dùng. Một số sản phẩm hoàn chỉnh đã được đưa vào thị trường như: Teleporthq [5], Builder.io [6], Quarkly.io [7],... đặc biệt là Wix [8] và Wordpress [9] đang được ưa chuộng và sử dụng khá phổ biến. Các

vấn đề nhóm nhận thấy khi trải nghiệm sử dụng các sản phẩm này:

- Giao diện: vẫn còn khó khăn, cứng nhắc trong việc kéo thả elements.
- Phát sinh mã nguồn chưa thân thiện với người dùng về mặt tổ chức mã nguồn, đặt tên... dẫn đến việc khó chỉnh sửa và phát triển thêm sau này.
- Giá cả.
- Có quá nhiều tính năng nên hiệu suất thấp, gây trở ngại khi sử dụng.

**Cách tiếp cận của nhóm để giải quyết các vấn đề trên:**

## 1. Hệ thống Buildify

- Xây dựng Buildify với giao diện đơn giản, dễ sử dụng.
- Thiết kế sẵn các elements, templates. Người dùng chỉ việc kéo thả và tùy chỉnh các thông số.
- Cho phép người dùng cấu hình theme với các thuộc tính tự định nghĩa như primary-color, font-size, font-weight, font-family, spacing, style button... và dễ dàng nhúng vào giá trị thuộc tính của elements.
- Cho phép người dùng tạo dữ liệu động và nhúng dữ liệu vào các elements.
- Cho phép thêm sự kiện cho các elements như: hover, click (navigate, scroll, hiện pop-up)...
- Cho phép thiết kế nhiều trang và chuyển trang (routing) trong website.

## 2. Phát sinh mã nguồn

- Tiến hành phát sinh mã nguồn với các thông tin nodes, pages, theme nhận được từ front-end.
- Mã nguồn phát sinh có cấu trúc của một ReactJS điển hình, và có thể phát triển mở rộng thêm cho các framework khác.
- Mỗi component sẽ có các props nhất định, giá trị truyền vào được nhận từ thông số người dùng tùy chỉnh.

- Cho phép website có nhiều trang, mỗi trang gắn với một đường dẫn duy nhất và hỗ trợ chuyển trang.
- Hướng dẫn người dùng cách chạy mã nguồn ở file README.

## 2.5 Kết quả dự kiến của đề tài

- Sản phẩm đầu ra: một sản phẩm website builder hoàn chỉnh với đầy đủ các thành phần của một trang web cơ bản như: landing page, các trang xác thực người dùng, phần quản trị người dùng (admin), đặc biệt là phần website builder và tích hợp được tính năng phát sinh mã nguồn từ giao diện thiết kế.
- Số liệu định lượng:
  - Đối với website builder:
    - \* Số lượng basic element (thành phần cơ bản), built-in template (mẫu thiết kế dựng sẵn) được thiết kế sẵn: đa dạng và đủ để thiết kế một website cơ bản như landing page.
    - \* Tùy chỉnh các thuộc tính giao diện của element: có một bộ các tùy chỉnh thuộc tính hỗ trợ người dùng thay đổi thuộc tính element phù hợp với mỗi loại element, giúp người dùng đạt được giao diện như ý muốn.
    - \* Tùy chỉnh sự kiện element: hỗ trợ tạo website động với các sự kiện cơ bản như: navigate, mở pop-up, scroll tới một element cụ thể...
    - \* Các thao tác cơ bản của editor: thêm/sửa/xoá/di chuyển các element (hỗ trợ thao tác kéo thả một selector vào editor); xem cấu trúc cây của các element (dạng Layer); hoàn tác và tái thực hiện (undo và redo) dựa trên lịch sử (history) được lưu lại...
    - \* Cơ sở dữ liệu (database): cho phép tạo và quản lý riêng cho từng dự án của từng người dùng.
    - \* Hỗ trợ lưu trữ và tải lại dữ liệu đã thiết kế của người dùng.

- Đối với chức năng phát sinh mã nguồn:
  - \* Mã nguồn ReactJS có bộ component riêng và có kiến trúc của một dự án ReactJS điển hình, dễ đọc và dễ tái sử dụng.
  - \* Code chạy không lỗi và giao diện đúng với bản thiết kế ban đầu.
  - \* Hỗ trợ phân trang và một vài sự kiện cơ bản.
  - \* Tốc độ phát sinh mã nguồn: < 3s.
- Công trình khoa học liên quan: viết một bài báo khoa học về phương pháp xây dựng hệ thống website builder hỗ trợ tự động phát sinh mã nguồn front-end.

## 2.6 Kế hoạch thực hiện

Bảng 1: Kế hoạch chi tiết khoá luận

Giai đoạn	Giai đoạn nhỏ	Vũ	Ý
Giai đoạn 1: (01/02/2023 - 01/03/2023)  Tìm hiểu và khảo sát đề tài	Không	<ul style="list-style-type: none"> <li>- Xác định đề tài và phạm vi chung của khoá luận.</li> <li>- Khảo sát nhu cầu người dùng.</li> <li>- Tìm hiểu và phân tích các nghiên cứu liên quan.</li> <li>- Tìm hiểu các sản phẩm có liên quan.</li> <li>- So sánh các giải pháp hiện tại.</li> <li>- Chọn vấn đề cần tập trung giải quyết.</li> </ul>	<ul style="list-style-type: none"> <li>- Xác định đề tài và phạm vi chung của khoá luận.</li> <li>- Khảo sát nhu cầu người dùng.</li> <li>- Tìm hiểu và phân tích các nghiên cứu liên quan.</li> <li>- Tìm hiểu các sản phẩm có liên quan.</li> <li>- So sánh các giải pháp hiện tại.</li> <li>- Chọn vấn đề cần tập trung giải quyết.</li> </ul>

<p>Giai đoạn 2: (01/03/2023 - 20/03/2023)</p> <p>Khảo sát công nghệ và lập kế hoạch</p>	<p>Không</p>	<ul style="list-style-type: none"> <li>- Tìm hiểu công nghệ front-end: React, TypeScript, các kiến thức chung và kiến trúc áp dụng cho front-end.</li> <li>- Tìm hiểu các kiến trúc xây dựng website builder hiện tại.</li> <li>- Tìm hiểu kiến trúc hỗ trợ quản lý editor.</li> <li>- Phương pháp phát sinh mã nguồn.</li> <li>- Lên ý tưởng các tính năng chính.</li> <li>- Thiết kế giao diện sản phẩm.</li> <li>- Lập kế hoạch và viết đề cương khoá luận.</li> </ul>	<ul style="list-style-type: none"> <li>- Tìm hiểu công nghệ back-end: Go, Protobuf, gRPC, Bazel, MongoDB.</li> <li>- Tìm hiểu microservices, cấu trúc monorepo.</li> <li>- Phương pháp phát sinh mã nguồn.</li> <li>- Lên ý tưởng các tính năng chính.</li> <li>- Thiết kế giao diện sản phẩm.</li> <li>- Lập kế hoạch và viết đề cương khoá luận.</li> <li>- Tìm hiểu Kubernetes và Docker để triển khai sản phẩm.</li> </ul>
---	--------------	---	--

<p>Giai đoạn 3: (21/03/2023 - 06/06/2023)</p> <p>Cài đặt sản phẩm</p>	<p>3.1</p> <p>Cài đặt</p>	<ul style="list-style-type: none"> <li>- Khởi tạo mã nguồn front-end.</li> <li>- Cài đặt các tính năng của website builder Buildify.</li> <li>- Cài đặt bộ component UI và các cài đặt cho phép chỉnh sửa tương ứng với các thuộc tính CSS (từ config sẵn có).</li> <li>- Cài đặt tính năng phát sinh mã nguồn.</li> <li>- Cài đặt tính năng cho phép người dùng tạo và quản lý cơ sở dữ liệu.</li> </ul>	<ul style="list-style-type: none"> <li>- Khởi tạo mã nguồn back-end.</li> <li>- Áp dụng microservices, gRPC.</li> <li>- Thiết lập quy trình CI/CD sử dụng GitHub Actions.</li> <li>- Hỗ trợ cài đặt các tính năng của website builder Buildify.</li> <li>- Cài đặt service phát sinh mã nguồn.</li> <li>- Cài đặt service cho phép người dùng tạo và quản lý cơ sở dữ liệu.</li> </ul>
	<p>3.2</p> <p>Kiểm thử</p>	<ul style="list-style-type: none"> <li>- Kiểm thử website builder và ghi nhận lại các lỗi liên quan.</li> <li>- Sửa lỗi giao diện và logic liên quan.</li> </ul>	<ul style="list-style-type: none"> <li>- Kiểm thử website builder và ghi nhận lại các lỗi liên quan.</li> <li>- Kiểm thử white-box các tính năng quan trọng.</li> <li>- Sửa lỗi logic.</li> </ul>

	3.3 Triển khai	<ul style="list-style-type: none"> <li>- Hỗ trợ triển khai sản phẩm cuối cùng, nghiên cứu việc thuê các server liên quan (lưu ảnh, icon, mã nguồn của người dùng...)</li> </ul>	<ul style="list-style-type: none"> <li>- Tìm hiểu và triển khai sản phẩm cuối cùng lên GKE cluster.</li> </ul>
Giai đoạn 4: (07/06/2023 - 30/06/2023) Viết báo cáo	4.1 Viết bài báo khoa học	<ul style="list-style-type: none"> <li>- Tổng hợp kết quả và nộp bài báo khoa học về chủ đề website builder hỗ trợ phát sinh mã nguồn front-end.</li> </ul>	<ul style="list-style-type: none"> <li>- Tổng hợp kết quả và nộp bài báo khoa học về chủ đề website builder hỗ trợ phát sinh mã nguồn front-end.</li> </ul>
	4.2 Báo cáo khoa luận và các tài liệu liên quan	<ul style="list-style-type: none"> <li>- Tổng hợp và liên tục hoàn thiện báo song song với việc cài đặt sản phẩm.</li> </ul>	<ul style="list-style-type: none"> <li>- Tổng hợp và liên tục hoàn thiện báo song song với việc cài đặt sản phẩm.</li> </ul>
Giai đoạn 5: (01/07/2023 - 21/07/2023) Bảo vệ khoá luận	5.1 Chuẩn bị tài liệu	<ul style="list-style-type: none"> <li>- Chuẩn bị đầy đủ các tài liệu theo yêu cầu để phục vụ việc bảo vệ khoá luận.</li> </ul>	<ul style="list-style-type: none"> <li>- Chuẩn bị đầy đủ các tài liệu theo yêu cầu để phục vụ việc bảo vệ khoá luận.</li> </ul>

	<p>5.2</p> <p>Chuẩn bị slide và lên kế hoạch thuyết trình</p>	<ul style="list-style-type: none"> <li>- Thiết kế slide bảo vệ.</li> <li>- Lên ý tưởng và tập luyện thuyết trình.</li> </ul>	<ul style="list-style-type: none"> <li>- Thiết kế slide bảo vệ.</li> <li>- Lên ý tưởng và tập luyện thuyết trình.</li> </ul>
--	---	--	--

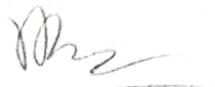
## Tài liệu

- [1] T. Beltramelli, “pix2code: Generating code from a graphical user interface screenshot,” in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pp. 1–6, 2023.
- [2] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, “Machine learning-based prototyping of graphical user interfaces for mobile apps,” *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196–221, 2023.
- [3] A. Lee, “Generating webpages from screenshots,” 2023.
- [4] Figma, “Design to code plugin.” <https://www.figma.com/community/tag/designtocode/plugins>, 2023.
- [5] Teleporthq, “A collaborative front-end platform with integrated ui development and content modelling tools..” <https://teleporthq.io>, 2023.
- [6] Builder.io. <https://www.builder.io/>, 2023.
- [7] Quarkly.io. <https://quarkly.io/>, 2023.
- [8] Wix, “Create a website without limits.” <https://www.wix.com>, 2023.
- [9] Wordpress, “Open your online store with a powerful, flexible platform designed to grow with you.” <https://wordpress.com>, 2023.

XÁC NHẬN  
CỦA NGƯỜI HƯỚNG DẪN  
(Ký và ghi rõ họ tên)

  
Phan Thanh Minh

TP. Hồ Chí Minh, ngày 31 tháng 03 năm 2023  
NHÓM SINH VIÊN THỰC HIỆN  
(Ký và ghi rõ họ tên)

  
Đam Thị Xuân Vy

  
Võ Hoàng Vũ

# Mục lục

Nhận xét của giảng viên hướng dẫn	i
Nhận xét của giảng viên phản biện	ii
Lời cảm ơn	iii
Đề cương chi tiết	iv
Mục lục	xx
Danh sách thuật ngữ	xxvii
Tóm tắt	xxxv
<b>1 Giới thiệu</b>	<b>1</b>
1.1 Đặt vấn đề . . . . .	1
1.2 Mục tiêu của đề tài . . . . .	2
1.3 Phạm vi đề tài . . . . .	4
1.4 Nội dung của luận văn . . . . .	5
<b>2 Khảo sát các công nghệ tương tự</b>	<b>7</b>
2.1 Các phương pháp phát sinh mã nguồn front-end hiện nay .	7
2.1.1 Ứng dụng học máy để phát sinh mã nguồn . . . . .	8
2.1.2 Sử dụng plugin để phát sinh mã nguồn từ bản thiết kế Figma . . . . .	8

2.1.3	Phát sinh mã nguồn với sự hỗ trợ từ website builder	9
2.2	Các hệ thống website builder liên quan . . . . .	9
2.2.1	Wix . . . . .	9
2.2.2	WordPress . . . . .	11
2.2.3	TeleportHQ . . . . .	14
2.3	Tổng kết . . . . .	16
2.3.1	Vấn đề tồn đọng của các hệ thống tương tự . . . . .	16
2.3.2	Phương pháp đề xuất . . . . .	16
<b>3</b>	<b>Phân tích yêu cầu và thiết kế hệ thống</b>	<b>19</b>
3.1	Phân tích yêu cầu . . . . .	19
3.1.1	Yêu cầu chức năng . . . . .	19
3.1.2	Yêu cầu phi chức năng . . . . .	23
3.1.3	Sơ đồ use case . . . . .	24
3.2	Thiết kế kiến trúc hệ thống . . . . .	24
3.2.1	Sơ đồ kiến trúc tổng quan . . . . .	24
3.2.2	Các công nghệ được sử dụng ở front-end . . . . .	27
3.2.3	Các mô hình và công nghệ được sử dụng ở back-end	39
3.3	Triển khai ứng dụng . . . . .	50
3.3.1	Triển khai front-end . . . . .	50
3.3.2	Triển khai back-end . . . . .	50
3.4	Thiết kế cơ sở dữ liệu . . . . .	51
<b>4</b>	<b>Cài đặt hệ thống Buildify</b>	<b>53</b>
4.1	Tổng quan giao diện Buildify . . . . .	53
4.2	Các thành phần của trang thiết kế . . . . .	56
4.3	Cài đặt trang thiết kế . . . . .	60
4.3.1	Cấu trúc editor . . . . .	61
4.3.2	Cấu trúc node . . . . .	62
4.3.3	Cấu trúc layers . . . . .	64
4.3.4	Cấu trúc history . . . . .	66
4.3.5	Kéo thả một selector vào trong editor . . . . .	67

4.3.6	Các thao tác với element trong editor . . . . .	68
4.3.7	Kéo thả một layer trên layers . . . . .	69
4.3.8	Cơ chế xác định vị trí thả . . . . .	72
4.3.9	Cơ chế undo và redo . . . . .	74
4.3.10	Cấu hình theme . . . . .	75
4.3.11	Cơ chế quản lý cơ sở dữ liệu động . . . . .	76
4.3.12	Cơ chế multiple-page và routing . . . . .	77
4.3.13	Tùy chỉnh thuộc tính cho element . . . . .	77
4.3.14	Cơ chế lưu và load trạng thái . . . . .	81
4.3.15	Cơ chế phát sinh selector và các tùy chỉnh thuộc tính tương ứng . . . . .	82
4.4	Cài đặt back-end . . . . .	86
4.4.1	Các services của back-end . . . . .	86
4.4.2	Phát sinh mã nguồn ReactJS . . . . .	86
<b>5</b>	<b>Quá trình thực hiện</b>	<b>89</b>
5.1	Tổng quan . . . . .	89
5.2	Vấn đề gặp phải khi phát triển ứng dụng . . . . .	90
5.2.1	Vấn đề chung gặp phải . . . . .	90
5.2.2	Vấn đề gặp phải khi tìm hiểu Website Builder . . . . .	91
5.2.3	Vấn đề gặp phải khi cài đặt front-end . . . . .	91
5.2.4	Vấn đề gặp phải khi cài đặt back-end . . . . .	92
5.3	Kiến thức học được khi thực hiện đề tài . . . . .	95
5.3.1	Kiến thức chung . . . . .	95
5.3.2	Kiến thức front-end . . . . .	95
5.3.3	Kiến thức back-end . . . . .	97
5.3.4	Kỹ năng . . . . .	98
5.3.5	Kiến thức về lĩnh vực website builder . . . . .	98
<b>6</b>	<b>Kết luận và hướng phát triển</b>	<b>100</b>
6.1	Kết quả đạt được . . . . .	100
6.1.1	Website Builder . . . . .	100

6.1.2	Paper . . . . .	107
6.2	Ý nghĩa của kết quả . . . . .	107
6.3	Các vấn đề tồn đọng . . . . .	108
6.3.1	Vấn đề tồn đọng của website builder . . . . .	108
6.3.2	Vấn đề tồn đọng của chức năng phát sinh mã nguồn front-end . . . . .	109
6.4	Hướng phát triển của đề tài . . . . .	110
6.4.1	Mở rộng theo hướng ứng dụng . . . . .	110
6.4.2	Mở rộng theo hướng nghiên cứu . . . . .	111
	<b>Tài liệu tham khảo</b>	<b>113</b>

# Danh sách hình

2.1	Giao diện thiết kế của Wix . . . . .	10
2.2	Giao diện thiết kế của WordPress . . . . .	12
2.3	Giao diện thiết kế của TeleportHQ . . . . .	14
3.1	Sơ đồ use case . . . . .	24
3.2	Sơ đồ kiến trúc tổng quan . . . . .	25
3.3	Sơ đồ CI/CD ở back-end . . . . .	51
3.4	Sơ đồ thiết kế cơ sở dữ liệu . . . . .	52
4.1	Giao diện của landing page . . . . .	53
4.2	Giao diện của trang projects dashboard . . . . .	54
4.3	Giao diện của trang builder . . . . .	54
4.4	Giao diện của trang quản lý thông tin cá nhân . . . . .	55
4.5	Giao diện của trang quản lý cơ sở dữ liệu . . . . .	55
4.6	Giao diện của trang đăng ký . . . . .	56
4.7	Giao diện của trang đăng nhập . . . . .	56
4.8	Các thành phần của giao diện editor . . . . .	57
4.9	Các thành phần của giao diện toolbox . . . . .	58
4.10	Các thành phần của giao diện side panel . . . . .	59
4.11	Các thành phần của giao diện header . . . . .	60
4.12	Cấu trúc của editor . . . . .	62
4.13	Cấu trúc layers . . . . .	64
4.14	Hiển thị thông tin layer . . . . .	65
4.15	Cấu trúc history . . . . .	66

4.16 Kéo thả một selector vào trong editor . . . . .	67
4.17 Di chuyển một node đến vị trí khác trong editor . . . . .	69
4.18 Luồng kéo và thả layer . . . . .	71
4.19 Kéo một layer vào ranh giới của layer khác . . . . .	71
4.20 Kéo một layer vào bên trong một layer khác . . . . .	72
4.21 Ví dụ về cơ chế hoạt động của history . . . . .	75
4.22 Giao diện quản lý theme . . . . .	76
4.23 Cấu hình CSS của font size . . . . .	79
4.24 Ví dụ trực quan về tính năng tùy chỉnh thuộc tính giao diện	80
 6.1 Giao diện thiết kế người dùng của Buildify . . . . .	101
6.2 Cấu trúc thư mục và một đoạn code ReactJS component .	101
6.3 Đoạn mã nguồn phát sinh cho Home page . . . . .	102
6.4 Giao diện chạy website trên localhost của trang web được tạo ra từ mã nguồn phát sinh . . . . .	103
6.5 Cấu trúc thư mục và một đoạn code ReactJS component của TeleportHQ . . . . .	104
6.6 Đoạn mã nguồn phát sinh cho một page của TeleportHQ .	105

# Danh sách bảng

1	Danh sách thuật ngữ chung . . . . .	xxviii
2	Danh sách thuật ngữ lĩnh vực website builder . . . . .	xxxii
3.1	So sánh monorepo với polyrepo . . . . .	41
3.2	So sánh goroutine với thread . . . . .	43
3.3	Các khái niệm liên quan đến gRPC . . . . .	44

# Danh sách mã nguồn

4.1	Hàm <code>findPosition()</code> dùng để tính toán indicator . . . . .	72
4.2	Tự động rendering JSX cho selector . . . . . . . . . . .	84

# Danh sách Thuật ngữ

Bảng 1: Danh sách thuật ngữ chung

Thứ tự	Thuật ngữ	Giải thích
1	admin	Giao diện dành cho quản trị viên quản lý và kiểm soát các hoạt động trong một hệ thống hoặc ứng dụng.
2	API - Application Programming Interface	Một tập hợp các quy tắc, giao thức và công cụ được xây dựng để front-end và back-end có thể giao tiếp và trao đổi dữ liệu với nhau.
3	back-end	Phần của một ứng dụng hoặc trang web chịu trách nhiệm xử lý và quản lý dữ liệu, logic kinh doanh và giao tiếp với cơ sở dữ liệu và các thành phần khác.
4	base64	Phương thức mã hóa dữ liệu nhị phân thành chuỗi các ký tự ASCII. Quá trình này giúp truyền dữ liệu qua mạng hoặc lưu trữ dữ liệu trong các định dạng văn bản mà không bị mất mát thông tin và đảm bảo tính toàn vẹn của dữ liệu.
5	backup	Sao lưu dữ liệu hoặc hệ thống để đảm bảo an toàn và khôi phục sau khi xảy ra sự cố.

6	build	Quá trình tạo ra một ứng dụng hoặc sản phẩm từ mã nguồn và tài nguyên liên quan. Nó bao gồm các bước như biên dịch, gộp mã, tạo file thực thi và xử lý các tài nguyên như hình ảnh, stylesheets.
7	CRUD	Các thao tác cơ bản trên cơ sở dữ liệu hoặc hệ thống thông tin, bao gồm tạo (create), đọc (read), cập nhật (update) và xóa (delete).
8	CI/CD	Một phương pháp phát triển phần mềm tự động, kết hợp việc tích hợp liên tục (Continuous Integration) và triển khai liên tục (Continuous Deployment), nhằm đảm bảo rằng mã nguồn được xây dựng và triển khai một cách tự động và liên tục, giúp tăng tốc độ phát triển và đảm bảo chất lượng phần mềm.
9	cache	Một vùng lưu trữ tạm thời dùng để lưu trữ dữ liệu hoặc kết quả tính toán để truy cập nhanh chóng trong các hoạt động sau. Nó giúp cải thiện hiệu suất và tăng tốc độ truy cập dữ liệu bằng cách giảm thời gian truy xuất từ nguồn dữ liệu chính.
10	dashboard	Một trang giao diện tổng quan và trực quan, hiển thị các thông tin, số liệu và báo cáo quan trọng để giúp người dùng quản lý và theo dõi hoạt động của hệ thống hoặc ứng dụng một cách thuận tiện.
11	debug	Quá trình tìm và sửa lỗi hoặc vấn đề trong mã nguồn hoặc chương trình để đảm bảo hoạt động chính xác và ổn định.

12	hook	Một tính năng trong ReactJS cho phép sử dụng các trạng thái (state) và vòng đời (life-cycle) trong functional Component.
13	interceptor	Một thành phần trong hệ thống phần mềm, giúp ghi lại, thay đổi hoặc kiểm tra các yêu cầu và phản hồi giữa các thành phần khác nhau trong một quá trình xử lý.
14	localhost	Máy chủ cục bộ, thường dùng để chạy trong lúc phát triển phần mềm, mặc định trả đến 127.0.0.1.
15	library	Thư viện.
16	NoSQL	Một hệ quản trị cơ sở dữ liệu không phụ thuộc vào mô hình dữ liệu quan hệ truyền thống, thường được sử dụng để lưu trữ và truy xuất dữ liệu phi cấu trúc hoặc dữ liệu có cấu trúc linh hoạt.
17	framework	Một cấu trúc và mô hình đã được thiết kế sẵn để phát triển ứng dụng hoặc hệ thống phần mềm.
18	front-end	Phần giao diện của một ứng dụng hoặc trang web mà người dùng cuối tương tác và nhìn thấy.
19	functional component	Một loại component trong React được viết dưới dạng hàm JavaScript và không sử dụng class. Nó nhận vào các props và trả về các phần tử UI để hiển thị trên giao diện người dùng.
20	plugin	Một phần mở rộng hoặc module có thể được thêm vào một ứng dụng hoặc hệ thống để cung cấp các tính năng và chức năng bổ sung.

21	pure function	Một hàm mà đầu ra chỉ phụ thuộc vào giá trị đầu vào và không có tác động bên ngoài.
22	proxy gateway	Một cổng trung gian được sử dụng để tiếp nhận và chuyển tiếp các yêu cầu và phản hồi giữa một hệ thống và các hệ thống hoặc dịch vụ bên ngoài, để cung cấp tính năng như bảo mật, định tuyến, kiểm soát truy cập, và quản lý tài nguyên.
23	registry	Một dịch vụ quản lý và lưu trữ các hình ảnh container trong Google Cloud, cho phép người dùng lưu trữ, quản lý và phân phối các hình ảnh container một cách dễ dàng và an toàn.
24	repository	Một kho lưu trữ trên GitHub, một không gian trực tuyến để lưu trữ, quản lý và chia sẻ mã nguồn của một dự án phần mềm.
25	server	Máy chủ.
26	SSL - Secure Sockets Layer	Một giao thức bảo mật được sử dụng để tạo một kết nối an toàn giữa máy khách và máy chủ trên mạng Internet, đảm bảo rằng thông tin gửi đi và đến được mã hóa và bảo vệ khỏi các mối đe dọa bên ngoài.
27	website / trang web	Một tài nguyên trên internet, cung cấp thông tin, nội dung hoặc dịch vụ cho người dùng thông qua trình duyệt web. Thuật ngữ "Web" thường được sử dụng để chỉ đến các công nghệ và nguyên tắc liên quan đến việc xây dựng và phát triển trang web.

Bảng 2: Danh sách thuật ngữ lĩnh vực website builder

1	collection	Một tập hợp hoặc nhóm các đối tượng hoặc dữ liệu được tổ chức và lưu trữ cùng nhau trong cơ sở dữ liệu.
2	component	Là một thành phần của trang web, là khối xây dựng độc lập có thể tái sử dụng trong phát triển phần mềm, được sử dụng để tạo ra giao diện người dùng và xử lý logic riêng biệt.
3	click	Sự tương tác của người dùng bằng cách nhấp chuột lên một vị trí hoặc thành phần trên một trang web hoặc ứng dụng.
4	CMS - Content Management System	Hệ thống quản lý nội dung cho phép người dùng tạo, quản lý và công bố nội dung trên website một cách dễ dàng.
5	CSS inline style	Cách áp dụng kiểu CSS trực tiếp vào một phần tử HTML bằng cách sử dụng thuộc tính "style" trong thẻ mở của phần tử đó.
6	CSS - Cascading Style Sheets	Ngôn ngữ định dạng sử dụng để trình bày và tạo kiểu cho các phần tử trong trang web.
7	database	Cơ sở dữ liệu.
8	document	Một đơn vị thông tin cơ bản trong cơ sở dữ liệu, thường được biểu diễn dưới dạng bản ghi hoặc tài liệu có cấu trúc.
9	DOM - Document Object Model	Mô hình đối tượng tài liệu, là một biểu diễn phân cấp của các thành phần trên một trang web hoặc tài liệu HTML, cho phép truy cập và tương tác với các thành phần này thông qua mã JavaScript.
10	editor	Trình chỉnh sửa.
11	element	Phần tử trên trang web.

12	form	Biểu mẫu, dùng để thu thập thông tin người dùng.
13	header / heading	Phần đầu của một trang web hoặc tài liệu/Phần tiêu đề.
14	hydration	Quá trình tải dữ liệu giao diện website ban đầu và kích hoạt tính năng tương tác trên phía máy khách sau khi trang đã được tải từ máy chủ. Quá trình này giúp trang có khả năng tương tác ngay lập tức và cải thiện trải nghiệm người dùng.
15	HTML - HyperText Markup Language	Ngôn ngữ đánh dấu sử dụng để tạo và cấu trúc các trang web.
16	menu	Danh sách các lựa chọn.
17	navigate	Điều hướng hoặc chuyển đổi giữa các trang hoặc các phần khác nhau trong một ứng dụng hoặc trang web.
18	page	Trong ngữ cảnh website hoặc ứng dụng web, page là một khái niệm dùng để chỉ một trang web cụ thể hoặc một phần tử giao diện người dùng có chức năng riêng biệt và hiển thị nội dung độc lập.
19	path	Một chuỗi ký tự hoặc đường dẫn đại diện cho vị trí hoặc địa chỉ của một trang web.
20	pop-up	Cửa sổ hoặc hộp thoại xuất hiện trên một trang web hoặc ứng dụng để hiển thị thông tin bổ sung hoặc yêu cầu tương tác từ người dùng.

21	render	Quá trình hiển thị giao diện người dùng trên màn hình. Nó thường được sử dụng trong ngữ cảnh lập trình web để chỉ việc tạo ra và hiển thị các thành phần, phần tử, hoặc nội dung trên trang web dựa trên dữ liệu hoặc logic đã được xử lý.
22	scroll	Cuộn hoặc di chuyển nội dung trên một trang web hoặc ứng dụng để hiển thị phần nội dung ẩn hoặc tiếp theo.
23	SEO - Search Engine Optimization	Quá trình tối ưu hóa trang web để cải thiện vị trí và hiển thị trên các công cụ tìm kiếm.
24	template	Mẫu được thiết kế sẵn, đáp ứng một nội dung nào đó.
25	theme	Chủ đề.
26	UI - user interface	Giao diện người dùng.
27	website builder	Công cụ cho phép người dùng tạo và xây dựng trang web mà không cần kiến thức lập trình hoặc thiết kế đồ họa.

# Tóm tắt

Trong thời đại số hóa ngày nay, sở hữu một trang web chuyên nghiệp đã trở thành một yếu tố cực kỳ quan trọng trong việc quảng bá sản phẩm, dịch vụ hay thương hiệu của cả doanh nghiệp lẫn cá nhân. Tuy nhiên, việc thiết kế và xây dựng một trang web chất lượng đòi hỏi kiến thức chuyên môn về lập trình và thiết kế, đây là rào cản đáng kể đối với những người không có kinh nghiệm hoặc muốn tạo ra trang web nhanh chóng trong thời gian ngắn.

Để giải quyết vấn đề này, đã xuất hiện nhiều giải pháp phát sinh mã nguồn front-end, bao gồm phát sinh mã từ hình ảnh, bản vẽ phát họa, bản thiết kế từ các công cụ như Figma hay sử dụng website builder. Nhận thấy sự hạn chế của các hướng tiếp cận học máy hiện có và các công cụ thiết kế như Figma, nhóm quyết định xây dựng một website builder nhằm hỗ trợ phát sinh mã nguồn front-end có tính tùy chỉnh và tái sử dụng cao.

Buildify là hệ thống website builder mạnh mẽ đã được nhóm phát triển, với đầy đủ tính năng cần thiết, cho phép người dùng thiết kế nhiều trang, dễ dàng kéo thả các phần tử đã được thiết kế sẵn để tạo thành trang web, cũng như điều chỉnh thuộc tính, xử lý sự kiện, cấu hình theme và tích hợp dữ liệu động. Hệ thống cũng hỗ trợ phát sinh mã nguồn nhanh chóng, tạo ra mã có cấu trúc chuẩn của một React App, giúp mã nguồn dễ đọc, dễ hiểu, dễ chỉnh sửa và tái sử dụng, giao diện của trang web được phát sinh cũng hoàn toàn khớp với bản thiết kế mà người dùng mong muốn. Với sự hỗ trợ đáng kể từ Buildify, việc xây dựng trang web chuyên nghiệp không còn là trở ngại và tiết kiệm thời gian đáng kể cho người dùng.

# Chương 1

## Giới thiệu

*Chương 1 sẽ làm rõ đe tài nghiên cứu, mục tiêu, phạm vi của đe tài cũng như giới thiệu khái quát nội dung của từng chương trong luận văn.*

### 1.1 Đặt vấn đề

Trong thời đại số hóa hiện nay, việc sở hữu một trang web chuyên nghiệp đã trở thành một yếu tố vô cùng quan trọng trong việc quảng bá sản phẩm, dịch vụ hay thương hiệu của một doanh nghiệp hay cá nhân. Tuy nhiên, để thiết kế và xây dựng một trang web chuyên nghiệp đòi hỏi kiến thức chuyên môn về lập trình và thiết kế, gây khó khăn cho những người không có kinh nghiệm hoặc muốn tạo ra trang web nhanh chóng trong thời gian ngắn. Website builder ra đời và được đề xuất như là một giải pháp hữu hiệu và phổ biến giúp nhanh chóng tạo ra các trang web đẹp mắt mà không cần quá nhiều thời gian và chi phí. Tuy nhiên, một điểm yếu của các website builder hiện nay đó là hạn chế sở hữu mã nguồn khiến cho người dùng không thể phát triển trang web của mình một cách linh hoạt. Hơn nữa, mã nguồn phát sinh từ các website builder hiện nay thường không được tổ chức hợp lý, gây khó khăn cho việc đọc và hiểu mã nguồn. Việc tìm kiếm một giải pháp để sở hữu mã nguồn của trang web và tiếp tục phát triển trong tương lai vẫn đang là một thách thức đối với

người dùng.

Để giải quyết vấn đề này, đề tài "Buildify: Hệ thống website builder hỗ trợ phát sinh mã nguồn front-end" ra đời. Với giao diện đơn giản, dễ sử dụng và thân thiện với người dùng cùng với các mẫu thiết kế website đa dạng và chất lượng, Buildify cho phép người dùng xây dựng trang web nhanh chóng và dễ dàng chỉ bằng việc kéo thả. Đặc biệt, khả năng phát sinh mã nguồn cho các trang web này mang lại cho người dùng khả năng tùy chỉnh và tái sử dụng cao. Người dùng có thể kết hợp nó với chuyên môn và kinh nghiệm của những lập trình viên để phát triển các tính năng nâng cao và phức tạp hơn cho trang web của mình.

Đề tài không chỉ giúp cho nhiều người dùng, đặc biệt là các doanh nghiệp và cá nhân có nhu cầu xây dựng trang web, tiết kiệm được thời gian và chi phí đáng kể trong việc tạo ra các trang web chất lượng cao, mà còn có tiềm năng phát triển và ứng dụng rộng rãi trong lĩnh vực phát triển web.

## 1.2 Mục tiêu của đề tài

Sau thời gian tìm hiểu và nghiên cứu lĩnh vực website builder, nhóm đã nhận thấy rằng nhiều website builder hiện tại thường tạo ra mã nguồn còn khá thô, đơn giản hoặc không thân thiện với lập trình viên khi cần phát triển tiếp trong tương lai. Ngoài ra, nhóm cũng nhận thấy có khá ít nghiên cứu được công bố (có giải thích chi tiết) về phương pháp xây dựng một website builder hỗ trợ với việc phát sinh mã nguồn giao diện front-end dạng library/framework. Với mục tiêu giải quyết các vấn đề này, nhóm quyết định đề xuất đề tài nghiên cứu và phát triển hệ thống website builder hỗ trợ phát sinh mã nguồn front-end - Buildify với các mục tiêu sau:

- Nghiên cứu, tổng hợp và đề xuất một phương pháp xây dựng website builder hoàn chỉnh hướng đến việc phát sinh mã nguồn front-end.

Nhóm sẽ đóng góp một mã nguồn mở cho phương pháp này, kèm theo những giải thích về kiến trúc và các luồng xử lý chính.

- Xây dựng hệ thống website builder hoàn thiện với bộ tính năng đầy đủ, giao diện thân thiện và dễ sử dụng. Buildify sẽ giúp người dùng có thể nhanh chóng tạo ra trang web chỉ bằng cách kéo và thả các thành phần đã được thiết kế sẵn và cho phép chỉnh sửa các thông số thuộc tính theo mong muốn. Không chỉ hỗ trợ thiết kế về mặt giao diện, Buildify còn cho phép người dùng gắn một số sự kiện tương tác cho trang web, đồng thời hỗ trợ thêm việc triển khai các khía cạnh khác như: cơ sở dữ liệu, cấu hình theme, phân trang và chuyển trang.
- Phát triển tính năng phát sinh mã nguồn front-end với thư viện ReactJS từ bản giao diện thiết kế. Mã nguồn cần có cấu trúc tổ chức tốt, dễ dàng đọc hiểu và dễ mở rộng. Phát sinh mã nguồn là tính năng quan trọng của website builder, tạo cơ hội cho người dùng xem và sửa đổi trực tiếp mã nguồn trang web mà họ đã tạo, hỗ trợ nhu cầu phát triển thêm các tính năng phức tạp sau này. Hơn nữa, việc cho phép người dùng sở hữu mã nguồn trang web giúp họ có thể tự tin và dễ dàng triển khai trang web mà họ đã thiết kế lên các nền tảng khác nhau mà không phụ thuộc vào bất kỳ công cụ hay dịch vụ nào khác.

Nhóm sẽ áp dụng các kỹ thuật phát triển phần mềm và ứng dụng công nghệ hiện đại để đảm bảo rằng mã nguồn được phát sinh ra là thân thiện, tổ chức tốt, dễ đọc hiểu và mở rộng, từ đó giúp người dùng dễ dàng sửa đổi và phát triển trang web theo ý muốn, tiết kiệm thời gian, tăng năng suất và giảm chi phí cho người dùng trong quá trình phát triển trang web. Ngoài ra, đây sẽ là một nguồn tham khảo chất lượng cho các lập trình viên về phương pháp cài đặt và xây dựng một website builder hoàn thiện hướng đến việc phát sinh mã nguồn front-end. Từ đó, lập trình viên có thể chọn

1 trong 2 cách: tiếp tục phát triển thêm tính năng dựa trên mã nguồn mở hiện tại hoặc tham khảo về ý tưởng cài đặt để tự tổng hợp được phương pháp riêng, nhằm tạo ra một sản phẩm website builder cho mình.

Về mặt học thuật, đề tài tạo cơ hội để nhóm nghiên cứu sâu hơn về lĩnh vực website builder và phát sinh mã nguồn. Từ đó, nhóm tiến đến xây dựng một sản phẩm hoàn chỉnh, đồng thời rèn luyện và hoàn thiện các kỹ năng chuyên môn của mình trong lĩnh vực kỹ thuật phần mềm.

## 1.3 Phạm vi đề tài

- Nội dung nghiên cứu của đề tài:
  - Công nghệ:
    - \* Front-end: NextJS, ReactJS, TypeScript, HTML, CSS, JavaScript, Material-UI, Styled-components, Ant Design, Tailwind CSS.
    - \* Back-end: Go, Bazel, gRPC, MongoDB.
    - \* DevOps: Docker, Kubernetes phục vụ việc đóng gói ứng dụng.
  - Website builder: toàn bộ lý thuyết và kiến trúc xây dựng các thành phần của website builder, bao gồm: editor, node, element, layer, history...
  - Nghiên cứu và áp dụng hệ thống phân tán:
    - \* Microservices: các services ở back-end giao tiếp với nhau bằng gRPC.
    - \* Git submodules.
- Đối tượng hướng đến:
  - Bất kỳ đối tượng nào (cá nhân, doanh nghiệp vừa và nhỏ...) có nhu cầu thiết kế nhanh chóng website cho mục đích riêng với chi phí thấp.

- Những lập trình viên muốn thiết kế nhanh chóng giao diện người dùng, sau đó tiếp tục phát triển các xử lý logic phức tạp hơn dựa trên mã nguồn được sinh ra.
- Không gian: website - thiết kế trực tiếp trên giao diện desktop.
- Thời gian: 01/02/2023 - 14/07/2023

## 1.4 Nội dung của luận văn

Nội dung của luận văn bao gồm các chương sau:

- **Chương 1: Giới thiệu:** giới thiệu về đề tài nghiên cứu, mục tiêu và phạm vi của đề tài cũng như trình bày tổng quát nội dung của luận văn.
- **Chương 2: Khảo sát các công nghệ tương tự:** tìm hiểu các phương pháp phát sinh mã nguồn front-end hiện nay. Tiếp theo, tập trung khảo sát và đánh giá các hệ thống website builder tương tự đã có sẵn và nổi tiếng trên thị trường. Qua đó, nghiên cứu những tính năng, ưu điểm và hạn chế của các hệ thống này để có cái nhìn tổng quan về lĩnh vực nghiên cứu, cuối cùng là giới thiệu giải pháp đề xuất của nhóm.
- **Chương 3: Phân tích yêu cầu và thiết kế hệ thống:** phân tích các yêu cầu chức năng và phi chức năng của hệ thống, đồng thời trình bày thiết kế kiến trúc tổng quan, các công nghệ được sử dụng để xây dựng hệ thống. Sơ đồ thiết kế cơ sở dữ liệu cũng sẽ được trình bày ở phần này.
- **Chương 4: Hệ thống website builder Buildify:** trình bày chi tiết về cách cài đặt các thành phần cũng như cơ chế hoạt động của các chức năng tương ứng của hệ thống Buildify.

- **Chương 5: Quá trình thực hiện:** mô tả quá trình thực hiện khóa luận của nhóm, đồng thời đưa ra những vấn đề nhóm gặp phải và kiến thức thu được khi thực hiện khóa luận.
- **Chương 6: Kết luận và hướng phát triển:** trình bày những kết quả đạt được của khoá luận (ứng dụng, bài báo khoa học), từ đó rút ra được những kết luận chung và bàn luận về những hướng mở rộng mới cho đề tài trong tương lai.

## Chương 2

# Khảo sát các công nghệ tương tự

*Chương 2 sẽ tìm hiểu các phương pháp phát sinh mã nguồn front-end hiện nay và chỉ ra ưu thế của website builder. Từ đó, tiến hành khảo sát và đánh giá các hệ thống website builder phổ biến trên thị trường, tổng kết và đưa ra giải pháp cho hệ thống của nhóm.*

## 2.1 Các phương pháp phát sinh mã nguồn front-end hiện nay

Phát sinh mã nguồn front-end hiện nay có những cách tiếp cận phổ biến sau:

- Phát sinh mã nguồn từ hình ảnh: trích xuất thông tin từ hình ảnh màn hình và phát sinh mã nguồn tương ứng.
- Phát sinh mã nguồn từ bản vẽ phác thảo: trích xuất thông tin từ các bản vẽ phác thảo để phát sinh mã nguồn.
- Phát sinh mã nguồn từ công cụ thiết kế - nổi bật với Figma: sử dụng công cụ thiết kế đồ họa Figma để thiết kế giao diện người dùng, sau đó sử dụng plugin để sinh mã nguồn.

- Phát sinh mã nguồn từ website builder: sử dụng các công cụ thiết kế trang web để tạo ra giao diện, sau đó sinh mã nguồn tương ứng.

Nhóm sẽ lần lượt làm rõ các phương pháp trên ở những phần tiếp theo và lý do vì sao nhóm lựa chọn phương pháp sử dụng website builder để hỗ trợ phát sinh mã nguồn front-end.

### **2.1.1 Ứng dụng học máy để phát sinh mã nguồn**

Nhóm đã tiến hành khảo sát các bài báo nghiên cứu khoa học về lĩnh vực phát sinh mã nguồn hiện nay, một vài bài báo có thể kể đến như: pix2code: Generating Code from a Graphical User Interface Screenshot [11], Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps [22], Generating webpages from screenshots [19].

Tuy nhiên đặc điểm chung của các cách tiếp cận trên là cần lượng dữ liệu training đáng kể và chỉ đạt hiệu quả cao ở trường hợp website đơn giản, cấu trúc phổ biến. Hơn nữa, mã nguồn phát sinh có thể phức tạp, khó hiểu và thiếu tổ chức, điều này gây khó khăn cho việc sửa đổi, tùy chỉnh giao diện kết quả và phát triển thêm sau này. Vấn đề quan trọng nhất là tính cá nhân hoá của người dùng (theme, primary-color, font-family...) gần như không có.

### **2.1.2 Sử dụng plugin để phát sinh mã nguồn từ bản thiết kế Figma**

Đối với việc phát sinh mã nguồn từ Figma, hầu hết các công cụ được phát triển là các plugin đi kèm theo Figma, một vài cái tên nổi bật [13] như Anima, TeleportHQ, Locofy, pxCode. Figma không dễ sử dụng thành thạo và để phát sinh được mã nguồn đủ tốt từ các plugins này, người dùng phải cực kỳ tỉ mỉ và tuân thủ các quy tắc (như đặt tên, nhóm các elements, phân chia layout...) và hiểu cơ chế phát sinh mã nguồn của plugin. Việc này rất tốn thời gian và không dễ để thực hiện được.

### **2.1.3 Phát sinh mã nguồn với sự hỗ trợ từ website builder**

Nhận thấy vấn đề từ các hướng đi trên, nhóm quyết định sử dụng website builder để hỗ trợ việc phát sinh mã nguồn được chính xác, hiệu quả với cả những website phức tạp và đáp ứng nhu cầu dễ sử dụng, dễ tùy chỉnh của người dùng.

Trong phần tiếp theo, nhóm sẽ tập trung vào các hệ thống website builder phổ biến nhất hiện nay để có cái nhìn tổng quan và đưa ra được phương hướng giải quyết cũng như các tính năng mà nhóm lựa chọn để phát triển cho hệ thống của mình.

## **2.2 Các hệ thống website builder liên quan**

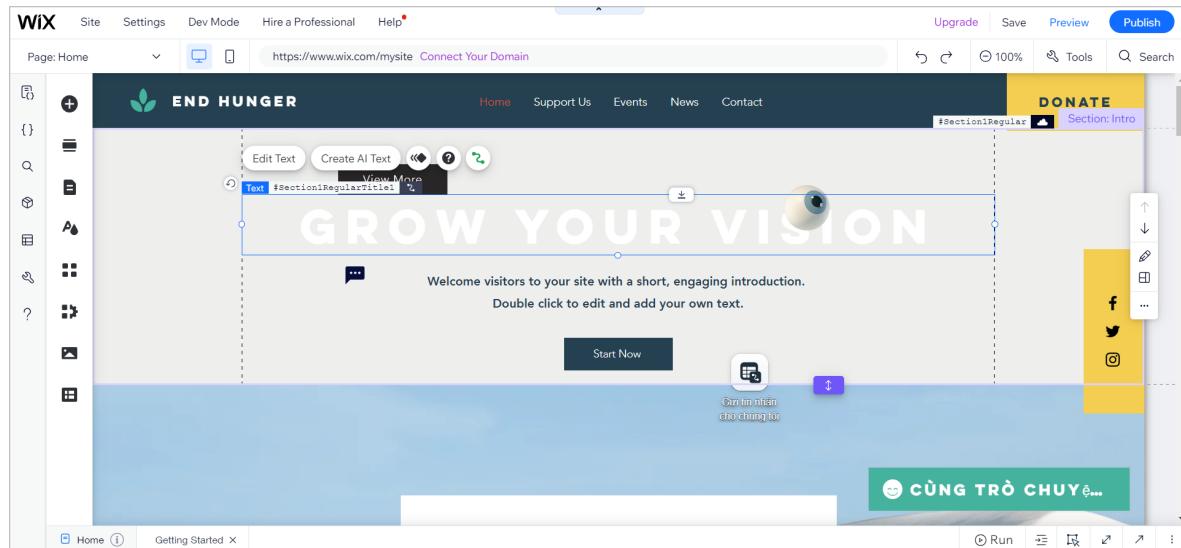
Sau khi tìm hiểu, nhóm quyết định lựa chọn khảo sát 3 hệ thống sau: Wix, Wordpress (2 hệ thống website builder phổ biến nhất hiện nay) và TeleportHQ (hệ thống website builder rất mạnh về tính năng phát sinh mã nguồn giao diện front-end). Khi khảo sát các hệ thống trên, nhóm có tham khảo các nguồn sau để đưa ra nhận xét và sự so sánh chính xác nhất: Wix với WordPress [29, 30], Wix với TeleportHQ [28], TeleportHQ với WordPress [25].

### **2.2.1 Wix**

Link website: <https://www.wix.com> [27].

Wix là một nền tảng xây dựng website trực tuyến rất nổi tiếng cho phép người dùng tạo ra các trang web chuyên nghiệp mà không cần kiến thức về lập trình. Nó cung cấp một giao diện trực quan và dễ sử dụng, cho phép người dùng kéo và thả các thành phần, tùy chỉnh giao diện và nội dung trang web theo ý muốn.

Giao diện thiết kế của Wix được thể hiện ở hình 2.1.



Hình 2.1: Giao diện thiết kế của Wix

### Ưu điểm:

- Dễ sử dụng: giao diện trực quan, có một danh sách liệt kê các element theo từng loại (như nút bấm, văn bản, hình ảnh, video...) bên trái cho phép lựa chọn để sử dụng.
- Công cụ kéo và thả thông minh: rất tự do trong việc di chuyển, kéo thả và sắp xếp các element một cách linh hoạt và nhanh chóng. Có hiển thị các đường chỉ dẫn hỗ trợ trong lúc kéo thả.
- Số lượng element và template giao diện sẵn có khá nhiều và đẹp mắt: người dùng có nhiều lựa chọn để tạo giao diện trang web mà không cần tùy chỉnh nhiều, chúng được phân loại theo nhóm chức năng giúp dễ tìm kiếm hơn.
- Cho phép thiết kế nhiều trang cho website, hỗ trợ tính năng phân trang và liên kết trang.
- Tính năng kết nối dữ liệu: Wix hỗ trợ người dùng tạo và quản lý bộ các dữ liệu để kết nối vào nội dung dạng văn bản của trang web.
- Tích hợp nhiều tiện ích: Wix cung cấp nhiều tính năng tiện ích như hình ảnh, video, biểu đồ, form liên hệ, tích hợp mạng xã hội và nhiều

tính năng khác, giúp tăng tính tương tác và chuyên nghiệp cho trang web.

- Nhanh chóng: Wix cung cấp các công cụ và tính năng giúp người dùng nhanh chóng thiết lập và tạo ra sản phẩm với chất lượng ổn.
- Hỗ trợ hai chế độ màn hình: desktop và mobile.

Hạn chế:

- Khả năng mở rộng hạn chế: không dễ dàng thay đổi template hoặc thêm các tính năng bên thứ ba vào trang web.
- Không hỗ trợ chuyển đổi nền tảng: khi xây dựng trang web trên Wix, người dùng gắn kết với nền tảng của Wix và không thể chuyển đổi sang một nền tảng khác một cách dễ dàng. Điều này có thể gây rắc rối và hạn chế quyền tự do của người dùng khi muốn thay đổi nền tảng xây dựng trang web.
- Chưa có tính năng phát sinh mã nguồn: người dùng không thể truy cập, tải xuống hoặc tùy chỉnh mã nguồn của trang web.
- Chưa hỗ trợ nhiều sự kiện tương tác của người dùng: hạn chế trong việc cung cấp các tùy chọn linh hoạt để xử lý các sự kiện tương tác phức tạp. Điều này có thể gây khó khăn cho những người muốn tạo ra trang web động.

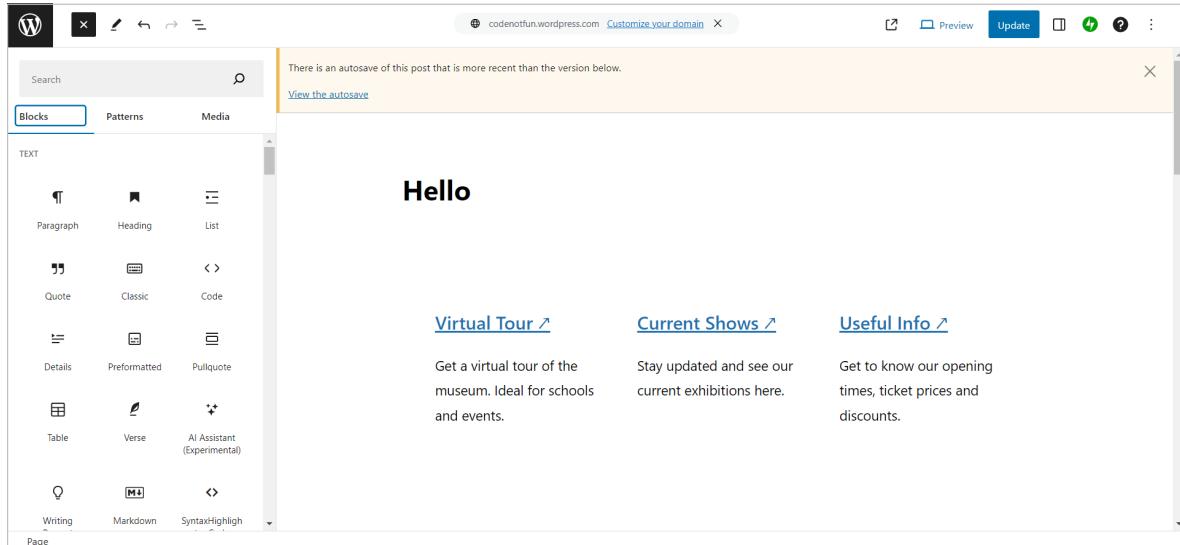
## 2.2.2 WordPress

Link website: <https://wordpress.com> [31].

WordPress là một trong những nền tảng thiết kế web đang rất được ưa chuộng hiện nay, và cũng là một hệ thống CMS phổ biến được sử dụng để xây dựng và quản lý các trang web. Nó cho phép người dùng tạo, chỉnh sửa và tổ chức nội dung trên trang web một cách dễ dàng mà không cần có kiến thức về lập trình. WordPress cung cấp một giao diện người dùng

thân thiện và nhiều tính năng mở rộng, bao gồm cả giao diện trực quan để thiết kế trang web và cài đặt các chức năng bổ sung thông qua các plugin.

Giao diện thiết kế của WordPress được thể hiện ở hình 2.2.



Hình 2.2: Giao diện thiết kế của WordPress

Ưu điểm:

- Dễ sử dụng: WordPress có một giao diện người dùng thân thiện và dễ sử dụng, thao tác sử dụng đơn giản, dễ hiểu và dễ vận hành.
- Da dạng theme và plugin: WordPress cung cấp một thư viện lớn theme và plugin cho phép người dùng tùy chỉnh giao diện và mở rộng chức năng của trang web một cách linh hoạt và dễ dàng.
- Cho phép chỉnh sửa giao diện dạng mã code HTML, người biết lập trình có thể tự nhúng code theo ý muốn.
- Cho phép thiết kế nhiều trang cho website, hỗ trợ tính năng phân trang và liên kết trang.
- Hỗ trợ tốt từ cộng đồng: với một cộng đồng người dùng rộng lớn, WordPress được hỗ trợ mạnh mẽ qua các diễn đàn, tài liệu và tài nguyên trực tuyến. Người dùng có thể tìm kiếm và nhận được giúp đỡ từ cộng đồng này.

- Tính dễ mở rộng và tùy chỉnh cao: ngoài plugin, WordPress cũng hỗ trợ ngôn ngữ lập trình PHP, hệ thống theme và template, cho phép người dùng tạo ra giao diện tùy chỉnh và thiết kế trang web theo ý muốn của mình.
- Tối ưu hóa SEO: có các công cụ mặc định để giúp SEO trang web dễ dàng hơn và nhanh hơn.
- Cho phép thay đổi template: linh hoạt trong việc chuyển đổi.
- Cho phép xem mối quan hệ tổng quát giữa các element.

Hạn chế:

- Giao diện không có khả năng kéo, thả và di chuyển element vẫn còn hơi cứng nhắc, điều này có thể làm giới hạn sự sáng tạo và khả năng tùy chỉnh của trang web.
- Cần có kiến thức cho các tùy chỉnh nâng cao: để tận dụng hiệu quả các chức năng, người dùng cần có kiến thức về mã nguồn PHP, HTML, CSS và có khả năng sử dụng hệ thống WordPress API. Điều này có nghĩa là người dùng muốn tùy chỉnh nhiều thì phải học và áp dụng các nguyên tắc lập trình và nắm vững kiến thức về bố cục giao diện.
- Mã nguồn phát sinh đơn giản: tuy có hỗ trợ phát sinh mã nguồn trang web nhưng mã nguồn này không tường minh, dễ hiểu và thân thiện với lập trình viên hiện nay (với xu hướng là sử dụng các framework front-end như ReactJS). Mã nguồn phát sinh này còn đơn giản (HTML, CSS thuần dạng inline style) và chưa đáp ứng được nhu cầu muôn phát triển tiếp trên thiết kế đó của lập trình viên.
- Khó khăn trong việc quản lý server: người dùng phải tự quản lí server và việc backup dữ liệu. Điều này đòi hỏi kiến thức kỹ thuật và sự chịu khó từ phía người dùng.

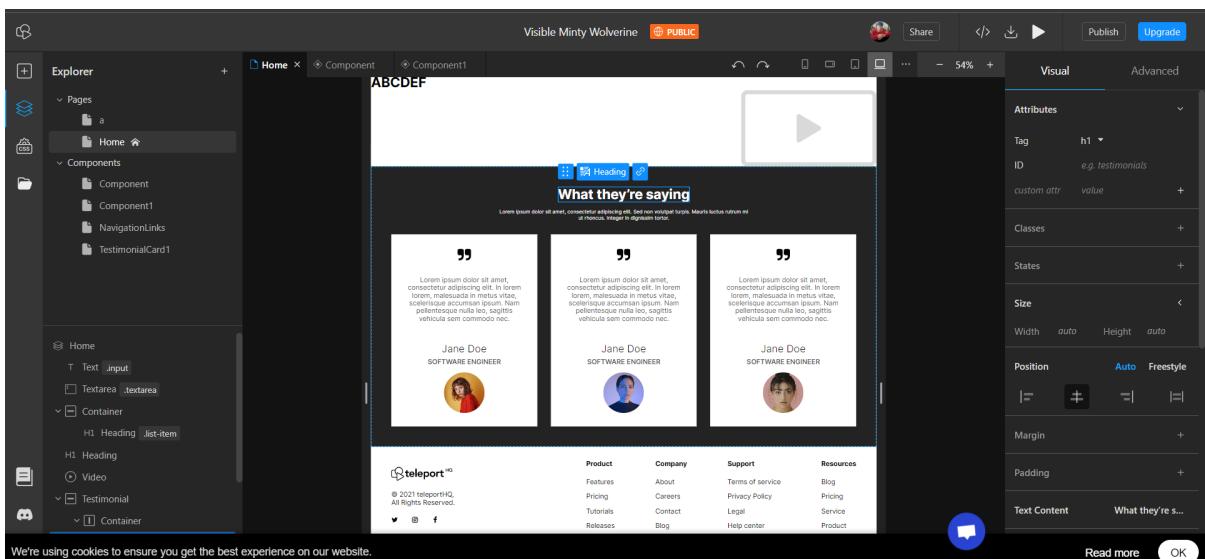
- Số lượng template sẵn có không quá nhiều so với một số website builder khác.

### 2.2.3 TeleportHQ

Link website: <https://teleporthq.io> [9].

TeleportHQ là một nền tảng phát triển front-end dùng để xây dựng và quản lý giao diện người dùng của các ứng dụng web. Nó cung cấp các công cụ tích hợp để phát triển, thiết kế và xuất bản giao diện người dùng một cách nhanh chóng và dễ dàng. TeleportHQ giúp tăng cường hiệu suất và hiệu quả trong việc xây dựng các trang web và ứng dụng web tương tác. Mục tiêu hướng tới của TeleportHQ là trở thành một công cụ phát triển dành cho lập trình viên và các tính năng thiết kế dành cho người có kiến thức lập trình.

Giao diện thiết kế của TeleportHQ được thể hiện ở hình 2.3.



Hình 2.3: Giao diện thiết kế của TeleportHQ

Ưu điểm:

- Giao diện trực quan, đơn giản, dễ nhìn và sử dụng với lập trình viên.

- Tư duy thiết kế hiện tại: hỗ trợ thiết kế các component nhỏ, cho phép chia nhỏ thành phần trang và tái sử dụng. Các component sẽ được dùng để kéo và thả vào các trang.
- Cho phép định nghĩa hệ thống CSS dạng class: định nghĩa class và bộ thuộc tính CSS đi kèm cho mỗi class, gần với lập trình website, hỗ trợ tốt cho các lập trình viên.
- Cho phép chỉnh sửa một số giá trị của element như: id, class, cấu trúc HTML, chỉnh các trạng thái khi focus hay hover...
- Cho phép thiết kế nhiều trang cho website.
- Hỗ trợ thiết kế website responsive trên nhiều mốc giới hạn về chiều rộng màn hình (width).
- Hỗ trợ triển khai và phát hành trang web.
- Có hỗ trợ xem tổng quát mối quan hệ giữa các element thuộc website dưới dạng cây. Đồng thời, người dùng có thể chỉnh sửa tên, vị trí các element trên đây.
- Hỗ trợ tính năng phát sinh mã nguồn dưới dạng nhiều library/framework như ReactJS, Vue, Angular... Ví dụ với ReactJS, mã nguồn bao gồm các component và view (tương ứng với page) đáp ứng đủ tiêu chuẩn của library.
- Cho phép chia sẻ và cộng tác trên cùng một dự án.

Hạn chế:

- Số lượng element và template được thiết kế sẵn không nhiều và đa dạng.
- Khả năng tùy chỉnh như kéo thả, di chuyển element còn hạn chế. Một số tùy chỉnh kéo thả không tùy ý được mà phải phụ thuộc

layout (center, left, right....). Điều này có thể khiến cho việc xây dựng giao diện phức tạp hoặc tuỳ chỉnh đặc biệt trở nên khó khăn.

- Còn cứng nhắc trong căn chỉnh vị trí element.
- Chưa chia sẻ được thư viện các component thiết kế.
- Mã nguồn muôn tái sử dụng thì cần chỉnh sửa nhiều.

## 2.3 Tổng kết

### 2.3.1 Vấn đề tồn đọng của các hệ thống tương tự

Nhìn chung, có một số vấn đề mà nhóm nhận thấy khi trải nghiệm các sản phẩm trên:

- Giao diện: đôi khi vẫn còn cứng nhắc trong việc kéo thả element.
- Mã nguồn phát sinh chưa thân thiện với lập trình viên về mặt cấu trúc tổ chức, cách tách và phân chia các thành phần trang, cách đặt tên... dẫn đến việc khó chỉnh sửa và mở rộng.
- Một số mã nguồn phát sinh khá đơn giản và không áp dụng library/framework front-end mạnh mẽ mà chỉ đơn thuần là HTML, CSS, một chút JavaScript nên không có tính áp dụng triển khai cao.
- Chưa hỗ trợ nhiều sự kiện tương tác của người dùng.

### 2.3.2 Phương pháp đề xuất

Cách tiếp cận của nhóm để giải quyết các vấn đề trên và đồng thời phục vụ mục đích nghiên cứu là:

1. Xây dựng nền tảng website builder Buildify: đề xuất và phân tích được một phương pháp xây dựng website builder với đầy đủ các tính năng cần thiết như sau:

- Giao diện trực quan và dễ sử dụng.
- Thiết kế sẵn một bộ các element cơ bản và nâng cao (template dựng sẵn). Người dùng chỉ việc kéo thả và tùy chỉnh các thông số.
- Thao tác thiết kế dễ dàng. Điều này có nghĩa là cho phép người dùng tự do và linh hoạt trong việc kéo thả, di chuyển, xoá, sửa, thay đổi kích thước chiều dài, chiều rộng các element.
- Cho phép người dùng tự điều chỉnh các thông số CSS mặc định cho từng loại element như font-size, font-weight, font-family, spacing, padding, margin... Mỗi loại element sẽ có những đặc điểm riêng về giao diện, khả năng tương tác sự kiện.
- Hỗ trợ xem mối quan hệ giữa các element thuộc website dưới dạng cây. Đồng thời, người dùng có thể chỉnh sửa tên, thay đổi vị trí các element trên đây.
- Lưu trữ và quản lý lịch sử chỉnh sửa, hỗ trợ hai thao tác hoàn tác và tái thực hiện (undo và redo).
- Cho phép tạo cơ sở dữ liệu riêng và nhúng vào nội dung dạng văn bản của trang web. Ngoài ra, cho phép người dùng phát sinh cơ sở dữ liệu dưới dạng file JSON hoặc file script SQL.
- Cho phép tạo bộ theme riêng cho trang web mang phong cách người dùng.
- Cho phép người dùng lưu trữ và truy xuất lại giao diện thiết kế trang web khi cần thiết.
- Cho phép gắn và chỉnh sửa các sự kiện người dùng cho các element như: hover, click (navigate, scroll, mở giao diện pop-up)...
- Cho phép thiết kế nhiều trang và cho phép điều hướng trang trong website.

## 2. Phát sinh mã nguồn

- Tiến hành đóng gói toàn bộ dữ liệu ở front-end và phát sinh mã nguồn dựa trên dữ liệu đã đóng gói ở back-end.
- Mã nguồn phát sinh có cấu trúc tuân thủ library ReactJS điển hình, được tổ chức tốt - gần với các dự án thực tế, và trong tương lai có thể phát triển mở rộng thêm cho các library/framework khác.
- Phát sinh mã nguồn dưới dạng các component và chia giao diện theo từng page.
- Mỗi component sẽ có các props nhất định, giá trị truyền vào được nhận từ thông số mà người dùng đã tùy chỉnh.
- Cho phép website có nhiều trang (gắn với một path riêng) và hỗ trợ routing.
- Phát sinh các cơ sở dữ liệu và theme của người dùng.
- Hướng dẫn người dùng cách tải các gói cần thiết và chạy mã nguồn ở file README.

# Chương 3

## Phân tích yêu cầu và thiết kế hệ thống

*Chương 3 sẽ đi sâu vào phân tích các yêu cầu chức năng và phi chức năng cụ thể của hệ thống, trình bày kiến trúc tổng quan và các công nghệ cụ thể được sử dụng để xây dựng hệ thống và cuối cùng là trình bày sơ đồ thiết kế cơ sở dữ liệu.*

### 3.1 Phân tích yêu cầu

#### 3.1.1 Yêu cầu chức năng

##### Chức năng cơ bản của người dùng

- Đăng ký và đăng nhập: để thiết kế website, người dùng cần đăng ký thông tin và đăng nhập để quản lý các dự án website.
- Quản lý thông tin tài khoản: người dùng có thể xem các thông tin cơ bản (username, tên đầy đủ, email, ảnh đại diện) và cập nhật thông tin khi cần thiết.
- Quản lý các dự án website:

- Tạo mới: tạo một dự án website mới với các thông tin cơ bản (tên dự án, thể loại dự án). Mỗi dự án khi khởi tạo có thể chọn một trong các thể loại website đã được nhóm thiết kế sẵn như landing page, content management system... để có được giao diện thiết kế mặc định ứng với thể loại website muốn hướng tới.
  - Xem thông tin: danh sách các dự án website được hiển thị dưới dạng bảng, có hỗ trợ tìm kiếm theo tên, sắp xếp theo tên dự án và thể loại dự án.
  - Chính sửa: vào trang thiết kế để tiếp tục chỉnh sửa giao diện dự án.
  - Xoá: người dùng có thể xoá dự án khi nó không còn cần thiết nữa.
- Quản lý cơ sở dữ liệu động:
    - Dữ liệu động của mỗi dự án bao gồm danh sách các collections, mỗi collections bao gồm danh sách các documents, lấy ý tưởng từ cách thiết kế dữ liệu NoSQL tương tự như MongoDB.
      - \* Collections: hỗ trợ CRUD, dữ liệu bao gồm id, tên collection, danh sách keys và danh sách types tương ứng, có thể hỗ trợ type number và string. Ví dụ: (id: 1, name: "Product", keys: ["ProductName", "Price"], types: ["string", "number"]).
      - \* Documents: hỗ trợ CRUD, dữ liệu bao gồm các cặp key - value với key thuộc danh sách keys đã định nghĩa ở collection tương ứng. Ví dụ một document thuộc collection "Product": (id: 1, ProductName: "Laptop", Price: 20).
    - Cho phép tải xuống file script chứa các câu lệnh dùng để tạo bảng và dữ liệu cho dự án website, hỗ trợ các hệ thống quản trị cơ sở dữ liệu như MySQL, SQLite, SQL Server, Postgres.

## Chức năng của website builder

(Các chức năng được liệt kê theo từng phần của website builder)

1. Toolbox: gồm tập hợp các selector (elements đã được thiết kế sẵn), cho phép người dùng kéo thả vào editor để xây dựng website.
  - Hỗ trợ 2 loại selectors:
    - Basic element: là những thành phần nhỏ nhất cấu thành nên một trang web như div, button, text, input, link, image, video...)
    - Built-in template: là những mẫu giao diện thông dụng như headers, footers, menus, introductions, contents, banners,... được thiết kế sẵn bằng cách kết hợp các basic element, giúp người dùng nhanh chóng xây dựng được trang web mà không mất nhiều công sức thiết kế.
  - Cho phép người dùng xem trước (preview) giao diện của các selector ở toolbox với kích thước thu nhỏ.
2. Settings panel: sau khi kéo thả selector vào editor, editor sẽ render selector đó thành một element thuộc editor và cho phép chỉnh sửa thuộc tính, CSS của element đó ở settings panel như thay đổi màu sắc, phông chữ, kích thước, khoảng cách, đường dẫn...
  - Mỗi element sẽ có những cài đặt riêng ứng với từng thuộc tính của element đó, bao gồm các loại cài đặt như text input, number input, dropdown menu, slider, checkbox, radio, color picker, image upload...
  - Hỗ trợ tính năng theme: ngoài giá trị chỉnh tay fix cứng cho các thuộc tính của element, người dùng có thể nhúng giá trị từ theme. Những giá trị này sẽ tự động cập nhật khi giá trị thuộc tính của theme tương ứng có sự thay đổi.

- Hỗ trợ tùy chỉnh các events như navigate, scroll, hiện pop-up... tùy vào từng element.
3. Editor: là một canvas cho phép người dùng thao tác và tùy chỉnh giao diện trang web.
- Cho phép kéo thả các selector vào editor: người dùng có thể tùy chọn một selector từ toolbox để kéo thả vào vị trí thích hợp trong editor để tạo giao diện trang web theo ý muốn.
  - Thao tác với element thuộc editor:
    - Thay đổi thuộc tính element với settings panel.
    - Thay đổi vị trí element bằng cách kéo thả đến vị trí mới, bao gồm cả việc kéo thả vào trong một element khác (tạo mối quan hệ cha - con).
    - Nhân bản (clone) element.
    - Xoá element.
  - Hỗ trợ phản hồi real-time về vị trí thả trong quá trình kéo thả element trên editor: hỗ trợ người dùng trong việc xác định chính xác vị trí tối ưu cho element đang kéo.
  - Hỗ trợ thiết kế nhiều page cho một dự án website: mỗi trang có tên, đường dẫn và giao diện riêng biệt.
  - Hỗ trợ cài đặt theme cho dự án: theme gồm danh sách các thuộc tính với tên và giá trị do người dùng định nghĩa. Khi thay đổi giá trị của các thuộc tính này, những element liên quan ở hiện tại sẽ được cập nhật theo. Theme giúp người dùng dễ dàng quản lý và nhanh chóng cập nhật giao diện theo phong cách mới nhất của website
  - Cho phép nhúng dữ liệu động vào nội dung của element: người dùng có thể nhúng giá trị từ cơ sở dữ liệu động của dự án vào

nội dung của các element khác nhau, những nội dung này sẽ đồng loạt thay đổi khi dữ liệu động tương ứng có sự điều chỉnh.

4. Layers: trực quan hóa bối cảnh, cấp bậc (mỗi quan hệ cha - con) của các element hiện tại.

- Mỗi layer là một ánh xạ 1 - 1 với các element hiện tại.
- Người dùng có thể kéo thả layer để thay đổi vị trí của element trên Editor và có phản hồi real-time vị trí thả trên giao diện layers.

5. History: cung cấp chức năng undo và redo.

- Undo: cho phép người dùng trở về trạng thái trước đó của editor.
- Redo: khôi phục hành động đã được hoàn tác trước đó bằng lệnh undo.

6. Phát sinh mã nguồn front-end:

- Hỗ trợ phát sinh mã nguồn ReactJS, giao diện đảm bảo giống với giao diện người dùng thiết kế trong dự án.
- Mã nguồn hỗ trợ routing nhiều page, có tích hợp theme và dữ liệu động.

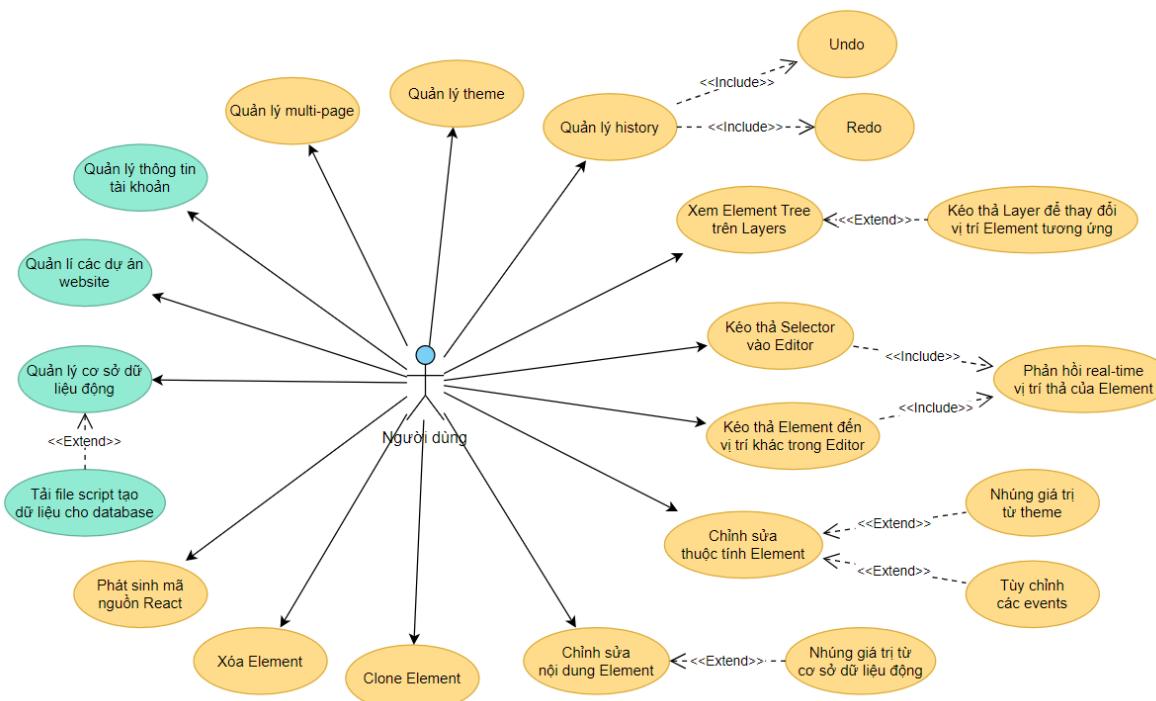
### 3.1.2 Yêu cầu phi chức năng

- Khả năng sử dụng:
  - Website builder: giao diện dễ hiểu, người dùng mới có thể nhanh chóng làm quen và sử dụng.
  - Mã nguồn ReactJS được phát sinh từ giao diện website: có cấu trúc tiêu chuẩn của mã nguồn ReactJS, dễ đọc, dễ hiểu, dễ chỉnh sửa và dễ mở rộng thêm tính năng khác.

- Hiệu suất:
  - Tốc độ phát sinh mã nguồn trung bình không quá 5 giây.
  - Các chức năng còn lại của website builder có tốc độ phản hồi không quá 2 giây .

### 3.1.3 Sơ đồ use case

Dựa trên yêu cầu đã được phân tích, nhóm đã mô tả lại các use case ở hình 3.1.



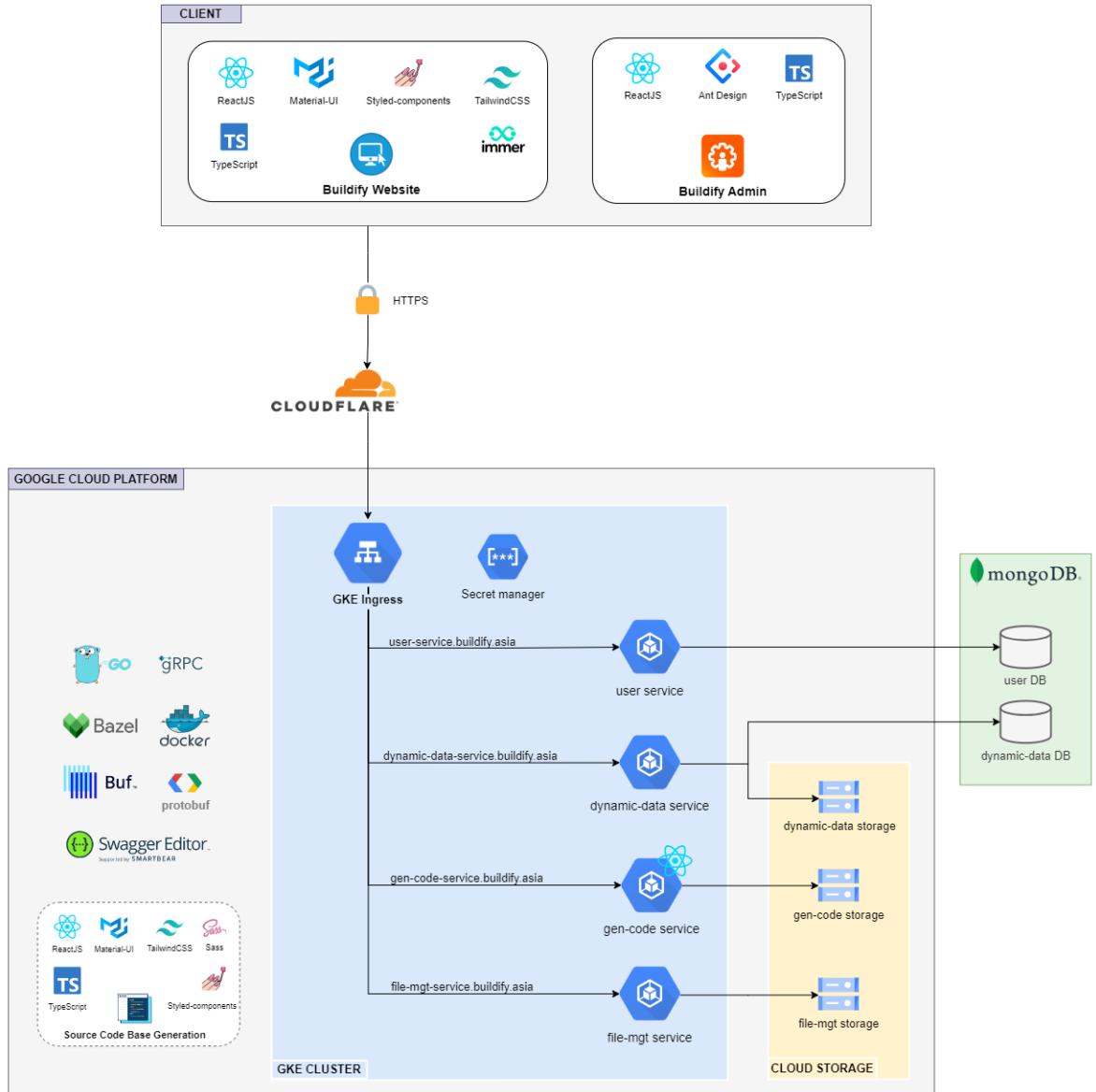
Hình 3.1: Sơ đồ use case

## 3.2 Thiết kế kiến trúc hệ thống

### 3.2.1 Sơ đồ kiến trúc tổng quan

Kiến trúc tổng quan của hệ thống được mô tả ở hình 3.2, với front-end là vùng client phía trên, bao gồm 2 thành phần là Buildify website (nơi

triển khai website builder) và Buildify admin. Back-end là vùng còn lại, triển khai mô hình microservices trên nền tảng Google Cloud với 4 services là user service, dynamic-data service, gen-code service và file-mgt service.



Hình 3.2: Sơ đồ kiến trúc tổng quan

## Front-end

1. Sử dụng ReactJS và TypeScript cho cả Buildify website và admin.
2. Sử dụng các thư viện Material-UI, Tailwind CSS, Styled-components để xây dựng giao diện người dùng cho Buildify website.

3. Sử dụng thư viện Ant Design để xây dựng giao diện người dùng cho Buildify admin.
4. Sử dụng thư viện Immer để quản lý trạng thái editor và xây dựng tính năng quản lý lịch sử.

## Back-end

1. GKE (Google Kubernetes Engine) cluster là một dịch vụ hàng đầu của GCP (Google Cloud Platform) được sử dụng để triển khai mô hình microservices. Với GKE, việc tạo và quản lý các services được đóng gói trong các container trở nên dễ dàng và linh hoạt, hỗ trợ tự động điều phối và cho phép mở rộng dễ dàng theo nhu cầu. Nhóm sử dụng các tài nguyên sau trong GKE:
  - GKE Ingress: đóng vai trò là ingress controller cho GKE cluster, hoạt động như một cổng kết nối với lưu lượng truy cập HTTP(S) từ bên ngoài, cung cấp tính năng định tuyến và cân bằng tải cho các services trong cluster.
  - Secret Manager: được sử dụng để lưu trữ và đảm bảo an toàn cho các thông tin cần được bảo mật, cho phép chia sẻ thông tin này cho các services trong cluster một cách an toàn và đáng tin cậy.
  - Deployment: sử dụng để triển khai và quản lý các phiên bản của các ứng dụng trong cluster.
  - Service: đại diện cho các services trong cluster, cung cấp một địa chỉ IP unique để truy cập vào service đó.
  - Managed Certificate: cung cấp khả năng quản lý chứng chỉ SSL cho các services trong cluster.
2. Đối với mỗi service, nhóm sử dụng các công nghệ và công cụ sau:

- Sử dụng ngôn ngữ lập trình Go để phát triển mã nguồn cho các services trong cluster.
  - Sử dụng framework gRPC để giao tiếp giữa các services trong cluster, cho phép truyền và nhận dữ liệu một cách hiệu quả và đáng tin cậy.
  - Sử dụng Buf để hỗ trợ biên dịch các file Protobuf thành các file Go và cung cấp giao diện Swagger UI để trực quan hóa API.
  - Sử dụng Bazel để biên dịch mã nguồn Go thành các image container.
3. Cloud Storage được sử dụng để lưu trữ và quản lý các tệp tin trong hệ thống.
  4. MongoDB là một cơ sở dữ liệu NoSQL được sử dụng để quản lý dữ liệu trong ứng dụng.
  5. Cloudflare là một dịch vụ quản lý DNS, proxy và cung cấp chứng chỉ SSL miễn phí, giúp tăng cường tính bảo mật và hiệu suất của hệ thống.

### 3.2.2 Các công nghệ được sử dụng ở front-end

#### ReactJS

ReactJS [4] là một thư viện JavaScript phổ biến và mạnh mẽ được sử dụng để xây dựng giao diện người dùng động trong ứng dụng web. Nó được phát triển bởi Facebook và hiện đang được cộng đồng lập trình viên rất ưa chuộng.

Một số lý do chọn ReactJS:

1. Hiệu suất cao: ReactJS sử dụng cây DOM ảo (virtual DOM) [7] để tối ưu hóa hiệu suất. Thay vì cập nhật toàn bộ DOM khi có thay

đổi giao diện, ReactJS chỉ cập nhật những phần cần thiết, giúp tăng tốc độ render và tương tác trên trang web.

2. Linh hoạt và dễ bảo trì: ReactJS sử dụng cấu trúc component, cho phép phân chia giao diện thành các thành phần độc lập (component) và tái sử dụng. Điều này giúp tạo ra mã nguồn rõ ràng, dễ đọc và dễ bảo trì.
3. Tư duy xây dựng component: ReactJS chia trang web thành từng component nhỏ, không giới hạn trong việc tùy chỉnh một component dựa trên danh sách props truyền vào được phép tự định nghĩa. Điều này rất phù hợp với cách triển khai ý tưởng của một website builder, dựa trên việc định nghĩa và quản lý danh sách các component (gọi là selector); cho phép người dùng kéo và thả trong danh sách này để thiết kế. Mỗi node trong trình editor cũng tương ứng là một thẻ hiện (instance) của một loại selector cụ thể. Hơn nữa, khi phát sinh mã nguồn, mỗi component page đều có mã JSX là một danh sách các component (ánh xạ tương ứng từ node) thuộc danh sách selector trên.
4. Cộng đồng lớn và hỗ trợ mạnh mẽ: ReactJS có một cộng đồng lập trình viên rất lớn và phong phú, với nhiều tài liệu, thư viện và công cụ hỗ trợ. Bạn có thể tìm thấy rất nhiều tài liệu học tập, ví dụ, và giải pháp cho các vấn đề phổ biến trong việc phát triển front-end.
5. Thư viện tham khảo website builder: tham khảo mã nguồn mở Craft.js [23] (phần lớn nguồn tham khảo về kiến trúc xây dựng và các xử lí logic chính) cũng được cài đặt bằng ReactJS.
6. Tích hợp dễ dàng: ReactJS có khả năng tích hợp với các thư viện và framework khác một cách mạnh mẽ, cộng đồng cực lớn nên có thể tích hợp đầy đủ framework cần thiết.
7. Khả năng quản lý trạng thái (state management) vô cùng mạnh mẽ:

ReactJS cung cấp một số công cụ cho việc quản lý trạng thái, nhất là trong các dự án lớn, trong đó context và hook useReducer là hai phương pháp phổ biến. Ý tưởng xây dựng ban đầu của website builder là quản lý tất cả trạng thái của editor trong một global state, cho phép truy cập dữ liệu từ bất kỳ component nào và một loạt các hành động (action) cho phép thay đổi từng phần nhỏ dữ liệu bên trong.

## TypeScript

TypeScript là một phiên bản mở rộng của JavaScript, được phát triển bởi Microsoft và cung cấp tính năng kiểu tĩnh (static typing) cho JavaScript. TypeScript cho phép xác định rõ kiểu dữ liệu của biến, tham số và giá trị trả về trong quá trình phát triển ứng dụng.

Một số lý do chọn TypeScript:

1. Kiểm tra lỗi tại thời điểm biên dịch: TypeScript giúp phát hiện và báo lỗi các lỗi ngữ cảnh, sai kiểu dữ liệu và các lỗi khác tại thời điểm biên dịch. Điều này giúp tăng tính tin cậy và giảm số lỗi (nhất là lỗi nhỏ) trong quá trình phát triển và lúc chạy ứng dụng (runtime).
2. Tiện lợi hơn khi phát triển: với TypeScript, có thể tường minh hóa kiểu dữ liệu và có các thông tin hữu ích về API của các thư viện và framework. Điều này giúp tăng tốc độ code trong quá trình phát triển, đồng thời giúp code dễ hiểu hơn, tốt cho nhóm khi làm việc chung.
3. Mở rộng khả năng tái sử dụng mã: TypeScript cho phép xác định interface, class và kiểu tùy chỉnh, tạo ra mã dễ dàng tái sử dụng và dễ bảo trì. Nó cung cấp tính năng module và namespace để tổ chức mã nguồn và giảm xung đột giữa các thư viện và module khác nhau.
4. Tích hợp với ESLint để tạo mã nguồn sạch và chuẩn: khi đi cùng với ESLint, một công cụ phân tích mã nguồn mã nguồn mở giúp kiểm

tra và báo lỗi về quy tắc lập trình trong dự án, giúp tạo ra mã nguồn chất lượng cao, tuân thủ các quy tắc lập trình phổ biến (convention) và kiểm soát tốt hơn về kiểu dữ liệu. Điều này giúp tăng tính đáng tin cậy và dễ bảo trì.

## NextJS

Next.js [8] là một framework ReactJS được phát triển dưới dạng mã nguồn mở, bổ sung các khả năng tối ưu hóa như render phía máy chủ (Server-side Rendering) và tạo trang web static. Next.js xây dựng dựa trên thư viện ReactJS, có nghĩa là các ứng dụng Next.js sử dụng core của ReactJS và chỉ thêm các tính năng bổ sung. Việc triển khai ứng dụng SSR cho phép máy chủ truy cập tất cả dữ liệu được yêu cầu và xử lý JavaScript cùng nhau để hiển thị trang. Sau đó, trang được gửi lại toàn bộ cho trình duyệt và ngay lập tức được hiển thị. SSR cho phép các trang web load trong thời gian nhỏ nhất và tăng trải nghiệm người dùng với khả năng phản hồi nhanh hơn.

Một số lý do chọn NextJS:

1. Rendering phía máy chủ (server-side rendering - SSR): Next.js cho phép ứng dụng web được render trước trên máy chủ trước khi được gửi đến trình duyệt, giúp cải thiện tốc độ tải trang và SEO. Điều này đảm bảo rằng nội dung của trang web có sẵn ngay từ ban đầu, không cần chờ đợi các yêu cầu AJAX hay tải tài nguyên từ máy chủ.
2. Routing tự động: Next.js cung cấp routing tự động dựa trên cấu trúc thư mục, giúp việc quản lý các tuyến đường trở nên dễ dàng và tự nhiên. Bằng cách đặt các file vào các thư mục tương ứng, có thể tạo các tuyến đường trong ứng dụng một cách tự động.
3. Tối ưu hóa hiệu suất: Next.js có các tính năng tối ưu hóa hiệu suất như tải chậm (code splitting), dự đoán để prefetching dữ liệu (predictive prefetching) và tối ưu hóa dữ liệu tĩnh (static optimization,

tối ưu hóa cho ứng dụng bằng cách tạo ra các tập tin HTML tĩnh trong quá trình build). Điều này giúp cải thiện tốc độ tải trang và trải nghiệm người dùng.

4. Cộng đồng phát triển đông đảo: Next.js có một cộng đồng phát triển rộng lớn, với nhiều tài liệu, ví dụ và gói mở rộng hữu ích. Có thể tìm thấy sự hỗ trợ và tư vấn từ cộng đồng khi gặp vấn đề trong quá trình phát triển.
5. Hot Module Replacement (HMR): Next.js hỗ trợ HMR, cho phép thay đổi mã nguồn của ứng dụng mà không cần tải lại trang hoặc mất trạng thái. Điều này giúp tăng năng suất phát triển và tốc độ phản hồi khi chỉnh sửa mã.
6. Hỗ trợ TypeScript: Next.js hỗ trợ TypeScript một cách nền tảng. Bạn có thể sử dụng TypeScript để viết mã trong dự án Next.js của mình, giúp tăng tính tự tin, xác thực và khả năng tái sử dụng của mã.

## State management với ReactJS context và useReducer

Trong ReactJS, Context [24] và useReducer [26] là hai công cụ quan trọng được sử dụng để quản lý trạng thái (state management) trong ứng dụng. Khi kết hợp với nhau, chúng tạo nên một hệ thống quản lý trạng thái mạnh mẽ và linh hoạt.

ReactJS Context là một tính năng cho phép chia sẻ trạng thái giữa các thành phần con trong cây thành phần mà không cần truyền qua các thành phần cha. Nó giúp giảm bớt sự phức tạp của việc truyền trạng thái qua nhiều cấp thành phần và làm cho mã nguồn trở nên sạch hơn. Để sử dụng Context, cần tạo một Context bằng cách sử dụng hàm createContext(). Sau đó, sử dụng giá trị Context.Provider trả về bọc (wrap) tất cả những phần của ứng dụng cần cung cấp trạng thái. Khi cần truy cập và sử dụng trạng thái từ các Component con bên trong, sử dụng Context.Consumer

để truy cập. Bằng cách này, các Component con có thể truy cập vào trạng thái chung và cập nhật nó khi cần thiết.

useReducer là một hook trong ReactJS được sử dụng để quản lý trạng thái và xử lý hành động (action). Nó là một hàm pure function nhận vào hai tham số là reducer và trạng thái ban đầu, và trả về trạng thái mới và dispatch function để gửi hành động tới reducer. Reducer là một hàm nhận vào trạng thái hiện tại và hành động, và dựa vào hành động đó, nó xử lý và trả về một trạng thái mới. Khi dispatch được gọi, reducer được kích hoạt và trạng thái được cập nhật tương ứng.

Khi kết hợp context và useReducer, có thể xây dựng một hệ thống quản lý trạng thái mạnh mẽ. Thông qua context, có thể chia sẻ trạng thái với các component con một cách dễ dàng. Với useReducer, có thể xử lý các hành động và cập nhật trạng thái một cách cấu trúc và nhất quán. Bằng cách này, việc quản lý trạng thái trở nên dễ dàng hơn, mã nguồn trở nên sạch hơn và khả năng mở rộng cũng được cải thiện.

Tuy nhiên, việc sử dụng context và useReducer không phải lúc nào cũng là lựa chọn tốt nhất cho mọi dự án, nó tỏ ra kém hiệu quả trong những dự án nhỏ và đơn giản, không cần quản lý và dùng chung nhiều dữ liệu.

## Material-UI

Material-UI là một thư viện giao diện người dùng (UI) cho ReactJS, được xây dựng dựa trên nguyên lý thiết kế của Material Design của Google. Nó cung cấp một bộ các thành phần UI sẵn có, rất đẹp và đa dạng, dễ dàng tùy chỉnh, các hệ thống lưới (grid) linh hoạt và các quy tắc thiết kế để giúp xây dựng giao diện web đẹp mắt và hiệu quả.

Lý do chọn Material-UI:

- Thiết kế hấp dẫn: các thành phần và giao diện được cung cấp bởi Material-UI đảm bảo có giao diện sạch, mạnh mẽ và hấp dẫn, giúp tạo ra trải nghiệm người dùng tốt.

2. Tích hợp tốt với ReactJS: Material-UI được xây dựng đặc biệt cho ReactJS, vì vậy nó tận dụng tối đa các tính năng và hiệu suất của ReactJS. Nó sử dụng cú pháp JSX và hỗ trợ tốt việc quản lý trạng thái và rendering hiệu quả của ReactJS.
3. Tiết kiệm thời gian và công sức: sử dụng những thành phần UI sẵn có để tạo ra giao diện nhanh chóng và tiết kiệm thời gian.
4. Tùy chỉnh linh hoạt: mặc dù Material-UI cung cấp các thành phần UI sẵn có, nhưng vẫn có thể tùy chỉnh chúng để phù hợp với nhu cầu thiết kế của mình.
5. Cung cấp sẵn rất nhiều tùy chỉnh về Theme.
6. Cộng đồng lớn và hỗ trợ tốt.

## Ant Design

Ant Design [2] cũng là một thư viện giao diện UI cho ReactJS (tương tự với Material-UI), cung cấp một bộ công cụ thiết kế giao diện người dùng đẹp và chuyên nghiệp cho ứng dụng web, cho phép sử dụng để triển khai sản phẩm hoàn chỉnh ngay lập tức. Nó cung cấp các thành phần UI đã được xây dựng sẵn như nút bấm, biểu mẫu, bảng, menu, tiện ích...

Lý do chọn Ant Design:

1. Ant Design rất mạnh trong xây dựng các trang admin: Ant Design hỗ trợ rất tốt trong việc tạo dashboard nhờ component Layout. Còn với thư viện Material-UI, nếu muốn tạo một dashboard thì lập trình viên sẽ phải tốn khá nhiều công sức xây dựng [5].
2. Thiết kế chuyên nghiệp: Ant Design cung cấp một giao diện người dùng thẩm mỹ và chuyên nghiệp. Các thành phần UI đã được thiết kế và kiểm tra kỹ lưỡng, giúp tạo ra một giao diện admin chất lượng cao và chuyên nghiệp.

3. Da dạng các thành phần UI: Ant Design cung cấp một bộ sưu tập đa dạng các thành phần UI sẵn có, rất phong phú và đầy đủ tính năng. Các thành phần này đáp ứng đủ nhu cầu xây dựng một trang admin cơ bản của nhóm.
4. Tùy chỉnh rất linh hoạt, đơn giản và dễ sử dụng.
5. Cộng đồng lớn và hỗ trợ tốt.

## Styled-Components

Styled-Components [15] là một thư viện CSS-in-JS cho phép viết CSS trong các component ReactJS bằng cú pháp tương tự như CSS thông thường nhưng được kết hợp và quản lý trực tiếp trong mã JavaScript.

Lý do chọn Styled-Components:

1. Tích hợp giao diện và logic: Styled-Components cho phép tạo ra các component ReactJS và định nghĩa các kiểu CSS cho chúng trong file JSX. Điểm đặc biệt này vô cùng phù hợp với **tính năng phát sinh mã nguồn** của nhóm, khi mà toàn bộ UI của component được đọc và tích hợp từ **props** của ReactJS component (sẽ được trình bày ở các phần sau).
2. Tự động tạo ra các class duy nhất: khi sử dụng Styled-Components, mỗi component sẽ được tự động tạo ra một class duy nhất. Điều này giúp **tránh xung đột tên class** và tạo ra một cấu trúc CSS rõ ràng và dễ bảo trì.
3. Hỗ trợ tái sử dụng và kế thừa: Styled-Components cho phép tái sử dụng các kiểu CSS và kế thừa từ các component khác. Có thể tạo ra các component tái sử dụng và một cấu trúc CSS module dễ quản lý.
4. Tính tương thích tốt với nhiều công cụ phát triển ReactJS khác nhau.
5. Cú pháp gần gũi: Sử dụng cú pháp tương tự CSS thông thường.

## Tailwind CSS

Tailwind CSS [1] là một thư viện CSS có cấu trúc lớp (class) được thiết kế để giúp phát triển web nhanh chóng hơn bằng cách cung cấp các lớp CSS trực tiếp có sẵn để tái sử dụng. Thay vì viết CSS từ đầu, Tailwind CSS cho phép tạo giao diện bằng cách kết hợp các lớp CSS để xây dựng nhanh chóng các thành phần UI.

Lý do chọn Tailwind CSS:

1. Tăng tốc phát triển: xây dựng các thành phần UI nhanh chóng và dễ dàng với các CSS được định nghĩa sẵn.
2. Tiết kiệm công sức và giảm số lượng file CSS: không cần phải nghĩ tên class để đặt cho thành phần (cần có ý nghĩa và không trùng lặp nhau, các class có hệ thống). Tránh việc thay đổi ít (đôi khi chỉ cần chỉnh 1 thuộc tính của một tag html) mà phải tạo cả một file CSS riêng.

## React Hook Form

React Hook Form [3] là một thư viện quản lý form trong ReactJS, được xây dựng dựa trên cơ chế hook. Nó cung cấp các công cụ và API giúp đơn giản hóa việc xử lý, kiểm tra hợp lệ (validation) và thông báo lỗi của Form trong ứng dụng ReactJS.

Lý do chọn React Hook Form:

1. Sử dụng hook: React Hook Form sử dụng cơ chế hook trong ReactJS, giúp quản lý trạng thái và tương tác với biểu mẫu một cách đơn giản và hiệu quả hơn, đồng thời cũng phù hợp với phong cách code **Functional Components** của dự án.
2. Validation linh hoạt: React Hook Form cung cấp cách tiếp cận đơn giản và linh hoạt để kiểm tra hợp lệ và cập nhật thông báo lỗi cho các trường dữ liệu trong form. Hỗ trợ đầy đủ các quy tắc kiểm tra

hợp lệ như kiểm tra bắt buộc, kiểm tra định dạng, kiểm tra độ dài và các **quy tắc tùy chỉnh khác** do người dùng tự định nghĩa. Ngoài ra, việc tự thiết kế một form từ đầu với cơ chế validation chung và quản lý dữ liệu của các trường tồn rất nhiều thời gian và công sức.

3. Hiệu suất tốt: thư viện tập trung vào việc cung cấp các API nhẹ nhàng và tối ưu hóa hiệu suất. React Hook Form sử dụng kỹ thuật uncontrolled component (kỹ thuật uncontrolled component là một phương pháp trong ReactJS để xử lý các thành phần form mà không cần theo dõi và kiểm soát giá trị của các input trong state của Component). Điều này giúp giảm tải và tối ưu hóa hiệu suất trong quá trình xử lý form.
4. Hỗ trợ cho việc xử lý tương tác: React Hook Form cung cấp các API để xử lý tương tác như điều hướng, xóa giá trị, cập nhật giá trị, theo dõi giá trị, xác nhận và kiểm tra hợp lệ của form. Điều này giải quyết được hầu hết các yêu cầu tương tác trong dự án.
5. Tài liệu phong phú: React Hook Form cung cấp một tài liệu đầy đủ và dễ hiểu, đi kèm với ví dụ và hướng dẫn chi tiết.
6. Cộng đồng lớn và hỗ trợ tốt.

## Immer

Immer [20] là một thư viện JavaScript được sử dụng để thay đổi dữ liệu không được thay đổi trực tiếp (immutable data) một cách dễ dàng và đơn giản. Nó cung cấp một cú pháp đơn giản và dễ hiểu để thay đổi trạng thái của đối tượng, mảng hoặc bất kỳ cấu trúc dữ liệu nào mà không làm thay đổi trực tiếp đối tượng gốc.

Bằng cách sử dụng Immer, có thể tạo ra các hàm thay đổi dữ liệu một cách dễ dàng, dễ đọc và không gây hiệu ứng phụ. Thay vì phải thực hiện các hoạt động như sao chép đối tượng, tạo ra các bản sao và thực hiện các

phép toán thay đổi, chỉ cần mô tả các thay đổi muốn áp dụng và Immer sẽ xử lý những công việc phức tạp đó một cách tự động. Immer được áp dụng trong dự án như sau:

1. **Ứng dụng tốt trong hệ thống state management:** Immer giúp thực hiện việc thay đổi trạng thái một cách an toàn và dễ dàng, phù hợp cơ chế quản lý toàn bộ thông tin của editor thông qua một state management của dự án.
2. **Hỗ trợ cơ chế quản lý lịch sử (history):** Immer giúp theo dõi các thay đổi trong lịch sử và xử lý việc hoàn tác (undo) hoặc làm lại (redo) các thay đổi. Thay vì lưu toàn bộ các bản sao của trạng thái tại mỗi thay đổi, Immer sử dụng kỹ thuật copy-on-write để chỉ lưu các thay đổi và tạo ra các phiên bản mới khi cần thiết (kỹ thuật copy-on-write là một phương pháp trong lập trình để tạo ra bản sao mới của một đối tượng hoặc cấu trúc dữ liệu khi thực hiện thay đổi. Thay vì thay đổi trực tiếp trạng thái hiện tại, một bản sao mới được tạo ra và các thay đổi chỉ được áp dụng trên bản sao đó, giữ nguyên trạng thái ban đầu). Điều này giúp tối ưu bộ nhớ và tăng hiệu suất khi quản lý lịch sử của ứng dụng. Ứng dụng dùng Immer để cài đặt tính năng này.

## Vite

Vite là một công cụ phát triển ứng dụng web hiệu suất cao và tối ưu cho các dự án sử dụng JavaScript và TypeScript. Nó được tạo ra để cung cấp một quá trình phát triển nhanh chóng và trải nghiệm tương tác mượt mà cho các nhà phát triển.

Vite sử dụng ES modules (mô-đun ECMAScript) trong trình duyệt để tải các tệp JavaScript một cách nhanh chóng và hiệu quả. Thay vì đóng gói toàn bộ mã nguồn thành một tệp duy nhất như các công cụ build truyền thống, Vite sẽ phân tách các module thành các tệp riêng biệt, giúp tăng tốc quá trình tải và build ứng dụng.

Nhóm chọn Vite vì một số lý do sau:

1. Hiệu suất tốt: việc dùng ES modules (một cách sử dụng và quản lý module trong JavaScript theo chuẩn ECMAScript để tạo và tải các phần mở rộng (extensions) một cách hiệu quả và nhanh chóng) trong trình duyệt để xây dựng và chạy ứng dụng, giúp giảm thời gian khởi động và tăng tốc hiệu suất của ứng dụng. Nó tận dụng các tính năng của trình duyệt như tải động (dynamic loading) và thay thế module nóng (hot module replacement) để cung cấp trải nghiệm phát triển nhanh chóng.
2. Phát triển nhanh chóng: với Vite, có thể tận dụng ngay các tính năng mới của ES modules, như import on demand (tải module theo yêu cầu), để xây dựng ứng dụng một cách nhanh chóng. Việc sử dụng Vite không đòi hỏi quá trình đóng gói trước khi chạy ứng dụng, giúp tiết kiệm thời gian phát triển.
3. Tích hợp dễ dàng: Vite hỗ trợ sẵn các công cụ phát triển như TypeScript, CSS pre-processors (SASS, LESS), và JSX, giúp viết mã và quản lý dự án dễ dàng hơn. Bạn có thể tùy chỉnh và mở rộng Vite theo nhu cầu của dự án.
4. Hỗ trợ tốt cho ReactJS.

## Vercel

Vercel [6] là một nền tảng dành cho nhà phát triển cung cấp các công cụ, quy trình làm việc và cơ sở hạ tầng cần thiết để xây dựng và triển khai ứng dụng web nhanh mà không cần cấu hình phức tạp. Vercel hỗ trợ các framework phổ biến của front-end một cách dễ dàng và không cần thêm bất kỳ cấu hình bổ sung nào. Cơ sở hạ tầng của Vercel có khả năng mở rộng và bảo mật, được phân phối toàn cầu để phục vụ nội dung từ các trung tâm dữ liệu gần người dùng, mang lại tốc độ tối ưu.

Vercel được chọn vì tính năng cao, hiệu suất tốt, tích hợp dễ dàng và hỗ trợ triển khai tự động tốt không cần cấu hình (zero configuration) cho các công nghệ mà nhóm đang sử dụng (NextJS cho Buildify website, Vite cho Buildify admin) khi chỉ cần kết nối với repository tương ứng trên GitHub. Như vậy, quy trình CI/CD trở nên đơn giản hơn bao giờ hết. Ngoài ra, Vercel còn hỗ trợ những tính năng sau cho trang web:

1. Giám sát hiệu suất tự động (automatic performance monitoring).
2. Tự động HTTPS và chứng chỉ SSL.
3. Tự động CI/CD (through GitHub, GitLab, Bitbucket, v.v.).
4. Hỗ trợ biến môi trường.
5. Hỗ trợ tên miền tùy chỉnh.
6. Hỗ trợ tối ưu hóa hình ảnh.
7. Triển khai ứng dụng tức thì thông qua lệnh "git push".

### 3.2.3 Các mô hình và công nghệ được sử dụng ở backend

#### Bazel

Bazel [10] là một công cụ build phần mềm mã nguồn mở, phát triển bởi Google và được sử dụng phổ biến trong cộng đồng phát triển phần mềm. Bazel hỗ trợ đa ngôn ngữ và được thiết kế để hỗ trợ việc build đa nền tảng, phát triển phần mềm có cấu trúc lớn, cho phép quản lý và kiểm soát các phụ thuộc giữa các thành phần phần mềm khác nhau, giúp đảm bảo tính nhất quán và độ tin cậy trong quá trình phát triển phần mềm. Hơn nữa, Bazel được thiết kế để tận dụng tối đa việc sử dụng bộ nhớ cache và các kỹ thuật tối ưu hóa khác để mang lại tốc độ và hiệu suất cao cho quá trình

bien dịch. Với những ưu điểm trên, nhóm lựa chọn Bazel là công cụ biên dịch phần mềm cho các services của back-end, giúp quá trình triển khai server được dễ dàng và nhanh chóng hơn.

## Kiến trúc Microservices

Microservices là một kiến trúc phần mềm linh hoạt cho phép phân tách ứng dụng thành các module service độc lập. Mỗi service được phát triển và triển khai độc lập so với các service khác. Để thiết kế được kiến trúc microservices cho dự án cụ thể, lập trình viên cần có kinh nghiệm và khả năng phân tích tốt để chia tách service, cơ sở dữ liệu sao cho hợp lý và giảm sự phụ thuộc lẫn nhau nhất, đảm bảo được các tính chất của microservices. Lợi ích của việc sử dụng kiến trúc microservices bao gồm:

- Khả năng mở rộng với quy mô lớn.
- Dễ bảo trì và nâng cấp hệ thống.
- Nếu một service gặp sự cố, hệ thống vẫn hoạt động bình thường.
- Mỗi service có thể sử dụng công nghệ, framework, ngôn ngữ và cơ sở dữ liệu khác nhau.
- Dễ dàng hơn trong việc tích hợp các bên thứ ba.
- Mỗi service được phát triển bởi một nhóm nhỏ, giảm thiểu xung đột và sự phụ thuộc, chờ đợi lẫn nhau, đồng thời cũng dễ dàng hơn cho các thành viên trong nhóm hiểu rõ nghiệp vụ của service.
- Phù hợp với các hệ thống lớn, quy trình nghiệp vụ có yêu cầu thay đổi thường xuyên.

Do vậy, hiện nay kiến trúc microservices được áp dụng rộng rãi tại các công ty lớn như Netflix, Amazon, Twitter, eBay, PayPal... Tuy nhiên một số nhược điểm đáng quan tâm của microservices là phức tạp hơn monolithic

rất nhiều, gây khó khăn trong việc thực hiện kiểm thử tích hợp, quy trình triển khai phức tạp hơn kèm theo đó là khá nhiều sự cố như lỗi kết nối chậm, không vẹn toàn cơ sở dữ liệu, thông điệp gửi đến không đúng thứ tự... Do đó cần cân nhắc kỹ khi lựa chọn kiến trúc microservices và chỉ áp dụng khi thực sự cần. Nhóm đã lựa chọn kiến trúc microservices cho back-end với mục đích học tập và làm quen với nhiều công nghệ mới hiện nay.

## Mô hình monorepo

Monorepo và polyrepo là hai mô hình quản lý mã nguồn phổ biến hiện nay. Sau khi cân nhắc ưu nhược điểm của hai mô hình trên (bảng 3.1), nhóm lựa chọn cấu trúc monorepo cho mã nguồn của back-end với mục đích dễ quản lý, dễ bảo trì và tái cấu trúc dễ dàng hơn.

Bảng 3.1: So sánh monorepo với polyrepo

	Monorepo	Polyrepo
Định nghĩa	Là kiểu cấu trúc dự án trong đó tất cả module đều nằm trong cùng một git repository.	Là kiểu cấu trúc dự án trong đó mỗi module được chứa ở những git repository riêng lẻ.
Ưu điểm	<ul style="list-style-type: none"> <li>• Dễ dàng chia sẻ code giữa các modules.</li> <li>• Việc tổ chức mã nguồn được liền mạch và nhất quán, giảm chi phí quản lý.</li> <li>• Cải thiện văn hóa làm việc chung giữa các team</li> <li>• Tái cấu trúc dễ dàng hơn.</li> </ul>	<ul style="list-style-type: none"> <li>• Phân quyền rõ ràng.</li> <li>• Giảm xung đột code.</li> <li>• Codebase không quá lớn nên dễ dàng quản lý.</li> </ul>

Khuyết điểm	<ul style="list-style-type: none"> <li>• Tất cả các module trong cùng một repo khiến lịch sử commit bị rối khi có commit của những module không liên quan.</li> <li>• Khi phân quyền truy cập repository nghĩa là phân quyền truy cập cho tất cả module, mặc dù người được phân quyền không có trách nhiệm, bỗn phận trong các module mà họ “vô tình” được phân quyền.</li> <li>• Nằm chung trong một repository khiến tất cả module phải áp dụng cùng một git workflow. Điều này làm mất đi tính linh hoạt, vì tùy tính chất của từng team, đặc điểm của từng module mà sẽ áp dụng những workflow khác nhau.</li> <li>• Kích thước của repository sẽ càng ngày càng phình to một cách dư thừa vì chứa tất cả module và những thay đổi, branch trên tất cả module.</li> </ul>	<ul style="list-style-type: none"> <li>• Các team trở nên phân tán và có thể không biết rõ mục tiêu chính của dự án mà mình tham gia và do đó không bận tâm về những gì người khác đang làm miễn là công việc của họ được hoàn thành.</li> <li>• Cách lưu trữ và quản lý này mất khá nhiều thời gian cũng như tài nguyên.</li> <li>• Không thống nhất về style code.</li> </ul>
-------------	---	---

## Go

Go [16] (hay còn gọi là Golang) là một ngôn ngữ lập trình mã nguồn mở được phát triển bởi Google vào năm 2007 và được phát hành chính thức vào năm 2009. Go giúp đơn giản hóa quá trình phát triển phần mềm, tăng tốc độ biên dịch và thực thi chương trình, và đặc biệt phù hợp cho các ứng dụng web, cloud, và các hệ thống phân tán. Go được thiết kế với mục đích tối ưu hiệu suất, đơn giản và dễ dàng sử dụng, đồng thời hỗ trợ đa nền tảng và dễ dàng tích hợp với các công nghệ khác.

Điều khiến Go trở nên khác biệt chính là goroutine. Goroutine là một hàm có thể chạy đồng thời với các hàm khác, được quản lý bởi goroutine scheduler của Go runtime. Bảng 3.2 cho thấy rõ ưu điểm của goroutine so với thread.

Bảng 3.2: So sánh goroutine với thread

Tiêu chí	Goroutine	Thread
Kích thước	Kích thước nhỏ hơn rất đáng kể so với thread. Goroutines chỉ sử dụng 2KB memory stack.	Các OS thread có thể lên đến 2MB.
Bộ nhớ	Linh động tăng giảm bộ nhớ sử dụng.	Cố định.
Số lượng	Một chương trình Go có thể có hàng trăm nghìn goroutine.	Từ vài trăm đến hàng nghìn.
Thời gian khởi động	Nhanh hơn.	Chậm hơn.

Giao tiếp	Goroutines có khả năng giao tiếp an toàn với nhau thông qua các kênh (channel). Các kênh này hỗ trợ cơ chế mutex lock (cơ chế đồng bộ hóa trong lập trình đa luồng), giúp ngăn chặn các lỗi liên quan đến ghi và đọc đồng thời trên các vùng dữ liệu chia sẻ (data race). Nhờ vậy, Go đảm bảo tính an toàn và độ tin cậy trong việc truyền thông giữa các goroutines.	Chia sẻ dữ liệu khó khăn hơn và phải tự quản lý nên có thể xảy ra data race.
-----------	---	--

## gRPC

gRPC [17] là một framework mã nguồn mở được phát triển bởi Google dùng để hiện thực hóa RPC (Remote Procedure Calls) bằng cách làm cho nó có thể tương thích, hiện đại và hiệu quả nhờ sử dụng các công nghệ như Protocol buffers và HTTP/2. Các khái niệm liên quan được trình bày ở bảng 3.3.

Bảng 3.3: Các khái niệm liên quan đến gRPC

Khái niệm	Định nghĩa
-----------	------------

RPC	Là một kiểu giao tiếp client-server trong đó client gọi một hàm (procedure) trên server và nhận kết quả trả về. Khác với REST API sử dụng HTTP, RPC không giới hạn phương thức giao tiếp và có thể sử dụng các giao thức khác như TCP, UDP, hay giao thức tùy chỉnh. RPC đã tồn tại từ lâu và được sử dụng rộng rãi trong các hệ thống phân tán, cho phép các thành phần trong hệ thống giao tiếp và làm việc với nhau thông qua các hàm gọi từ xa.
HTTP/2	Là phiên bản tiếp theo của giao thức HTTP được phát triển nhằm cải thiện hiệu suất và tăng cường tính năng so với HTTP/1. Một số đặc điểm của HTTP/2: <ul style="list-style-type: none"> <li>• Multiplexing: HTTP/2 hỗ trợ multiplexing, cho phép gửi nhiều request đồng thời trên cùng một kết nối TCP giúp tăng tốc độ tải trang web đáng kể.</li> <li>• Nén header với HPACK: HTTP/2 sử dụng HPACK để nén các header request và response. HPACK sử dụng một cơ chế đơn giản và bảo mật để nén lại các header, client và server duy trì danh sách header đã trao đổi, từ đó HTTP/2 chỉ gửi các header khác biệt so với các header trước đó. Điều này giúp giảm lượng dữ liệu truyền tải và tăng hiệu suất.</li> <li>• Server push: HTTP/2 cho phép server tự động gửi thông tin tới client mà không cần client yêu cầu, giúp giảm độ trễ và cải thiện hiệu suất bằng cách đẩy các tài nguyên liên quan trước khi client yêu cầu.</li> </ul>

Protocol buffers (protobuf)	<p>Là một ngôn ngữ độc lập với cơ chế serialize dữ liệu phát triển bởi Google, cho phép lập trình viên định nghĩa cấu trúc dữ liệu trong một file proto, sau đó sử dụng Protoc compiler để biên dịch sang mã nguồn của một ngôn ngữ lập trình cụ thể mà protobuf hỗ trợ. Khi chạy, dữ liệu được đóng gói và chuẩn hóa sang dạng nhị phân theo định dạng đã được định nghĩa trong file proto.</p> <p>Protobuf đem lại nhiều lợi ích, bao gồm việc giảm kích thước dữ liệu, tăng tốc độ truyền tải và xử lý dữ liệu, đồng thời cung cấp tính linh hoạt và tương thích đa ngôn ngữ.</p>
--------------------------------	--

Nhóm chọn gRPC vì những ưu điểm sau:

- Nhiều lựa chọn kiểu kết nối: gRPC hỗ trợ truyền dữ liệu theo kiến trúc event-driven với các kiểu server-side streaming, client-side streaming và bidirectional streaming.
- Dữ liệu nhỏ gọn: trung bình kích thước dữ liệu của gRPC nhỏ hơn khoảng 30% so với định dạng JSON.
- Hiệu suất cao: gRPC có tốc độ giao tiếp nhanh hơn nhiều lần so với JSON hoặc XML.
- Da ngôn ngữ: gRPC cho phép tự động tạo mã nguồn cho nhiều ngôn ngữ lập trình khác nhau chỉ từ file .proto.
- Chủ động hủy kết nối: gRPC client có khả năng hủy kết nối gRPC khi nó không cần nhận phản hồi từ server nữa, giúp tăng hiệu suất cho các server với nhiều yêu cầu.

gRPC cũng có nhiều nhược điểm, tuy nhiên đây là những nhược điểm khi muốn áp dụng gRPC ở trình duyệt web, do đó nhóm vẫn sẽ sử dụng

RESTful API cho giao tiếp giữa front-end và back-end. Còn đối với việc giao tiếp giữa các services trong mô hình microservices ở back-end, nhóm cho rằng những nhược điểm nêu ra dưới đây có thể là lợi thế:

- **Hạn chế hỗ trợ trên trình duyệt:** gRPC phụ thuộc vào giao thức HTTP/2, nhưng hiện tại các trình duyệt chưa hỗ trợ gọi trực tiếp tới giao thức HTTP/2, điều này làm hạn chế sử dụng gRPC trực tiếp trên trình duyệt. Thay vào đó, cần sử dụng proxy để chuyển tiếp gọi gRPC, gây ra một số hạn chế.
- **Dữ liệu không thân thiện:** dữ liệu trong gRPC được nén lại dạng nhị phân nên không thân thiện với lập trình viên. Để debug, phân tích payload hay tạo request thủ công, lập trình viên cần sử dụng thêm các công cụ hỗ trợ.

## MongoDB

MongoDB [21] là một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL) mã nguồn mở được thiết kế để lưu trữ và truy xuất dữ liệu một cách linh hoạt và mở rộng. Nó sử dụng kiểu dữ liệu JSON-like (BSON) để lưu trữ dữ liệu, cho phép lưu trữ các cấu trúc phức tạp, linh hoạt và thay đổi dễ dàng. Nhóm chọn MongoDB vì các ưu điểm sau:

- **Tính linh hoạt và dễ sử dụng:** MongoDB cho phép lưu trữ dữ liệu không đồng nhất và có cấu trúc thay đổi dễ dàng. Không cần định nghĩa cấu trúc tĩnh trước khi lưu trữ dữ liệu, giúp linh hoạt trong quá trình phát triển ứng dụng. Nó cũng cung cấp một ngôn ngữ truy vấn linh hoạt và dễ sử dụng để truy xuất dữ liệu.
- **Khả năng mở rộng:** MongoDB hỗ trợ mở rộng cơ sở dữ liệu theo chiều dọc bằng cách thêm các trường vào tài liệu hiện có, đồng thời cũng hỗ trợ mở rộng ngang bằng cách chia dữ liệu thành nhiều server, cho phép xử lý khối lượng lớn dữ liệu và tăng khả năng chịu tải của ứng dụng.

- Hiệu suất cao và tốc độ truy vấn: MongoDB được thiết kế để cung cấp hiệu suất cao và tốc độ truy vấn nhanh. Với cơ chế lưu trữ dựa trên bộ nhớ và các truy vấn tối ưu hóa, nó có thể xử lý các tải công việc lớn mà không gây tắc nghẽn.
- Cộng đồng mạnh mẽ: MongoDB có một cộng đồng lớn và phát triển năng động. Điều này có nghĩa là có sự hỗ trợ và tài liệu phong phú từ cộng đồng, giúp giải quyết vấn đề và tìm hiểu về MongoDB một cách dễ dàng.

## Docker

Docker [14] là một nền tảng ảo hóa mức hệ điều hành (OS-level virtualization) cho phép đóng gói ứng dụng và các phụ thuộc của chúng vào các container nhẹ nhưng độc lập. Container là một môi trường chạy độc lập, bao gồm tất cả các thành phần cần thiết để chạy một ứng dụng, bao gồm cả thư viện, phần mềm và các tài nguyên. Nhờ vậy, Docker giúp đơn giản hóa việc triển khai ứng dụng và đảm bảo tính nhất quán trên nhiều môi trường chạy khác nhau. Một số ưu điểm của Docker bao gồm:

- Dóng gói độc lập: Docker cho phép đóng gói ứng dụng và các phụ thuộc của nó vào container độc lập, đảm bảo tính di động và khả năng chạy ứng dụng trên nhiều môi trường mà không cần lo lắng về các khác biệt hệ thống.
- Tiết kiệm tài nguyên: Docker sử dụng cùng một kernel hệ điều hành chung cho các container, giúp tiết kiệm tài nguyên hệ thống so với việc chạy các máy ảo đầy đủ.
- Tính nhất quán và dễ triển khai: Docker đảm bảo tính nhất quán của môi trường chạy ứng dụng trên mọi môi trường, từ máy tính cá nhân đến máy chủ và đám mây. Điều này giúp dễ dàng triển khai và mở rộng ứng dụng.

- Tích hợp và linh hoạt: Docker được tích hợp với các công cụ phát triển phổ biến như Kubernetes, CI/CD pipelines và các công cụ quản lý hạ tầng. Nó cũng cho phép tùy chỉnh và mở rộng thông qua các Dockerfile và Docker Compose.
- Cộng đồng và hệ sinh thái lớn: Docker có một cộng đồng lớn và hệ sinh thái phát triển đa dạng, cung cấp hỗ trợ và tài liệu phong phú.

## Kubernetes

Kubernetes [18] (còn được viết tắt là K8s) là một nền tảng mã nguồn mở giúp tự động hóa việc quản lý, scaling và triển khai ứng dụng dưới dạng container. K8s là một lựa chọn tốt cho các dự án có nhu cầu:

- Scalability (khả năng mở rộng): K8s cho phép tăng hoặc giảm số lượng các container trong hệ thống một cách dễ dàng và linh hoạt.
- Availability (khả năng sẵn sàng): K8s cung cấp các tính năng như replication, self-healing, rollbacks tự động và horizontal scaling để đảm bảo rằng hệ thống luôn hoạt động mạnh mẽ và không bị gián đoạn.
- Continuous Deployment/Delivery (chuyển giao liên tục): K8s cho phép triển khai và cập nhật mã nguồn một cách nhanh chóng và dễ dàng.
- Portability (khả năng di động): K8s giúp triển khai và quản lý ứng dụng một cách nhất quán và đáng tin cậy, hỗ trợ việc chuyển đổi giữa các cloud provider hoặc môi trường on-premise dễ dàng.

Tóm lại, đối với hệ thống phức tạp chứa nhiều container hay microservices, sử dụng K8s giúp quản lý và triển khai hệ thống một cách dễ dàng hơn.

## 3.3 Triển khai ứng dụng

### 3.3.1 Triển khai front-end

Build ứng dụng: nhóm sử dụng lệnh tích hợp sẵn "next build" của NextJS để build Buildify website và dùng Vite với lệnh "vite build" cho Buildify admin.

Sử dụng nền tảng Vercel (được phát triển bởi đội ngũ tạo ra NextJS) để triển khai cho cả Buildify website (chọn cấu hình cho NextJS) và admin (chọn cấu hình cho Vite). Tích hợp CI/CD bằng cách kết nối các repository từ Github vào Vercel.

Nhóm có cấu hình file "vercel.json" với thuộc tính "rewrites" để mặc dù Buildify admin được deploy ở một domain khác (admin-buildify.vercel.app) với Buildify website (buildify.vercel.app), nhưng khi người dùng chuyển trang đến admin tại Buildify website thì vẫn thấy cùng một domain cũ (buildify.vercel.app) với đường dẫn /admin.

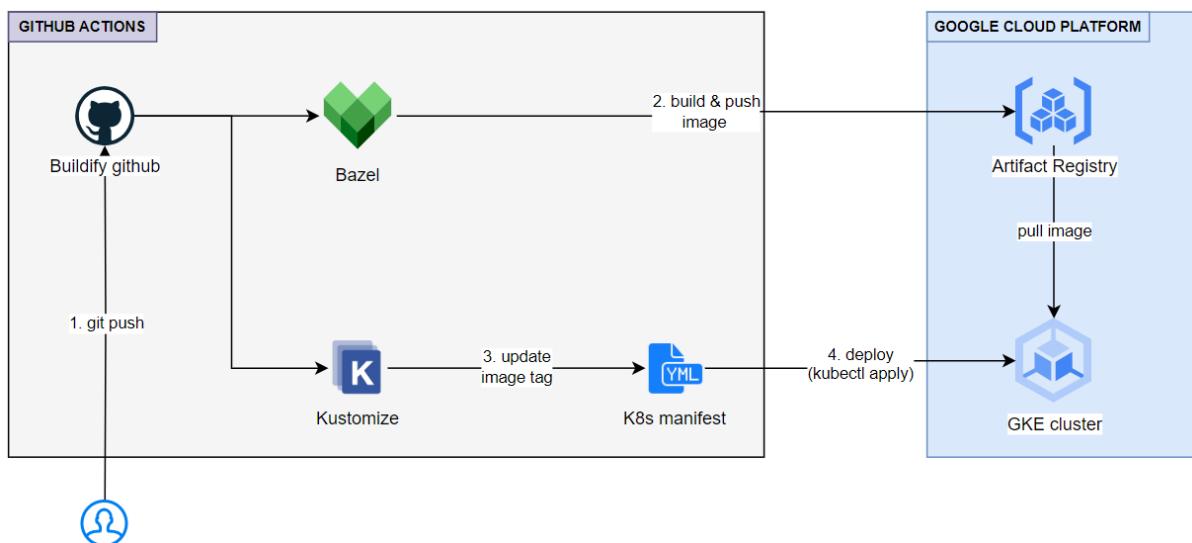
### 3.3.2 Triển khai back-end

Nhóm sử dụng GitHub Actions để triển khai CI/CD, giúp phát hiện sớm các lỗi và tự động hóa quá trình triển khai, chi tiết quy trình được mô tả ở hình 3.3, nhóm sử dụng Actions Secret để lưu trữ những thông tin bảo mật liên quan đến việc triển khai các services. Mỗi service có quy trình triển khai riêng lẻ độc lập nhau, chỉ những service có thay đổi code khi thực hiện merge code vào nhánh main thì mới trigger quy trình triển khai.

Bazel đóng vai trò quan trọng trong việc tạo ra image ở Artifact Registry. Dựa vào các định nghĩa, quy tắc, cấu hình trong file BUILD.bazel của từng service, Bazel sẽ tự động phân tích và xác định các phụ thuộc, tiến hành build image mới cho ứng dụng. Sau đó, dựa vào cấu hình đường dẫn registry, Bazel sẽ đẩy image lên Artifact Registry tương ứng để chuẩn

bị cho việc triển khai lên GKE cluster.

Nhóm sử dụng Kustomize để cập nhật image tag bằng biến môi trường "run\_number" của GitHub Actions (được tự động tạo ra và có giá trị là số thứ tự của mỗi lần chạy workflow) cho deployment K8s resource tương ứng, nhóm cũng cần phải lưu lại image tag này vào repo GitHub, cụ thể là file deployment yaml. Do đó, để tránh làm rối lịch sử commit trên nhánh main, nhóm đã tạo một nhánh release, thực hiện cập nhật image tag mới và commit vào nhánh release này.



Hình 3.3: Sơ đồ CI/CD ở back-end

### 3.4 Thiết kế cơ sở dữ liệu

Sơ đồ thiết kế cơ sở dữ liệu được mô tả như hình 3.4, có cấu trúc khá đơn giản nhưng hiệu quả và bảo mật vì toàn bộ dữ liệu cho dự án website của người dùng sẽ được mã hóa base64 trước khi gửi xuống server, cụ thể sẽ được lưu trữ trong trường projects.compress\_string tại cơ sở dữ liệu của service user.

1. User service: chịu trách nhiệm quản lý tin người dùng ("users"), các dự án tương ứng ("projects") và lưu sẵn "compress\_string" của các loại dự án mặc định ("default\_projects").

2. Dynamic data service: chịu trách nhiệm quản lý cơ sở dữ liệu cho từng dự án, bao gồm các "collections" và "documents". Trường "id" của "collections" và "documents" sẽ được phát sinh tự động tăng dần và chỉ duy nhất trên từng người dùng, nhằm đảm bảo tính thân thiện và thay thế cho kiểu dữ liệu ObjectId của trường "\_id" mặc định trong MongoDB.



Hình 3.4: Sơ đồ thiết kế cơ sở dữ liệu

# Chương 4

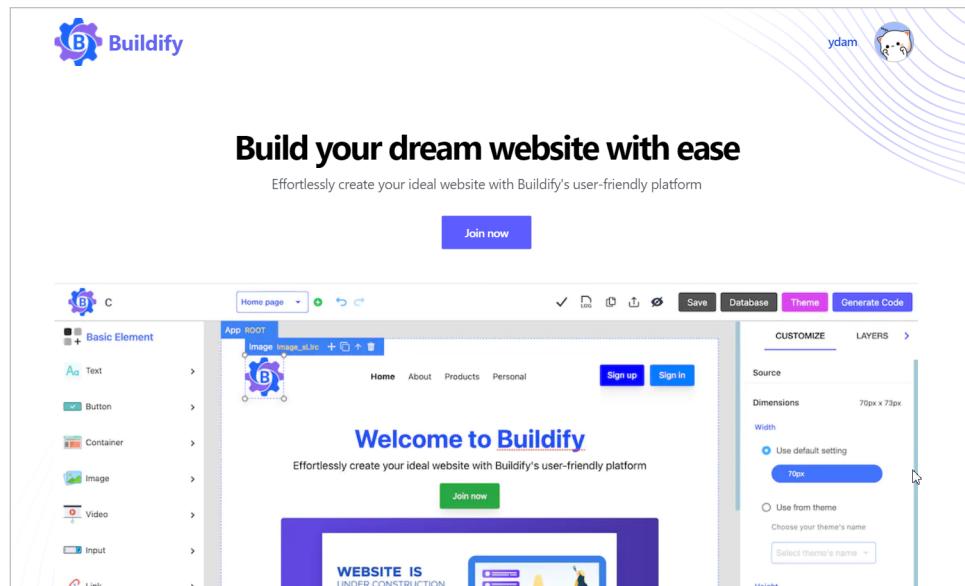
## Cài đặt hệ thống Buildify

Chương 4 sẽ trình bày chi tiết cách cài đặt các thành phần cũng như cơ chế hoạt động của các chức năng tương ứng trong hệ thống Buildify.

### 4.1 Tổng quan giao diện Buildify

Về mặt giao diện, hệ thống Buildify gồm những trang sau:

- Landing page: trang chủ của Buildify, giúp giới thiệu về sản phẩm, thông tin tác giả, thông tin liên hệ... với giao diện như hình 4.1



Hình 4.1: Giao diện của landing page

- Projects dashboard: trang hiển thị danh sách các dự án website của người dùng dưới dạng bảng, hỗ trợ việc sắp xếp, tìm kiếm và hỗ trợ phân trang, với giao diện như hình 4.2

The screenshot shows the 'My projects' section of the Buildify dashboard. At the top, there's a search bar labeled 'Search project' and a blue button '+ New Project'. Below the search bar is a table with the following data:

Name ↑	Type	Created Time	Updated Time	Operations
new				
project 2				
ydam				

At the bottom of the table, there are pagination controls: 'Rows per page: 5', '1–3 of 3', and arrows for navigating through the pages.

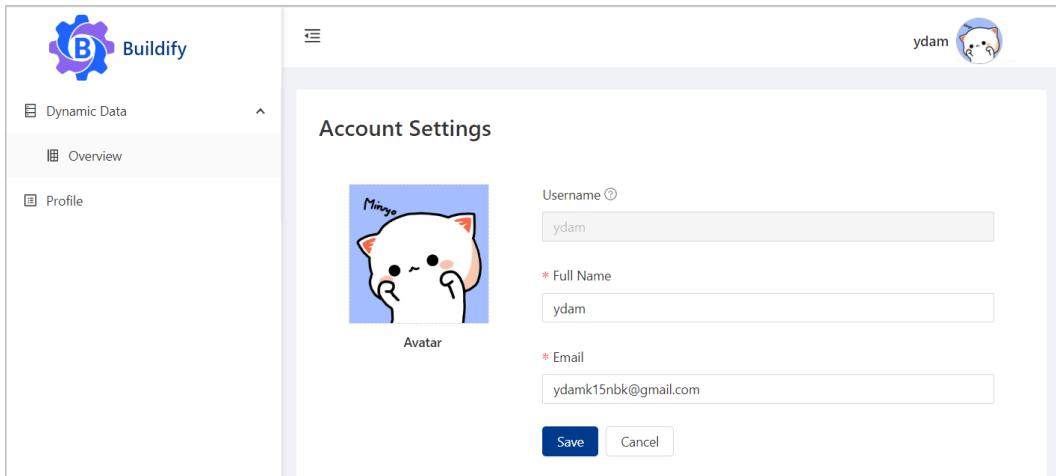
Hình 4.2: Giao diện của trang projects dashboard

- Builder: là trang quan trọng nhất, chứa toàn bộ các tính năng chính của Buildify. Đây là trang dành cho người dùng thiết kế giao diện cho dự án website của mình, giao diện như hình 4.3.

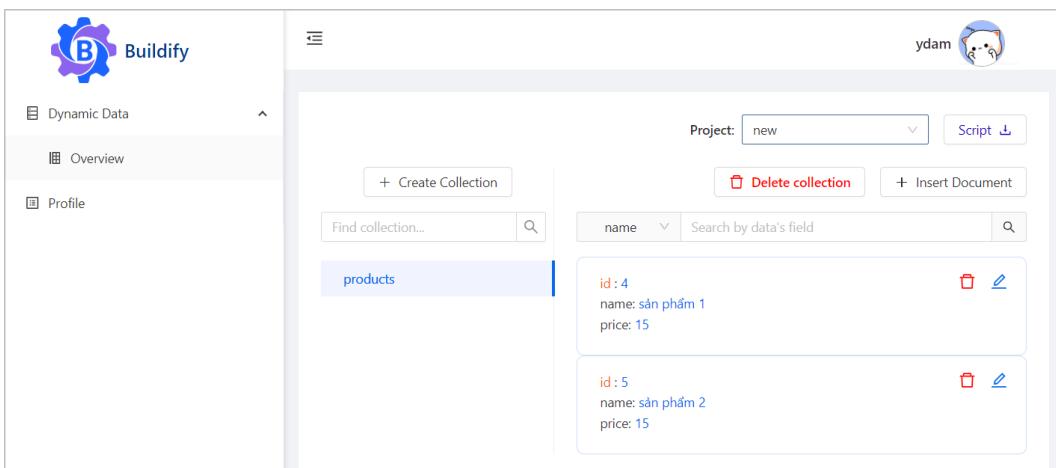
The screenshot shows the main builder interface. On the left, there's a sidebar with sections for 'Basic Element' and 'Built-in Template'. The central area displays a preview of a website with the title 'Welcome to Buildify' and a message 'Effortlessly create your ideal website with Buildify's user-friendly platform'. Below this is a large image of two workers building a website. On the right, there's a 'CUSTOMIZE' panel with various settings. Under 'Typography', it shows 'h1, 42, Bold, Center'. Under 'Font Size', it has a slider set at 42 and a dropdown menu with 'Select value'. There are also other options like 'Use suggestion style' and size selection buttons for 'Small', 'Medium', and 'Large'.

Hình 4.3: Giao diện của trang builder

- Admin: là các trang cho phép người dùng quản lý thông tin, hiện tại bao gồm quản lý thông tin cá nhân (hình 4.4) và quản lý cơ sở dữ liệu động dưới dạng các collections và documents (hình 4.5).



Hình 4.4: Giao diện của trang quản lý thông tin cá nhân



Hình 4.5: Giao diện của trang quản lý cơ sở dữ liệu

- Authentication: chứa các trang liên quan đến xác thực người dùng gồm đăng ký, đăng nhập với giao diện tương ứng ở hình 4.6 và 4.7

Hình 4.6: Giao diện của trang đăng ký

Hình 4.7: Giao diện của trang đăng nhập

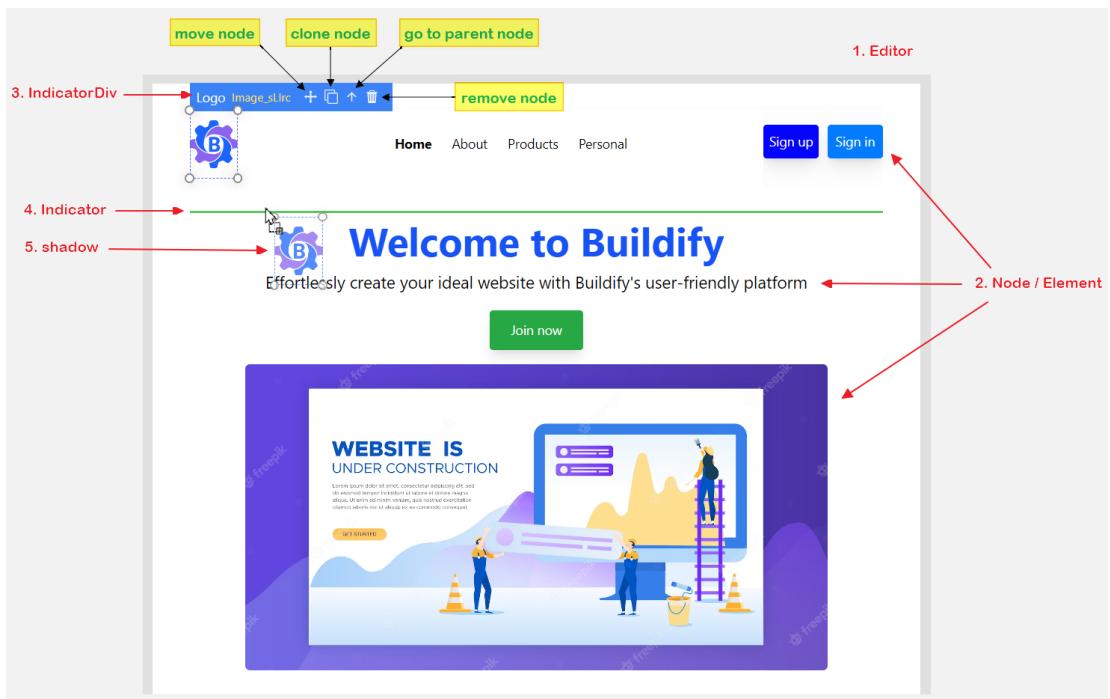
## 4.2 Các thành phần của trang thiết kế

Trang thiết kế website gồm 4 thành phần chính:

1. Editor: là một vùng canvas cho phép người dùng thao tác và tùy chỉnh giao diện trang web, bao gồm các thành phần:
  - Element: là giao diện của node, nói cách khác, editor render node thành element. Các elements được sắp xếp một cách trật

tự và có chủ đích trên editor theo ý muốn và thao tác của người dùng để tạo thành giao diện website.

- IndicatorDiv: chỉ xuất hiện khi người dùng hover/select element trên editor, dùng để hiển thị thông tin và các actions có thể thực hiện trên element tương ứng. Tùy thuộc vào từng loại element, các actions đó có thể là: di chuyển element, nhân bản element, xóa element, đi đến (select) element cha, nhúng dữ liệu động.
- Indicator và shadow: sẽ xuất hiện liên tục trong suốt quá trình người dùng kéo thả element/selector trên editor.
  - Shadow: dấu hiệu để người dùng xác định được có đang kéo thả đúng element hay không.
  - Indicator: giúp người dùng xác định được vị trí thả cho element. Nếu vị trí thả là hợp lệ, indicator có màu xanh, ngược lại nó có màu đỏ.

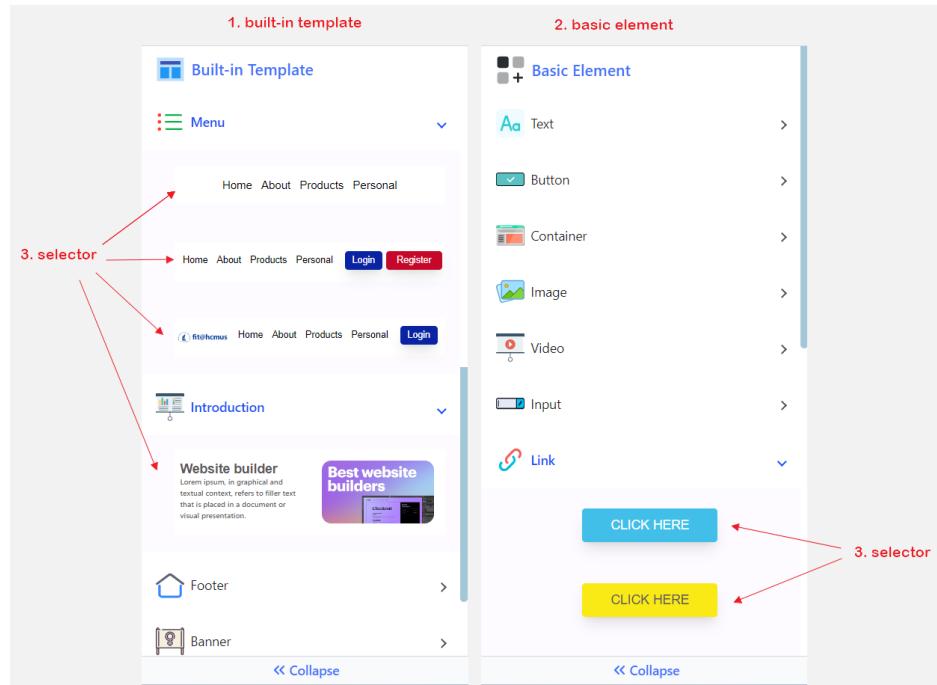


Hình 4.8: Các thành phần của giao diện editor

## 2. Toolbox: gồm tập hợp các selector thuộc một trong hai nhóm basic

element hoặc built-in template.

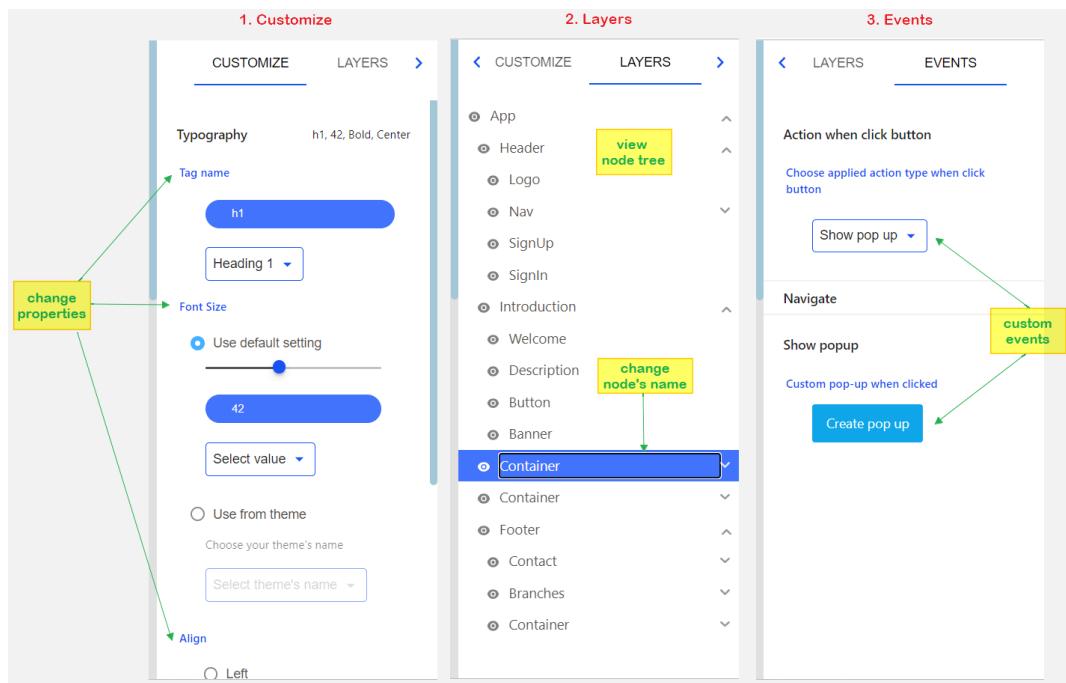
- Selector: là bản preview giao diện của element.
- Basic element: là những thành phần nhỏ nhất để cấu thành một trang web như container, button, text, input, anchor, image, video...
- Built-in template: là những mẫu giao diện được thiết kế sẵn bằng cách kết hợp từ nhiều basic element về thứ tự, vị trí, kích thước, màu sắc, nội dung,... để tạo nên một khối giao diện đẹp và đóng một vai trò nhất định trong trang web. Đây là những mẫu giao diện được chọn lọc, thiết kế phù hợp dựa trên yêu cầu thực tiễn. Chỉ cần chọn và kéo thả các template như header, footer, menu, introduction, content, banner,... người dùng có thể nhanh chóng xây dựng được trang web một cách hiệu quả và tiết kiệm thời gian.



Hình 4.9: Các thành phần của giao diện toolbox

### 3. Side panel: gồm 3 thành phần:

- Customize: cho phép tùy chỉnh thuộc tính của các element trên editor.
- Layers: trực quan hóa thứ tự, cấp bậc (mỗi quan hệ cha - con) của các element trên editor.
- Events: tùy chỉnh hành động xử lý sự kiện cho các element.

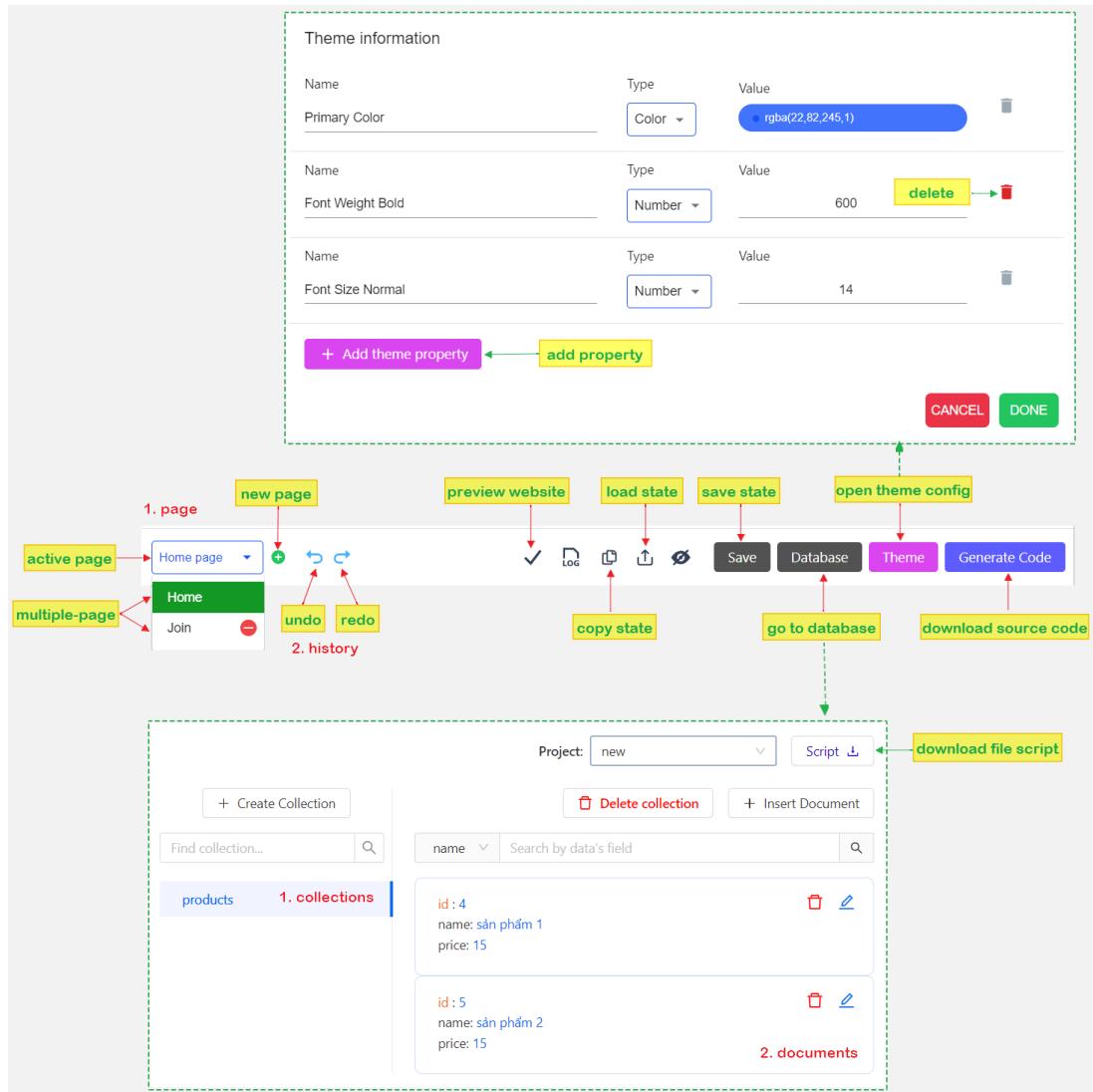


Hình 4.10: Các thành phần của giao diện side panel

#### 4. Header: chứa các chức năng như:

- Quản lý trang: bao gồm hiển thị trang hiện tại, cho phép thay đổi trang, tạo trang mới, xóa trang.
- Quản lý lịch sử: cho phép undo, redo.
- Xem trước website.
- Sao chép trạng thái dự án hiện tại, tải trạng thái dự án mới, lưu trạng thái dự án hiện tại.
- Di đến trang quản lý cơ sở dữ liệu động.
- Quản lý cấu hình theme.

- Phát sinh mã nguồn.



Hình 4.11: Các thành phần của giao diện header

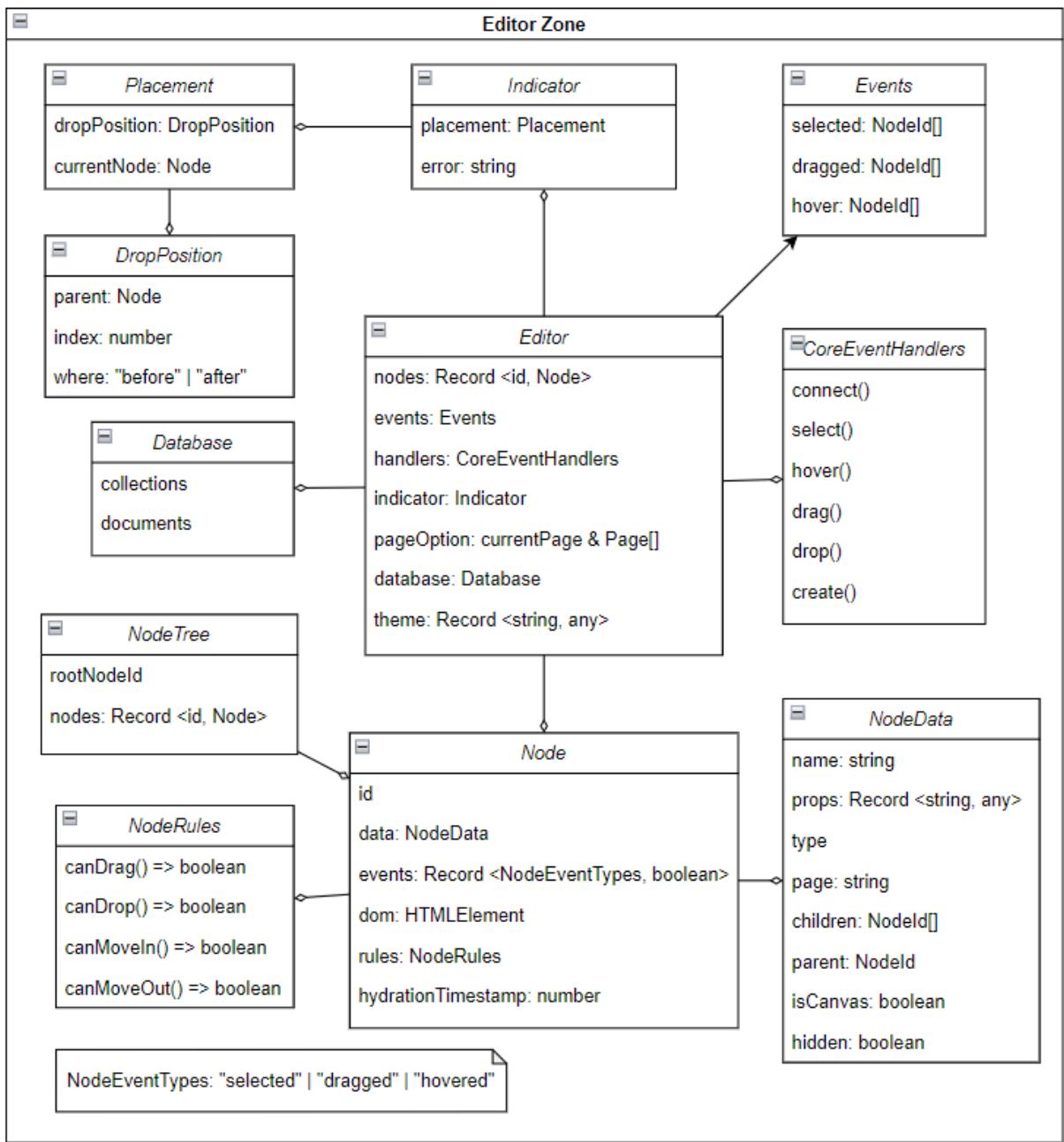
### 4.3 Cài đặt trang thiết kế

Trong phần này, nhóm sẽ lần lượt mô tả kiến trúc của 3 thành phần chính lưu trữ toàn bộ thông tin của dự án là editor, layers và history. Tiếp theo đó, nhóm sẽ trình bày cơ chế cũng như các luồng xử lý chính của các tính năng website builder mà nhóm hỗ trợ.

### 4.3.1 Cấu trúc editor

Bắt nguồn từ vai trò của browser rendering engine trong việc quản lý và hiển thị giao diện trang web như quản lý cây DOM, CSS, layout, render element và xử lý sự kiện, nhóm xây dựng một editor mạnh mẽ có cấu trúc được mô tả như hình 4.12. Đây là thành phần quan trọng nhất của trang thiết kế, lưu giữ gần như toàn bộ thông tin, trạng thái hiện tại của dự án website, bao gồm:

1. nodes: danh sách các nodes.
2. events: danh sách id của các nodes ứng với từng event như "selected", "dragged", "hovered".
3. handlers: bao gồm các hàm hỗ trợ xử lý sự kiện như connect(), select(), hover(), drag(), drop(), create().
4. indicator: gồm vị trí thả và lõi nếu có. Vị trí thả được xác định bởi node cha và vị trí trước hay sau chỉ số index (đây sẽ là thông tin node cha và vị trí mới của node đang được kéo thả nếu người dùng thả tay).
5. pageOption: gồm trang hiện tại (active page) và danh sách các pages.
6. database: cơ sở dữ liệu động.
7. theme: gồm danh sách các thuộc tính dưới dạng key - value.



Hình 4.12: Cấu trúc của editor

### 4.3.2 Cấu trúc node

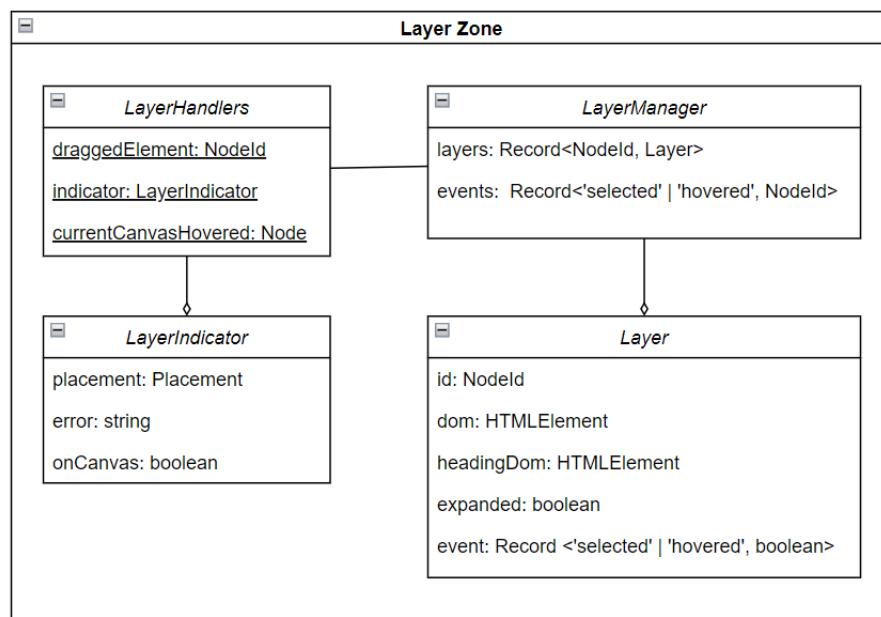
Thông tin của mỗi node cũng được mô tả ở hình 4.12, bao gồm:

1. `id`: định danh duy nhất cho mỗi node.
2. `data`: gồm các thông tin sau:
  - `name`: tên node.

- props: danh sách các thuộc tính dưới dạng key - value của node, có ý nghĩa tương tự như props của React Component, hỗ trợ tính năng thay đổi giá trị thuộc tính của element trên side panel.
  - type: có kiểu dữ liệu là string hoặc React.ElementType, xác định React Component tương ứng mà node thuộc về.
  - page: đường dẫn (path) của page mà node thuộc về, mỗi node chỉ thuộc duy nhất một page.
  - children: danh sách các node con.
  - parent: node cha.
  - isCanvas: xác định node có phải là canvas node hay không. Canvas node là loại node có khả năng chứa các node khác, cho phép người dùng thả element vào bên trong giao diện element tương ứng của canvas node đó.
  - hidden: ẩn node trên editor.
3. events: danh sách key - value cho phép xác định được node này có đang được "selected", "dragged", "hovered" hay không.
4. dom: HTML element tương ứng của node trên editor.
5. rules: bao gồm danh sách các hàm cho phép xác định người dùng có được thực hiện thao tác drag, drop, movein, moveout trên node này hay không. Cụ thể:
- canDrag(): trả về true nếu được phép kéo node.
  - canDrop(): trả về true nếu được phép thả node khác vào bên trong nó.
  - canMoveIn(): trả về true nếu cho phép di chuyển node khác vào bên trong nó.
  - canMoveOut(): trả về true nếu cho phép di chuyển phần tử con của node ra khỏi nó.
6. hydrationTimestamp: thời điểm mà node xảy ra quá trình hydration.

### 4.3.3 Cấu trúc layers

Tiếp theo, nhóm sẽ trình bày cấu trúc của layers, một cách tiếp cận đơn giản để hiển thị được giao diện layers là dựa vào danh sách nodes của page hiện tại trong editor, tiến hành tính toán node tree và hiển thị lên màn hình. Tuy nhiên cách tiếp cận này sẽ dẫn đến việc tăng chi phí tính toán và chi phí re-render vì các node trong editor sẽ thường xuyên thay đổi trạng thái. Điều quan trọng là cách tiếp cận này sẽ hạn chế việc mở rộng các tính năng khác cho layers như tính năng kéo thả layer để thay đổi vị trí của node tương ứng hay đơn giản là chỉnh sửa tên node hoặc ẩn node trên editor. Những tính năng này đòi hỏi phải ánh xạ layer 1 - 1 với node tương ứng, cho phép quản lý thuộc tính và sự kiện trên từng layer, cấu trúc layers được nhóm mô tả ở hình 4.13 hoàn toàn đáp ứng được yêu cầu này.

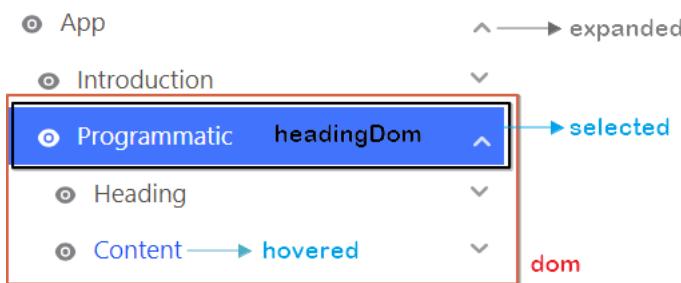


Hình 4.13: Cấu trúc layers

Một layer bao gồm các thuộc tính sau và được minh họa trực quan ở hình 4.14:

1. id: định danh duy nhất của layer, cũng là id của node ánh xạ tương ứng.

2. dom: HTML element của layer hiện tại bao gồm cả HTML element của toàn bộ layer con.
3. headingDom: HTML element của layer hiện tại.
4. expanded: xác định trạng thái mở rộng hay thu gọn các layer con của layer hiện tại.
5. events: danh sách key - value cho phép xác định được layer này có đang được "selected", "hovered" hay không.



Hình 4.14: Hiển thị thông tin layer

Ngoài ra, để hỗ trợ thao tác kéo thả layer (được mô tả chi tiết ở phần 4.3.7), cấu trúc layers còn có các thành phần sau:

1. LayerManager: dùng để quản lý toàn bộ layers và các trạng thái events, gồm:
  - layers: danh sách các layers.
  - events: danh sách id của các layers ứng với từng event như "selected", "dragged", "hovered".
2. LayerHandlers: trực tiếp tham gia xử lý thao tác kéo thả, gồm:
  - draggedElement: một biến toàn cục lưu id của layer đang được kéo.
  - indicator: một biến toàn cục lưu trạng thái của indicator hiện tại.
  - currentCanvasHovered: một biến toàn cục lưu node tương ứng của layer đang được người dùng hover.
  - handlers: bao gồm các hàm phục vụ cho quá trình kéo thả bằng cách lắng nghe và xử lý các sự kiện như "dragstart", "dragen-

ter", "dragover", "dragend"...

3. LayerIndicator: lưu trữ trạng thái hiện tại của indicator tính toán được, bao gồm:

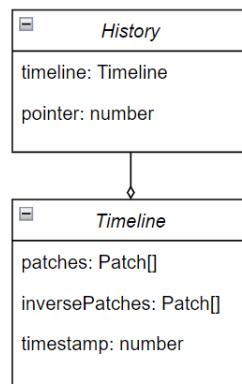
- placement: vị trí thả.
- error: lỗi nếu vị trí thả không hợp lệ.
- onCanvas: dùng để phân biệt trường hợp "dragover" với "dragenter".

#### 4.3.4 Cấu trúc history

Buildify quản lý toàn bộ thông tin của editor bằng cách sử dụng state management, sử dụng các actions để cập nhật thông tin ứng với từng loại thao tác của người dùng. Với mỗi action được kích hoạt khi người dùng có tương tác sửa đổi thông tin của editor, nhóm sử dụng history có cấu trúc được mô tả như hình 4.15 để ghi lại thông tin lịch sử chỉnh sửa này.

Thông tin được lưu trữ bao gồm:

1. **timeline**: là một mảng lưu lại thông tin thay đổi ứng với từng mốc thời gian, bao gồm patches và inversePatches (được quản lý bởi thư viện immer) chứa thông tin liên quan đến action cùng với timestamp là thời điểm xảy ra action.
2. **pointer**: là index trỏ tới thời điểm đại diện cho trạng thái hiện tại của editor trong timeline.

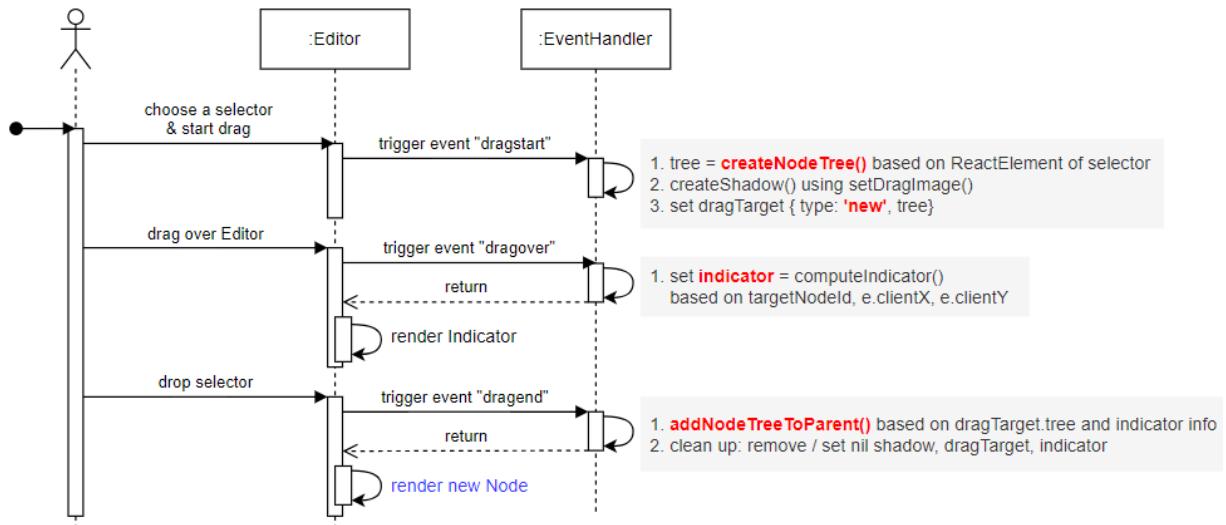


Hình 4.15: Cấu trúc history

Dựa vào những thông tin này, nhóm dễ dàng cài đặt được tính năng undo và redo, được đề cập chi tiết ở phần 4.3.9

### 4.3.5 Kéo thả một selector vào trong editor

Selector là một bản preview của element nằm trên toolbox. Về thực chất, node mới là nơi lưu trữ thông tin đầy đủ cũng như cung cấp các cơ chế xử lý sự kiện tương tác của người dùng với element trên editor. Do đó quy trình xử lý việc kéo thả một selector vào trong editor liên quan trực tiếp đến node, được mô tả ở hình 4.16, kết quả của quá trình này là tạo mới một node (trường hợp selector thuộc basic element) hoặc node tree (trường hợp selector thuộc built-in template) và render thành các elements nằm trên editor.



Hình 4.16: Kéo thả một selector vào trong editor

Để render node thành giao diện element trên editor, nhóm sử dụng hàm createElement() của ReactJS với 3 tham số tương ứng là type, props và children với children là mã JSX của tất cả các node con thuộc node đó.

Suốt quá trình kéo thả luôn có sự xuất hiện của một thành phần quan trọng là indicator, được tính toán và cập nhật liên tục để hỗ trợ phản hồi

real-time cho người dùng về vị trí thả, nhóm sẽ trình bày chi tiết ở phần 4.3.8.

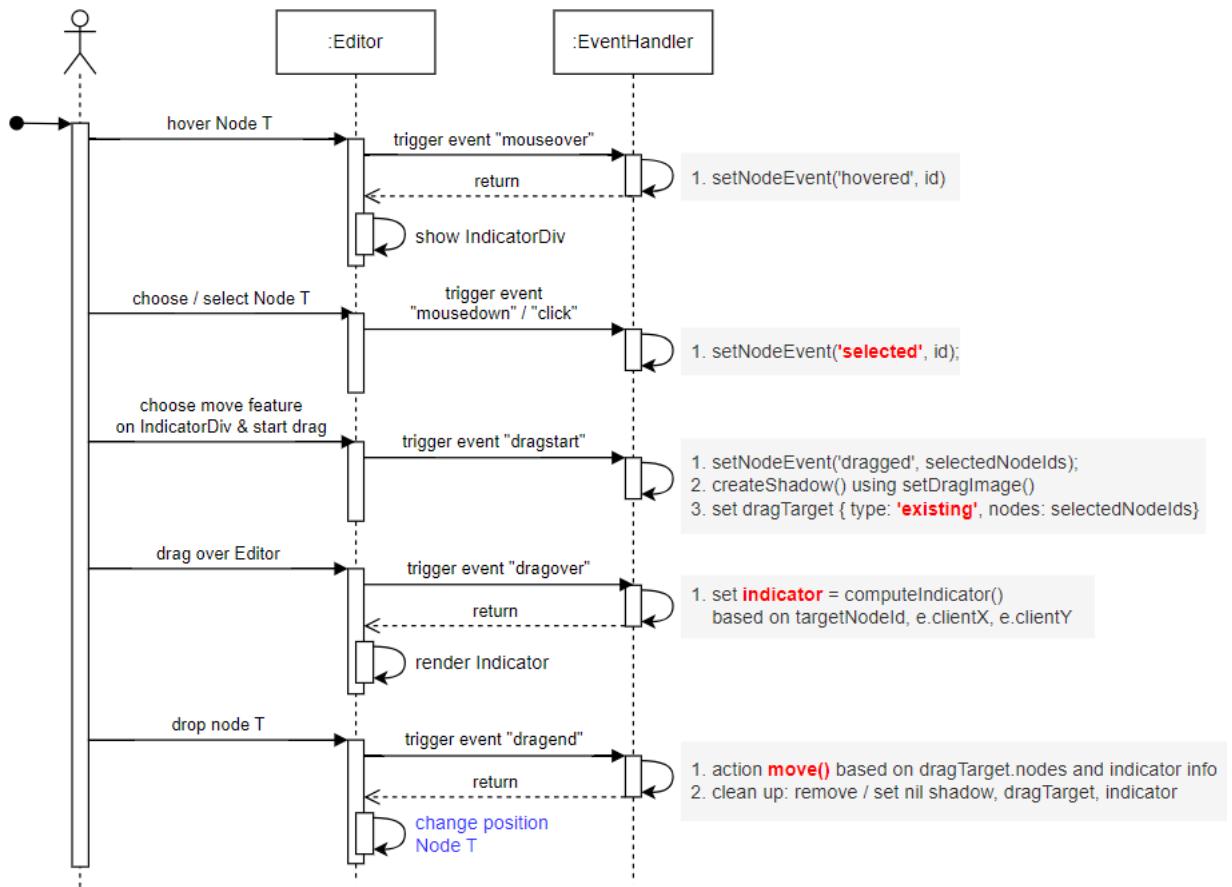
#### 4.3.6 Các thao tác với element trong editor

Do node là logic bên dưới của element nên thao tác với element trong editor thực chất là thao tác với node, bao gồm các thao tác sau:

1. Di chuyển: cho phép kéo và di chuyển node đến một vị trí khác.
2. Sao chép: cho phép sao chép node hiện tại với giao diện và tính năng giống hoàn toàn. Node mới được thêm vào sẽ có cùng node cha và được tự động thêm vào cuối danh sách node con. Tính năng này không áp dụng cho node root của trang.
3. Chọn node cha của node hiện tại: tính năng này giúp người dùng nhanh chóng xác định được node cha của node hiện tại, nhất là khi số lượng node quá nhiều và bối cảnh phức tạp, đôi khi các node chồng chéo lên nhau, rất khó để tìm thấy node cha. Tính năng này không áp dụng cho node gốc của trang vì nó không có node cha.
4. Xoá: xoá toàn bộ thông tin node hiện tại khỏi editor (bao gồm luôn các thông tin tham chiếu) và cũng đồng thời xoá các node con bên trong. Tính năng này không áp dụng cho node gốc của trang vì mỗi trang cần có ít nhất một canvas node.
5. Kết nối dữ liệu động: cho phép thay thế text hiện tại của node bằng dữ liệu tham chiếu từ cơ sở dữ liệu động do người dùng tự định nghĩa. Tính năng này sẽ hiển thị ra danh sách collections và documents để người dùng chọn.

Một số node thuộc loại đặc thù (ví dụ như container) còn cho phép người dùng kéo thả để thay đổi kích thước element (chiều dài, chiều rộng). Tính năng này được cài đặt nhờ vào thư viện re-resizable [12].

Trong các thao tác kể trên, di chuyển là thao tác phức tạp nhất, cho phép người dùng kéo thả node được chọn ("selected") vào trong một node khác. Thao tác này được nhóm mô tả chi tiết ở hình 4.17.



Hình 4.17: Di chuyển một node đến vị trí khác trong editor

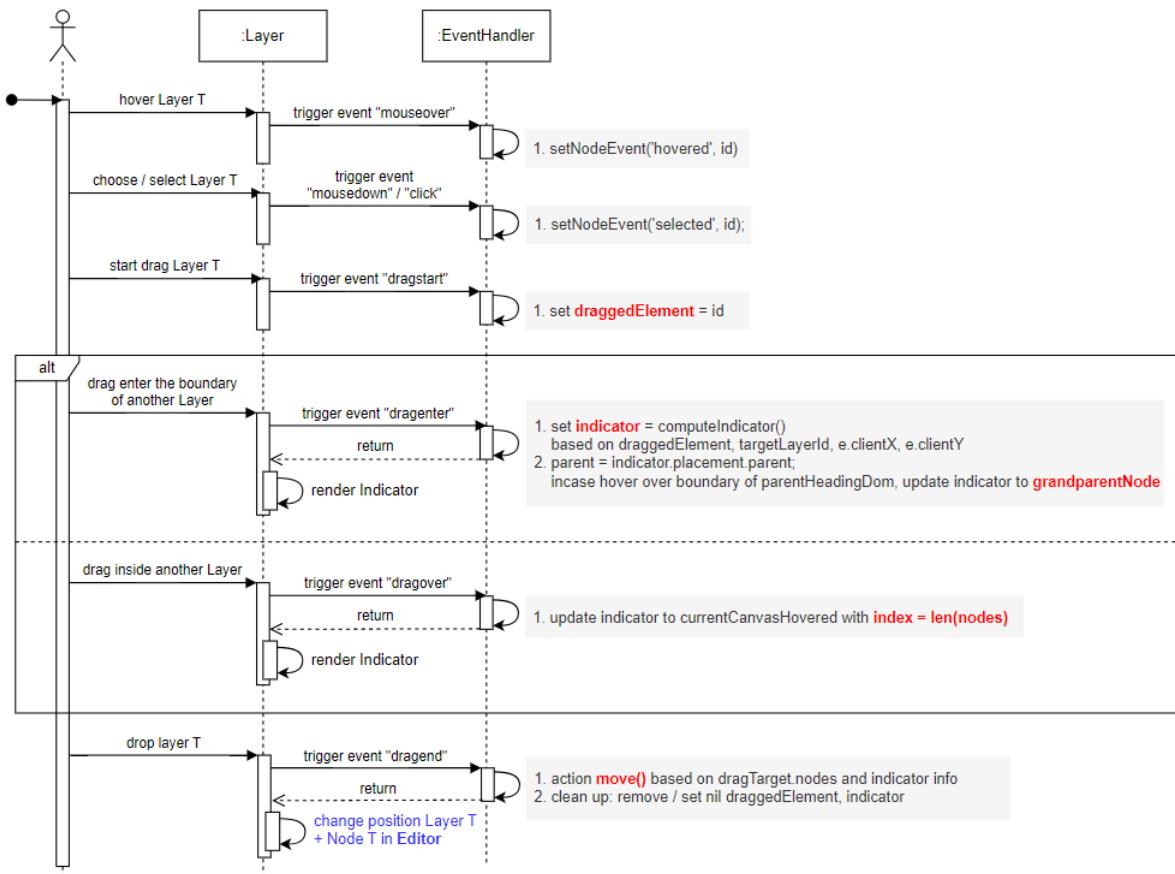
#### 4.3.7 Kéo thả một layer trên layers

Quá trình kéo thả layer được mô tả ở hình 4.18, tập trung vào 4 sự kiện quan trọng gồm "dragstart", "dragenter", "dragover" và "dragend", một số ý tưởng xử lý chính như sau (giả sử người dùng kéo layer T và thả vào layer Q):

- dragstart: khởi tạo quá trình kéo bằng cách gán draggedElement là id của layer được chọn (layer T).

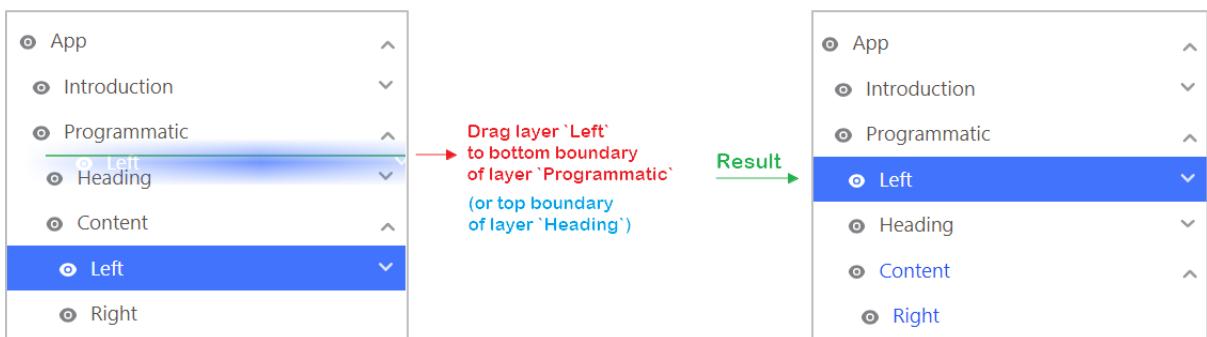
- dragenter: xử lý cập nhật thông tin indicator dựa trên vị trí hiện tại của con trỏ chuột. Nếu con trỏ chuột đang hover trên phần biên của layer Q thì ta xem như ý định của người dùng là đổi vị trí của node T sang vị trí mới kề với node Q (kề trước hay kề sau phụ thuộc vào việc hover vào biên trên hay biên dưới của node Q), do đó cần cập nhật lại indicator.parent là node cha của node Q.
- dragover: để tối ưu hiệu suất, ta không cần tính toán lại indicator vì người dùng không thể kéo layer T vào bên trong layer Q mà không kéo vào phần biên của layer Q ngay trước đó. Trường hợp này, indicator.parent chính là node Q và ta chỉ cần thêm node T vào danh sách node con của node Q bằng cách cập nhật indicator.index = len(node Q.children).
- dragend: hoàn tất quá trình kéo bằng cách di chuyển node tương ứng dựa trên thông tin draggedElement và indicator.

Cần lưu ý là trường hợp "dragenter" thì node cha của node Q phải là canvas node, còn ở trường hợp "dragover" thì bản thân node Q phải là canvas node.

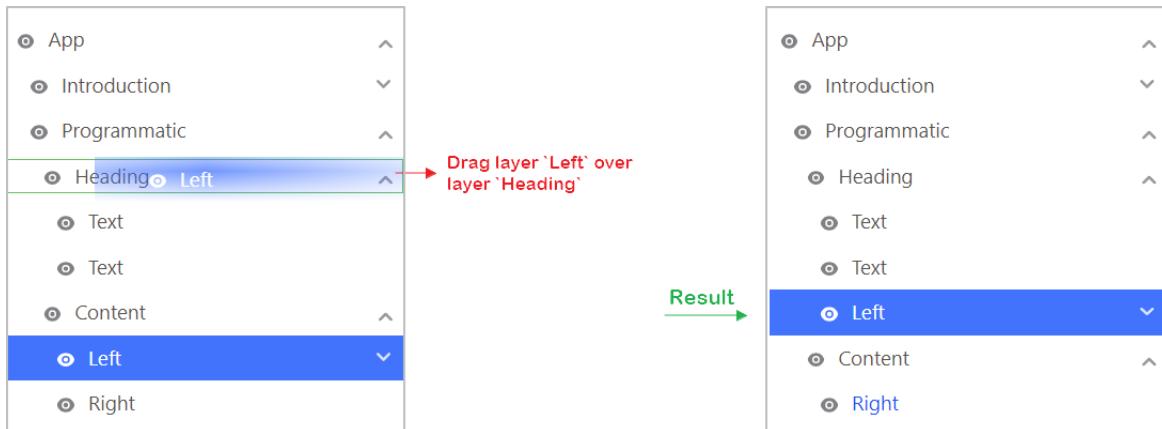


Hình 4.18: Luồng kéo và thả layer

Mô tả trực quan cho 2 trường hợp này được vẽ ở hình 4.19 và hình 4.20.



Hình 4.19: Kéo một layer vào ranh giới của layer khác



Hình 4.20: Kéo một layer vào bên trong một layer khác

### 4.3.8 Cơ chế xác định vị trí thả

Trong quá trình người dùng kéo thả selector hay element trên editor, indicator đóng vai trò quan trọng trong việc cung cấp phản hồi real-time về vị trí thả hiện tại được tính toán từ con trỏ chuột, hỗ trợ người dùng xác định chính xác vị trí thả cho selector/element đang được kéo.

Để hiểu rõ hơn về quá trình tính toán indicator, cụ thể hơn là cách xác định được vị trí của indicator trên editor, nhóm cung cấp đoạn mã nguồn mẫu 4.1 để minh họa cho hàm `findPosition()` với đầu vào gồm

- parent: là `indicator.parent`.
- dims: danh sách thông tin của các node con (gồm id và thông tin dom như x, y, top, left, bottom, right, width, height, outerWidth, outerHeight).
- x, y: vị trí của con trỏ chuột.

```

1 function findPosition(parent:Node, dims:NodeInfo[], x,y:number) {
2   result: DropPosition = { parent, index: 0, where: "before" };
3   leftLimit = rightLimit = bottomLimit = xCenter = yCenter = 0;
4   for (let i = 0, len = dims.length; i < len; i++) {
5     dim = dims[i];
6     xCenter = dim.left + dim.outerWidth / 2;
7     yCenter = dim.top + dim.outerHeight / 2;
8     // Skip if over the limits
9     if ((rightLimit && dim.left > rightLimit) ||
10       (leftLimit && dim.right < leftLimit) ||

```

```

11     (bottomLimit && yCenter >= bottomLimit))
12     continue;
13
14     result.index = i;
15
16     if (dim.inFlow) {
17         if (y < yCenter) {
18             result.where = "before";
19             break; // yCenter covariate over loop
20         } else result.where = "after"; // after last element
21     } else {
22         if (y < dim.bottom) bottomLimit = dim.bottom;
23         if (x < xCenter) {
24             rightLimit = xCenter;
25             result.where = "before";
26         } else {
27             leftLimit = xCenter;
28             result.where = "after";
29         }
30     }
31 }
32 return result;
33 }
```

Mã nguồn 4.1: Hàm findPosition() dùng để tính toán indicator

(Giả sử người dùng di chuyển node T vào bên trong node Q).

Danh sách các node con được liệt kê theo thứ tự từ trái qua phải , từ trên xuống dưới và có thể phân bố theo chiều dọc, chiều ngang, vừa dọc vừa ngang hoặc có thể nằm chồng lên nhau.

Để đơn giản, indicator dc chia làm hai loại: indicator ngang và indicator dọc, đại diện cho vị trí của node T giữa các node con của node Q. Nếu các node con của node Q đang được phân bố theo chiều dọc thì indicator sẽ nằm ngang và ngược lại. Do đó, ý tưởng của hàm findPosition() là tính toán xem các element con của node Q có đang được phân bố theo chiều dọc hay không.

Một node con được đánh dấu là inFlow = true (tức là nó được phân bố dọc giữa các node con của node Q) nếu nó thoả mãn 2 điều kiện sau đây:

1. CSS style của node Q: thuộc tính display của node Q không được là "`float`" hoặc "`grid`" (có thể làm cho các node con nằm chồng lên nhau). Node Q có thể "`flex`", nhưng "`flex-direction`" phải là "`column`".
2. CSS style của node con: thuộc tính position của node con không được là "`fixed`" hoặc "`absolute`", và thuộc tính display của node con phải tạo được ngắt dòng (**line break**) trước và sau element của nó.

Đối với các node con mà `inFlow = true`, vì các node con liệt kê theo thứ tự từ trên xuống dưới, giá trị của `yCenter` sẽ tăng dần qua các vòng lặp. Do đó, vòng lặp sẽ kết thúc ngay lập tức nếu tìm thấy một node con thỏa mãn  $y < yCenter$  vì đây là vị trí phù hợp nhất cho indicator.

Đối với các node con mà `inFlow = false`, có thể xảy ra trường hợp các node con nằm chồng lên nhau. Do đó ngay cả khi tìm được một node con thỏa mãn yêu cầu, vòng lặp vẫn sẽ tiếp tục để tìm ra node con phù hợp nhất, cùng với đó, cần **giới hạn** lại các ranh giới bên trái, bên phải và phía dưới để tìm ra index phù hợp nhất.

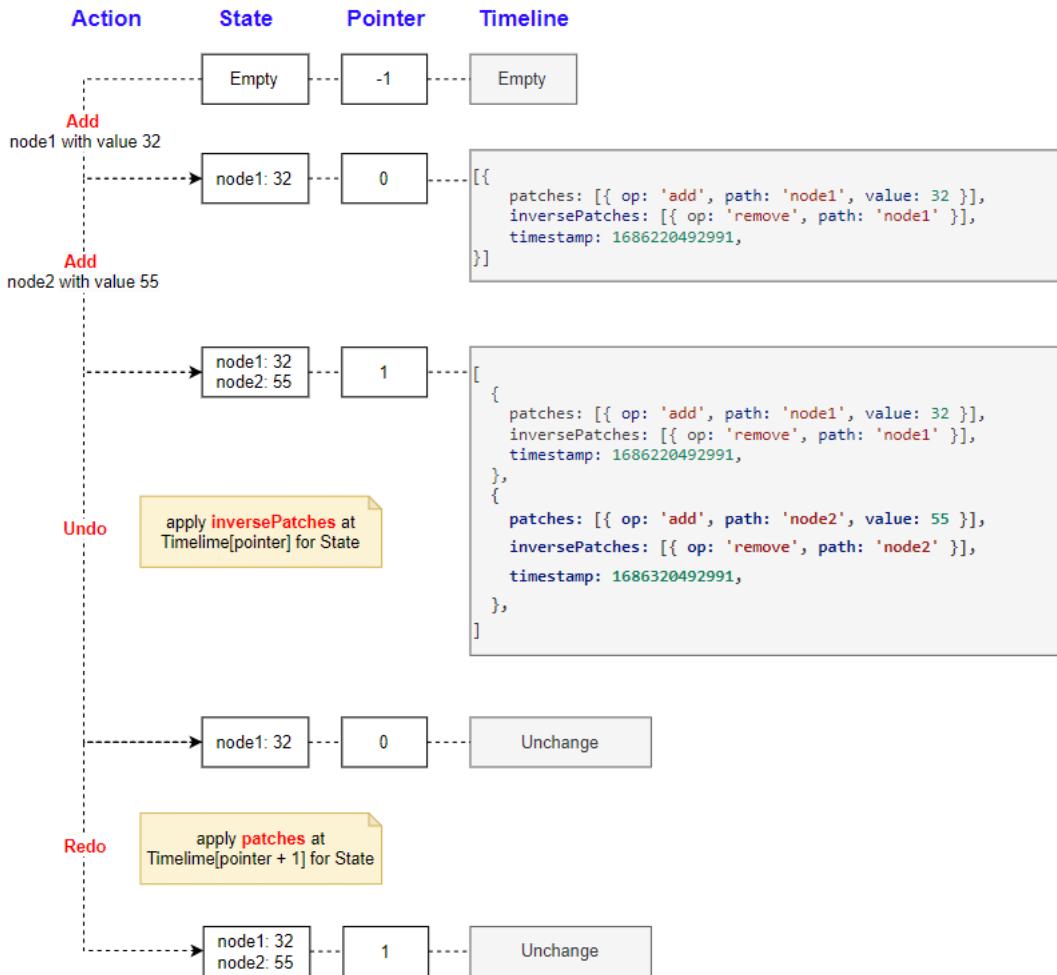
#### 4.3.9 Cơ chế undo và redo

Để cài đặt được tính năng undo và redo, nhóm sử dụng thư viện immer [20], dùng để lưu trữ danh sách các thay đổi theo thứ tự thời gian (timeline) thay vì chỉ lưu trạng thái cuối cùng. Điều này cho phép theo dõi lịch sử các actions và các thay đổi tương ứng của editor theo mốc thời gian.

Khi người dùng yêu cầu thực hiện action undo, immer truy xuất trạng thái hiện tại của editor và áp dụng các thay đổi trước đó từ timeline theo thứ tự ngược lại (`inversePatches`). Tương tự với action redo, immer truy xuất trạng thái hiện tại và áp dụng các thay đổi tiếp theo được lưu trữ trong timeline (gọi là patches).

Cách biến đổi trạng thái và lưu trữ lịch sử của timeline và pointer) trong quá trình thực hiện tuân tự các action thêm node trong editor, cũng

như cách mà lịch sử ghi lại thông tin để hỗ trợ các action redo và undo, được minh họa bằng một ví dụ trong hình 4.21 (dữ liệu của một node trong ví dụ đã được đơn giản hóa nhằm phục vụ mô tả cơ chế).



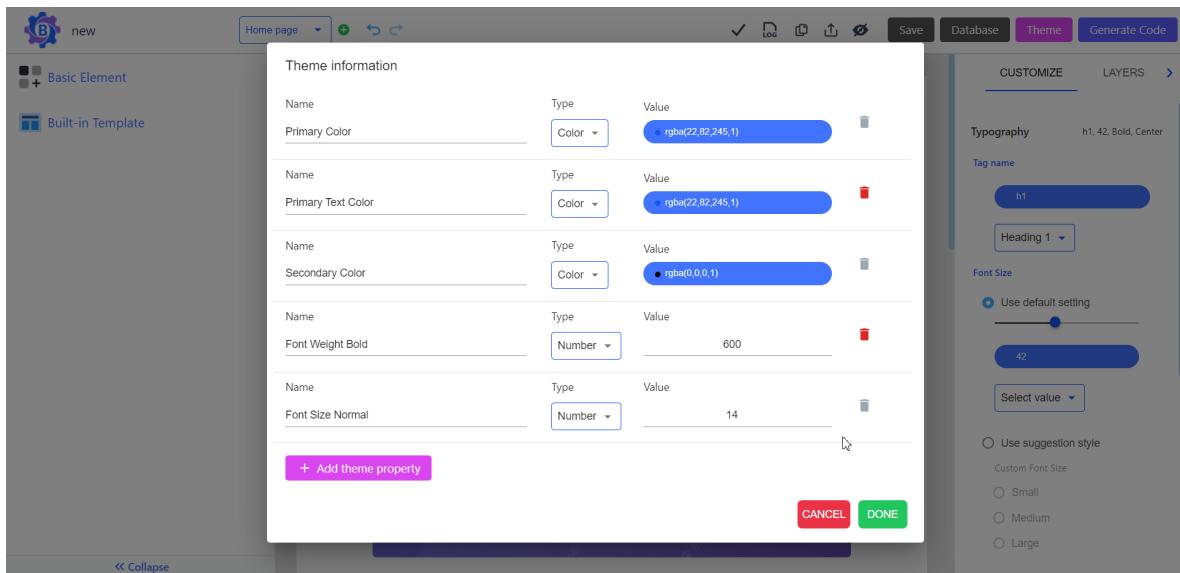
Hình 4.21: Ví dụ về cơ chế hoạt động của history

### 4.3.10 Cấu hình theme

Cấu hình theme bao gồm danh sách các thuộc tính do người dùng tự định nghĩa dưới dạng key - value, với key là tên duy nhất của thuộc tính, value là giá trị của thuộc tính. Hiện tại, nhóm hỗ trợ 2 kiểu dữ liệu cho giá trị của các thuộc tính trong theme, bao gồm color (màu sắc) và number (số).

- Giá trị thuộc tính hoàn toàn có thể thay đổi, đồng thời sẽ tự động cập nhật giá trị mới cho tất cả các thuộc tính của các node có tham chiếu tới nó.
- Người dùng có thể thêm, xoá, sửa các thuộc tính của theme một cách tùy ý. Tuy nhiên những thuộc tính đang được bất kỳ node nào tham chiếu đến sẽ không được phép xóa, cần phải xoá hết các tham chiếu này trước khi thực hiện xoá thuộc tính.

Một ví dụ về giao diện quản lý theme được mô tả ở hình 4.22.



Hình 4.22: Giao diện quản lý theme

### 4.3.11 Cơ chế quản lý cơ sở dữ liệu động

Mỗi dự án website builder sẽ có cơ sở dữ liệu động riêng do người dùng tự định nghĩa. Do tính chất linh động của cơ sở dữ liệu động, giao diện và cách cài đặt nó được nhóm mô phỏng từ cơ sở dữ liệu NoSQL với các collections và documents.

- Người dùng hoàn toàn có thể thực hiện thêm, xóa, sửa cho cả collections và documents.

- Khi thêm mới collection, người dùng cần định nghĩa danh sách các thuộc tính tương ứng với các trường dữ liệu của documents.
- Khi thêm mới documents, người dùng sẽ điền giá trị của các trường dữ liệu đã được định nghĩa ở collection tương ứng.
- Hỗ trợ thao tác tìm kiếm document dựa trên giá trị của một trường dữ liệu bất kỳ (được phép chọn).

Để nhúng dữ liệu động vào element, người dùng sẽ thực hiện kết nối dữ liệu động và lựa chọn một trong các giá trị thuộc tính của document để thay thế giá trị text hiện tại của node tương ứng.

#### **4.3.12 Cơ chế multiple-page và routing**

Multiple-page là tính năng cho phép thiết kế website với nhiều trang trên cùng một dự án, mỗi trang có một vai trò và giao diện riêng biệt. Routing là quá trình điều hướng người dùng đến các trang khác nhau của dự án dựa trên đường dẫn hiện tại. Tính năng được triển khai như sau:

- Mỗi trang có tên và đường dẫn duy nhất do người dùng tự đặt.
- Mỗi node sẽ thuộc về một trang duy nhất.
- Mỗi trang luôn có duy nhất một node root có id bắt đầu bằng "ROOT", thuộc loại container dùng để chứa tất cả các node khác của trang, và tất nhiên node root phải là một canvas node để có thể kéo thả các selector vào bên trong node root.

#### **4.3.13 Tùy chỉnh thuộc tính cho element**

Mỗi element sẽ có những tùy chỉnh thuộc tính riêng tùy thuộc vào loại component mà node tương ứng thuộc về, chung quy lại có thể chia làm 2 loại tùy chỉnh thuộc tính:

- Tuỳ chỉnh thuộc tính giao diện.
- Tuỳ chỉnh thuộc tính sự kiện.

Nhóm sẽ lần lượt trình bày cấu hình của từng loại tùy chỉnh.

## Tuỳ chỉnh thuộc tính giao diện

Các tùy chỉnh thuộc tính giao diện sẽ nằm ở khu vực customize trên side panel. Về cơ bản, mỗi loại component sẽ có các tùy chỉnh thuộc tính khác nhau, dựa trên những đặc điểm giao diện của element và giới hạn chọn lọc các thuộc tính CSS tương ứng mà lập trình viên thường điều chỉnh khi code front-end. Đây cũng chính là những thuộc tính props của component và được dịch thành CSS tương ứng nhờ thư viện styled-components [15].

Để dễ dàng hơn cho người dùng trong việc thay đổi giá trị thuộc tính của node, nhóm tiến hành phân chia, sắp xếp các tùy chỉnh vào từng cụm tùy chỉnh có ý nghĩa riêng ví dụ như Typography, Dimensions, Margin, Padding, Appearance, Decoration... Khi người dùng thay đổi giá trị thuộc tính, giá trị props của node cũng sẽ thay đổi ngay lập tức và re-render lại giao diện element tương ứng để người dùng theo dõi được sự thay đổi của element.

Giá trị của một thuộc tính có thể là một giá trị đơn hoặc một mảng các giá trị (ví dụ như margin lưu một mảng các giá trị theo thứ tự tương ứng với margin top, left, right, bottom). Có 3 cách để chỉnh sửa giá trị thuộc tính:

1. Default setting: người dùng sẽ tự nhập giá trị fix cứng cho thuộc tính. Buildify hỗ trợ đa dạng các loại nhập liệu như: text, number, select drop-down menu, slider, checkbox, radio, color picker, upload image.
2. Suggestion style: Buildify có xây dựng một bộ cấu hình sẵn dạng CSS tương tự như TailwindCSS, ví dụ như cấu hình font-size được mô tả ở hình 4.23. Khi người dùng chọn một trong những giá trị

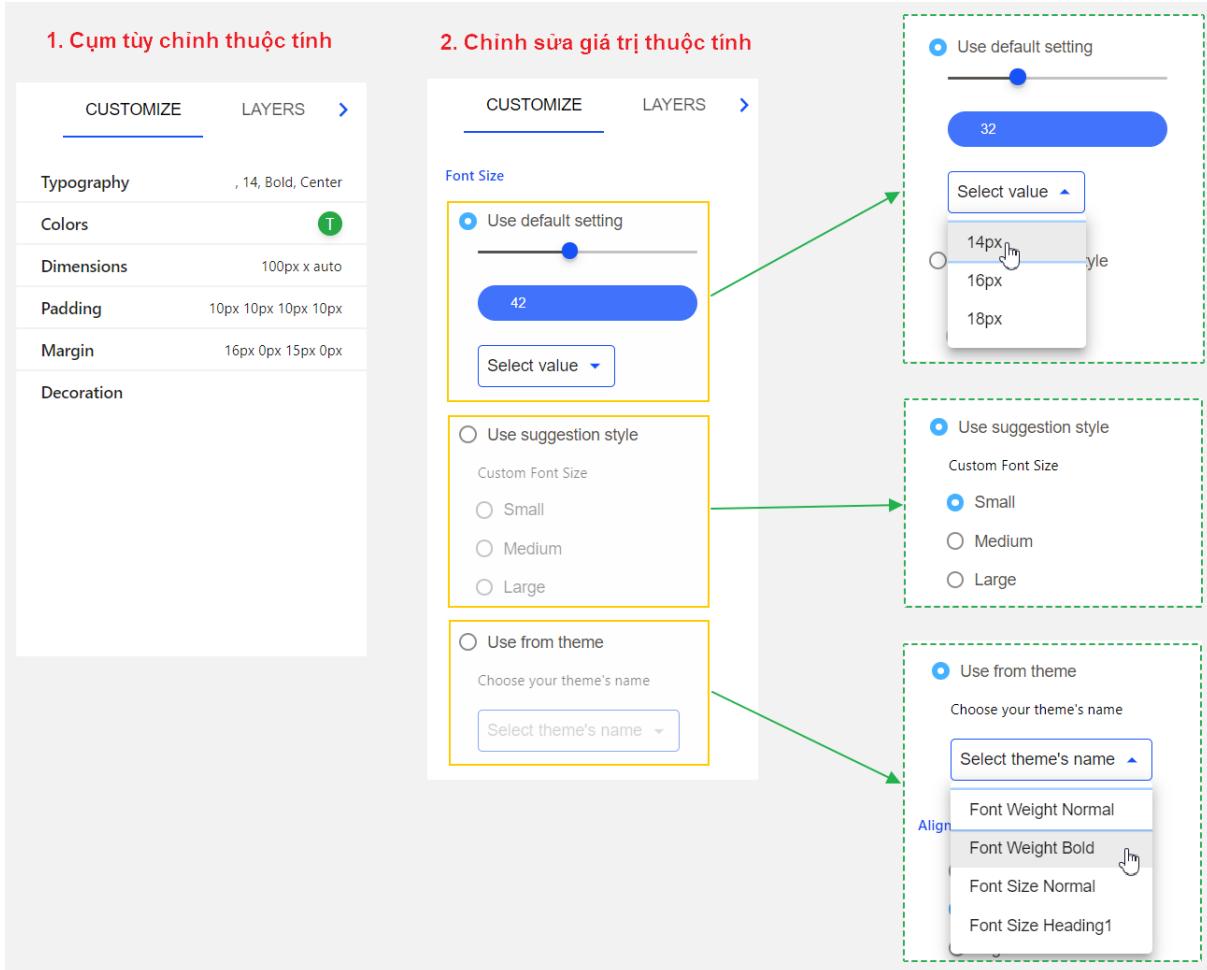
này, Buildify sẽ tự động thêm tên class tương ứng với giá trị được chọn vào props "styledClassNamesValues" của node, đây là thuộc tính chứa danh sách các class tùy chỉnh, các class này sẽ được tự động thêm vào HTML element khi chạy mã nguồn phát sinh).

```
.custom-fs-small{  
    font-size: 12px !important;  
}  
  
.custom-fs-medium{  
    font-size: 16px !important;  
}  
  
.custom-fs-large{  
    font-size: 22px !important;  
}
```

Hình 4.23: Cấu hình CSS của font size

3. Theme style: cho phép tham chiếu tới một giá trị thuộc tính nằm trong cấu hình theme đã được định nghĩa. Giá trị này phải có cùng kiểu dữ liệu với kiểu dữ liệu của thuộc tính cần chỉnh sửa.

Hình 4.24 mô tả trực quan các cụm tùy chỉnh thuộc tính cho loại button và 3 cách để chỉnh sửa giá trị cho thuộc tính font-size.



Hình 4.24: Ví dụ trực quan về tính năng tùy chỉnh thuộc tính giao diện

Ngoài ra, Buildify còn hỗ trợ thuộc tính lồng nhau (gọi là nested prop), tức là một node có thể chứa một thuộc tính mà giá trị của nó là toàn bộ props của một node con bên trong. Ví dụ: Node thuộc loại pop-up có chứa các node con bên trong như: title, content, first button, second button,... thì tương ứng có chứa các nested prop như titleComponent, contentComponent, firstButtonComponent, secondButtonComponent... Mỗi nested prop này lại chứa giá trị hoàn toàn giống như một node bình thường.

## Tùy chỉnh thuộc tính sự kiện

Ngoài các thuộc tính CSS liên quan đến giao diện của node, nhóm còn định nghĩa sẵn một bộ các thuộc tính giúp xử lý sự kiện cho một số loại

component nhất định nhằm tăng tính tương tác cho trang web. Ví dụ: với node thuộc loại button sẽ có thuộc tính sự kiện liên quan đến event "click" (tương ứng với việc gắn onClick cho element tương ứng). Hiện tại, Buildify hỗ trợ những thuộc tính sau:

- Navigate: click vào element sẽ chuyển trang nội bộ website (internal routing) hoặc mở một link bên ngoài (external link).
- Href: click vào element sẽ scroll màn hình tới một element được chọn (thuộc cùng một trang).
- Mở pop-up: click vào element sẽ mở giao diện pop-up đã được người dùng thiết kế.

Tính năng tùy chỉnh sự kiện này thực sự có ý nghĩa quan trọng vì hầu hết các trang web hiện đại đều có tính động và khả năng tương tác cao.

#### 4.3.14 Cơ chế lưu và load trạng thái

Lưu và load trạng thái là hai tính năng quan trọng của Buildify, cho phép lưu trữ, tải lại và tiếp tục làm việc trên dự án, thậm chí là chia sẻ thông tin thiết kế cho người khác.

1. Lưu trạng thái: sử dụng phương thức JSON.stringify để chuyển đổi toàn bộ dữ liệu hiện tại của editor (bao gồm danh sách nodes, danh sách pages, cấu hình theme) thành kiểu string. Tiếp theo, sử dụng mã hóa base64 cho chuỗi string này và gọi API để gửi về server. Server sẽ cập nhật trạng thái mới nhất của dự án thành chuỗi dữ liệu mã hóa này.
2. Load trạng thái: từ chuỗi nén base64, tiến hành giải mã và load lại toàn bộ thông tin editor để tiếp tục thiết kế trang web.

Ngoài ra, người dùng còn có thể sử dụng tính năng sao chép trạng thái để lấy được thông tin trạng thái hiện tại (dưới dạng chuỗi nén base64) của dự án và chia sẻ thông tin thiết kế cho người dùng khác. Sau đó, kết hợp với tính năng load trạng thái, người dùng tiến hành dán chuỗi dữ liệu đã chia sẻ/lưu trữ để thay thế trạng thái hiện tại của editor bằng dữ liệu mới giải mã được.

#### 4.3.15 Cơ chế phát sinh selector và các tùy chỉnh thuộc tính tương ứng

Nhằm đơn giản hóa quá trình thiết kế khi thêm một selector mới vào hệ thống của Buildify trong tương lai, cũng như tạo một nền tảng kiến trúc vững chắc, có hệ thống, dễ mở rộng và dễ chỉnh sửa, tránh việc lặp lại code, nhóm đã thiết kế các mẫu dữ liệu cho việc cấu hình cùng các hàm đọc và phát sinh dữ liệu tự động cho Buildify như sau:

##### Cấu hình phát sinh các selector

Ở các phần trên, nhóm trình bày selector như là một bản preview của element trên toolbox. Tuy nhiên trên thực tế, bản thân mỗi selector sẽ chứa nhiều thông tin hơn, mục đích là để phục vụ việc tạo ra node mới và render thành element trên editor. Cụ thể, mỗi selector sẽ chứa thông tin 2 phiên bản React Component tương ứng như sau:

1. View element: là phiên bản React Component bình thường, không cho phép thực hiện các tính năng chỉnh sửa (text, resize...), đây cũng là phiên bản React Component mà người dùng có được khi sử dụng tính năng phát sinh mã nguồn. Phiên bản này cho phép người dùng xem trước giao diện của element trước khi kéo thả vào editor.
2. Craft element: là phiên bản React Component có sử dụng các hook để kết nối với editor, cho phép editor được quản lý node tương ứng và thực hiện các tính năng chỉnh sửa trên element được render.

Ngoài ra, giao diện của phiên bản này sẽ có đôi chút khác biệt với view element về mặt kích thước nhưng vẫn sẽ giữ đúng tỉ lệ và vẻ ngoài (màu sắc, nội dung, độ đậm nhạt...). Điều này là cần thiết vì phiên bản view element được hiển thị trên một vùng toolbox có kích thước giới hạn với mục đích xem trước giao diện element, còn craft element sẽ là phiên bản thật sự được tạo ra nằm trong editor với một vùng thiết kế rộng hơn nhiều nên sẽ có kích thước to hơn để cân đối với vùng chứa.

Để có được danh sách selector được sắp xếp theo menu trên toolbox, cần định nghĩa cấu hình cho từng selector với các thuộc tính như sau (gọi là một **SelectorItemProps**):

1. CraftElement
2. ViewElement
3. overwritePropsView: các thuộc tính props được thiết kế riêng cho view element (sẽ được ghi đè với default props của React Component). Việc ghi đè này cho phép có nhiều biến thể của cùng một element. Ví dụ: với button có nhiều biến thể về màu sắc, kích thước, border...
4. overwritePropsCraft: tương tự với overwritePropsView nhưng dành cho craft element.
5. label: mô tả về selector, thường sẽ là tên của loại selector hoặc tên một phân loại sub-menu của selector đó.
6. Icon: biểu tượng của selector.
7. subItems: Array<SelectorItemProps> đây là thuộc tính để phục vụ tính năng sub-menu cho selector, với dữ liệu **SelectorItemProps** chính là bộ thuộc tính của một selector bên trong sub-menu đó.
8. isTemplate: có giá trị là boolean, xác định selector này có phải là built-in template hay không. Với built-in template sẽ có cách xử lý riêng trong phần tạo mới một craft element.

Một selector với cấu hình trên được render dưới dạng JSX như đoạn code 4.2: với logic xử lý của hàm "**fnCreateCraftItem**" đã được mô tả

```

1 <div
2   className='flex justify-center'
3   ref={(ref) => {
4     if (isTemplate) fnCreateCraftItem(ref, CraftElement);
5     else fnCreateCraftItem(ref, <CraftElement {...overwritePropsCraft} />);
6   }}>
7   <Tooltip title='Drag and drop to use this' placement='right'>
8     <ViewElement className='cursor-move' {...overwritePropsView} />
9   </Tooltip>
10 </div>

```

Mã nguồn 4.2: Tự động rendering JSX cho selector

trong hình 4.16. Cần lưu ý là "`ref`" ở đây dùng để tạo shadow khi kéo một selector từ menu. Khi gọi hàm "`fnCreateCraftItem`", với basic element thì chỉ cần truyền tham số thứ 2 là đoạn JSX có truyền "`overwritePropsCraft`" là đủ. Riêng với template là một tổ hợp của các basic element với nhau, đoạn JSX của `CraftElement` được tính toán từ một hàm bên ngoài.

### Cấu hình phát sinh các tùy chỉnh thuộc tính cho element

Ở phần 4.3.13 đã trình bày về các thiết kế hỗ trợ cho tính năng tùy chỉnh thuộc tính, phần này sẽ tập trung giới thiệu các cài đặt cấu hình liên quan. Tùy chỉnh thuộc tính giao diện và sự kiện đều có chung một cấu hình duy nhất, và để tiện trình bày trong phần này, chúng sẽ được gọi chung là thuộc tính. Như đã trình bày, mỗi node sẽ gắn với một cấu hình tùy chỉnh riêng để phù hợp với từng đặc điểm cụ thể, được định nghĩa bởi 2 thuộc tính "`settings`" (về giao diện) và "`events`" (về sự kiện) nằm trong trường "`related`" khi định nghĩa React Component của selector. Để các tùy chỉnh thuộc tính giữa các selector đồng bộ, thống nhất, tạo ra một hệ thống tùy chỉnh cho phép sử dụng chung, dễ thay đổi và mở rộng thì các tùy chỉnh thuộc tính cơ bản và cụm tùy chỉnh thuộc tính được thiết kế như sau:

- Tùy chỉnh thuộc tính cơ bản: gồm các tùy chỉnh cho một thuộc tính, bao gồm các thông tin như:

1. propKey: tên key tương ứng trong props của Component, dùng để cập nhật và hiển thị đúng giá trị từng thuộc tính của node. Ví dụ: "color".
  2. type: danh sách kiểu hiển thị nhập liệu tương ứng cho người dùng. Bao gồm: "text", "color", "bg", "number", "imageUpload", "slider", "radio", "select", "checkbox". Mỗi thuộc tính có thể có nhiều cách thức nhập giá trị giúp người dùng linh hoạt hơn.
  3. label: mô tả thuộc tính cho người dùng hiểu.
  4. styledCustomOptions: bộ các suggestion style bao gồm label, value.
  5. themeTypes: danh sách kiểu dữ liệu theme có thể chọn.
  6. radioChildren: mảng các label, value cho phép người dùng chọn theo dạng radio button.
  7. checkboxChildren: mảng các label, value cho phép người dùng chọn theo dạng checkbox button.
  8. selectChildren: mảng các label, value cho phép người dùng chọn dưới dạng danh sách các lựa chọn.
- Cụm tùy chỉnh thuộc tính: gồm danh sách các tùy chỉnh thuộc tính cơ bản, bao gồm các thông tin như:
    1. items: danh sách tên các tùy chỉnh thuộc tính cơ bản trong cụm, dùng tham chiếu đến cấu hình của tùy chỉnh thuộc tính cơ bản.
    2. props: danh sách các props React Component tương ứng của node mà cụm này quản lý, dùng để đọc giá trị từ node, tính toán và hiển thị cho người dùng dễ theo dõi.
    3. summary: hiển thị các giá trị thuộc tính hiện tại trong cụm.
    4. title: tên cụm tùy chỉnh thuộc tính.

Ngoài ra, việc cấu hình cho một selector mới là rất linh hoạt vì hàm đọc và phát sinh cấu hình tùy chỉnh thuộc tính còn **cho phép ghi đè** các

cấu hình mới so với cấu hình mặc định ban đầu nếu người dùng định nghĩa thêm. Nhờ vào việc cấu hình tự động như trên, khi thiết kế một selector mới, ví dụ như với text, lập trình viên chỉ cần điền đúng tên của các **cụm tùy chỉnh thuộc tính**: "typography", "margin", "appearanceText" nếu đã được định nghĩa trước đây, và có thể tạo mới/ghi đè các tùy chỉnh thuộc tính nếu cần.

## 4.4 Cài đặt back-end

### 4.4.1 Các services của back-end

Back-end hiện tại có 4 services, giao tiếp thông qua framework gRPC, bao gồm:

- User service (`user-service.buildify.asia`): thực hiện các API liên quan đến đăng ký, đăng nhập, quản lý thông tin người dùng và quản lý dự án.
- Dynamic data service (`dynamic-data-service.buildify.asia`): chịu trách nhiệm quản lý dữ liệu động, bao gồm các collections và documents, cung cấp API để tạo ra file script để khởi tạo dữ liệu cho database của người dùng.
- File management service (`file-mgt-service.buildify.asia`): quản lý các vấn đề liên quan đến upload, lưu trữ file.
- Generate code service (`gen-code-service.buildify.asia`): có nhiệm vụ sinh mã nguồn front-end, cụ thể là ReactJS App.

### 4.4.2 Phát sinh mã nguồn ReactJS

Mã nguồn được phát sinh dựa trên thông tin về danh sách nodes, danh sách pages, cấu hình theme và cơ sở dữ liệu động của dự án website. Quy trình cụ thể như sau:

- Bước 1: Khởi tạo ReactJS App bằng cách copy mã nguồn cơ sở đã được config sẵn từ trước.
- Bước 2: Phát sinh mã nguồn cho theme với thông tin nhận từ client.
- Bước 3: Phát sinh mã nguồn cho database với thông tin từ dynamic data service.
- Bước 4: Routing dựa trên thông tin về danh sách pages, gồm tên page và path tương ứng.
- Bước 5: Phát sinh mã nguồn cho từng page của project: bắt đầu từ node root của page đó, cụ thể là node có id bắt đầu bằng "ROOT", sử dụng giải thuật đệ quy merge lần lượt các nodes con của node root, đến các nodes con của các nodes con đó... Cuối cùng tạo được một element tree JSX của page đó. Nhóm cũng lưu ý trích xuất các thông tin như text, name từ thuộc tính để tăng khả năng nhận diện cho element khi người dùng đọc nội dung code của page.
- Bước 6: Format code với Prettier.
- Bước 7: Nén file zip và upload lên Cloud Storage.

Mã nguồn cơ sở đã được config sẵn là mã nguồn theo chuẩn cấu trúc ReactJS điển hình, bao gồm các folder, file sẽ không thay đổi trong suốt quá trình phát sinh mã nguồn. Một vài config đáng lưu ý như sau:

- Sử dụng một global context là ApplicationContext để lưu trữ dữ liệu theme và database, đảm bảo tất cả components của website luôn truy cập được theme và database.
- Sử dụng hook để truy xuất giá trị của theme, database từ thông tin thuộc tính của node.

- Đối với components của dự án, nhóm nhận thấy các components ở front-end quá phức tạp khi phải sử dụng hook để connect với editor, lắng nghe và xử lý các sự kiện khi người dùng thao tác, cho phép chỉnh sửa thuộc tính... Trong khi đó, mã nguồn cơ sở cần một component đơn giản hơn, có cấu trúc của một component điển hình trong ReactJS. Do đó nhóm thực hiện chia 2 versions cho front-end và back-end, front-end cần chia tách component để có cấu trúc giống version back-end và xử lý các sự kiện phức tạp hơn ở một file khác. Nhóm sử dụng Git submodule để quản lý phần chung giữa 2 version, giúp dễ dàng tích hợp, phát triển và bảo trì code bằng cách pull version mới nhất từ submodule khi components có sự thay đổi.

Để tăng tốc độ phát sinh mã nguồn, nhóm tận dụng goroutine trong Golang để xử lý đồng thời cho các bước 2, 3, 4, 5 đề cập ở trên vì những bước này có thể xử lý độc lập nhau với dữ liệu đầu vào cố định đã được xác định từ trước. Hơn nữa ở bước 6, nhóm chỉ format prettier với các file đã được chỉnh sửa hoặc phát sinh thêm trong quá trình phát sinh mã nguồn.

## Chương 5

# Quá trình thực hiện

*Chương 5 mô tả tổng quan quá trình thực hiện khóa luận của nhóm, đồng thời đưa ra những vấn đề nhóm gặp phải khi thực hiện khóa luận.*

## 5.1 Tổng quan

Sau khi nhận được các đề tài gợi ý từ giáo viên hướng dẫn, nhóm đã tiến hành tìm hiểu và chốt đề tài xây dựng một website builder để hỗ trợ phát sinh mã nguồn front-end. Quá trình thực hiện khóa luận bắt đầu từ ngày 01/02/2023 và được chia thành ba giai đoạn như sau:

1. Giai đoạn đầu tiên: tìm hiểu đề tài.

- Tiến hành tìm hiểu các mã nguồn mở liên quan đến website builder và phát sinh mã nguồn front-end để nắm được ý tưởng và cách triển khai.
- Khảo sát các sản phẩm hiện có trên thị trường và đưa ra ý tưởng về các tính năng cho sản phẩm mà nhóm đang xây dựng.

2. Giai đoạn thứ hai: cài đặt sản phẩm.

- Nghiên cứu, lựa chọn các công nghệ áp dụng cho phần front-end và back-end của website.

- Khởi tạo mã nguồn và tiến hành việc tìm hiểu song song với việc lên ý tưởng và triển khai các tính năng của sản phẩm.
3. Giai đoạn thứ ba: triển khai sản phẩm, viết bài báo khoa học và báo cáo khóa luận.
- Tiến hành triển khai sản phẩm lên môi trường thực tế.
  - Thực hiện một báo cáo khoa học giới thiệu về phương pháp xây dựng website builder có hỗ trợ phát sinh mã nguồn ReactJS.
  - Viết báo cáo khóa luận.

Về việc họp nhóm, định kỳ mỗi tuần nhóm sẽ có hai buổi họp vào thứ sáu và thứ bảy:

- Buổi họp thứ sáu là buổi họp nội bộ trong nhóm, nhằm báo cáo tiến độ công việc trong tuần vừa qua, tổng kết và lên kế hoạch công việc cho tuần tiếp theo.
- Buổi họp thứ bảy là buổi họp với giáo viên hướng dẫn, trong đó nhóm trình bày công việc đã thực hiện và kế hoạch cho tuần tiếp theo. Buổi họp này cũng là cơ hội để nhóm trao đổi với giáo viên về các vấn đề gặp phải, nhận ý kiến và góp ý từ giáo viên về cả mặt kỹ thuật và yêu cầu chức năng.

## 5.2 Vấn đề gặp phải khi phát triển ứng dụng

### 5.2.1 Vấn đề chung gặp phải

1. Kỹ năng ước lượng thời gian thực hiện cho một công việc còn chưa tốt: thời gian đầu, nhóm chưa có nhiều kinh nghiệm trong việc tính toán thời gian làm việc, dẫn đến đôi khi phải dời kế hoạch, cắt giảm tính năng hoặc thức khuya làm việc, ảnh hưởng đến tiến độ và cả sức khoẻ của các thành viên trong nhóm.

Ví dụ: với tính năng cho phép tạo nhiều trang thiết kế của Buildify, nhóm đã nghĩ nó sẽ nhanh và ước lượng chỉ 3 ngày, trong khi thực tế phải mất rất nhiều thời gian đọc hiểu, debug các luồng hiển thị và đọc dữ liệu của thư viện, dẫn đến mất khoảng gần 1 tuần.

2. Chưa có nhiều kinh nghiệm cho việc kiểm thử sản phẩm: đôi lúc nhóm lại tốn thời gian kiểm thử lại cho những phần không cần thiết khi thay đổi code, fix bug và có thể bỏ sót một số bug bên trong sản phẩm.

### **5.2.2 Vấn đề gặp phải khi tìm hiểu Website Builder**

1. Ít nghiên cứu liên quan: không có nhiều nghiên cứu về chủ đề website builder, đặc biệt chủ đề website builder có kèm phát sinh mã nguồn gần như không thấy có công trình nghiên cứu nào được công bố.
2. Tiếp xúc với nhiều khái niệm mới.
3. Độ phức tạp: việc hiểu và làm quen với tất cả các khía cạnh của website builder có thể mất rất nhiều thời gian và công sức.
4. Thiết kế bộ selector và trải nghiệm người dùng: website builder cần cung cấp một giao diện người dùng dễ sử dụng với một bộ selector phong phú và hấp dẫn để người dùng có thể thiết kế và tùy chỉnh trang web một cách thuận tiện. Điều này đòi hỏi kiến thức về thiết kế giao diện người dùng (UI/UX) và khả năng hiểu nhu cầu của người dùng.

### **5.2.3 Vấn đề gặp phải khi cài đặt front-end**

1. Thiếu kinh nghiệm với một số công nghệ và thư viện như immer, use-methods...: để làm quen với các công nghệ mới cần phải dành thời gian nghiên cứu và tìm hiểu cách sử dụng chúng. Điều này đôi

khi gây khó khăn và tốn nhiều thời gian thử và sai, đôi lúc cần phải đọc đến phần cài đặt bên trong.

2. Khó khăn trong việc tìm kiếm, lựa chọn kiến trúc và công nghệ phù hợp: việc tìm kiếm và lựa chọn kiến trúc, công nghệ phù hợp để xây dựng website builder không phải là điều dễ dàng. Có một vài lựa chọn có sẵn lúc đầu, tuy nhiên độ lớn và độ phức tạp để đáp ứng nhu cầu của nhóm ở những lựa chọn này còn hạn chế, tìm hiểu và đánh giá chúng để rút trích và tìm ra giải pháp tốt nhất là một thách thức, đòi hỏi nhóm phải mở rộng việc nghiên cứu sâu hơn nữa.
3. Hạn chế về số lượng thư viện mã nguồn mở: trong quá trình tìm hiểu, nhóm đã gặp khó khăn với việc tìm kiếm thư viện mã nguồn mở để tham khảo. Số lượng thư viện phù hợp có sẵn khá ít, điều này đòi hỏi nhóm phải sáng tạo và tự tìm giải pháp khi muốn xây dựng thêm các tính năng mới cho website builder.
4. Khó khăn trong việc đọc mã nguồn của thư viện đã được xây dựng gần 4 năm: khi đã tìm được nguồn tham khảo phù hợp nhất, nhóm gặp một thách thức khác là mã nguồn của nó đã được phát triển trong một thời gian dài với những logic phức tạp đã được tinh giản. Điều này tạo ra khó khăn trong việc đọc và hiểu logic của tác giả. Nhóm phải dành thời gian nghiên cứu và phân tích từng phần của mã nguồn, đôi khi phải vẽ sơ đồ và debug từng dòng để hiểu rõ ý tưởng và cách thức hoạt động.

#### **5.2.4 Vấn đề gặp phải khi cài đặt back-end**

Thử sức với mô hình microservices và các công nghệ mới, nhóm gặp khá nhiều vấn đề khi triển khai server, cụ thể:

1. Xây dựng thư viện sử dụng chung cho các service: nhóm mất khá nhiều thời gian để nghiên cứu, cấu hình cho phần này, các công việc

cần thực hiện bao gồm:

- Load cấu hình với Viper.
  - Cấu hình log với Logrus.
  - Xác thực interceptor với JWT, recovery interceptor.
  - Xây dựng proxy gateway: vì front-end không sử dụng gRPC mà sử dụng RESTful API nên back-end phải mở hai cổng cho mỗi service: một cổng HTTP và một cổng gRPC. Nhóm đã cố gắng sử dụng thư viện grpc-gateway để tạo ra proxy gateway. Ban đầu, nhóm chọn tích hợp các file proto của Google API để cấu hình đường dẫn HTTP trên proto, nhưng hay gặp lỗi và sau đó chuyển sang sử dụng Buf. Tiếp theo, nhóm phải xây dựng server gRPC và gateway sao cho các services không cần quá quan tâm đến sự tồn tại của proxy gateway, chỉ cần triển khai các API đã được định nghĩa trong proto là đủ.
2. Cấu hình Bazel: trong quá trình cấu hình Bazel, nhóm gặp nhiều khó khăn, cụ thể:
- Dependency: nhóm gặp vấn đề trong việc cấu hình các dependency, thường xuyên gặp lỗi mặc dù đã làm đúng hướng dẫn và log lỗi cũng không dễ hiểu. Sau đó nhóm nhận ra rằng có xung đột hoặc không tương thích giữa các phiên bản của các dependency. Sau khi tìm thấy một cấu hình đã được thử nghiệm trên GitHub với các dependency sẵn có, nhóm mới có thể chạy được.
  - Service gặp khó khăn nhiều nhất là service gen-code khi service này có sử dụng mã nguồn cơ sở (ở thư mục base) để phát sinh mã nguồn ReactJS. Mặc định bazel chỉ build file go, nhóm đã cấu hình để image build ra bao gồm cả mã nguồn cơ sở này, tuy nhiên thư mục base luôn không được tìm thấy khi chạy container. Nhóm phải tìm kiếm và đọc thông tin bazel đã build ra thì mới

cấu hình đúng được data\_path cho thư mục base. Vẫn đề tiếp theo là nhóm sử dụng Prettier để format code. Go image mặc định của Bazel không hỗ trợ cài đặt thêm package prettier và nhóm phải sử dụng dockerfile để có được image golang hỗ trợ chạy lệnh prettier. Có thể nói Bazel còn mới và tài liệu hướng dẫn chưa đủ nhiều, phải thử rất nhiều phương án khác nhau mới thành công config được image này.

### 3. Triển khai GKE cluster:

- Trước khi chuyển sang GKE, nhóm đã sử dụng Microsoft Azure. Azure có rất nhiều dịch vụ hỗ trợ triển khai container và cũng có rất nhiều video hướng dẫn config. Nhóm bị bối rối khi chọn dịch vụ phù hợp với nhu cầu, cũng đã thử qua khá nhiều dịch vụ và tốn rất nhiều thời gian nhưng không triển khai thành công.
- Sau khi chuyển sang GKE, mặc dù không có nhiều video hướng dẫn bằng Azure nhưng triển khai dễ dàng hơn. Tuy nhiên GKE lại giới hạn miễn phí 300\$ dùng thử trong vòng 3 tháng, nên nhóm phải dời sang tài khoản khác để được tiếp tục xài miễn phí. Do nhóm sử dụng HTTP kết hợp với cả cổng gRPC là một phương thức khá mới mẻ nên cũng không tìm được hướng dẫn đầy đủ, phải vừa làm vừa kết hợp tìm hiểu Kubenertes để hoàn thành được bước này.

### 4. Cấu hình workflow GitHub Actions cho CI/CD: đây cũng là một kiến thức mới với nhóm khi vừa build image, push image lên Artifact Registry và triển khai phiên bản mới cho các service tương ứng trên GKE cluster. Nhóm đã tìm hiểu và sử dụng công cụ Kustomize để dễ dàng cập nhật image tag, sử dụng biến có sẵn là github.run\_number để đánh số tag cho image.

Nhìn chung, triển khai mô hình Microservices phức tạp hơn mô hình monolithic rất nhiều, nhóm đã nỗ lực nghiên cứu để hoàn thành tốt nhiệm

vụ được giao, tuy đối mặt với nhiều vấn đề khó khăn nhưng kiến thức thu được là rất ý nghĩa.

## 5.3 Kiến thức học được khi thực hiện đề tài

### 5.3.1 Kiến thức chung

1. Kiến thức về quản lý trạng thái: website builder có nhiều thành phần tương tác và bộ phận phức tạp. Việc quản lý trạng thái để có thể truy xuất từ nhiều nơi, cập nhật và theo dõi giá trị cùng rất nhiều tính năng liên quan một cách hiệu quả và cho hiệu suất cao là rất quan trọng.
2. Kiến thức về kiến trúc phần mềm: việc xây dựng một website builder đòi hỏi kiến thức về kiến trúc phần mềm để tổ chức và quản lý các thành phần, tương tác giữa các Element và xử lý dữ liệu. Từ đó, nhóm đã tìm hiểu và nắm được các kiến thức này.
3. Kiến thức về tối ưu hóa hiệu suất: website builder có nhiều tính năng phức tạp nên cần xử lý tối ưu hóa hiệu suất để ứng dụng chạy mượt mà, có thời gian phản hồi nhanh, đảm bảo trải nghiệm người dùng.
4. Kiến thức về sử dụng Git để cộng tác trong một dự án, quản lý tính năng theo từng nhánh (branch) và cách hệ thống các commit để dễ truy xuất khi cần thiết. Ví dụ: cách đặt tên commit có ý nghĩa, commit bắt đầu bằng loại commit (feature/fix/refactor), tách commit theo tính năng nhỏ, gộp commit khi cần thiết...

### 5.3.2 Kiến thức front-end

1. Nghiên cứu rộng và sâu hơn về HTML, CSS: khi làm tính năng Tuỳ chỉnh giao diện cho element, cần nghiên cứu qua tất cả các thuộc tính CSS, hiểu bản chất của chúng để bộ cài đặt đa dạng cho từng

loại selector và để phục vụ một số tuỳ chỉnh phức tạp liên quan về bố cục trang web, các tính toán liên quan về vị trí thả selector vào một canvas element với trong nhiều trường hợp.

2. Cách thiết kế một cấu trúc (dạng json) cho phép cấu hình động (dynamic configuration) các loại dữ liệu mang tính lặp lại và sử dụng chung 1 logic, ví dụ với website builder: bộ Selector trên một menu hỗ trợ sub-menu (gồm 2 loại basic element và built-in template, mỗi loại lại khác nhau một chút về logic hiển thị), bộ các tuỳ chỉnh node về **giao diện** và **sự kiện** chung cho toàn website. Đồng thời học cách viết 1 hàm tổng quát để phát sinh tự động từ những cấu hình đó.
3. Kiến thức về tương tác sự kiện người dùng và kinh nghiệm xử lý các sự kiện: một website builder tốt phải có khả năng tương tác tốt với người dùng, nhóm học được cách triển khai các tính năng như kéo thả, chỉnh sửa nội dung trực tiếp trên editor, quản lý layer, và các tính năng tương tác khác. Trong đó cần linh hoạt xử lý các sự kiện như: dragover, dragenter, dragend, hover, focus, select, mouseover, mousedown, click.
4. Kiến thức về thiết kế giao diện người dùng (UI/UX): một website builder thành công phải có giao diện người dùng hấp dẫn và dễ sử dụng, nhóm học cách thiết kế và triển khai giao diện người dùng hợp lý để tối ưu trải nghiệm người dùng.
5. Kiến thức về ReactJS và TypeScript cùng với các công nghệ liên quan đã được liệt kê ở hình sơ đồ tổng quan 3.2 để xây dựng nên một website hoàn chỉnh.
6. Kiến thức về lựa chọn công nghệ và thư viện sao cho tối ưu nhất với dự án, RESTful API và giao tiếp với back-end.
7. Cách đóng gói và triển khai sản phẩm trên môi trường thực tế.

8. Kiến thức về sử dụng Git submodule để quản lý bộ Selector phục vụ tính năng phát sinh mã nguồn.
9. Phương pháp cài đặt tính năng quản lý lịch sử cho các chức năng undo và redo.
10. Cách thiết kế và chia nhỏ component hiệu quả, sử dụng các props của component để phục vụ tất cả các loại tuỳ chỉnh của một node từ giao diện, xử lí sự kiện, lấy giá trị từ context, nested prop...

### 5.3.3 Kiến thức back-end

1. Triển khai mô hình microservices: nhóm áp dụng mô hình quản lý mã nguồn monorepo trên nền tảng GitHub và triển khai các server trên GKE cluster. Điều này giúp nhóm dễ dàng quản lý, theo dõi và triển khai các phiên bản của các services một cách thống nhất.
2. Nắm vững ngôn ngữ GoLang và sử dụng goroutines để tối ưu hiệu suất của các services: Bằng cách sử dụng goroutines, nhóm có thể thực hiện các tác vụ đồng thời một cách hiệu quả, cải thiện hiệu suất và tăng khả năng đáp ứng của hệ thống.
3. Sử dụng gRPC để thực hiện việc giao tiếp giữa các services: gRPC là một phương thức giao tiếp hiệu quả và đáng tin cậy để truyền tải dữ liệu giữa các thành phần của hệ thống.
4. Sử dụng Bazel để build và đẩy image lên registry. Bazel giúp đóng gói các ứng dụng phần mềm một cách hiệu quả và cũng hỗ trợ quá trình triển khai các container image một cách dễ dàng, nhất quán.
5. Triển khai CI/CD bằng GitHub Actions: nhóm đã cấu hình và triển khai quy trình CI/CD trên nền tảng GitHub. Điều này giúp tự động hóa các quy trình, tiết kiệm thời gian và đảm bảo sự ổn định trong quá trình phát triển hệ thống.

6. Phân tích cơ sở dữ liệu và sử dụng MongoDB: nhóm rèn luyện được khả năng phân tích cơ sở dữ liệu và lựa chọn MongoDB làm hệ quản trị cơ sở dữ liệu. MongoDB cung cấp tính năng linh hoạt và khả năng truy xuất dữ liệu với hiệu suất cao, phù hợp với yêu cầu của dự án.

#### **5.3.4 Kỹ năng**

1. Kỹ năng quản lí thời gian, sắp xếp và phân công công việc cũng như xử lý linh hoạt để hoàn thành công việc đúng thời hạn với chất lượng tốt.
2. Kỹ năng giao tiếp và làm việc nhóm.
3. Kỹ năng lập kế hoạch dài hạn, chia nhỏ từng mục tiêu (liên quan đến việc đặt ra cột mốc) và sắp xếp thứ tự ưu tiên từng mục tiêu để phù hợp với thời gian cũng như nguồn nhân lực hiện có.
4. Kỹ năng tự học và tự nghiên cứu các công nghệ, thư viện hiện có.
5. Khả năng đọc hiểu mã nguồn của thư viện và tuỳ chỉnh, mở rộng theo nhu cầu.
6. Kỹ năng phân tích yêu cầu phần mềm và lên ý tưởng, thiết kế giao diện cho tính năng mới.

#### **5.3.5 Kiến thức về lĩnh vực website builder**

1. Kiến thức về khái niệm, bản chất, phân loại, các thành phần chủ yếu của một website builder.
2. Ý nghĩa và vai trò quan trọng của website builder trong việc xây dựng website cơ bản cho người chưa có kinh nghiệm lập trình và lập trình viên.

3. Trải nghiệm các sản phẩm website builder nổi tiếng trên thị trường hiện nay, các ưu và nhược điểm của chúng cùng khả năng áp dụng trong tương lai.
4. Kiến thức về kiến trúc xây dựng một phần mềm website builder: Editor, Node (dữ liệu của một Node, lưu Node props để gắn vào React Component, Canvas Node, Node Tree), các quy tắc (rule) liên quan quá trình kéo và thả, các dạng sự kiện (event) lưu trữ, danh sách Selector, Layer, quản lí History...

# Chương 6

## Kết luận và hướng phát triển

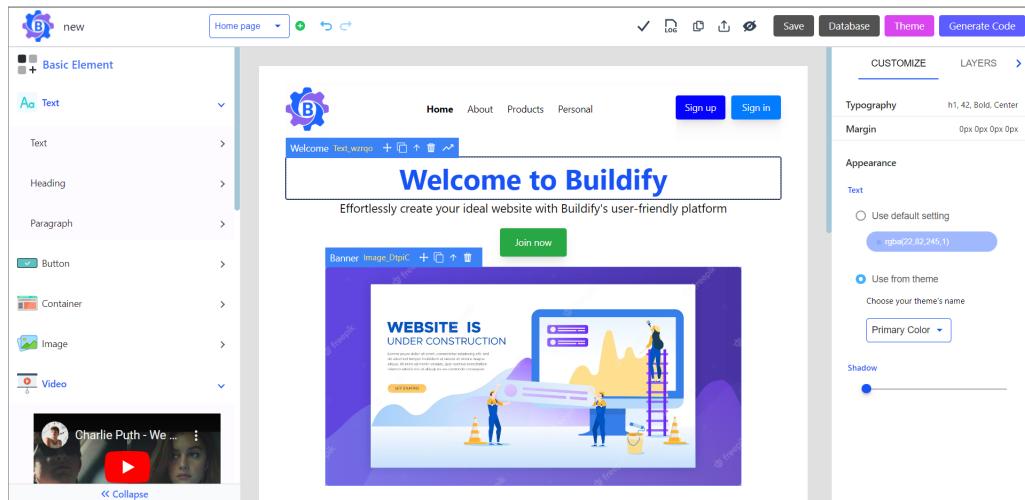
*Chương 6 tổng kết lại kết quả nhóm đạt được và ý nghĩa của các kết quả đó, đồng thời nêu lên các vấn đề tồn đọng và hướng phát triển của đề tài.*

### 6.1 Kết quả đạt được

#### 6.1.1 Website Builder

Trong khoá luận này, nhóm đã triển khai thành công một sản phẩm website builder hoàn chỉnh với đầy đủ các thành phần của một trang web cơ bản như: landing page, các trang xác thực người dùng, phần quản trị người dùng (admin), đặc biệt là **phần website builder** và tích hợp được tính năng **phát sinh mã nguồn** từ giao diện thiết kế. Những phần giao diện khác đã được trình bày ở phần tổng quan giao diện 4.1, mục này sẽ trình bày kết quả của tính năng phát sinh mã nguồn như sau: giao diện của thiết kế, cấu trúc của mã nguồn được phát sinh và giao diện kết quả của trang web khi chạy mã nguồn.

Hình 6.1 thể hiện giao diện một thiết kế người dùng của Buildify, mang tính trực quan và hấp dẫn, dễ dàng trong việc điều hướng và tương tác.



Hình 6.1: Giao diện thiết kế người dùng của Buildify

Hình 6.2 minh họa mã nguồn phát sinh từ giao diện thiết kế trang web ở hình 6.1, cụ thể là danh sách các components và mã nguồn chi tiết của một component (Anchor) được xây dựng và phát sinh trong ReactJS sau khi được tải về và giải nén.

```

EXPLORER
YDAM_NEW_SOURCE_CODE
  node_modules
  public
  src
    components
      Anchor
        index.tsx
        props.ts
        styled.ts
      Button
      Container
      Image
      Input
      Popup
      Text
      Video
        constant.ts
        index.ts
        constants
        context
        database
        hooks
        pages
          Home
            index.tsx
            props.tsx
          Join
            index.ts
          routes
          styles
          theme
          utils
            App.css
            App.test.tsx
            App.tsx
            index.css
            index.tsx
            logo.svg
        OUTLINE
        TIMELINE

index.tsx ... \Home
index.tsx ... \Anchor

src > components > Anchor > index.tsx > (e) Anchor > (e) mapClickEvent
4 import { Text, TRIM } from './text';
5 import { AnchorProps, EventKeys, defaultProps } from './props';
6
7 import { StyledAnchor } from './styled';
8 export const Anchor = (props: AnchorProps) =>
9 > | const [...]
  > |   = props;
10
11 const styledClassNamesValues = (Object.values(styledClassNames) as string[]).flat();
12
13 const { pageNavigate, absoluteUrlNavigate, href, clickType } = events;
14 const handleNavigate = () =>
15   if (isUsedHref) return;
16   if (pageNavigate || absoluteUrlNavigate) {
17     const desUrl = pageNavigate || absoluteUrlNavigate;
18     window.location.href = desUrl;
19   }
20
21 };
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

```

Hình 6.2: Cấu trúc thư mục và một đoạn code ReactJS component

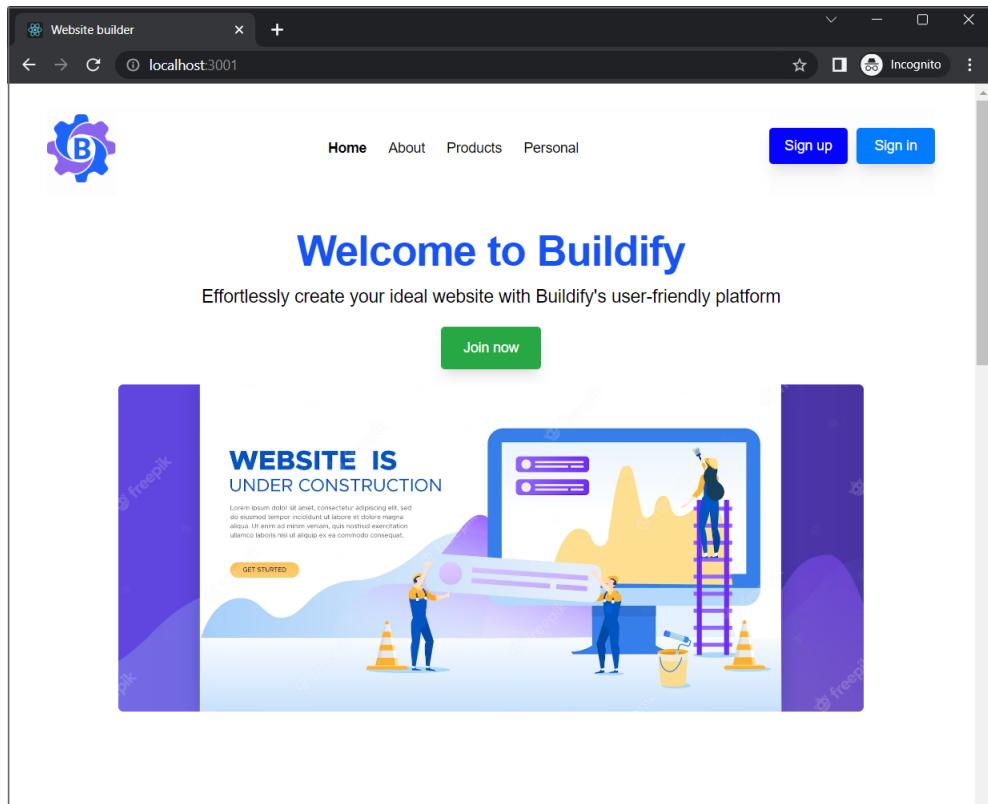
Thời gian phát sinh mã nguồn ở môi trường localhost trên máy chỉ tầm 2s, trong khi đó với môi trường thực tế thì thời gian phản hồi có thể lên đến 15s.

Hình 6.3 thể hiện cấu trúc các component JSX được phát sinh và sắp xếp theo đúng thứ tự thiết kế ở mỗi trang.

```
src > pages > Home > index.tsx > [e] Home
1 import React, { FC, ReactElement } from "react";
2 import { Container, Text, Button, Image, Anchor } from "src/components";
3 import { refProps } from "./props";
4 import { useGetValuesFromReferencedProps } from "src/hooks/useGetValuesFromReferencedProps";
5
6 const Home: FC = (): ReactElement => {
7   const props = useGetValuesFromReferencedProps(refProps);
8   return [
9     <Container name="App" {...props.ROOT}>
10    <Container name="Header" {...props.Container_FMvVO}>
11      <Image name="Logo" {...props.Image_sLIrc} />
12      <Container name="Nav" {...props.Container_nJSWU}>
13        <Anchor text="Home" {...props.Anchor_IHwSE} />
14        <Anchor text="About" {...props.Anchor_jmfNK} />
15        <Anchor text="Products" {...props.Anchor_gLywl} />
16        <Anchor text="Personal" {...props.Anchor_pmgpo} />
17    </Container>
18    <Button name="SignUp" text="Sign up" {...props.Button_VDpnc} />
19    <Button name="SignIn" text="Sign in" {...props.Button_juijI} />
20  </Container>
21  <Container name="Introduction" {...props.Container_wfSGL}>
22    <Text text="Welcome to Buildify" {...props.Text_wzrqa} />
23    <Text
24      name="Description"
25      text="Effortlessly create your ideal website with Buildify's user-friendly platform"
26      {...props.Text_KCykI}
27    />
28    <Button text="Join now" {...props.Button_ZYaya} />
29    <Image name="Banner" {...props.Image_DtpiC} />
30  </Container>
31  <Container {...props.Container_IYQDo}>
32    <Text name="Products" text="Our products" {...props.Text_EncCY} />
33    <Container {...props.Container_rTUbI}>...
34    </Container>
35  </Container>
36  <Container {...props.Container_cjpKd}>
37    <Container {...props.Container_DgiqA} />
38  </Container>
39  <Container name="Footer" {...props.Container_VTNky}>
40    <Container name="Contact" {...props.Container_lKSMs}>
41      <Text text="Contact us" {...props.Text_wJEZx} />
42      <Container {...props.Container_UiqFO}>
43        <Text text="Phone: 0836256444" {...props.Text_qJUpL} />
44        <Text
45          text="Address: 227 Nguyễn Văn Cừ, Quận 5, TP Hồ Chí Minh, Việt Nam"
46        />
47    </Container>
48  </Container>
```

Hình 6.3: Đoạn mã nguồn phát sinh cho Home page

Sau khi cài đặt tất cả các gói (packages) cần thiết và chạy mã nguồn trên localhost, có thể thấy được giao diện của trang web được xây dựng từ mã nguồn như trong hình 6.4, giống với giao diện đã thiết kế trước đó trong hình 6.1.



Hình 6.4: Giao diện chạy website trên localhost của trang web được tạo ra từ mã nguồn phát sinh

### So sánh mã nguồn phát sinh với TeleportHQ

Sau khi đã trải nghiệm thiết kế và phát sinh một mã nguồn trang web từ TeleportHQ, nhóm ghi nhận lại các kết quả như sau:

The screenshot shows a code editor interface with two main sections: an Explorer sidebar on the left and a code editor on the right.

**EXPLORER**

- > public
- > src
  - > components
    - doctor.css
    - doctor.js
    - features.css
    - features.js
    - practice.css
    - practice.js
  - > views
    - home.css
    - home.js
    - index.js
    - style.css
  - craco.config.js
  - package.json

**JS features.js**

```

src > components > JS features.js > Features
1 import React from 'react'
2
3 import PropTypes from 'prop-types'
4
5 import './features.css'
6
7 const Features = (props) => {
8   return [
9     <div className="features-section quick-links">
10       <div className="features-heading">
11         <h3>{props.Title}</h3>
12         <img alt="image" src={props.Icon} className="features-icon" />
13       </div>
14       <p>{props.Description}</p>
15       <div className="features-divider"></div>
16     </div>
17   ]
18 }
19
20 Features.defaultProps = {
21   Description: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.',
22   Icon: '/Icons/arrow.svg',
23   Title: 'Virtual Assistant',
24 }
25
26 Features.propTypes = {
27   Description: PropTypes.string,
28   Icon: PropTypes.string,
29   Title: PropTypes.string,
30 }
31
32 export default Features
33

```

Hình 6.5: Cấu trúc thư mục và một đoạn code ReactJS component của TeleportHQ

```

js home.js x
src > views > js home.js > ...
1 import React from 'react'
2 import { Link } from 'react-router-dom'
3
4 import Script from 'dangerous-html/react'
5 import { Helmet } from 'react-helmet'
6
7 import Features from '../components/features'
8 import Practice from '../components/practice'
9 import Doctor from '../components/doctor'
10 import './home.css'
11
12 const Home = (props) => {
13   return (
14     <div className="home-container">
15       <Helmet>
16         <title>Medica template</title>
17         <meta property="og:title" content="Medica template" />
18       </Helmet>
19       <div data-modal="practices" className="home-modal">
20         <div className="home-practices">
21           <div className="home-heading">
22             <span className="home-header">Our practices</span>
23             <svg
24               viewBox="0 0 1024 1024"
25               data-close="practices"
26               className="home-close"
27             >
28               <path d="M225.835 286.165l225.835 225.835-225.835 225.835c-16.683 16.683 43.691 0 60.331s4
29                 835c16.683 16.683 43.691 16.683 60.331 0s16.683-43.691 0-60.331l-225.835-225.835 225.835-225.835c
30                 683-60.331 0l-225.835 225.835-225.835c-16.683-43.691-16.683-60.331 0s-16.683 43.691 0 60.331s4
31             </svg>
32           </div>
33           <div className="home-grid">
34             <div className="home-section">
35               <div className="home-heading01">
36                 <span className="home-header01">Cardiology</span>
37                 <span className="home-caption">
38                   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
39                   do eiusmod tempor incididunt.
                 </span>
               </div>
             </div>
           </div>
         </div>
       </div>
     </div>
   )
}

```

Hình 6.6: Đoạn mã nguồn phát sinh cho một page của TeleportHQ

Từ những kết quả trên, có thể rút ra một số kết luận sau:

- Về mặt cấu trúc thư mục, mã nguồn phát sinh của Buildify ở hình 6.2 đầy đủ, chuyên nghiệp và gần với dự án ReactJS thực tế hơn TeleportHQ ở hình 6.5. Có thể thấy TeleportHQ chỉ phát sinh các component và page (với 2 file js và css), đây là cấu trúc đơn giản nhất của một dự án ReactJS khi khởi tạo. Trong khi đó, ngoài các component và page, Buildify phát sinh đầy đủ các thư mục khác như constants (chứa các hằng số), context (định nghĩa React context), hooks (một thư mục quan trọng trong phát triển ReactJS theo phong cách functional component), routes (chứa các định nghĩa cấu hình cho

routing của website và hiển thị các trang theo path), utils (các hàm tiện ích), theme (bộ chủ đề website), database (bộ các cơ sở dữ liệu) và thư mục styles sử dụng SCSS (nơi tập trung các CSS dự án). Như vậy, cấu trúc thư mục mã nguồn phát sinh của Buildify có ý nghĩa thực tiễn và khả năng áp dụng, mở rộng và tiếp tục phát triển trong tương lai cao hơn TeleportHQ. Với TeleportHQ, lập trình viên sẽ cần phải tái cấu trúc lại mã nguồn dự án.

- Về mặt chi tiết mã nguồn của một component, có thể thấy Buildify có nhiều hỗ trợ xử lý động dựa trên các thuộc tính trong props bằng logic JavaScript, đặc biệt liên quan đến các xử lý sự kiện (event) như "click"/"navigate" hoặc thuộc tính "href" ở hình 6.2... thay vì chỉ đơn giản là đọc từ props và hiển thị giao diện như TeleportHQ.
- Về mặt chi tiết mã nguồn của một page, Buildify hiển thị một danh sách có thứ tự của các component ở hình 6.3, mang lại một cái nhìn tổng quan về các thành phần của trang web theo thứ tự từ trên xuống dưới. Khi cần thêm thông tin chi tiết về một component, lập trình viên có thể click vào xem cài đặt mã nguồn hoặc xem danh sách props được gắn vào tương ứng.

Với TeleportHQ (ở hình 6.6), có thể thấy mã nguồn của một page được hiển thị chi tiết hơn, không chỉ là danh sách các component như Buildify mà còn có các thẻ HTML, được đặt tên class với ngữ nghĩa đi kèm. Ở điểm này, TeleportHQ cho thấy mã nguồn tự nhiên và gần gũi hơn Buildify; tuy nhiên điều đó cũng có một hạn chế là làm mã nguồn một page trở nên dài hơn và khó nắm bắt cấu trúc của page dưới một cái nhìn tổng quan như Buildify.

- So với TeleportHQ, Buildify có hỗ trợ tích hợp sẵn TypeScript vào dự án để khai báo các kiểu dữ liệu tường minh hơn, giúp việc đọc hiểu mã nguồn trở nên dễ dàng. Ngoài ra, Buildify còn tích hợp sẵn thư viện Tailwind CSS, người dùng có thể tận dụng trong tương lai

nếu cần.

### 6.1.2 Paper

Từ những nghiên cứu trong khoá luận, nhóm đã viết một bài báo khoa học với chủ đề "A Methodological Approach to Building a ReactJS Code Generation Website Builder". Bài báo là kết quả của quá trình nghiên cứu và xây dựng Buildify, tập trung vào việc giới thiệu và phân tích cài đặt chi tiết cho một phương pháp xây dựng website builder hoàn chỉnh với tính năng phát sinh mã nguồn front-end ReactJS.

## 6.2 Ý nghĩa của kết quả

- Ý nghĩa trong học tập: nhóm đã tìm hiểu và thực hành cách xây dựng một hệ thống website builder phức tạp với tính năng phát sinh mã nguồn, đầy đủ tính năng hoàn chỉnh với nhiều xử lý sự kiện và áp dụng nhiều công nghệ mới.
- Phân tích và tổng hợp một phương pháp xây dựng website builder hoàn chỉnh từ nhiều nguồn và đóng góp cho cộng đồng một mã nguồn mở về cách xây dựng một website builder cho phép phát sinh mã nguồn giao diện cho ReactJS. Điều này giúp cộng đồng có thể tận dụng và phát triển các phần mềm xây dựng website builder dựa trên kiến thức và kinh nghiệm của Buildify.
- Khi công bố mã nguồn mở Buildify, nhóm có kèm giải thích chi tiết các cài đặt bên trong website builder (kiến trúc xây dựng và các luồng xử lý phức tạp), cung cấp nguồn kiến thức xây dựng và hướng dẫn sử dụng mã nguồn. Điều này cho phép lập trình viên tùy chỉnh mã nguồn theo nhu cầu của họ và có khả năng tự thực hiện các cải tiến và tuỳ chỉnh trong tương lai.

- Cung cấp một hệ thống website builder cho phép người dùng không có kiến thức về lập trình có thể tự thiết kế một sản phẩm website đầy đủ tính năng cho riêng mình nhờ vào bộ thư viện các selector được thiết kế sẵn.

## 6.3 Các vấn đề tồn đọng

### 6.3.1 Vấn đề tồn đọng của website builder

Trang web hiện tại còn một số vấn đề tồn đọng cần được giải quyết để cải thiện trải nghiệm người dùng như sau:

1. Thiếu tính responsive: hiện tại, trang web chưa hỗ trợ tính năng thiết kế website trên các thiết bị khác nhau và kích thước màn hình khác nhau.
2. Thư viện selector chưa phong phú: bộ selector thiết kế sẵn còn chưa thật sự đủ nhiều và đẹp so với các website builder hiện có trên thị trường như Wix, Squarespace... và các tùy chỉnh tương tác sự kiện người dùng vẫn chưa đầy đủ so với một website thông thường.
3. Dữ liệu động hiện nay chỉ hỗ trợ kiểu chuỗi (string) và số (number) đơn giản, điều này có thể hạn chế khả năng lưu trữ và quản lý dữ liệu phức tạp hơn như mảng (array), đối tượng (object) và các kiểu dữ liệu đặc biệt khác.
4. Chưa hỗ trợ triển khai và hosting website của người dùng lên Internet.
5. Chưa hỗ trợ tích hợp dữ liệu động từ API của người dùng hay bên thứ ba như Airtable, AWS...

### **6.3.2 Vấn đề tồn đọng của chức năng phát sinh mã nguồn front-end**

Hiện tại, chức năng phát sinh mã nguồn front-end của nhóm còn một số hạn chế sau:

1. Thời gian phát sinh mã nguồn chưa tối ưu trong môi trường production: khi triển khai website builder ở môi trường thực tế, thời gian phản hồi có thể lên đến 15s, trong khi thời gian thử nghiệm ở môi trường local (cấu hình: vi xử lý i7-1255U 1.70 GHz, 10 cores, 16GB RAM) chỉ tầm 2s. Một trong những nguyên nhân có thể là do network latency khi triển khai GKE cluster và hạn chế về phân bổ tài nguyên cho service gen-code (CPU: 200 milliCPU, 500MiB RAM). Sử dụng goroutines của GoLang để xử lý đồng thời nhiều tác vụ có thể gấp hạn chế khi số lõi CPU bị giới hạn, dẫn đến chênh lệch đáng kể trong thời gian phản hồi giữa môi trường thử nghiệm và môi trường thực tế.
2. Hiện chỉ hỗ trợ tính năng phát sinh mã nguồn cho ReactJS, điều này có thể gây hạn chế cho người dùng muốn sử dụng các library/framework khác.
3. Mã nguồn chưa hỗ trợ responsive: hiện tại, website builder chưa hỗ trợ responsive, và việc này ảnh hưởng trực tiếp đến mã nguồn phát sinh. Điều này có nghĩa là mã nguồn của người dùng sẽ không đáp ứng tốt trên các thiết bị di động hoặc kích thước màn hình khác nhau.
4. Chưa hỗ trợ phát sinh file CSS: nhóm hiện sử dụng Styled-Components để thuận tiện cho việc thay đổi thuộc tính CSS thông qua giá trị bên trong props và chưa hỗ trợ phát sinh file CSS.
5. Chưa tích hợp gọi API để lấy dữ liệu động cho dự án, có thể gây khó

khăn cho người dùng muốn tạo ra các trang web động và tương tác với dữ liệu từ các nguồn khác nhau.

## 6.4 Hướng phát triển của đề tài

### 6.4.1 Mở rộng theo hướng ứng dụng

Để cung cấp trải nghiệm tốt hơn và linh hoạt hơn cho người dùng, nhóm có thể phát triển thêm những tính năng sau đây trong tương lai:

1. Tích hợp công cụ kiểm tra và tối ưu hóa SEO: cho phép dự án tích hợp với các công cụ kiểm tra và tối ưu hóa SEO để giúp người tăng khả năng tìm thấy trang web của họ trên Internet.
2. Tích hợp tính năng chia sẻ và cộng tác (collaboration) vào công cụ website builder. Điều này cho phép nhiều người cùng làm việc trên một dự án website tại một thời điểm.
3. Tích hợp tính năng phân tích và theo dõi người dùng: có thể theo dõi lưu lượng truy cập, xu hướng sử dụng và hiệu suất của trang web, từ đó cải thiện và tối ưu hóa trải nghiệm người dùng.
4. Hỗ trợ responsive: giúp trang web của người dùng trông chuyên nghiệp và hấp dẫn trên mọi thiết bị. Điều này mở rộng khả năng tiếp cận và tương tác của khách hàng đối với dự án của người dùng trên nhiều nền tảng thiết bị.
5. Tích hợp tính năng tự động triển khai và host website: ứng dụng hỗ trợ người dùng đăng ký tên miền và triển khai server để chia sẻ website của mình với mọi người một cách nhanh chóng và dễ dàng thay vì phải tự tìm hiểu và sử dụng các dịch vụ hosting bên ngoài.

6. Hỗ trợ đa ngôn ngữ: cung cấp khả năng dịch và hỗ trợ đa ngôn ngữ cho trang web, giúp khách hàng từ các quốc gia và khu vực khác nhau có thể sử dụng được website của người dùng.
7. Tích hợp tính năng tương tác xã hội: hỗ trợ tích hợp các tính năng tương tác xã hội như chia sẻ nội dung trên mạng xã hội, bình luận và đánh giá từ khách hàng, tạo ra một môi trường tương tác đa dạng và thú vị trên website của người dùng.
8. Tích hợp plugin: tính năng plugin cho phép tích hợp không giới hạn các selector mới từ cộng đồng mở, bao gồm cả người dùng và lập trình viên muốn đóng góp hoặc cộng tác cho website builder. Lúc này, hệ thống cần cung cấp các API và template chuẩn để mọi người có thể đóng góp xây dựng bộ selector từ đơn giản đến phức tạp cho người dùng cuối. Các selector chỉ cần tuân thủ các chuẩn quy tắc được công bố sẽ được xét duyệt để tích hợp vào bộ selector của hệ thống Buildify.
9. Da dạng các sự kiện và hiệu ứng: ứng dụng cung cấp nhiều hơn nữa khả năng tùy chỉnh sự kiện và các hiệu ứng bắt mắt để người dùng có thể tạo ra những trải nghiệm động và hấp dẫn trên website của mình.

### 6.4.2 Mở rộng theo hướng nghiên cứu

Dưới đây là một số hướng nghiên cứu mở rộng có thể được khám phá và phát triển trong tương lai:

1. Phát triển tính năng user component: người dùng có khả năng hoàn toàn tùy chỉnh và định nghĩa các components riêng từ các components có sẵn. Điều này cho phép người dùng tạo ra các component cá nhân hóa để tái sử dụng trên toàn bộ website. Đồng thời, những

components này cũng sẽ được tự động phát sinh thành các file components khi người dùng sử dụng tính năng phát sinh mã nguồn của website builder

2. Hỗ trợ tải dữ liệu động từ nhiều nguồn: Ứng dụng hỗ trợ nhúng dữ liệu từ các API có sẵn và người dùng cần cung cấp các token (mã chứng thực người dùng) để gọi API. Điều này cho phép người dùng nhúng dữ liệu từ các dịch vụ lưu trữ bên thứ ba như Airtable, AWS và nhiều nguồn dữ liệu khác vào dự án website.
3. Bổ sung các kiểu dữ liệu phức tạp hơn như mảng, đối tượng... cho các trường dữ liệu động thay vì chỉ hỗ trợ chuỗi (string) và số (number) như hiện tại.
4. Tích hợp các công cụ thiết kế đồ họa: cung cấp tích hợp với các công cụ thiết kế đồ họa như Photoshop, Sketch hoặc Figma để người dùng có thể nhúng các giao diện vẽ sẵn của mình từ nguồn bên ngoài.

# Tài liệu tham khảo

## Tiếng Việt

- [1] *4 tiện ích mở rộng (extensions) trên VSCode giúp bạn khi phát triển với TailwindCSS.* URL: <https://viblo.asia/p/4-tien-ich-mo-rong-extensions-tren-vscode-giup-ban-khi-phat-trien-voi-tailwindcss-WR5JRxoDVGv> (visited on 12/06/2023).
- [2] *[Front-End] Cùng tìm hiểu về Ant Design, một thư viện đặc lực của Front-End.* URL: <https://viblo.asia/p/front-end-cung-tim-hieu-ve-ant-design-mot-thu-vien-dac-luc-cua-front-end-1VgZv00M5Aw> (visited on 12/06/2023).
- [3] *React Hook Form - xử lý form dễ dàng hơn bao giờ hết.* URL: <https://viblo.asia/p/react-hook-form-xu-ly-form-de-dang-hon-bao-gio-het-RnB5pAdDKPG> (visited on 12/06/2023).
- [4] *Reactjs là gì? Những thành phần chính của Reactjs.* URL: <https://bizflycloud.vn/tin-tuc/reactjs-la-gi-20220511171943895.htm> (visited on 12/06/2023).
- [5] *So sánh thư viện UI Ant Design và Material UI.* URL: <https://c0thuysontnhp.edu.vn/so-sanh-thu-vien-ui-ant-design-va-material-ui-15tavape> (visited on 12/06/2023).
- [6] *Triển khai app sử dụng Vercel trong 2 phút.* URL: <https://www.komaster.dev/post/trien-khai-app-su-dung-vercel-trong-2-phut> (visited on 12/06/2023).

- [7] *Tìm hiểu virtual dom là gì và cách hoạt động.* URL: <https://xaydungso.vn/blog/tim-hieu-virtual-dom-la-gi-va-cach-hoat-dong-vi-cb.html> (visited on 12/06/2023).
- [8] *Tìm hiểu về NextJS.* URL: <https://techmaster.vn/posts/37519/tim-hieu-ve-nextjs> (visited on 12/06/2023).

## Tiếng Anh

- [9] *A collaborative front-end platform with integrated UI development and content modelling tools.* URL: <https://teleporthq.io> (visited on 12/06/2023).
- [10] *Bazel build.* URL: <https://bazel.build/> (visited on 12/06/2023).
- [11] Beltramelli, Tony. “pix2code: Generating code from a graphical user interface screenshot”. In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 2023, pp. 1–6.
- [12] Bokuweb. *re-resizable - A resizable component for React.* URL: <https://github.com/bokuweb/re-resizable> (visited on 12/06/2023).
- [13] *Design to Code Plugin.* URL: <https://www.figma.com/community/tag/designtocode/plugins> (visited on 12/06/2023).
- [14] *Docker.* URL: <https://www.docker.com/> (visited on 12/06/2023).
- [15] Evan Jacobs Phil Pluckthun, Glen Maddern Max Stoiber et al. *Visual primitives for the component age. Use the best bits of ES6 and CSS to style your apps without stress.* URL: <https://styled-components.com> (visited on 12/06/2023).
- [16] *Go language.* URL: <https://go.dev/> (visited on 12/06/2023).
- [17] *gRPC framework.* URL: <https://grpc.io/> (visited on 12/06/2023).
- [18] *Kubernetes.* URL: <https://kubernetes.io/> (visited on 12/06/2023).

- [19] Lee, Andrew. *Generating Webpages from Screenshots*. 2023.
- [20] Michel Weststrate Alec Larson, Gregory Assasie et al. *immerjs - Create the next immutable state by mutating the current one*. URL: <https://github.com/immerjs/immer> (visited on 12/06/2023).
- [21] *MongoDB*. URL: <https://www.mongodb.com/> (visited on 12/06/2023).
- [22] Moran, Kevin et al. “Machine learning-based prototyping of graphical user interfaces for mobile apps”. In: *IEEE Transactions on Software Engineering* 46.2 (2023), pp. 196–221.
- [23] Prev Wong Andy Krings-Stern, Mateusz Drulis et al. *A React Framework for building extensible drag and drop page editors*. URL: <https://github.com/prevwong/craft.js> (visited on 12/06/2023).
- [24] *ReactContext*. URL: <https://legacy.reactjs.org/docs/context.html> (visited on 12/06/2023).
- [25] *TeleportHQ vs Wordpress*. URL: <https://www.saashub.com/compare-teleporthq-vs-wordpress> (visited on 12/06/2023).
- [26] *useReducer*. URL: <https://legacy.reactjs.org/docs/hooks-reference.html#usereducer> (visited on 12/06/2023).
- [27] *Wix - Create a website without limits*. URL: <https://www.wix.com> (visited on 12/06/2023).
- [28] *Wix vs TeleportHQ*. URL: <https://www.saashub.com/compare-wix-vs-teleporthq> (visited on 12/06/2023).
- [29] *Wix vs Wordpress.com*. URL: <https://www.webcreate.io/wix-vs-wordpress-com-int> (visited on 12/06/2023).
- [30] *Wix vs Wordpress.com*. URL: <https://www.forbes.com/advisor/business/software/wix-vs-wordpress/> (visited on 12/06/2023).
- [31] *Wordpress - Open your online store with a powerful, flexible platform designed to grow with you*. URL: <https://wordpress.com> (visited on 12/06/2023).