

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Hệ thống tự động sinh mã code dựa trên mẫu thiết kế vẽ tay

TẠ CAO CẢNH

canh.tc160394@sis.hust.edu.vn

Ngành Kỹ thuật phần mềm

Giảng viên hướng dẫn: PGS. TS. Cao Tuấn Dũng

Bộ môn: Công nghệ phần mềm

Viện: Công nghệ thông tin và truyền thông

HÀ NỘI, 01/2021

ĐỀ TÀI TỐT NGHIỆP

Biểu mẫu của Đề tài/khóa luận tốt nghiệp theo qui định của viện, tuy nhiên cần đảm bảo giáo viên giao đề tài ký và ghi rõ họ và tên.

Trường hợp có 2 giáo viên hướng dẫn thì sẽ cùng ký tên.

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Lời cam kết

Họ và tên sinh viên: Tạ Cao Cảnh

Điện thoại liên lạc: 086 8787 398

Email: canh.tc98@gmail.com

Lớp: CNTT2.01

Bộ môn: Công nghệ phần mềm

Em – *Tạ Cao Cảnh* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân em dưới sự hướng dẫn của *PGS.TS Cao Tuấn Dũng*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng em, không sao chép theo bất kỳ công trình nào khác. Mọi tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, dữ liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Em xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Lời cảm ơn

Đi qua những năm tháng Bách Khoa, em mới biết tuổi trẻ đáng trân trọng như thế nào. Trân trọng vì những lúc được thầy cô nhiệt tình chỉ bảo, vì những lúc cùng bạn bè trải qua biết bao niềm vui nỗi buồn, vì những lúc khó khăn tưởng chừng như gục ngã nhưng đều đã vượt qua để rồi trưởng thành hơn rất nhiều.

Cảm ơn Bách Khoa ! 5 năm, một quãng thời gian có lẽ chẳng đáng gì so với cả cuộc đời nhưng có thể đã là tất cả của tuổi thanh xuân. Em không thể nhớ rõ Bách Khoa đã cho mình bao nhiêu cũng như mình đã cống hiến cho Bách Khoa được gì, chỉ biết rằng tuổi trẻ có Bách Khoa và chắc chắn em sẽ không bao giờ quên điều đó.

Em xin được gửi lời cảm ơn chân thành nhất đến các thầy cô giáo trong trường Đại học Bách Khoa Hà Nội cùng các thầy cô trong Viện Công nghệ Thông tin và Truyền thông và đặc biệt là thầy giáo PGS.TS Cao Tuấn Dũng đã truyền dạy cho em những kiến thức bổ ích, giúp em xây dựng nền móng vững chắc cho sự phát triển trong sự nghiệp và trong cuộc sống sau này.

Và lời cảm ơn cuối cùng xin dành cho gia đình em, những con người luôn âm thầm nuôi nấng, đùm bọc và cho em ăn học tới thời điểm ngày hôm nay.

Tóm tắt nội dung đề án

Ngày nay, các hệ thống ứng dụng web và di động ngày càng trở lớn và phức tạp hơn, đòi hỏi phải có sự phối hợp chặt chẽ giữa nhiều bên liên quan. Một hệ thống được tạo nên không chỉ đáp ứng tốt các nghiệp vụ của người dùng mà nó còn đòi hỏi đáp ứng tốt trải nghiệm của người dùng khi tương tác với hệ thống. Khi các nghiệp vụ của phía server được xây dựng lên phục vụ cho cái mà người dùng “cần” thì giao diện người dùng là cái làm cho người dùng trở nên hứng thú khi làm việc. Vì vậy, song với việc phát triển nghiệp vụ tốt, việc phát triển giao diện người dùng cũng không kém phần quan trọng.

Đối với một quá trình phát triển giao diện người dùng, việc đầu tiên luôn là tạo ra các bản vẽ thiết kế giao diện, hay còn gọi là wireframe. Wireframe được các nhà thiết kế web tạo ra nhằm mô tả giao diện người dùng của hệ thống. Sau khi hoàn thành, bản thiết kế này sẽ được chuyển đến tay các lập trình viên để xây dựng giao diện người dùng bằng các dòng mã code. Việc chuyển đổi từ wireframe sang mã code này đòi hỏi người lập trình viên phải có kinh nghiệm cũng như sẽ tiêu tốn rất nhiều thời gian. Để có thể cắt giảm chi phí về nhân lực cũng như thời gian, đề án này hướng tới xây dựng hệ thống chuyển đổi trực tiếp wireframe thành mã code cho các nền tảng khác nhau (ứng dụng web, ứng dụng di động) phát triển dựa trên bài toán Sketch2code của tác giả Alexander Robinson.

Hiện tại, hệ thống Sketch2code có thể chuyển đổi bản thiết kế wireframe thành mã code HTML đơn giản bao gồm 5 phần tử (element) như: title, image, button, input, paragraph. Đề án này hướng tới ba nhiệm vụ chính:

- Chuyển đổi wireframe thành code Reactjs cho ứng dụng web
- Chuyển đổi wireframe thành code Reactnative cho phát triển ứng dụng di động
- Mở rộng thêm một số phần tử HTML phổ biến khác
- Cải tiến bài toán sinh mã code

Thông qua đề án này, chúng ta sẽ biết cách để áp dụng trí tuệ nhân tạo vào việc giải quyết các vấn đề thực tế, đồng thời nắm bắt xu hướng phát triển ứng dụng web và ứng dụng di động hiện nay.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU CHUNG.....	1
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu và phạm vi đề tài.....	2
1.3 Định hướng giải pháp	3
1.4 Bố cục đồ án.....	4
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ	5
2.1 Bài toán Sinh mã code tự động	5
2.1.1 Quy trình thiết kế ứng giao diện người dùng	5
2.1.2 Wireframes	5
2.1.3 Quy trình sinh mã code tự động.....	6
2.1.4 Các nghiên cứu liên quan	8
CHƯƠNG 3. CÁC CÔNG NGHỆ SỬ DỤNG.....	13
3.1 Các kĩ thuật học máy	13
3.1.1 Mạng nơ-ron nhân tạo	13
3.1.2 Bài toán Phân vùng hình ảnh.....	15
3.2 Các công nghệ xây dựng ứng dụng website và di động	16
3.2.1 ASP.NET Core.....	16
3.2.2 Razor.....	16
3.2.3 Cấu trúc của một website	17
3.2.4 Reactjs	18
3.2.5 Thiết kế nguyên tử (Atomic design)	18
3.2.6 Bootstrap.....	18
3.2.7 React Native.....	18
3.3 Microsoft Azure: Dịch vụ tính toán đám mây	19
3.3.1 Thị giác máy tính	19
3.3.2 Blob storage	20
CHƯƠNG 4. PHÂN TÍCH VÀ TRIỂN KHAI ỨNG DỤNG	21
4.1 Phân tích yêu cầu phần mềm.....	21
4.1.1 Khảo sát hiện trạng.....	21
4.1.2 Tổng quan chức năng	23
4.1.3 Đặc tả chức năng.....	28
4.1.4 Yêu cầu phi chức năng	30

4.2	Kiến trúc hệ thống	30
4.3	Thiết kế chi tiết gói	32
4.4	Thiết kế lớp.....	34
4.5	Thiết kế cơ sở dữ liệu.....	40
4.6	Xây dựng ứng dụng	43
4.6.1	Thư viện và công nghệ sử dụng.....	43
4.6.2	Kết quả đạt được	44
CHƯƠNG 5. CÁC ĐÓNG GÓP NỔI BẬT		55
5.1	Xây dựng cấu trúc cây HTML	55
5.2	Sinh mã code dựa trên cấu trúc cây HTML	59
5.2.1	Quy trình sinh mã code từ cấu trúc HTML	59
5.2.2	Sử dụng cú pháp Razor cho khởi tạo file	60
5.2.3	Tạo code HTML và code Reactjs nhúng vào code HTML	61
5.2.4	Tạo code Reactjs theo kiến trúc Atomic	63
5.2.5	Tạo code Reactnative cho nền tảng di động.....	65
5.3	Mở rộng đối tượng nhận dạng.....	67
5.4	Kết quả đạt được.....	69
CHƯƠNG 6. KẾT LUẬN		70
6.1	Kết luận	70
6.2	Giải pháp và hướng phát triển.....	70
TÀI LIỆU THAM KHẢO		72

DANH MỤC BẢNG VÀ HÌNH VẼ

Hình 1.1 Quy trình thiết kế giao diện người dùng	1
Hình 1.2 Kiến trúc hệ thống.....	3
Hình 2.1 Bản vẽ thiết kế Wireframe	5
Hình 2.2 Các phần tử xuất hiện trong wireframe	6
Hình 2.3 Phần tử mới như link, pagination, carousel, table trong wireframe	6
Hình 2.4 Quy trình sinh mã code tự động	7
Hình 2.5 Mô tả mạng huấn luyện từ bài báo pix2code	9
Hình 2.6 Quy trình sinh mã code của pix2code.....	9
Hình 2.7 Mô hình SketchCode sử dụng chuỗi token mô tả làm đầu vào.....	10
Hình 2.8 Mỗi bản vẽ có thể tạo ra nhiều kiểu thể hiện khác nhau.....	11
Hình 2.9 Xác định vị trí các phần tử xuất hiện trong bản thiết kế.....	11
Hình 2.10 Sketch2code chuyển đổi wireframe thành mã code HTML	12
Hình 3.1 Một đơn vị xử lý của mạng nơ-ron nhân tạo.....	13
Hình 3.2 Mạng nơ-ron truyền thẳng nhiều lớp	14
Hình 3.3 So sánh nhận dạng đối tượng và phân vùng đối tượng.....	15
Hình 3.4 Kiến trúc mô hình DeepLab	16
Hình 3.5 Cây cấu trúc HTML	17
Hình 3.6 Các thành phần trong Atomic design.....	18
Hình 3.7 Nhận dạng kí tự quang học.....	20
Hình 4.1 Figma hỗ trợ sinh mã code CSS cho các phần tử.....	21
Hình 4.2 Hệ thống Sketch2code	22
Hình 4.3 Biểu đồ use case tổng quan	23
Bảng 4.1 Mô tả use case tổng quan với tác nhân người dùng	24
Bảng 4.2 Mô tả use case tổng quan với tác nhân quản trị viên	25
Hình 4.4 Biểu đồ phân rã “Tải ảnh lên hệ thống”	25
Hình 4.5 Biểu đồ phân rã “Xem kết quả nhận dạng đối tượng”	26
Hình 4.6 Biểu đồ phân rã “Sinh mã code tự động”	27
Hình 4.7 Biểu đồ phân rã “Sinh mã code tự động”	28
Hình 4.8 Luồng sự kiện “Sinh mã code tự động”	29
Hình 4.9 Luồng sự kiện “Quản lý phản hồi người dùng”	30
Hình 4.10 Mô hình MVC.....	31
Hình 4.11 Thiết kế tổng quan hệ thống	32
Hình 4.12 Thiết kế gói “Core”	33
Hình 4.13 Thiết kế gói “Webserver”	33
Hình 4.14 Biểu đồ phần tử liên kết	40
Bảng 4.3 Cấu trúc bảng PredictedDetails	41

Bảng 4.4 Cấu trúc bảng Image.....	41
Bảng 4.5 Cấu trúc bảng Slide_Image.....	42
Bảng 4.6 Cấu trúc bảng GroupBox	42
Bảng 4.7 Cấu trúc bảng BoundingBox.....	42
Bảng 4.8 Cấu trúc bảng PredictedObject	43
Bảng 4.9 Cấu trúc bảng Admin.....	43
Hình 4.15 Giao diện trang chủ.....	45
Hình 4.16 Giao diện preview kết quả sinh mã code	45
Hình 4.17 Giao diện cập nhật dữ liệu.....	46
Hình 4.18 Giao diện các lựa chọn sinh mã code.....	46
Hình 4.19 Mã code Reactjs nhúng trong file HTML	48
Hình 4.20 Mã code HTML và kết quả hiển thị tương ứng.....	49
Hình 4.21 Mã code Reactjs thiết kế theo module Atomic.....	50
Hình 4.22 Mã code JSX và giao diện tương ứng.....	50
Hình 4.23 File View chính trong ReactNative được sinh tự động	52
Hình 4.24 Giao diện danh sách lịch sử sinh mã code	52
Hình 4.25 Giao diện chi tiết lịch sử sinh mã code	53
Hình 4.26 Giao diện quản lý phản hồi người dùng.....	53
Hình 4.27 Giao diện quản lý mô hình dự đoán.....	54
Hình 5.1 Toạ độ và kích thước boundingBox của các đối tượng	56
Hình 5.2 Biểu đồ dữ liệu xuất phát từ nút gốc và từ Group [6].....	56
Hình 5.3 Bản vẽ thiết kế huấn luyện cho mô hình sketchcode và pix2code	57
Hình 5.4 So sánh các vị trí các đối tượng theo chiều ngang và chiều dọc.....	57
Hình 5.5 Các đối tượng chồng chéo lẫn nhau.....	58
Hình 5.6 Độ chồng chéo của các đối tượng.....	58
Hình 5.7 Hệ thống element do Bootstrap cung cấp	61
Hình 5.8 Layout sinh mã code HTML	62
Hình 5.9 Model tạo element Dropdown.....	62
Hình 5.10 Layout file HTML nhúng ReactJs	63
Hình 5.11 Kiến trúc mã nguồn Reactjs theo thiết kế Atomic.....	64
Hình 5.12 Layout file Page/index.js trong Reactjs Atomic.....	64
Hình 5.13 Component Atom cho Dropdown.....	65
Hình 5.14 Model render Dropdown tương ứng	65
Hình 5.15 Layout file View/index.js trong ReactNative.....	66
Hình 5.16 Component Dropdown cho code Reactjs.....	67
Hình 5.17 Công cụ đánh nhãn đối tượng của Azure.....	67
Hình 5.18 So sánh wireframe và mã code của pagination	68

Hình 5.19 So sánh wireframe và mã code của textarea	68
Hình 5.20 So sánh wireframe và mã code của carousel.....	68
Hình 5.21 So sánh wireframe và mã code của checkbox.....	68
Hình 5.22 So sánh wireframe và mã code của radio button.....	68
Hình 5.23 Hai dạng hiển thị table trong wireframe	69

Danh mục các từ viết tắt

HTML	HyperText Markup Language Ngôn ngữ đánh dấu siêu văn bản
CSS	Cascading Style Sheets Định dạng các phần tử được tạo ra bởi ngôn ngữ đánh dấu
JS	Javascript Ngôn ngữ lập trình kịch bản
DSL	Domain-specific language Ngôn ngữ cụ thể tên miền
API	Application programming interface Giao diện lập trình ứng dụng
GUI	Graphical user interface Giao diện đồ hoạ người dùng
CNTT	Công nghệ thông tin
ĐATN	Đồ án tốt nghiệp
ANN	Artificial neural network Mạng nơ-ron nhân tạo
CNN	Convolutional Neural Network Mạng nơ-ron tích chập

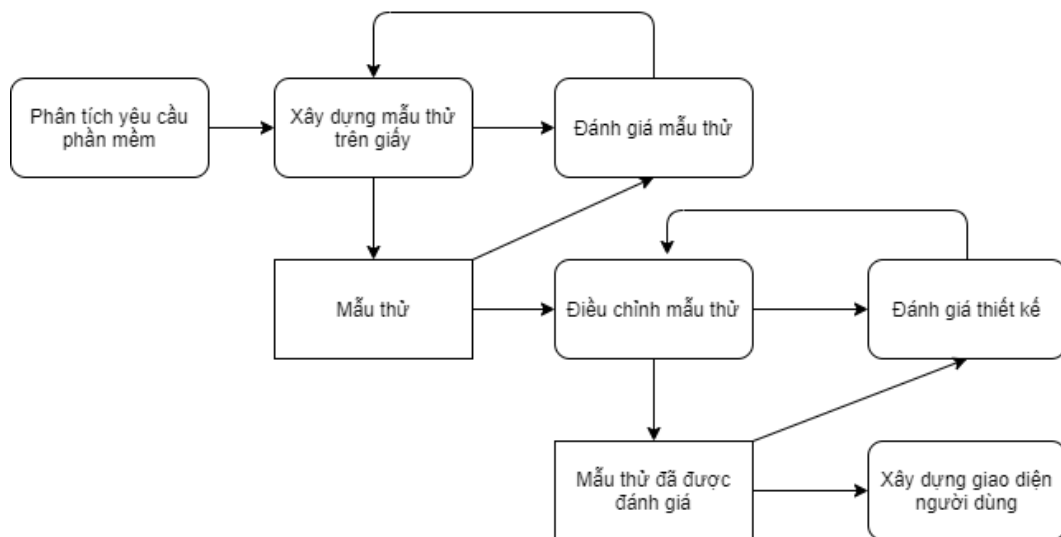
Danh mục thuật ngữ

Developer	Lập trình viên
Designer	Nhà thiết kế
Wireframe	Khung xương/ cấu trúc dây Công cụ trực quan để thiết kế trang web ở mức độ cấu trúc
Open-source	Mã nguồn mở
Styling	Tạo phong cách cho giao diện
Token	Một từ trong danh sách từ điển (mạng LSTM)

CHƯƠNG 1. GIỚI THIỆU CHUNG

1.1 Đặt vấn đề

Để tạo ra một giao diện người dùng có chức năng và hấp dẫn người dùng, các nhà thiết kế sẽ cố gắng phác thảo ra ý tưởng của họ trên bảng trắng, trên giấy, ... Khi bản phác thảo đạt được các tiêu chí đặt ra cả về giao diện và chức năng, nó được chụp lại dưới dạng hình ảnh và được cung cấp cho các lập trình viên UI chuyển đổi thành mã code HTML, CSS và Javascript để nó có chạy được trên các trình duyệt web. Khi quy trình chuyển đổi hoàn thành, các nhà phát triển web tương tác với ứng dụng mô phỏng của họ trên trình duyệt từ đó đưa ra các đánh giá và chỉnh sửa cần thiết. Bản phác thảo thứ hai ra đời dựa trên các thay đổi đó và các lập trình viên UI sẽ chỉnh sửa lại code để đáp ứng theo các yêu cầu mới này. Quy trình này được lặp đi lặp lại cho đến khi bản ứng dụng mô phỏng đạt được các tiêu chí đặt ra ban đầu. Cho dù quy trình này có thú vị tới đâu đi chăng nữa, nó cũng yêu cầu rất nhiều sự nỗ lực của cả đội phát triển và tất nhiên, sự chậm trễ trong quy trình phát triển phần mềm là không thể tránh khỏi.



Hình 1.1 Quy trình thiết kế giao diện người dùng

Để tiết kiệm chi phí và thời gian, các nhà nghiên cứu đã đưa ra các giải pháp chuyển đổi tự động bản vẽ thiết kế thành mã code một cách tự động. Việc xây dựng ứng dụng chuyển đổi bản vẽ thiết kế thành mã code đem lại lợi ích như:

- Vòng lặp nhanh hơn - việc chuyển đổi bản thiết kế thành mẫu thử chỉ cần tới sự tham gia của nhà thiết kế.
- Khả năng tiếp cận – cho phép những người không phải là lập trình viên có thể tạo nên ứng dụng cho riêng họ.
- Cho phép các nhà phát triển tập trung vào logic của ứng dụng hơn là giao diện người dùng.
- Cắt giảm thiểu chi phí và thời gian.

Từ những năm 2000, đã có những hệ thống chuyển đổi bản vẽ thành mã code, nổi bật như:

- SILK - Biến đổi bản vẽ kỹ thuật số thành mã ứng dụng bằng cử chỉ
- DENIM: hệ thống hỗ trợ phác thảo cho phép thiết kế ở các mức độ tinh chỉnh khác nhau và thống nhất các cấp độ thông qua thu phóng (Zooming User Interface)
- REMAUI: chuyển đổi ảnh chụp màn hình có độ trung thực cao thành ứng dụng dành cho di động

Hầu hết các hệ thống trên chỉ dừng lại ở việc áp dụng các kỹ thuật thị giác máy tính cổ điển nhằm giải quyết các bài toán con trọng điểm như phát hiện và phân loại đối tượng, ... vì thế chất lượng của các hệ thống này chưa thể đạt được kỳ vọng của tác giả đặt ra. Hiện tại, khi các kỹ thuật trí tuệ nhân tạo ngày càng phát triển, các bài toán liên quan tới lĩnh vực phát hiện và phân loại đối tượng nói trên đã đạt được những thành quả nhất định, từ đó tạo cơ hội cho việc tiếp tục phát triển bài toán này.

1.2 Mục tiêu và phạm vi đề tài

Bài toán chuyển đổi bản vẽ thiết kế thành mã code hiện nay không phải là quá mới, cho đến khi các kỹ thuật học máy được giới thiệu gần đây, nó mở ra một tiềm năng phát triển mới cho bài toán này. Các nghiên cứu và ứng dụng gần đây thay vì sử dụng các kỹ thuật thị giác máy tính kinh điển bằng việc áp dụng các kỹ thuật học máy mới, giúp nâng cao chất lượng của bài toán. Có 2 lĩnh vực nghiên cứu chính được xem xét trong bài toán này, đó là chuyển đổi các bản vẽ có độ tin tưởng thấp (bản vẽ tay hoặc tương tự) thành mã code và chuyển đổi khung hình/ảnh chụp màn hình có độ tin tưởng cao thành mã code.

Tuy nhiên, để xây dựng các bản vẽ có độ tin cậy cao đòi hỏi ở nhà thiết kế phải có kinh nghiệm cũng như tiêu tốn rất nhiều thời gian vào việc phác thảo này. Khi đó, với một khách hàng là một người hoàn toàn mới trong lĩnh vực thiết kế, rất khó để có thể mô tả được ý tưởng của họ trên giấy sao cho sát với yêu cầu đặt ra cho ứng dụng mà hệ thống chuyển đổi có thể sử dụng được. Hơn thế nữa, sự phát triển đáng kể trong lĩnh vực thị giác máy tính, việc triển khai bài toán dựa trên các bản vẽ có độ tin cậy thấp là hoàn toàn khả thi. Từ đó bất kỳ ai cũng có thể vẽ được bản phác thảo cho riêng mình một cách nhanh chóng, dễ dàng để tạo nên một ứng dụng hoạt động mà không cần đào tạo.

Trong phạm vi đề án này, em hướng tới phát triển 2 phần: tiếp tục cải tiến và mở rộng bài toán sinh mã code tự động cho việc xây dựng giao diện người dùng trên nền tảng web và nền tảng di động dựa trên nghiên cứu sketch2code; Xây dựng trang quản lý quá trình sinh mã code và huấn luyện dữ liệu.

Hiện tại, hệ thống sketch2code với đầu vào là các ảnh chụp bản phác thảo vẽ tay, đã có thể đưa ra các phần tử xuất hiện trong bản vẽ và tọa độ tương ứng bao gồm: title, image, button, input, paragraph và chuyển đổi thành mã code HTML, CSS đơn giản. Khi so sánh với hệ thống phần tử HTML thực tế hiện tại, con số này là khá nhỏ, chưa kể đến các phần tử rất phổ biến như table, link, dropdown, checkbox,

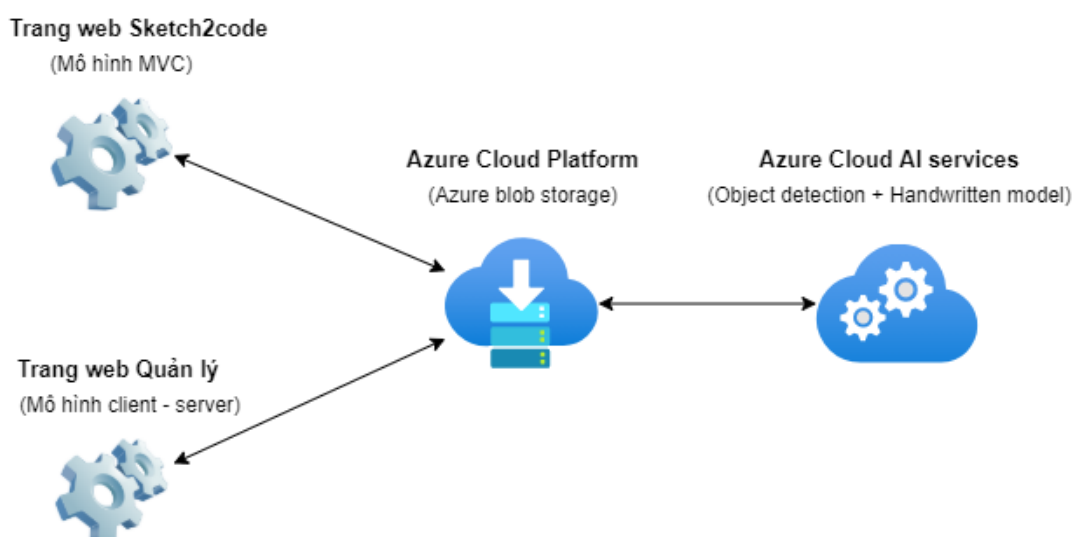
... Việc cải tiến bài toán cho phép mở rộng thêm các phần tử này cho sát với thực tế cũng như cải thiện việc hiển thị các phần tử tương ứng là hết sức cần thiết.

Khi các hệ thống ứng dụng web ngày càng trở nên phức tạp, công nghệ xây dựng nên các ứng dụng web cũng như các công nghệ xây dựng ứng dụng di động ngày càng phát triển và đổi mới liên tục. Một giao diện người dùng được xây dựng lên không đơn thuần chỉ sử dụng HTML, CSS, Javascript. Các thư viện và framework mới liên tục cập nhật cũng như những các phiên bản mới ra đời là công cụ cần thiết giúp đơn giản hoá quy trình xây dựng phần mềm. Vì thế, bài toán sinh mã code tự động có thể được mở rộng ra không chỉ sinh ra mã code HTML, CSS đơn thuần mà nó còn có thể triển khai được trên nhiều nền tảng và ngôn ngữ khác nhau. Trong đề án này, em hướng tới chuyển đổi bản vẽ thiết kế thành mã code ReactJs – framework phát triển Front-end hàng đầu hiện nay, để xây dựng lên module client trong mô hình client-server. Đồng thời, bài toán hướng tới sinh ra mã code Reactnative – framework xây dựng ứng dụng đa nền tảng các ứng dụng di động.

Đối với một bài toán áp dụng kỹ thuật học máy, dữ liệu huấn luyện là vô cùng quan trọng. Khi dữ liệu càng lớn, độ tin cậy càng cao. Việc tái sử dụng dữ liệu dự đoán trở thành một phần của dữ liệu huấn luyện đúng cách một phần nào đó sẽ cải thiện được chất lượng của mô hình. Ngoài việc mở rộng ứng dụng của bài toán sketch2code, và mở rộng tập dữ liệu huấn luyện theo cách thủ công, em sẽ xây dựng ứng dụng quản lý kết quả sinh mã code, từ đó giúp quản lý và đánh giá lại chất lượng của mô hình trong việc dự đoán kết quả cũng như mở rộng dữ liệu huấn luyện. Trang quản lý được xây dựng đơn giản, dễ sử dụng và hỗ trợ tối đa trong việc quản lý dữ liệu.

1.3 Định hướng giải pháp

Để cho phép người dùng có thể truy cập và sử dụng chương trình, hệ thống sinh mã code được mở rộng dựa trên mô hình của hệ thống sketch2code bao gồm:



Hình 1.2 Kiến trúc hệ thống

- Trang web sinh mã code tự động được xây dựng dựa trên mô hình MVC sử dụng framework ASP.NET Core – là một framework open-source mới hỗ trợ đa nền tảng. ứng dụng này sẽ hỗ trợ người dùng trong việc tải dữ liệu dạng ảnh

lên và từ đó hệ thống sẽ sinh ra mã code HTML, Reactjs hay React Native tương ứng. Người dùng có thể báo cáo lỗi do hệ thống sinh ra từ đó nâng cao chất lượng dự đoán sau này.

- Trang web hỗ trợ quản lý kết quả và dữ liệu sinh mã code được xây dựng dựa trên mô hình client – server, kết hợp giữa thư viện Nodejs cho phía server và ReactJs cho phía client. Hệ thống này dành riêng cho quản trị viên. Quản trị viên có thể theo dõi và quản lý kết quả dự đoán và dữ liệu sinh mã code sau mỗi lần dự đoán.
- Hệ thống sử dụng dịch vụ Azure Cloud AI, dịch vụ này giúp xử lý hai bài toán học máy chính là nhận dạng vật thể và nhận dạng chữ viết tay. Khi các bài toán học máy này đã trở nên phổ biến và đạt được những thành tựu rất tốt, việc triển khai và cải thiện hiệu năng đòi hỏi có kiến thức chuyên sâu trong lĩnh vực học máy và rất nhiều nỗ lực. Dưới vai trò là một người phát triển phần mềm, em hướng tới áp dụng các kỹ thuật học máy vào giải quyết ứng dụng thực tế hơn là đi sâu vào nghiên cứu học máy. Vì thế, thay vì xây dựng lại mô hình huấn luyện, em chỉ dừng lại tìm hiểu các mô hình thuật toán và sẽ sử dụng trực tiếp các dịch vụ của Azure để xử lý hai bài toán trên.
- Hệ thống sử dụng nền tảng Azure Cloud cho phép lưu trữ các dữ liệu dạng thô (video, hình ảnh, file, ...) và tương tác trực tiếp với các dịch vụ Azure Cloud AI dễ dàng hơn.

1.4 Bố cục đồ án

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

Chương 2 trình bày về cơ sở lý thuyết cũng như các nghiên cứu về bài toán sinh mã code tự động áp dụng các kỹ thuật học máy nổi tiếng hiện nay, từ đó đưa ra hướng phát triển của bài toán.

Chương 3 trình bày về các công nghệ xây dựng ứng dụng web và ứng dụng di động cùng với đó là các dịch vụ bên thứ 3 mà hệ thống sử dụng.

Chương 4 trình bày về quy trình xây dựng và triển khai phần mềm.

Chương 5 trình bày về các đóng góp trong bài toán tự động sinh mã code.

Chương 6 là chương kết của đồ án, để thông qua đó rút ra được kinh nghiệm trong quá trình làm đồ án cũng như hướng phát triển tiếp theo

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ

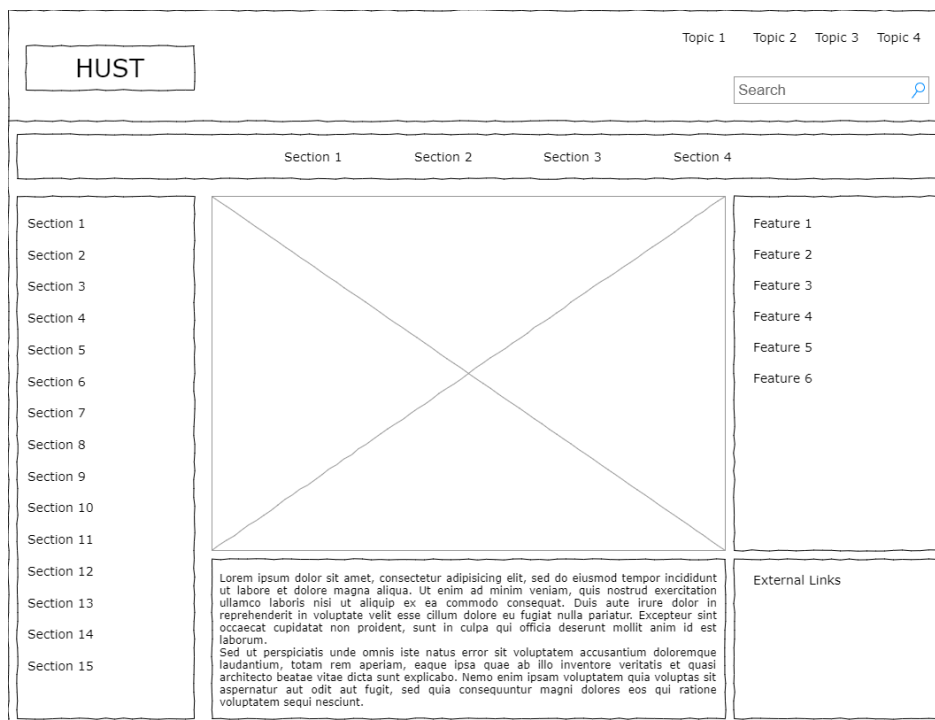
2.1 Bài toán Sinh mã code tự động

2.1.1 Quy trình thiết kế ứng giao diện người dùng

Quy trình thiết kế giao diện người dùng là một quy trình lặp dựa trên sự tương tác của người thiết kế, lập trình viên và khách hàng. Quy trình này bao gồm nhiều hoạt động, mỗi hoạt động đều giữ vai trò nhất định, các quy trình thiết kế thường mang tính chất cá nhân nhiều hơn là theo một tiêu chuẩn nhất định, mỗi cá nhân, công ty đều có một quy trình riêng cho mình. Tuy nhiên các quy trình đều được bắt đầu dưới dạng bản thiết kế kỹ thuật số hoặc phác thảo bằng tay. Các bản thiết kế này được gọi là wireframe - một tài liệu phác thảo cấu trúc cơ bản của ứng dụng, tài liệu thiết kế có độ trung thực thấp vì nó không xác định chi tiết cụ thể như màu sắc. Sau khi một wireframe được tạo ra, nó sẽ được xem xét và thêm chi tiết, tức là trở thành một bản thiết kế có độ tin cậy cao hơn. Sau đó, bản thiết kế có độ tin cậy cao này sẽ được chuyển cho các lập trình viên để xây dựng lên giao diện người dùng. Quá trình này tốn rất nhiều thời gian và liên quan tới nhiều bên.

Nếu một nhà thiết kế muốn tạo ra một trang web, họ phải tìm hiểu tất cả các chi tiết trước khi được lập trình viên triển khai, vì việc thử và thay đổi các ý tưởng trong bản vẽ thiết kế sẽ dễ dàng hơn rất nhiều khi đã chuyển thành mã code. Hơn nữa, việc chuyển đổi bản thiết kế thành mã code cũng tốn nhiều thời gian và nguồn lực. Từ đó ứng dụng sinh mã code tự động được đề xuất giúp giảm yếu tố về thời gian và chi phí bằng các chuyển đổi trực tiếp wireframe thành mã code trực tiếp.

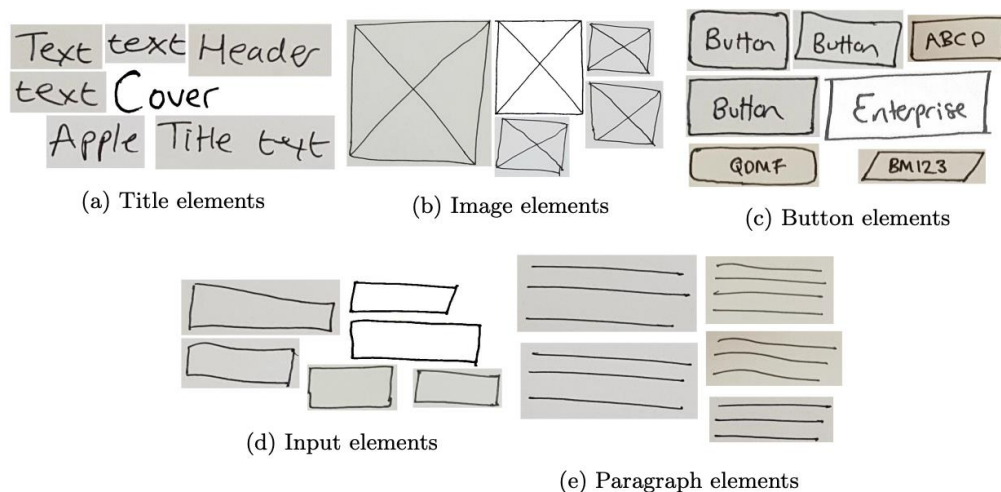
2.1.2 Wireframes



Hình 2.1 Bản vẽ thiết kế Wireframe

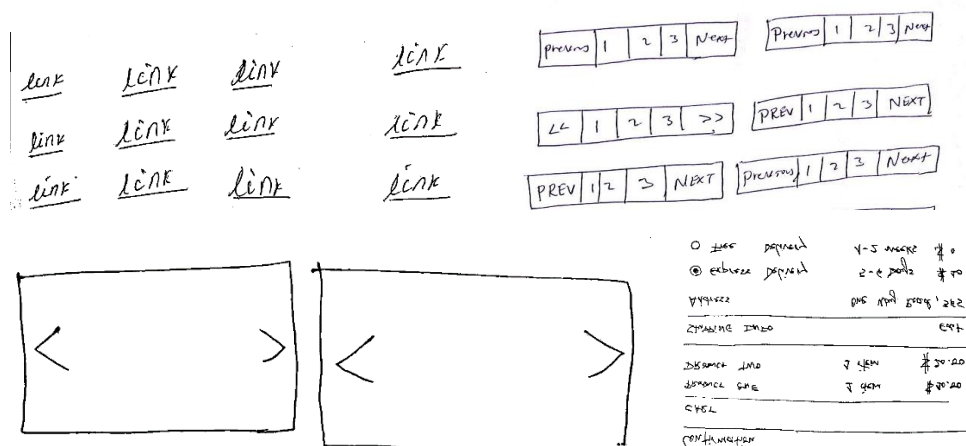
Wireframe hay còn được dịch ra là sơ đồ trang web hay bản thiết kế màn hình, là một hướng dẫn trực quan thể hiện khung cơ bản của một trang web hay của một

ứng dụng. Wireframe được tạo ra với mục đích sắp xếp các phần tử để hoàn thành tốt nhất một mục đích của thể nào đó. Wireframe mô tả bố cục của trang hoặc cách sắp xếp nội dung của một trang web, ứng dụng, bao gồm các yếu tố giao diện và hệ thống điều hướng và cách chúng hoạt động. Một wireframe được thiết kế thường bỏ qua các yếu tố như kiểu chữ, màu sắc, đồ họa vì trọng tâm chính nằm ở chức năng, hành vi và mức độ ưu tiên của nội dung.



Hình 2.2 Các phần tử xuất hiện trong wireframe

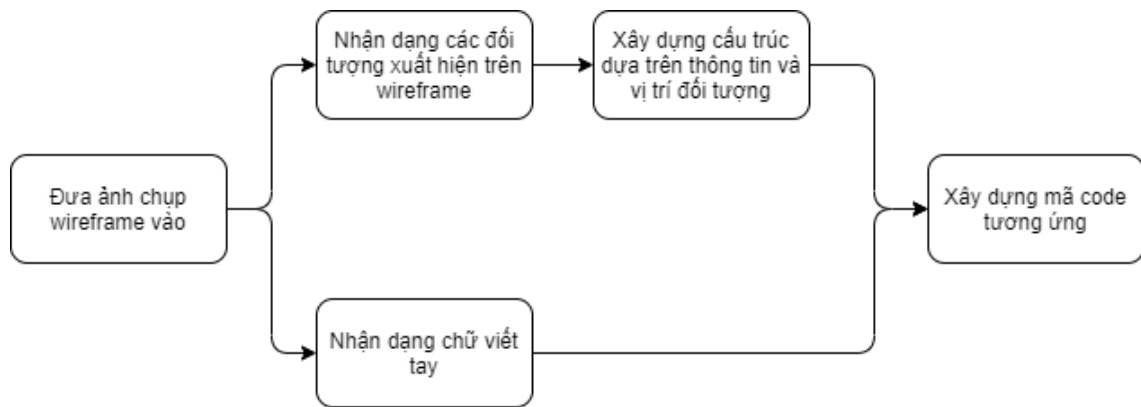
Mặc dù không có tiêu chuẩn thống nhất, wireframe thường sử dụng một bộ ký hiệu đơn giản có ý nghĩa thể hiện cho các phần tử xuất hiện trong màn hình. Trong đồ án này, chúng ta sẽ kế thừa và mở rộng số lượng các phần tử được sử dụng trong việc thiết kế ra wireframe, bao gồm: image, title, paragraph, button, input và mở rộng thêm các phần tử: pagination, link, dropdown, table, textarea, carousel, checkbox, radiobutton.



Hình 2.3 Phần tử mới như link, pagination, carousel, table trong wireframe

2.1.3 Quy trình sinh mã code tự động

Tuy có rất nhiều các phương pháp được áp dụng giải quyết bài toán sinh mã code tự động từ wireframe. Với bài toán tổng quát, quy trình giải quyết bài toán này như sau:



Hình 2.4 Quy trình sinh mã code tự động

Từ hình ảnh chụp wireframe mô tả bố cục của trang web, hệ thống sẽ sử dụng các kỹ thuật xử lý ảnh để xác định ra vị trí, kích thước, tọa độ của các phần tử xuất hiện trong wireframe. Đồng thời, hệ thống nhận dạng chữ viết tay sẽ đưa ra nội dung và vị trí của chữ viết xuất hiện trong hình ảnh thông. Khi tập các đối tượng xuất hiện trong bản vẽ được phát hiện, bao gồm loại, vị trí và kích thước, hệ thống chuyển qua giai đoạn phân tích và xây dựng cấu trúc cây phân cấp HTML. Mỗi bản thiết kế có tồn tại rất nhiều cách thể hiện cấu trúc cây phân cấp HTML khác nhau. Khi đã có cây cấu trúc phân cấp, hệ thống chuyển sang giai đoạn sinh mã code, tùy theo đầu ra của bài toán, hệ thống sẽ xây dựng lên bộ chuyển đổi code khác nhau. Trước đây, khi các kỹ thuật học máy còn chưa phổ biến và đủ mạnh mẽ, các kỹ thuật thị giác máy tính được coi là giải pháp tối ưu để giải quyết bài toán nhận dạng vật thể bao gồm các vấn đề như: phát hiện đường viền, phát hiện góc và phát hiện đường, ... Từ đó chúng ta có thể phân loại rõ được hình dạng thành các phần tử như button, text. Cách tiếp cận này gặp phải một số vấn đề nhất định:

- Khi sử dụng cách tiếp cận này, các ứng dụng tốn nhiều thời gian để mở rộng với các phần tử mới vì chúng dựa vào đánh giá của các lập trình viên để thiết kế các tính năng nhằm phân loại với tất cả các biểu diễn đã vẽ mà người dùng có thể nhập vào.
- Thông thường, các bản vẽ thô sơ chứa một vài sai sót nhỏ do vô tình của nhà thiết kế, việc dung lỗi về vị trí hay kích thước các phần tử sai khó có thể đạt được hiệu quả cao.

Cùng với sự phát triển vượt bậc của kỹ thuật học máy hiện tại, việc ứng dụng học máy vào giải quyết bài toán nhận dạng vật thể hay nhận dạng chữ viết là hoàn toàn khả thi và nâng cao chất lượng của hệ thống nói chung.

❖ Giới hạn trong sinh mã code cho hệ thống web và ứng dụng:

Trong hệ thống phần tử HTML, các phần tử khác nhau có thể biểu diễn cho một đối tượng giống nhau, ví dụ, phần tử image xuất hiện trong wireframe có thể được hiểu là , <svg> hay <video>. Việc hợp nhất các đối tượng tương tự nhau bằng thành một loại phần tử trong thể hiện wireframe giúp cho việc phân loại trở nên dễ dàng hơn. Tuy nhiên, điều này dẫn đến việc mô hình sinh code sẽ không thể hoàn toàn khớp với ý tưởng ban đầu của người thiết kế.

Các yếu tố liên quan tới styling bị bỏ qua trong bản thiết kế wireframe. Các yếu tố như: shadows, border, fonts, color, ... đều bị loại bỏ khỏi các phần tử thay vào đó là một giá trị đồng nhất.

Các hiệu ứng xuất hiện trong một trang web thực tế không thể biểu diễn được trong các bản thiết kế wireframe tĩnh được. Vì thế các yếu tố này cũng bị loại bỏ hoàn toàn.

2.1.4 Các nghiên cứu liên quan

a) Pix2code

Pix2code là một hệ thống có thể tự động tạo mã code dựa trên ảnh chụp màn hình GUI làm đầu vào. Hiện tại, mã được tạo cho ảnh chụp màn hình GUI có thể được lấy cho 3 nền tảng - android, iOS và dựa trên web (HTML).

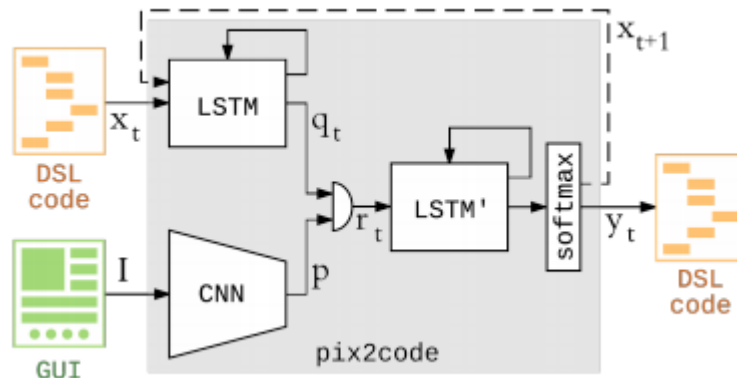
Tương tự như các nghiên cứu có từ trước, để có thể sinh được mã code tự động, hệ thống pix2code được chia thành 3 bài toán trọng tâm:

Tìm hiểu ảnh chụp màn hình GUI và suy ra các phần tử có mặt, nhãn, vị trí và tư thế của chúng

- Mô hình hoá ngôn ngữ - hiểu biết về mã code từ đó tạo ra các mẫu chính xác về mặt cú pháp và ngữ nghĩa
- Từ 2 bài toán trên, sinh ra mã code tương ứng

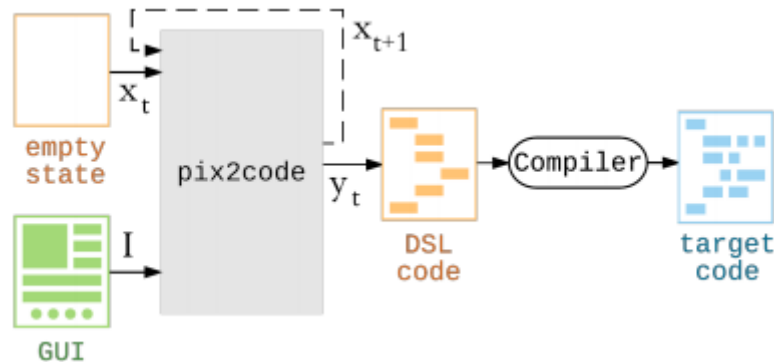
Tương ứng với mỗi bài toán, hệ thống pix2code là sự kết hợp của 3 mạng Nơ-ron

- Mạng CNN (Convolutional Neural Network) được sử dụng để tìm hiểu các tính năng và đối tượng trong hình ảnh đầu vào (GUI) và xuất ra một vector có độ dài cố định đã học tương ứng với hình ảnh đầu vào
- Mạng LSTM (Long-Short Term Memory) sử dụng một chuỗi token làm đầu vào, mỗi token là kết quả đầu cuối bằng ngôn ngữ đã chọn
- Mạng LSTM đã được sử dụng để học ngôn ngữ máy tính. Như chúng ta đã biết có nhiều loại ngôn ngữ máy tính, vì vậy để giữ cho mạng LSTM nói chung, nó đã được đào tạo trên một DSL mới được tạo ra (domain specific language).
- DSL được giới thiệu để giảm kích thước từ vựng, giảm số lượng mã thông báo được học bởi mạng LSTM. Do số token là cố định, đầu vào cho mô hình ngôn ngữ (LSTM) là một vector one-hot encoding.



Hình 2.5 Mô tả mạng huấn luyện từ bài báo pix2code

Sau quá trình huấn luyện, với mỗi bản vẽ thiết kế, hệ thống sẽ tạo ra mã code DSL tương ứng



Hình 2.6 Quy trình sinh mã code của pix2code

Trong quy trình trên, Empty state là một mảng gồm 48 vector thể với phần tử cuối cùng là token <START> và sau khi quá trình sinh kết thúc, nó sẽ được kết thúc với token <END>.

Mã DSL được tạo có thể được biên dịch thành mã đích như HTML (Web), Objective-C (iOS), Java (Android).

Hệ thống pix2code tuy đã sinh ra được mã code tự động, tuy nhiên, đầu vào của nó yêu cầu là ảnh chụp màn hình trang web, hay bài toán chỉ áp dụng cho các bản vẽ có độ tin cậy cao. Ngoài ra, hệ thống vẫn còn các hạn chế nhất định:

- Mã code được tạo bị hạn chế đối với một ngôn ngữ tập hợp con (DSL)
- Màu sắc, kích thước và văn bản bị bỏ qua trong quá trình sinh mã code

Hệ thống pix2code giới hạn ở 72 token, phụ thuộc vào độ dài bộ nhớ cố định của mạng LSTM, trong thực tế, đa phần các phần tử HTML đều bắt đầu bằng 1 thẻ mở và kết thúc bằng 1 thẻ đóng, ví dụ <p>Hello world </p>, tương ứng với mỗi phần tử này, sẽ cần có 2 token thể hiện cho <p> và </p>. Với tính chất lồng nhau của một cây HTML bao gồm hơn 100 loại phần tử, số lượng 72 token là quá ít so với thực tế.

Nếu mạng LSTM có thể triển khai với độ dài bộ nhớ cố định cao hơn đi chăng nữa thì vấn đề về dữ liệu huấn luyện sẽ nhân lên gấp nhiều lần.

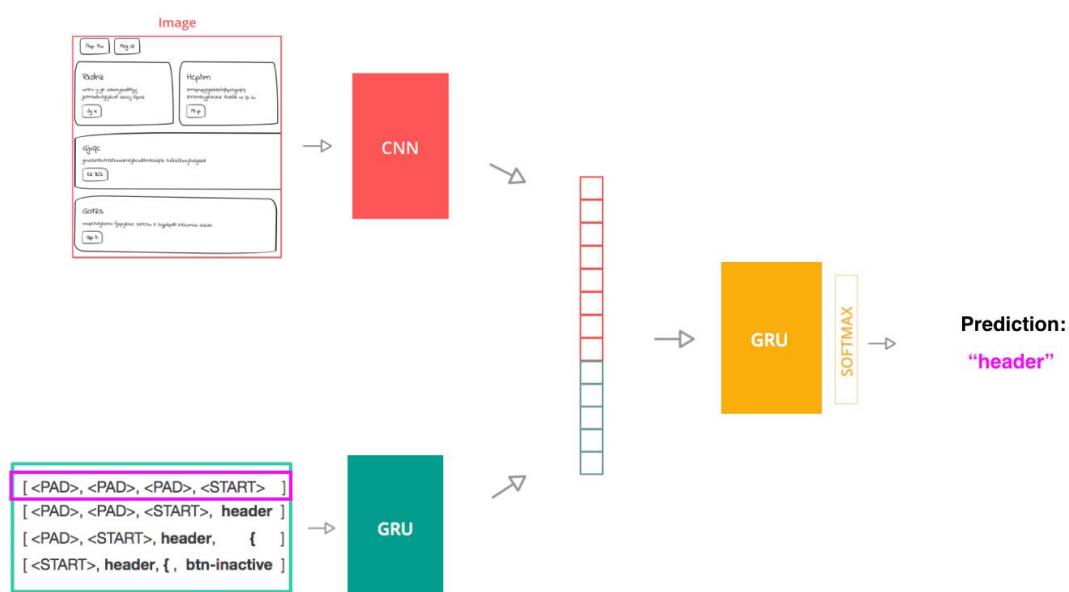
Mặc dù hệ thống pix2code đưa ra kết quả hứa hẹn trong tập dữ liệu tổng hợp của nó, tuy nhiên nó không khái quát hoá cho các trang web thực tế. Cả mã nguồn và tập dữ liệu hướng về thúc đẩy quá trình nghiên cứu trí tuệ nhân tạo vào ứng dụng thực tế hơn là hướng tới người dùng cuối. Vì thế việc mở rộng và triển khai hệ thống này để sinh mã code thực tế là không khả thi.

b) Ashnkumar/sketch-code

Bài toán SketchCode được Ashwin Kurmar lấy cảm hứng từ bài toán Dự đoán tiêu đề cho bức ảnh – mà một nhánh của thị giác máy tính trong lĩnh vực học máy. Việc sinh mã code từ bức ảnh chụp bản thiết kế wireframe cũng giống như việc tạo ra mô tả cho một bức ảnh của bài toán Dự đoán tiêu đề cho bức ảnh.

Mô hình này kế thừa bộ dữ liệu của mô hình pix2code, đồng nghĩa với việc, các wireframe được thể hiện đơn giản là tập các bộ phận tử Bootstrap đơn giản, nên nó cũng sẽ bị giới hạn về số lượng từ vựng nếu xem xét như bài toán Dự đoán tiêu đề cho bức ảnh.

Phát triển từ tập dữ liệu pix2code, tương ứng với mỗi bức ảnh là một chuỗi các token mô tả. Hình ảnh được duyệt theo chiều từ trên xuống dưới và từ trái sang phải. Với mỗi hàng ngang các phần tử, tương ứng là 1 mảng token mô tả các phần tử xuất hiện trong nó. Khi duyệt hết hàng, hàng tiếp theo sẽ được duyệt tương tự. Cho đến khi hệ thống gặp phải token <END> thì mới kết thúc.



Hình 2.7 Mô hình SketchCode sử dụng chuỗi token mô tả làm đầu vào

Như hình trên, ta có thể thấy, wireframe được chia thành 4 hàng, tương ứng mỗi hàng là một chuỗi các token mô tả các phần tử xuất hiện trong hàng đó. Mô hình sử dụng mạng CNN với hàm mất mát cross-entropy cost, đưa ra dự đoán chuỗi token xuất hiện trong wireframe.

Tương tự như bài toán pix2code, mô hình được huấn luyện dựa trên tập từ vựng gồm 16 loại phần tử nên nó không thể đưa ra dự đoán được các token xuất hiện bên ngoài tập dữ liệu huấn luyện.

Với cách sinh token theo hàng như trên, việc sinh mã code trở nên đơn giản hơn rất nhiều. Quy trình sinh mã code tương ứng là các vòng lặp duyệt token từ trên xuống dưới và từ trái qua phải.

Với các token thể hiện cho một phần tử, tác giả đã xây dựng nên các layout CSS khác nhau, từ đó làm đa dạng thêm đầu ra cho bài toán sinh mã code.



Hình 2.8 Mỗi bản vẽ có thể tạo ra nhiều kiểu thể hiện khác nhau

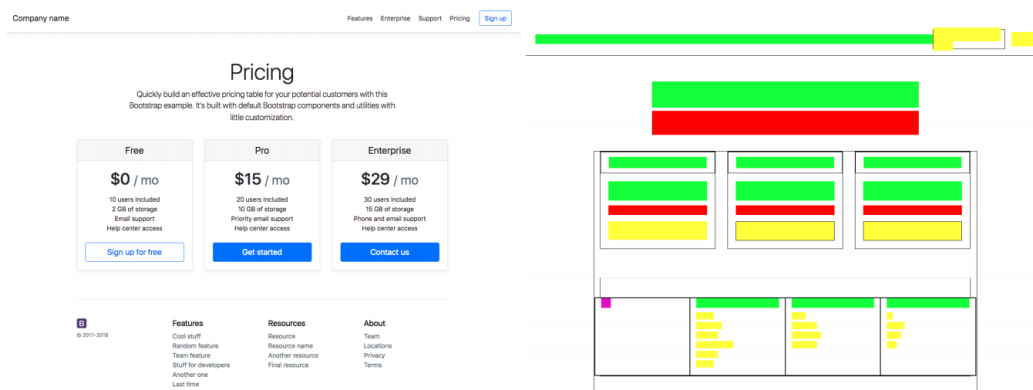
So với bài toán pix2code, tập ảnh đầu vào đã được tác giả chỉnh sửa cho giống với bản vẽ thiết kế wireframe hơn là ảnh chụp màn hình của một trang web. Đồng nghĩa, bài toán có thể đưa ra dự đoán từ các bản vẽ có độ tin cậy thấp.

Một bản vẽ wireframe thực tế khó có thể phân chia rạch ròi thành các hàng ngang như lý thuyết ban đầu, dẫn đến giới hạn trong việc dự đoán bài toán thực tế.

c) Microsoft/sketch2code

Hệ thống sketch2code được nghiên cứu và xây dựng với mục đích chuyển đổi tự động hình ảnh chụp wireframe với độ tin cậy thấp thành mã code HTML tương ứng.

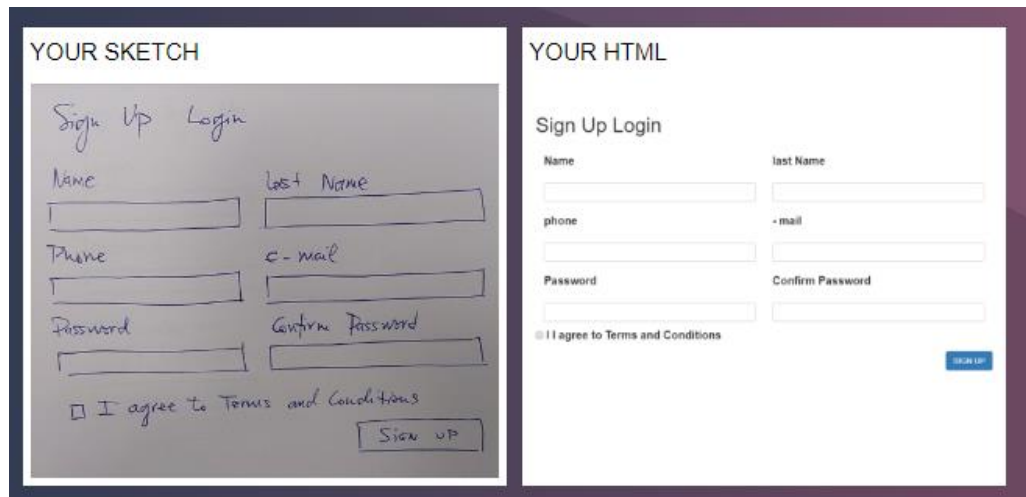
Sketch2code sử dụng mạng ANN (Artificial Neural Network) kết hợp với phân vùng hình ảnh (Image Segmentation) cho việc phát hiện, phân loại và chuẩn hoá các đối tượng xuất hiện trong bản thiết kế wireframe. Với việc áp dụng kỹ thuật phân vùng hình ảnh, cụ thể hơn là phân đoạn với tầng đối tượng, hệ thống có thể đưa ra chính xác vị trí của từng phần tử xuất hiện trong bản vẽ thiết kế, thay vì khung viền xung quanh phần tử.



Hình 2.9 Xác định vị trí các phần tử xuất hiện trong bản thiết kế

Do hệ thống sử dụng đầu vào là các bản vẽ có độ tin cậy thấp, vì thế khó có thể tránh khỏi các lỗi do người thiết kế tạo ra, các phần tử có thể chứa các lỗ hổng không liền mạch hay các góc cạnh không chính xác, ... Thông qua các kỹ thuật thị giác máy tính, các phần tử được chuẩn hoá các thuộc tính được áp dụng cho việc sinh mã code.

Từ danh sách các phần tử, hệ thống áp dụng thuật toán khởi tạo ra cây cấu trúc tương ứng. Tương tự như 2 mô hình trên, cây kế thừa DSL được khởi tạo ra và truyền vào hàm khởi tạo mã code để chuyển thành mã code HTML tương ứng.



Hình 2.10 Sketch2code chuyển đổi wireframe thành mã code HTML

So với 2 mô hình trước đó, sketch có thể đưa ra được cấu trúc cây DSL một cách linh hoạt hơn dựa trên vị trí của các phần tử, từ đó việc triển khai sinh mã code cho các bản thiết kế thực tế là hoàn toàn khả thi.

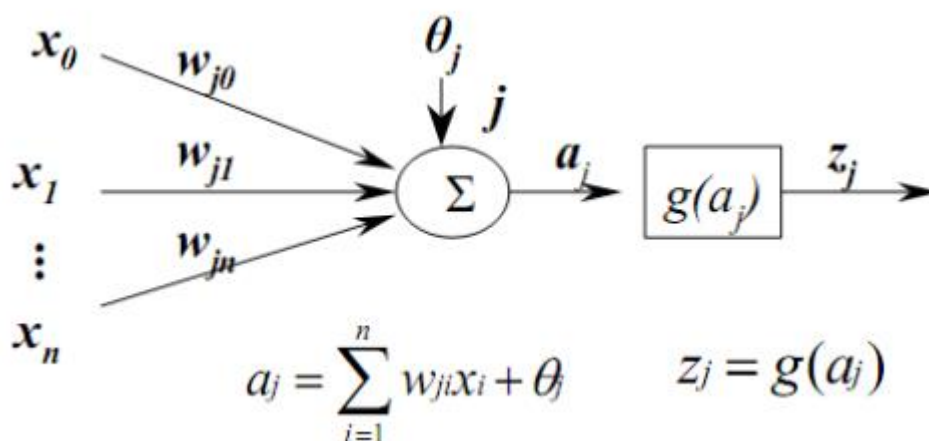
CHƯƠNG 3. CÁC CÔNG NGHỆ SỬ DỤNG

3.1 Các kĩ thuật học máy

3.1.1 Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo (ANNs) lấy cảm hứng từ mạng nơ-ron thần kinh là một cấu trúc khối gồm các đơn vị tính toán đơn giản được liên kết chặt chẽ với nhau trong đó các liên kết giữa các nơon quyết định chức năng của mạng.

Một mạng nơ-ron được cấu thành bởi các nơ-ron đơn lẻ được gọi là các perceptron. Các nơ-ron nó thực hiện nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng để tính tín hiệu ra sẽ được lan truyền sang các đơn vị khác.



Hình 3.1 Một đơn vị xử lý của mạng nơ-ron nhân tạo

Trong đó:

x_i : các đầu vào

w_{ji} : các trọng số tương ứng với các đầu vào

θ_j : độ lệch (bias)

a_j : đầu vào mạng (net-input)

z_j : đầu ra của nơon

$g(x)$: hàm chuyển (hàm kích hoạt).

Mỗi đơn vị j có thể có một hoặc nhiều đầu vào: $x_0, x_1, x_2, \dots, x_n$, nhưng chỉ có một đầu ra z_j . Một đầu vào tới một đơn vị có thể là dữ liệu từ bên ngoài mạng, hoặc đầu ra của một đơn vị khác, hoặc là đầu ra của chính nó.

Mỗi một đơn vị trong một mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với các đơn vị khác, sinh ra một giá trị gọi là net input. Hàm thực hiện nhiệm vụ này gọi là hàm kết hợp (combination function), được định nghĩa bởi một luật lan truyền cụ thể.

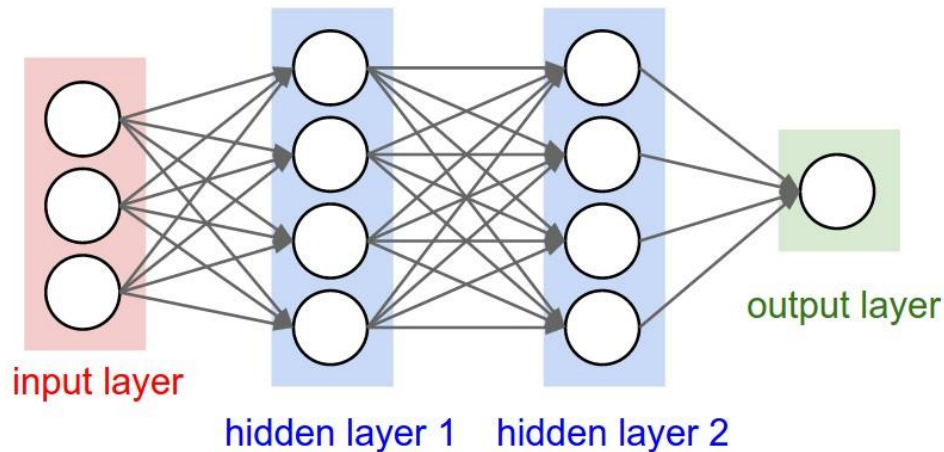
Phần lớn các đơn vị trong mạng nơ-ron chuyển net input bằng cách sử dụng một hàm vô hướng gọi là hàm kích hoạt, kết quả của hàm này là một giá trị gọi là mức độ kích hoạt của đơn vị (unit's activation).

3.1.1.1. Mạng truyền thẳng (Feed-forward neural network)

Dòng dữ liệu từ đơn vị đầu vào đến đơn vị đầu ra chỉ được truyền thẳng. Việc xử lý dữ liệu có thể mở rộng ra nhiều lớp, nhưng không có các liên kết phản hồi. Nghĩa là, các liên kết mở rộng từ các đơn vị đầu ra tới các đơn vị đầu vào trong cùng một lớp hay các lớp trước đó là không cho phép.

3.1.1.2. Mạng truyền thẳng đa tầng (Multilayer perceptron networks)

Mạng nơ-ron là sự kết hợp của các tầng perceptron hay còn được gọi là mạng truyền thẳng đa tầng (MLP)



Hình 3.2 Mạng nơron truyền thẳng nhiều lớp

Một mạng nơ-ron sẽ có 3 kiểu tầng:

- Tầng vào (input layer): Là tầng bên trái cùng của mạng thể hiện cho các đầu vào của mạng.
- Tầng ra (output layer): Là tầng bên phải cùng của mạng thể hiện cho các đầu ra của mạng.
- Tầng ẩn (hidden layer): Là tầng nằm giữa tầng vào và tầng ra thể hiện cho việc suy luận logic của mạng.

Các nơ-ron ở các tầng thường được liên kết đôi một với nhau tạo thành mạng kết nối đầy đủ (full-connected network).

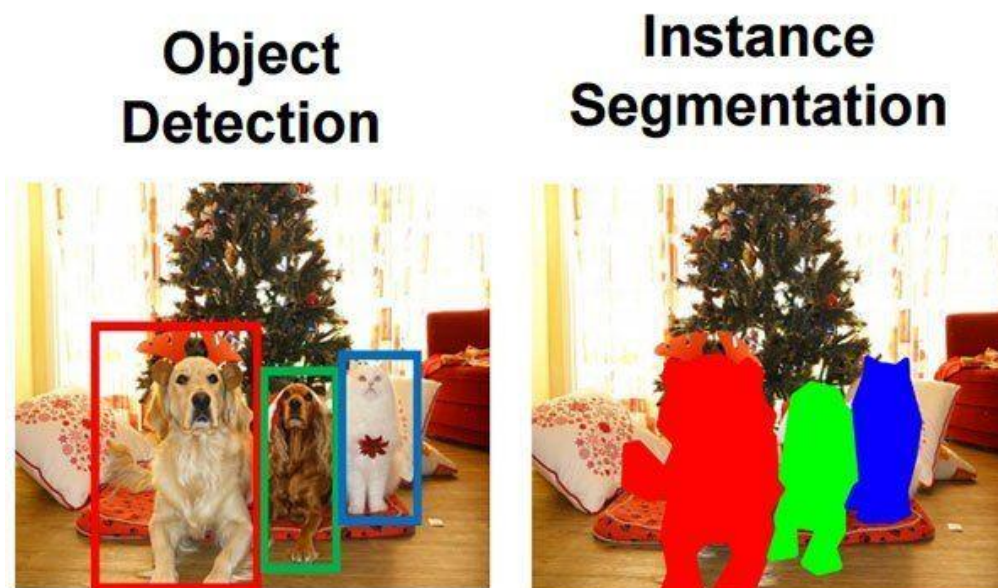
Khi một MLP đang được huấn luyện, các cặp dữ liệu đầu vào và đầu ra được đưa vào MLP sử dụng thuật toán lan truyền ngược (backpropagation) cập nhật trọng số giữa các nơ-ron đang cố gắng mô hình hóa phép biến đổi tính toán biến dữ liệu đầu vào thành dữ liệu đầu ra. Thường thì việc chuyển đổi càng khó, càng cần nhiều dữ liệu để học và mạng càng phức tạp. Khi MLP đạt đến hiệu suất có thể chấp nhận được, nó có thể được chạy bằng cách cung cấp dữ liệu cho lớp đầu vào và MLP tạo ra đầu ra.

Trong MLP, độ phức tạp của mô hình có thể được coi là số lượng tầng ẩn và số lượng nơ-ron trong các tầng ẩn.

Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu training. Về cơ bản, overfitting xảy ra khi mô hình quá phức tạp để mô phỏng training data. Điều này đặc biệt xảy ra khi lượng dữ liệu training quá nhỏ trong khi độ phức tạp của mô hình quá cao.

3.1.2 Bài toán Phân vùng hình ảnh

Hiện nay trong nghiên cứu về Computer Vision có rất nhiều khía cạnh cần để tìm hiểu như: Object Detection, Recognize Face, Semantic Segment, ... Bài toán về semantic segment hay còn gọi là bài toán về phân vùng ảnh. Nó khác với các bài toán về object detection là bài toán này có thể xác định rõ được vùng ảnh của đối tượng nào đó, còn object detection nó chỉ xác định được bounding box của đối tượng.



Hình 3.3 So sánh nhận dạng đối tượng và phân vùng đối tượng

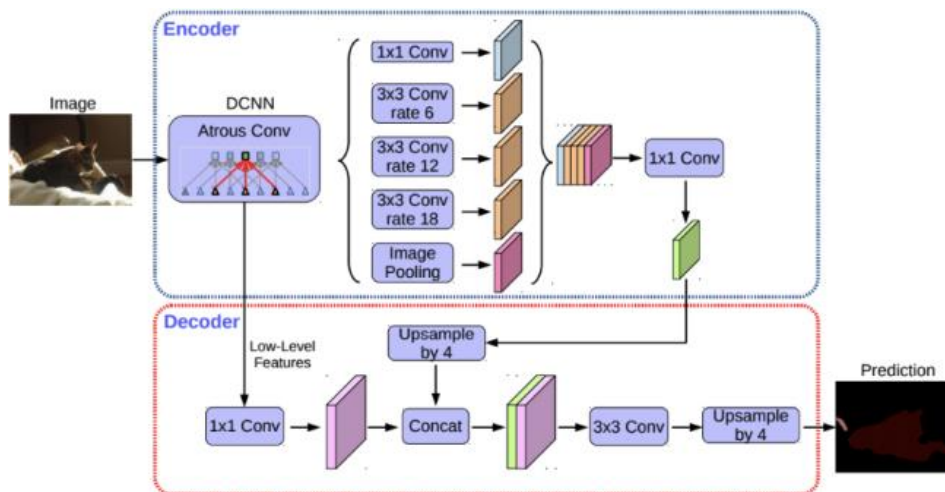
Phân vùng hình ảnh giúp làm đơn giản hóa và / hoặc thay đổi hình ảnh biểu diễn thành một thứ gì đó có ý nghĩa hơn. Nó liên quan đến việc gắn nhãn cho từng pixel sao cho các pixel có chung đặc điểm được gắn nhãn cùng nhau. Bên cạnh phương pháp phân vùng hình ảnh, có một số phương pháp giải quyết vấn đề tương tự như phương pháp phân cụm (clustering), Texton Forests, hoặc phát hiện cạnh (edge detection), ... nhưng phương pháp này được đánh giá đưa ra kết quả tốt nhất.

Bài toán phân vùng hình ảnh có thể được huấn luyện trên mô hình ANNs với đầu vào là các cặp dữ liệu: hình ảnh tương ứng với phân vùng được gắn nhãn của hình ảnh.

Hiện nay, có rất nhiều các mô hình mạng xử lý bài toán phân vùng hình ảnh, nếu xét về hiệu năng, DeepLab3+ được coi là mô hình hiệu quả nhất. DeepLab là một mô hình phân vùng ngữ nghĩa hiện đại được Google thiết kế và có nguồn mở vào năm 2016. Kể từ đó, nhiều cải tiến đã được thực hiện cho mô hình này, bao gồm DeepLab V2, DeepLab V3 và DeepLab V3 + mới nhất.

Mô hình DeepLab bao gồm hai giai đoạn:

- Giai đoạn mã hóa: Sử dụng mạng CNN trích xuất thông tin cần thiết từ hình ảnh.
- Giai đoạn giải mã: Thông tin trích xuất trong giai đoạn mã hóa được sử dụng để đưa ra phân vùng ngữ nghĩa của các đối tượng.



Hình 3.4 Kiến trúc mô hình DeepLab

3.2 Các công nghệ xây dựng ứng dụng website và di động

3.2.1 ASP.NET Core

ASP.NET Core là một web framework mã nguồn mở được tối ưu hóa cho cloud để phát triển các ứng dụng web chạy trên nhiều nền tảng như Windows, Linux và Mac. Hiện tại, nó bao gồm MVC framework được kết hợp các tính năng của MVC và Web API thành một web framework duy nhất. Các ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên .NET Framework hoàn chỉnh. Nó đã được thiết kế để cung cấp một framework tối ưu cho các ứng dụng để triển khai tới cloud hoặc chạy on-premises, những modular với các thành phần tối thiểu, do đó chúng ta giữ được tính linh hoạt trong quá trình xây dựng các giải pháp. Chúng ta có thể phát triển và chạy các ứng dụng đa nền tảng từ ASP.NET Core trên Windows, Mac và Linux.

Đã từ rất lâu thì lập trình web luôn là sân chơi của PHP tuy nhiên với sự xuất hiện ASP.NET Core đã chứng minh được sức mạnh của nó đủ để có thể cạnh tranh với ông lớn PHP. Với những lợi thế vượt trội như:

- Các ứng dụng Web mà chúng ta tạo ra có thể testing theo mô hình MVC (Model- View Controller).
- Razor cung cấp cho ta ngôn ngữ hiệu quả để tạo Views
- Tag Helper cho code server side tham gia vào việc tạo và render phần tử HTML.
- Tự động ánh xạ dữ liệu từ HTTP request tới tham số của method action trên Model Binding.
- Model Validation tự động thực hiện validate và serve

Chúng ta sẽ sử dụng framework này để xây dựng hệ thống chuyển đổi tự động mã code.

3.2.2 Razor

Razor là tên gọi của view engine sử dụng bởi ASP.NET Core, đồng thời cũng là tên gọi của loại ngôn ngữ đánh dấu sử dụng trong view engine này, là cú pháp kết

hợp C# và HTML để sinh ra HTML động trong ứng dụng ASP.NET Core. Razor được sử dụng trong Razor Pages, MVC và Blazor.

Việc sử dụng Razor engine nhằm kết hợp ngôn ngữ lập trình với HTML để dễ dàng sinh ra HTML. Nói cách khác, view engine hoạt động như một chương trình dịch (compiler hoặc interpreter) để chuyển đổi loại ngôn ngữ lai này thành HTML.

Razor có cú pháp đơn giản và mạnh mẽ giúp cho quá trình viết mã nguồn trở nên đơn giản hơn rất nhiều.

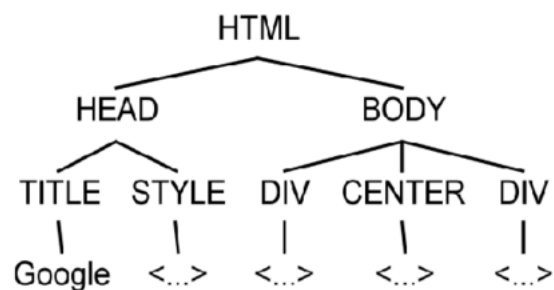
Chúng ta sẽ sử dụng cú pháp của Razor trong quá trình xây dựng giao diện cho hệ thống cũng như sinh mã code tự động.

3.2.3 Cấu trúc của một website

Trang web được cấu tạo theo cấu trúc với các thẻ phần tử HTML và được tạo style bằng CSS. Cấu trúc của một trang web bao gồm các loại phần tử, ví dụ: bảng, đoạn văn, nút, và cách chúng được định vị so với nhau. Style bao gồm màu sắc, phông chữ và đường viền, Tuy nhiên, HTML cũng có thể sử dụng để thể hiện style của các phần tử trong website, ví dụ như thẻ `` và CSS có thể sử dụng để định nghĩa một phần cấu trúc của website ví dụ như `display`, `position`.

HTML được xây dựng như một cây các đối tượng được gọi là Mô hình đối tượng tài liệu (DOM). Các nhánh của cây này đại diện cho các vùng chứa, ví dụ về chúng là `<div>`, `<footer>`, `<header>`, lá của cây là các phần tử chứa nội dung, ví dụ: ``, `<p>`, `<button>`.

```
<html>
  <head>
    <title>Google</title>
    <style>...</style>
  </head>
  <body>
    <div>...</div>
    <center>...</center>
    <div>...</div>
  </body>
</html>
```



Hình 3.5 Cây cấu trúc HTML

JavaScript là một ngôn ngữ kịch bản được chạy trên một trang web sau khi một trang web được hiển thị và có thể thao tác với DOM từ đó có thể thay đổi cấu trúc của trang web.

Do sự linh hoạt trong việc thể hiện cấu trúc và style của HTML, CSS và Javascript, nhiều trang web có cấu trúc khác nhau vẫn có thể hiển thị ra kết quả giống nhau. Điều này gây khó khăn trong việc đánh giá cấu trúc của cây HTML.

3.2.4 Reactjs

ReactJS là một thư viện JavaScript mã nguồn mở được Facebook thiết kế để tạo các ứng dụng web phong phú và hấp dẫn một cách nhanh chóng và hiệu quả với mã hóa tối thiểu. Mục tiêu cốt lõi của ReactJS là cung cấp hiệu suất hiển thị tốt nhất có thể. Sức mạnh của nó đến từ việc tập trung vào các thành phần riêng lẻ. Thay vì làm việc trên toàn bộ ứng dụng web, ReactJS cho phép nhà phát triển chia nhỏ giao diện người dùng phức tạp thành các thành phần đơn giản hơn.

3.2.5 Thiết kế nguyên tử (Atomic design)

Thiết kế nguyên tử (Atomic design) là một phương pháp luận lấy cảm hứng từ hóa học. Cũng giống như tất cả vật chất được tạo ra từ các nguyên tử kết hợp với nhau để tạo thành các phân tử, từ đó tạo nên các sinh vật phức tạp hơn, Thiết kế nguyên tử giúp xây dựng các hệ thống thiết kế nhất quán, chắc chắn và có thể tái sử dụng. Có 5 cấp độ thành phần trong phương pháp thiết kế này, các cấp độ thành phần sau là sự tổng hợp từ các thành phần có cấp độ trước đó.



Hình 3.6 Các thành phần trong Atomic design

3.2.6 Bootstrap

Bootstrap là một trong những framework front-end, mã nguồn mở, phổ biến nhất trên Web. Kể từ khi phát hành chính thức vào năm 2011, nó đã trải qua một số thay đổi và hiện là một trong những framework đáp ứng tốt và ổn định nhất hiện có. Nó được các nhà phát triển web ở mọi cấp độ yêu thích vì nó mang lại cho họ khả năng xây dựng một thiết kế trang web hấp dẫn, với chi phí bỏ ra ít. Một nhà phát triển mới làm quen với chỉ một số kiến thức cơ bản về HTML và một chút CSS có thể dễ dàng bắt đầu với Bootstrap.

3.2.7 React Native

React Native là một framework di động dựa trên JavaScript phổ biến cho phép bạn tạo các ứng dụng di động native cho iOS và Android. Framework cho phép bạn tạo ứng dụng cho các nền tảng khác nhau bằng cách sử dụng cùng một cơ sở mã nguồn.

React Native được Facebook phát hành lần đầu tiên dưới dạng một dự án mã nguồn mở vào năm 2015. Chỉ trong vài năm, nó đã trở thành một trong những giải pháp hàng đầu được sử dụng để phát triển di động.

Có một số lý do đằng sau thành công của React Native:

- Bằng cách sử dụng React Native, chúng ta có thể tạo mã chỉ một lần và sử dụng nó để xây dựng lên cả ứng dụng iOS và Android của họ. Điều này dẫn đến tiết kiệm thời gian và nguồn lực rất lớn.

- React Native được xây dựng dựa trên React - một thư viện JavaScript, vốn đã cực kỳ phổ biến khi framework di động được phát hành.
- Cho các nhà phát triển giao diện người dùng, những người trước đây chỉ có thể làm việc với các công nghệ dựa trên web, để tạo ra các ứng dụng mạnh mẽ, sẵn sàng sản xuất cho các nền tảng di động.

❖ **React và React Native**

Nói một cách đơn giản nhất, React Native không phải là một phiên bản React ‘mới hơn’, mặc dù React Native có sử dụng nó.

React (còn được gọi là ReactJS) là một thư viện JavaScript được sử dụng để xây dựng giao diện người dùng của một trang web. Tương tự như React Native, nó cũng được phát triển bởi nhóm kỹ sư của Facebook.

Trong khi đó, React Native - được cung cấp bởi React - cho phép các nhà phát triển sử dụng một tập hợp các thành phần giao diện người dùng để nhanh chóng biên dịch và khởi chạy các ứng dụng iOS và Android.

Cả React và React Native đều sử dụng hỗn hợp JavaScript và một ngôn ngữ đánh dấu đặc biệt, JSX. Tuy nhiên, cú pháp được sử dụng để hiển thị các phần tử trong các thành phần JSX khác nhau giữa React và React Native. Ngoài ra, React sử dụng HTML và CSS, trong khi React Native cho phép sử dụng các phần tử giao diện người dùng di động gốc.

3.3 Microsoft Azure: Dịch vụ tính toán đám mây

3.3.1 Thị giác máy tính

Dịch vụ Thị giác máy tính của Azure cung cấp cho chúng ta quyền truy cập vào các thuật toán nâng cao xử lý hình ảnh và trả lại thông tin dựa trên các đặc điểm hình ảnh cần quan tâm.

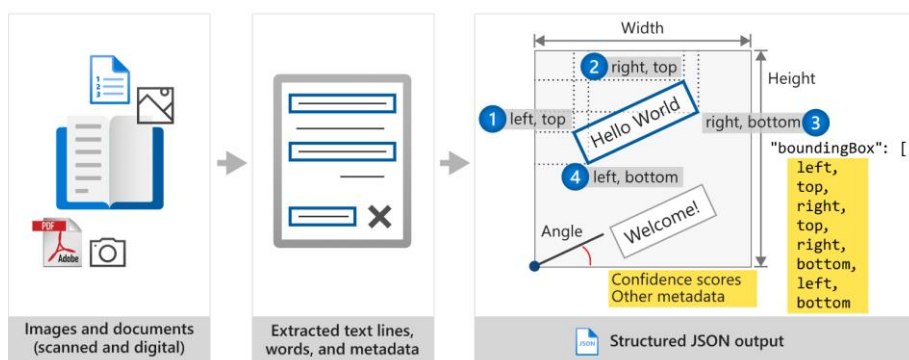
Chúng ta có thể tạo các ứng dụng Computer Vision thông qua SDK từ phía client hoặc bằng cách gọi trực tiếp API REST.

- Nhận dạng vật thể

Với dịch vụ Custom Vision cognitive, Azure cung cấp cho chúng ta giải pháp về thị giác máy tính và không yêu cầu chuyên môn về học sâu, đặc biệt là bài toán nhận dạng vật thể.

- Nhận dạng ký tự quang học (Optical Character Recognition)

Nhận dạng ký tự quang học (OCR) là công nghệ trích xuất văn bản, chữ viết tay, đánh máy, ... trong hình ảnh.



Hình 3.7 Nhận dạng kí tự quang học

Azure cung cấp giải pháp Thị giác máy tính có hỗ trợ công nghệ OCR mới nhất hiện nay bao gồm trích xuất văn bản in (bằng một số ngôn ngữ), văn bản viết tay (chỉ bằng tiếng Anh), chữ số và ký hiệu tiền tệ từ hình ảnh và tài liệu PDF nhiều trang. Nó được tối ưu hóa để trích xuất văn bản từ các hình ảnh nhiều văn bản và các tài liệu PDF nhiều trang với nhiều ngôn ngữ khác nhau, đồng thời hỗ trợ phát hiện cả văn bản in và viết tay trong cùng một hình ảnh hoặc tài liệu.

3.3.2 Blob storage

Azure Blob storage là giải pháp lưu trữ đối tượng của Microsoft trên nền tảng đám mây. Bộ lưu trữ Blob được tối ưu hóa để lưu trữ một lượng lớn dữ liệu phi cấu trúc - dữ liệu không tuân theo mô hình hoặc định nghĩa dữ liệu cụ thể, chẳng hạn như dữ liệu văn bản hoặc dữ liệu nhị phân.

Bộ nhớ Blob được thiết kế cho:

- Lưu trữ file cho truy cập phân tán
- Stream video và audio
- Ghi vào file log
- Lưu trữ dữ liệu để sao lưu và khôi phục
- Lưu trữ dữ liệu để phân tích bằng các dịch vụ khác của Azure

Người dùng hoặc ứng dụng client có thể truy cập các đối tượng trong bộ lưu trữ Blob qua HTTP/HTTPS. Các đối tượng trong bộ nhớ Blob có thể truy cập thông qua API Azure Storage REST, Azure PowerShell, Azure CLI hoặc thư viện Azure Storage.

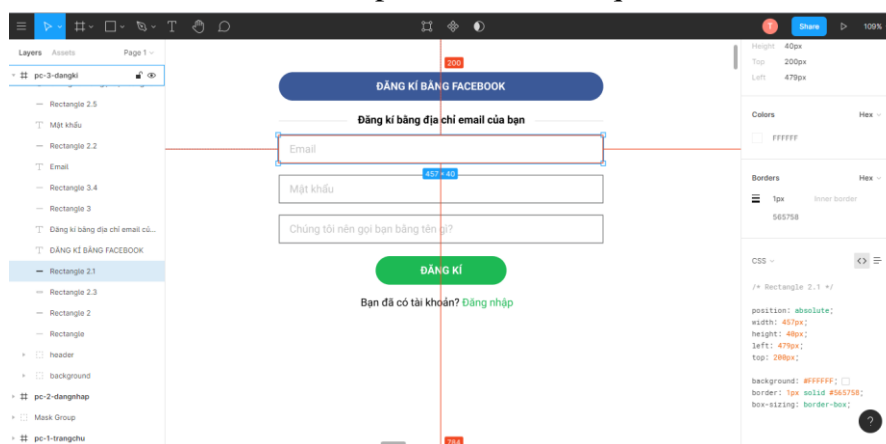
CHƯƠNG 4. PHÂN TÍCH VÀ TRIỂN KHAI ỨNG DỤNG

4.1 Phân tích yêu cầu phần mềm

4.1.1 Khảo sát hiện trạng

Hiện nay có rất nhiều các nghiên cứu, ứng dụng và các trang web hỗ trợ cho việc thiết kế và chuyển đổi bản thiết kế thành mã code. Chúng ta có thể chia các ứng dụng này hai lĩnh vực tách biệt bao gồm Hệ thống hỗ trợ thiết kế giao diện và Hệ thống chuyển đổi bản vẽ thiết kế thành mã code.

Với Hệ thống hỗ trợ thiết kế giao diện được biết đến như là các công cụ thiết kế dạng vector như Sketch và Figma. Từ cấu trúc dạng vector mà người thiết kế tạo ra, hệ thống sẽ đưa ra mã code cho từng thành phần nhỏ xuất hiện trong bản vẽ, tuy nhiên cấu trúc và vị trí của các phần tử sẽ bị bỏ qua.



Hình 4.1 Figma hỗ trợ sinh mã code CSS cho các phần tử

Với hệ thống chuyển đổi mã code tự động từ bản vẽ thiết kế, các bản vẽ thiết kế sẽ được chụp lại làm đầu vào cho hệ thống sinh mã code. Từ bản vẽ thiết kế, hệ thống phân tích và đưa ra mã code tương ứng cho thiết kế ban đầu. Các hệ thống chuyển đổi mã code có thể chia thành 2 lĩnh vực nhỏ hơn phụ thuộc vào đầu vào của hệ thống đó: Hệ thống sinh mã code dựa trên bản vẽ có độ tin cậy cao (Pix2code, Sketch2React, ...) và Hệ thống sinh mã code dựa trên bản vẽ có độ tin cậy thấp (Sketch2code, ...).

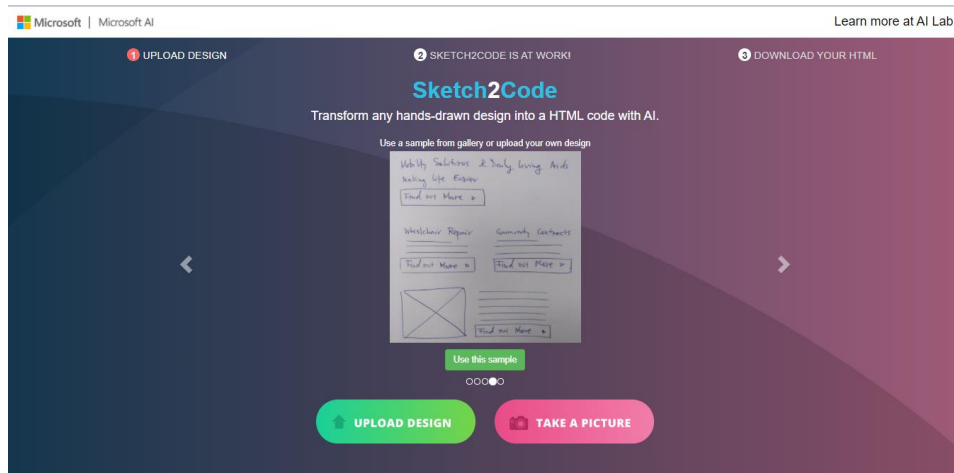
- Hệ thống Sketch2React là một công cụ hay một framework cho phép xây dựng các nguyên mẫu React và HTML trực tiếp trên ứng dụng Sketch giúp trích xuất ra mã code HTML hay React tương ứng.

Ưu điểm	Nhược điểm
<ul style="list-style-type: none">• Có giao diện đẹp• Hỗ trợ sinh mã code cho nhiều nền tảng• Cung cấp dạng plugin cho ứng dụng Sketch	Chỉ sinh mã code dựa trên bản thiết kế được người thiết kế tạo trên hệ thống Sketch, thay vì ảnh chụp bản thiết kế

- Hệ thống Pix2code là công cụ chuyển đổi ảnh chụp bản thiết kế có độ tin cậy cao thành mã code HTML, Object C, Java nhằm mục đích xây dựng các ứng dụng trên các nền tảng khác nhau.

Ưu điểm	Nhược điểm
<ul style="list-style-type: none"> Chỉ dừng lại ở mức độ bài nghiên cứu, chưa triển khai thành ứng dụng thực tế Hỗ trợ sinh mã code cho nhiều nền tảng khác nhau Sinh mã code dựa trên ảnh chụp bản vẽ có độ tin cậy cao 	<ul style="list-style-type: none"> Số lượng phần tử có thể sinh ra còn hạn chế Không hỗ trợ các bản vẽ thiết kế có độ tin cậy thấp Mã code sinh ra chưa phù hợp với các mô hình web thực tế

- Hệ thống Sketch2code cung cấp giải pháp chuyển đổi ảnh chụp bản thiết kế có độ tin cậy thấp thành mã code HTML tương ứng cho phép xây dựng ứng dụng trên nền tảng web.



Hình 4.2 Hệ thống Sketch2code

Ưu điểm	Nhược điểm
<ul style="list-style-type: none"> Có giao diện đẹp Sinh mã code dựa trên ảnh chụp bản vẽ có độ tin cậy thấp 	<ul style="list-style-type: none"> Số lượng phần tử có thể sinh ra còn hạn chế Chỉ hỗ trợ sinh ra mã code HTML thuần Chưa có hệ thống quản lý chung

Từ việc khảo sát các hệ thống sinh mã code trên, em đưa ra các tính năng cần thiết của hệ thống như sau:

- Hệ thống sinh mã code tự động**

Hệ thống được xây dựng và mở rộng dựa trên hệ thống Sketch2code hiện tại.

Tính năng dành cho người dùng:

- Tải và chụp hình ảnh bản thiết kế lên hệ thống phục vụ cho việc sinh mã code
- Sinh mã code HTML từ bản thiết kế và cho phép người dùng tải về

- Sinh mã code ReactJs và đóng gói mã nguồn kết hợp với công cụ create-react-app cho phép người dùng tải về cài đặt
- Sinh mã code Reactnative và đóng gói mã nguồn cho phép người dùng tải về cài đặt
- So sánh bản vẽ thiết kế ban đầu với ảnh chụp hình ảnh giao diện được xây dựng dựa trên mã code sinh ra
- Xem chi tiết kết quả dự đoán của mô hình, chỉnh sửa kết quả dự đoán sát với ý định thiết kế nhất từ đó sinh ra mã code tương ứng, kết quả này sẽ được hệ thống lưu lại và cập nhật lên mô hình huấn luyện.
- **Hệ thống quản lý kết quả sinh mã code**

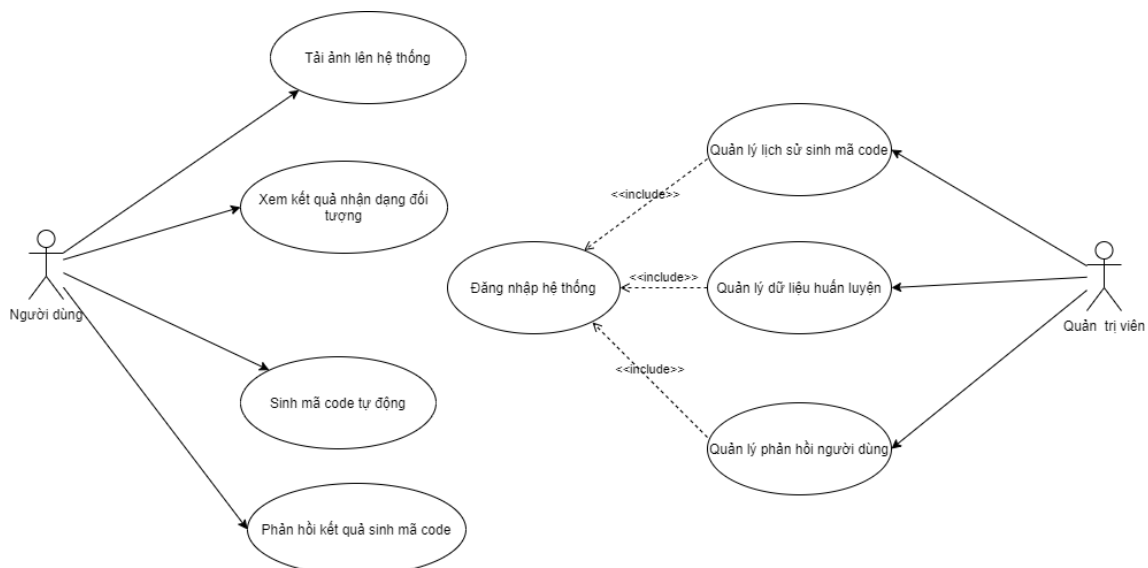
Hệ thống được xây dựng dành riêng cho quản trị viên hệ thống

Tính năng dành cho quản trị viên:

- Hệ thống chỉ dành riêng cho quản trị viên nên yêu cầu xác thực thông qua tài khoản Google.
- Quản lý lịch sử sinh mã code tự động: Xem, chỉnh sửa, cập nhật dữ liệu kết quả dự đoán.
- Xác thực các chỉnh sửa kết quả sinh mã code của người dùng: Chỉnh sửa, xoá, cập nhật lại thông tin người dùng gửi lên từ đó chuyển dữ liệu vào mô hình để cập nhật và huấn luyện lại.
- Quản lý dữ liệu huấn luyện.

4.1.2 Tổng quan chức năng

4.1.2.1. Biểu đồ usecase tổng quan



Hình 4.3 Biểu đồ use case tổng quan

Mô tả biểu đồ use case tổng quan:

- Tác nhân: Người dùng và quản trị viên
- Vai trò:
 - Người dùng sẽ sử dụng dịch vụ và các chức năng mà hệ thống cung cấp

- Quản trị viên sẽ quản lý dữ liệu liên quan tới quá trình nhận dạng và sinh mã code tự động

Với tác nhân người dùng: Người dùng không cần đăng nhập mà vẫn có thể sử dụng tất cả các chức năng của hệ thống sinh mã code.

Tên use case	Mô tả tóm tắt
Tải ảnh lên hệ thống	<ul style="list-style-type: none"> + Người dùng sẽ tải hình ảnh của bản thiết kế vẽ tay từ máy tính các nhân hoặc chụp ảnh trực tiếp từ webcam hoặc cũng có thể lựa chọn những bức ảnh ví dụ của hệ thống + Sau khi ảnh đã được tải lên, hệ thống sẽ tự động đẩy vào mô hình để tiến hành giai đoạn nhận dạng đối tượng phần tử HTML
Xem kết quả nhận dạng đối tượng	<ul style="list-style-type: none"> + Sau khi upload ảnh lên, hệ thống sẽ phân tích và đưa ra dự đoán về các phần tử xuất hiện trong bản thiết kế + Người dùng có thể xem các thông số mô hình đưa ra về các đối tượng xuất hiện trong bản thiết kế
Sinh mã code tự động	<ul style="list-style-type: none"> + Người dùng xem kết quả sinh mã code và tải mã nguồn đóng gói mà hệ thống sinh ra dựa trên tùy chọn của người dùng
Phản hồi kết quả sinh mã code	<ul style="list-style-type: none"> + Người dùng chỉnh sửa kết quả nhận dạng sai hoặc không đúng với thiết kế ban đầu . + Người dùng lựa chọn các đối tượng hợp lệ và loại bỏ các đối tượng nhận dạng sai với thiết kế. + Các phản hồi này sẽ được chuyển cho quản trị viên để tiến hành chuẩn hoá dữ liệu.

Bảng 4.1 Mô tả use case tổng quan với tác nhân người dùng

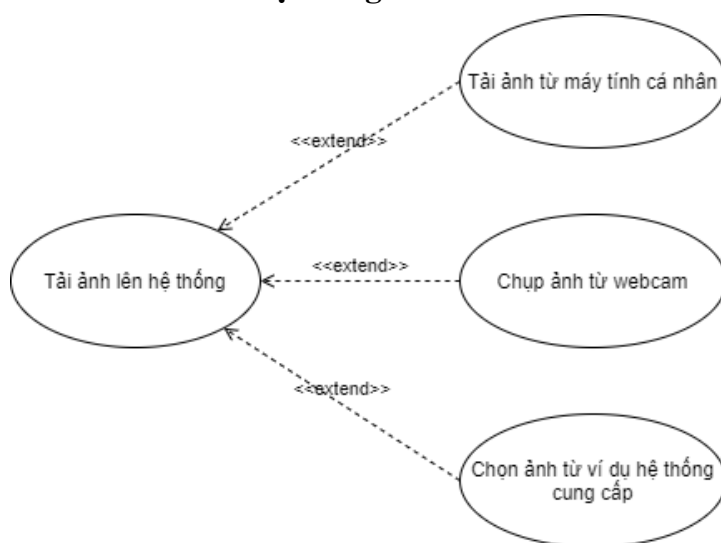
Với tác nhân là quản trị viên: Tất cả các chức năng của quản trị viên đều phải đăng nhập mới thực hiện được.

Tên use case	Mô tả tóm tắt
Quản lý lịch sử sinh mã code	<ul style="list-style-type: none"> + Quản trị viên có thể xem lịch sử sinh mã code của người dùng, bao gồm ảnh, mã code sinh ra, kết quả nhận dạng, kết quả dự đoán, ...
Quản lý dữ liệu huấn luyện	<ul style="list-style-type: none"> + Quản trị viên có thể xem tập dữ liệu sử dụng để huấn luyện mô hình. + Quản trị viên có thể thêm hoặc bớt dữ liệu huấn luyện của mô hình để tăng hiệu suất mô hình.
Quản lý phản hồi người dùng	<ul style="list-style-type: none"> + Quản trị viên xác thực các lỗi nhận dạng vật thể mà người dùng phản hồi để cập nhật lại dữ liệu cho mô hình dự đoán.

Bảng 4.2 Mô tả use case tổng quan với tác nhân quản trị viên

4.1.2.2. Biểu đồ usecase phân rã

Use case phân rã “Tải ảnh lên hệ thống”



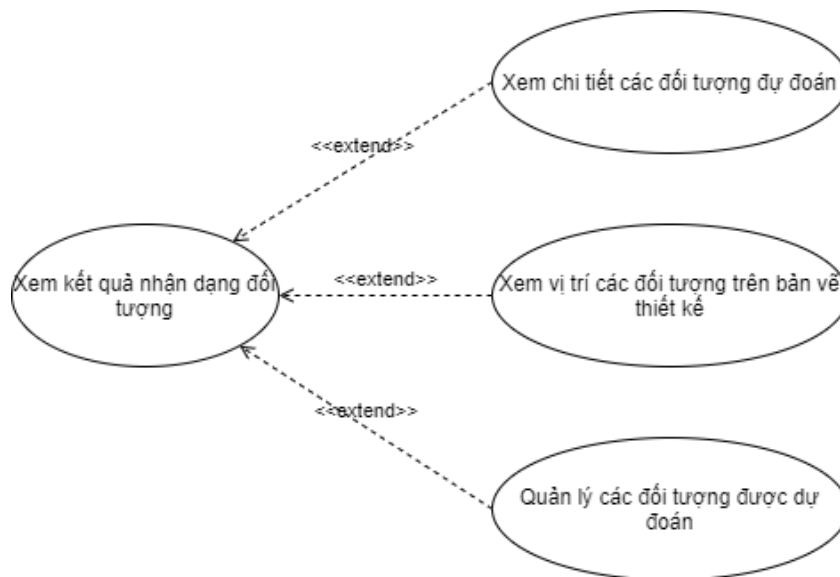
Hình 4.4 Biểu đồ phân rã “Tải ảnh lên hệ thống”

Tác nhân: Người dùng.

Mô tả use case:

Tên use case	Mô tả tóm tắt
Tải hình ảnh từ máy tính cá nhân	+ Người dùng sẽ tải hình ảnh của bản thiết kế vẽ tay từ máy tính cá nhân lên hệ thống
Chụp ảnh từ webcam	+ Người dùng cho phép trình duyệt sử dụng webcam và sử dụng webcam để chụp lại bản vẽ thiết kế đẩy lên hệ thống
Chọn ảnh từ ví dụ hệ thống cung cấp	+ Hệ thống sẽ cung cấp một số ảnh chụp các bản thiết kế phổ biến, người dùng có thể lựa chọn để thực hiện chuyển đổi sang mã code

Use case phân rã “Xem kết quả nhận dạng đối tượng”



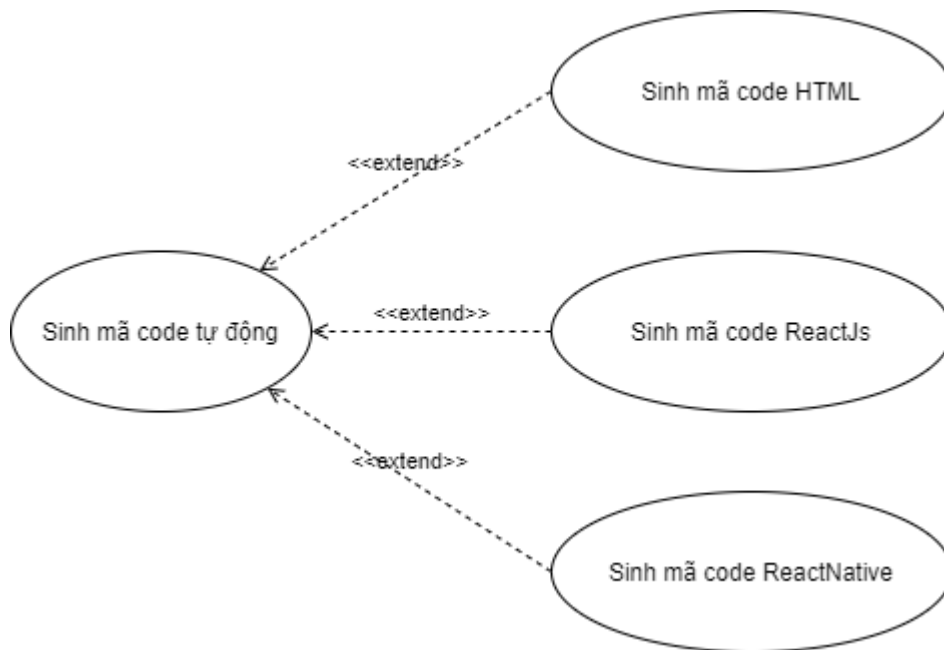
Hình 4.5 Biểu đồ phân rã “Xem kết quả nhận dạng đối tượng”

Tác nhân: Người dùng.

Mô tả use case:

Tên use case	Mô tả tóm tắt
Xem chi tiết các đối tượng dự đoán	+ Người dùng có thể xem chi tiết các thông số mà hệ thống xác định đối tượng sinh ra dựa trên bản vẽ thiết kế như nhãn, tọa độ, text nội bộ, xác suất, ...
Xem vị trí các đối tượng trên bản vẽ thiết kế	+ Dựa trên các thông số mô hình đưa ra, người dùng có thể xem ảnh thiết kế và các khoanh vùng đối tượng xuất hiện trong bản vẽ.
Quản lý các đối tượng dự đoán	+ Do hệ thống có thể không đưa ra chính xác được vị trí, nhãn, tọa độ, ... người dùng có thể tùy chỉnh theo ý tưởng thiết kế và cập nhật lên hệ thống và chuyển sang giai đoạn sinh mã code.

Use case phân rã “Sinh mã code tự động”



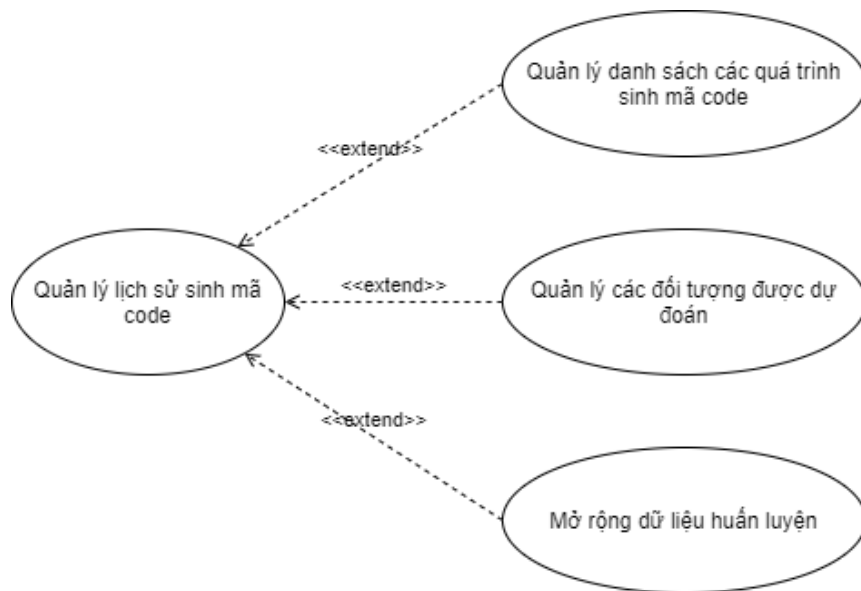
Hình 4.6 Biểu đồ phân rã “Sinh mã code tự động”

Tác nhân: Người dùng.

Mô tả use case:

Tên use case	Mô tả tóm tắt
Sinh mã code HTML	<ul style="list-style-type: none"> + Người dùng lựa chọn sinh mã code HTML + Hệ thống sẽ phân tích và sinh ra mã code HTML dựa trên các đối tượng mà người dùng đã xác thực + Hệ thống cho phép người dùng tải về dưới dạng file “index.html”
Sinh mã code ReactJs	<ul style="list-style-type: none"> + Hệ thống sinh ra các component của React và đóng gói cùng với mã code Reactjs cơ sở cho phép người dùng tải xuống và triển khai trên môi trường web
Sinh mã code React Native	<ul style="list-style-type: none"> + Hệ thống sinh ra các component của React Native và đóng gói cùng với mã code cơ sở cho phép người dùng tải xuống và triển khai trên môi trường Android hoặc IOS

Use case phân rã “Quản lý lịch sử sinh mã code”



Hình 4.7 Biểu đồ phân rã “Sinh mã code tự động”

Tác nhân: Quản trị viên.

Mô tả use case:

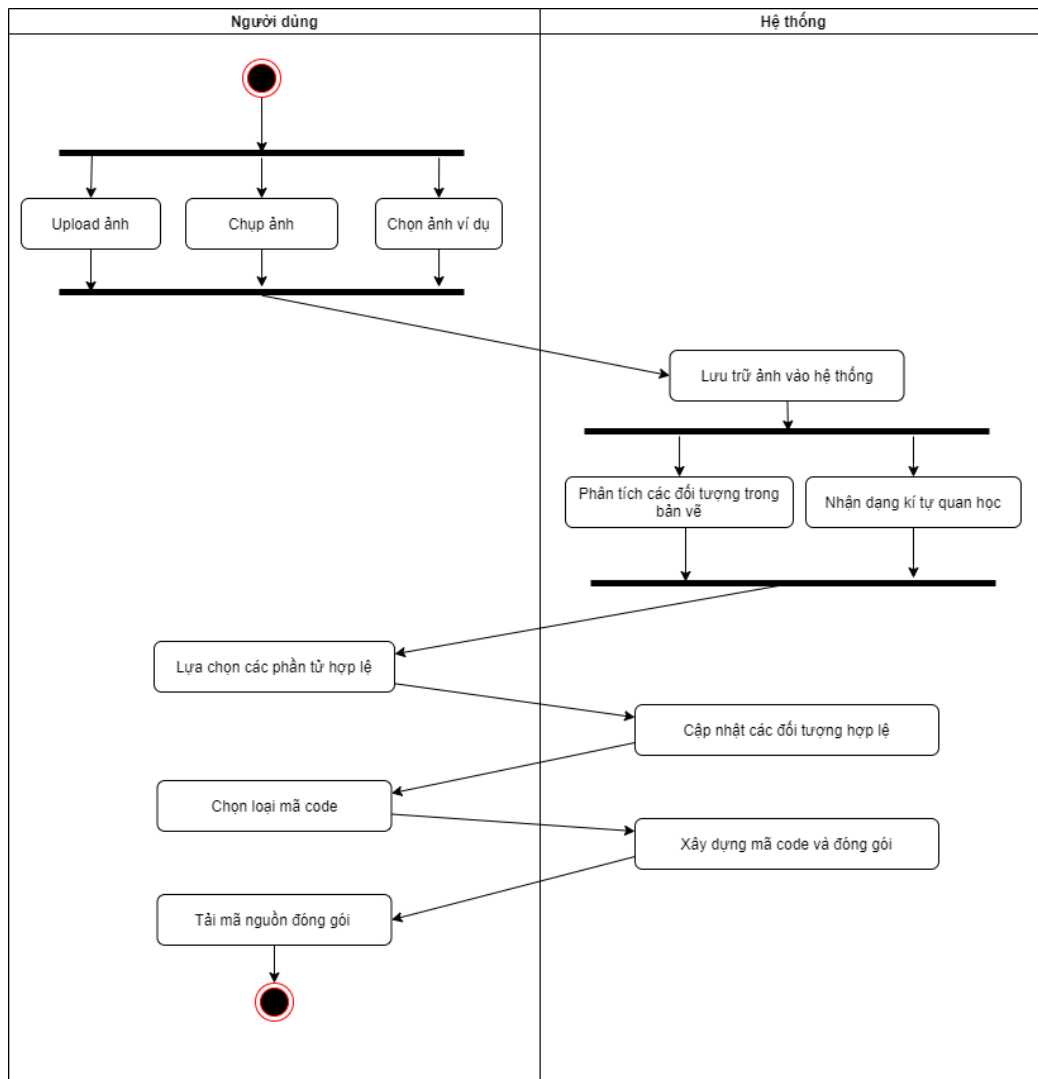
Tên use case	Mô tả tóm tắt
Quản lý danh sách các quá trình sinh mã code	+ Quản trị viên có thể xem danh sách các quá trình sinh mã code của người dùng hệ thống, từ ảnh đầu vào, mã code sinh ra, kết quả dự đoán, ...
Quản lý các đối tượng dự đoán	+ Hệ thống đưa các biểu đồ thống kê các đối tượng được dự đoán cùng với các thông số chi tiết từ giúp quản trị viên đánh giá độ hiệu quả của mô hình.
Mở rộng dữ liệu huấn luyện	+ Quản trị viên lựa chọn các quá trình có chất lượng tốt cập nhật vào dữ liệu huấn luyện của mô hình.

4.1.3 Đặc tả chức năng

Đặc tả use case “Sinh mã code tự động”

Tác nhân: Người dùng	Tiền điều kiện: Không
Dữ liệu vào: Ảnh chụp bản vẽ thiết kế	Dữ liệu ra: Mã code chương trình
Mô tả tóm tắt: Người dùng tải ảnh lên hệ thống, sau đó hệ thống tiến hành phân tích bức ảnh và đưa ra mã code chương trình tương ứng với bản thiết kế đã chụp	

Luồng sự kiện:

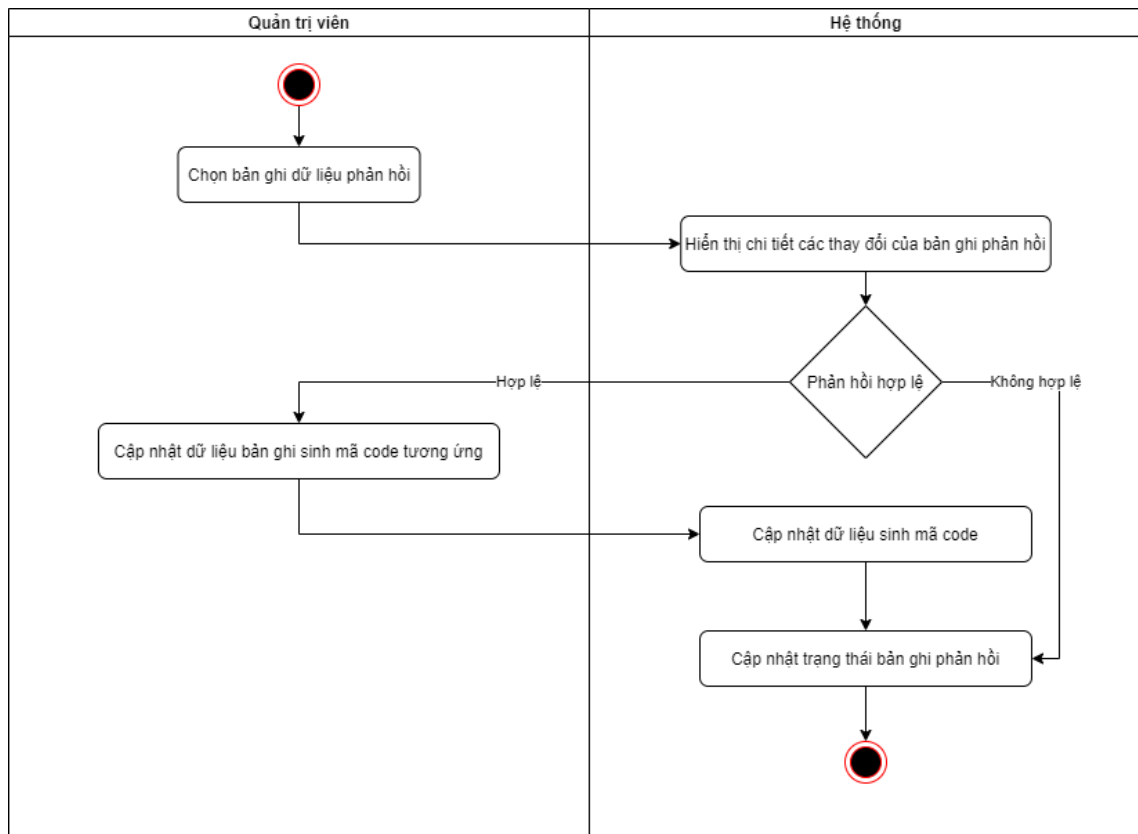


Hình 4.8 Luồng sự kiện “Sinh mã code tự động”

Đặc tả use case “Quản lý phản hồi người dùng”

Tác nhân: Quản trị viên	Tiền điều kiện: Yêu cầu đăng nhập
Dữ liệu vào: Lựa chọn bản ghi sinh mã code tự động được phản hồi	Dữ liệu ra: Phản hồi được xác thực
Mô tả tóm tắt: Người dùng sau khi lựa chọn ra các đối tượng hợp lệ so với bản vẽ thiết kế của mình, nếu có thay đổi so với kết quả dự đoán ban đầu của hệ thống sẽ gửi phản hồi về để quản trị viên phê duyệt và cập nhật lại dữ liệu	

Luồng sự kiện:



Hình 4.9 Luồng sự kiện “Quản lý phản hồi người dùng”

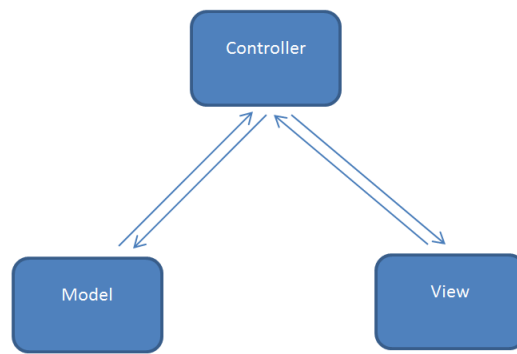
4.1.4 Yêu cầu phi chức năng

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

- Tính dễ dùng: Trang web có giao diện đẹp, thân thiện với người dùng, có sự thống nhất về màu sắc, vị trí, kích thước của các phần tử trong một trang và giữa các trang với nhau.
- Yêu cầu hiệu năng: Thời gian xác định các đối tượng trong bản vẽ trong khoảng thời gian dưới 1 phút.
- Yêu cầu tương thích: Trang web tương thích với tất cả các trình duyệt hiện nay như Chrome, Firefox, Safari.
- Yêu cầu bảo mật: Chỉ có quản trị viên mới có thể đăng nhập vào trong hệ thống quản lý/

4.2 Kiến trúc hệ thống

Hệ thống được thiết kế theo kiến trúc phần mềm MVC, viết tắt của Model-View-Controller, mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.



Hình 4.10 Mô hình MVC

Mô hình MVC được chia làm 3 lớp xử lý gồm Model – View – Controller:

- Model: là nơi chứa những nghiệp vụ tương tác với dữ liệu hoặc hệ quản trị cơ sở dữ liệu (mysql, mssql, ...); nó sẽ bao gồm các class/function xử lý nhiều nghiệp vụ như kết nối database, truy vấn dữ liệu, thêm – xóa – sửa dữ liệu...
- View: là nơi chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh... nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.
- Controller: là nơi tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class/ function xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu thông tin cần thiết nhờ các nghiệp vụ lớp Model cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp View.

Mô hình MVC có trình tự xử lý rõ ràng, các quy hoạch class/function vào các thành phần riêng biệt Controller - Model - View, việc đó làm cho quá trình phát triển - quản lý - vận hành - bảo trì web diễn ra thuận lợi hơn, tạo ra được các chức năng chuyên biệt hoá đồng thời kiểm soát được luồng xử lý.

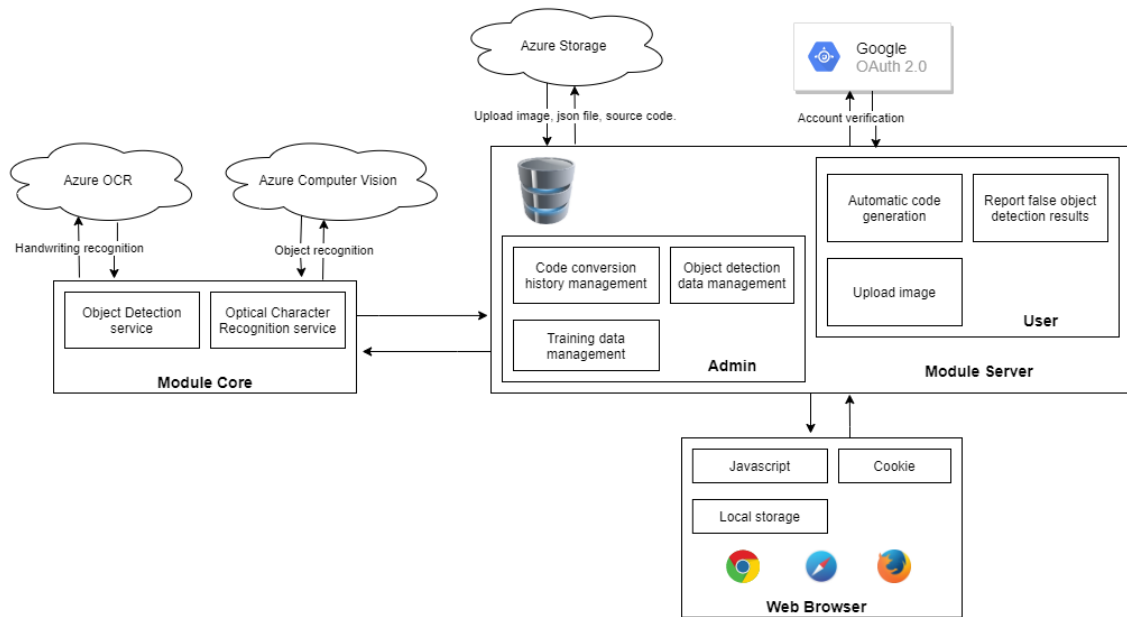
Hệ thống sinh mã code tự động được xây dựng tuân thủ theo kiến trúc MVC gồm 3 phần Model, Controller, View.

- Model: Gồm các thành phần quản lý như: PredictionModel, ContentViewModel, GeneratedHtmlModel, PredictionDetailsViewModel, PredictionViewModel, ... và thao tác với CSDL: class PredictedObject, class PredictionDetail, class Geometry, class GroupBox, ...
- Controller: Gồm các thành phần xử lý nghiệp vụ logic của hệ thống như: ApplicationController, TemplateController, LayoutController, ...
- View: Chứa các file thể hiện nội dung được Controller gửi đến. Các file này được viết theo cú pháp Razor – cú pháp kết hợp C# và HTML để sinh ra HTML động

Với hệ thống quản lý lịch sử sinh mã code, hệ thống cũng áp dụng kiến trúc MVC tương tự hệ thống sinh mã code, khác biệt duy nhất ở đây là công nghệ sử dụng để triển khai.

4.2.1.1. Thiết kế tổng quan

Hệ thống bao gồm Module Core, Module WebServer, Module giao diện ứng dụng



Hình 4.11 Thiết kế tổng quan hệ thống

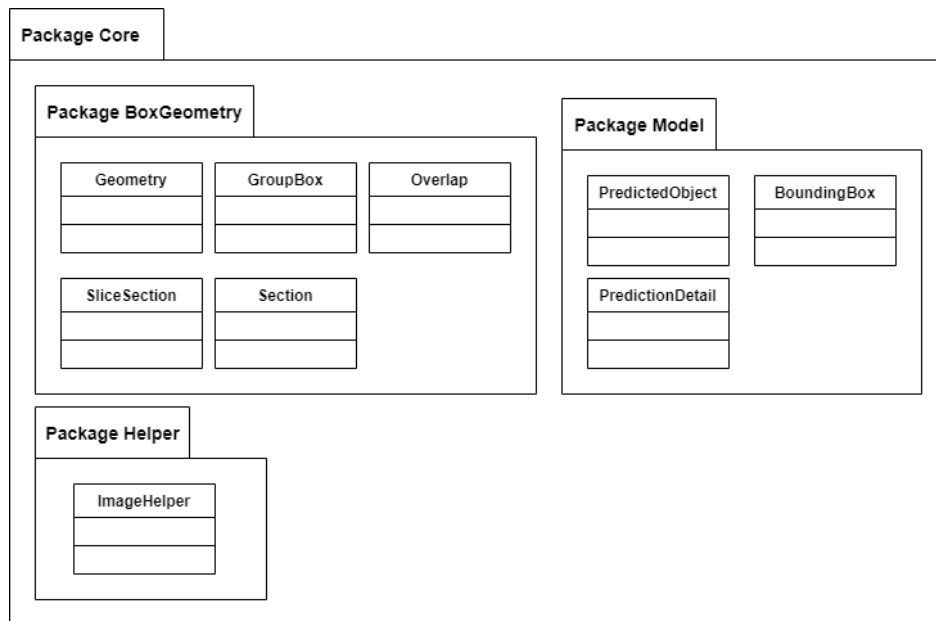
Module Core là module độc lập tương tác trực tiếp với các dịch vụ Thị giác máy tính của Azure. Module thiết lập kết nối tới dịch vụ Azure Optical Character Recognition và dịch vụ Azure Computer Vision từ đó cung cấp API hỗ trợ server trong bài toán sinh mã code tự động

Module Server cung cấp các chức năng của hệ thống dành cho người quản trị viên và người dùng. Module kết nối tới Module Core yêu cầu các dịch vụ liên quan tới nhận dạng đối tượng và nhận dạng chữ viết tay. Module kết nối tới dịch vụ Azure nhằm lưu trữ dữ liệu dạng blob như ảnh, file json, mã code HTML, file zip, ... Module này cũng kết nối tới dịch vụ Google Oauth nhằm xác thực người dùng thông qua tài khoản Google.

Trang web có thể hiển thị và sử dụng tốt trên các trình duyệt phổ biến hiện nay như Chrome, Safari, Firefox cung cấp giao diện đẹp mắt và thân thiện với người dùng. Trên trình duyệt, sử dụng Cookie kết hợp với Local storage để lưu thông tin xác thực người dùng của hệ thống.

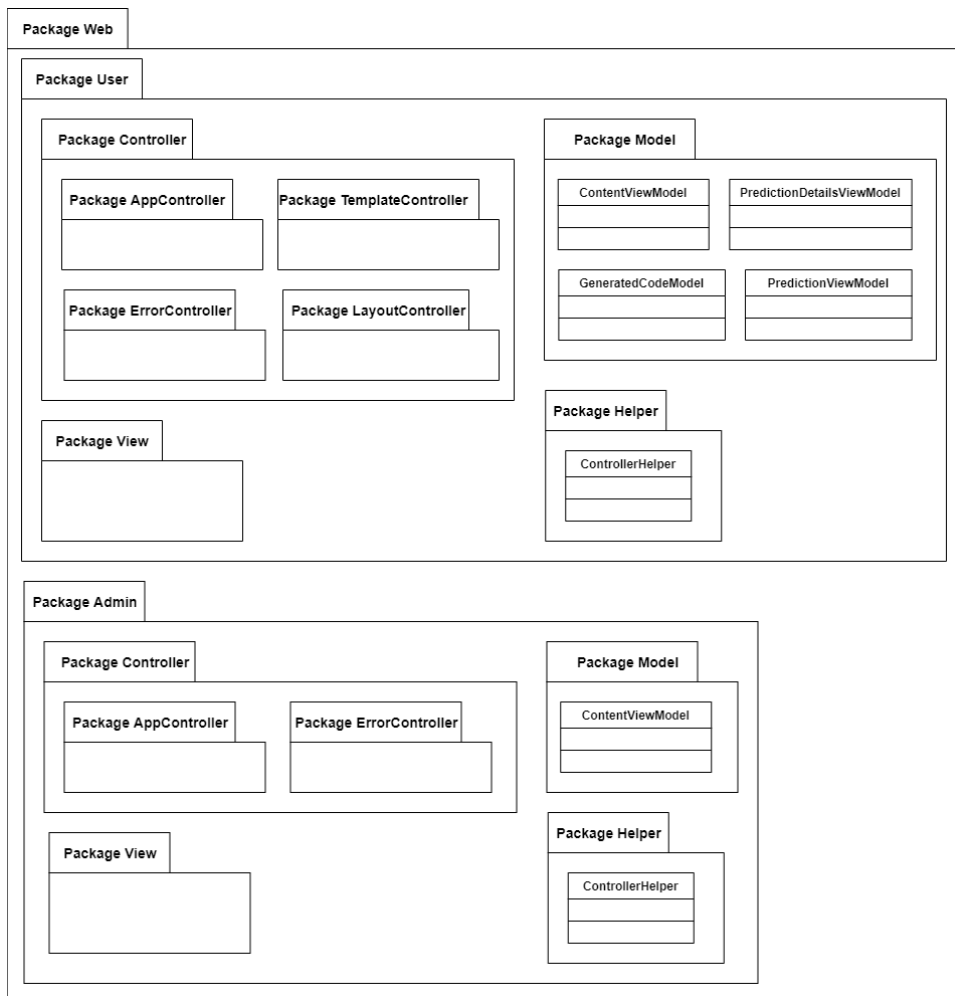
4.3 Thiết kế chi tiết gói

Module Core được tạo nên từ gói Core



Hình 4.12 Thiết kế gói “Core”

Webserver được chia thành 2 module: module User cho hệ thống sinh mã code dành cho người dùng và module Admin cho hệ thống quản trị dành cho quản trị viên



Hình 4.13 Thiết kế gói “Webserver”

❖ Đối với gói User

Trong gói Controller, gói ApplicationController xử lý luồng hệ thống theo yêu cầu của người dùng. Gói ErrorController xử lý luồng lỗi chung cho hệ thống. Gói TemplateController và Gói LayoutController được sử dụng trong quá trình sinh mã code từ đối tượng dự đoán.

Mỗi gói trong gói Model tương ứng với một bảng trong CSDL. Mỗi lớp bao gồm các thuộc tính và phương thức để định nghĩa kiểu liên kết với các bảng khác trong CSDL, Ngoài ra, mỗi lớp còn có thể chứa các phương thức truy vấn dữ liệu để các lớp trong gói Model sử dụng.

Gói View bao gồm các file hiển thị nội dung do Controller cung cấp

Gói Helper cung cấp các hàm hỗ trợ cho Controller, bao gồm các thuật toán, xử lý ảnh, xử lý dữ liệu thô, ...

❖ Đối với gói Admin

Cách tổ chức đóng gói tương tự đối với gói User.

4.4 Thiết kế lớp

❖ Gói “Core.BoxGeometry”

Thiết kế lớp “Geometry”

Các thuộc tính và phương thức:

Class Geometry
+ AlignTop (boxes): void + AlignLeft (boxes): void + MakeSameSize (boxes): void + IsBoxInVerticalRange (masterBox, childBox): boolean + IsBoxInHorizontalRange (masterBox, childBox): boolean + BuildProjectionRuler (boxes, axis): void + BuildGroups(boxes): GroupBox [] + RemoveOverLapping (boxes): void

Trong đó:

- **AlignTop (boxes):** căn lề trên cho tất cả các boxes về giá trị trung bình cộng lề phải
- **AlignLeft (boxes):** căn lề trái cho tất cả các boxes về giá trị trung bình cộng lề trái
- **MakeSameSize (boxes):** đưa tất cả các boxes về chung một kích thước là trung bình cộng của chiều cao và trung bình cộng của chiều dài
- **IsBoxInVerticalRange (masterBox, childBox):** kiểm tra xem childBox có nằm trong phạm vi theo chiều dọc của masterBox hay không
- **IsBoxInHorizontalRange (masterBox, childBox):** kiểm tra xem childBox có nằm trong phạm vi theo chiều ngang của masterBox hay không

- **BuildGroups(boxes):** Sử dụng thuật toán đệ quy phân chia các boxes thành các nhóm dựa trên toạ độ của chúng
- **RemoveOverLapping (boxes):** Loại bỏ các boxes bị các boxes khác khi đè xét treo trục X, trục Y, và diện tích ghi đè lên nhau

Thiết kế lớp “Overlap”

Các thuộc tính và phương thức:

Class Geometry
+ OverlapArea (box1, box2): double + OverlapAreaX (box1, box2): double + OverlapAreaY (box1, box2): double

Trong đó:

- **OverlapArea (box1, box2):** tính tỉ lệ ghi đè lên box nhỏ hơn theo diện tích
- **OverlapAreaX (box1, box2):** tính tỉ lệ ghi đè về độ dài xét theo trục X
- **OverlapAreaY (box1, box2):** tính tỉ lệ ghi đè về chiều cao xét theo trục Y

Thiết kế lớp “GroupBox”

Các thuộc tính và phương thức:

Class GroupBox
+ boxes: BoundingBox [] + direction: string + isEmpty: boolean + X: double + Y: double + Height: double + Width: double + Alignment: string
+ Boxes (boxes): void + Direction (box): void

Trong đó:

- **boxes:** các boxes được phân chia vào trong một nhóm dựa trên kích thước và toạ độ của chúng
- **direction:** chiều duyệt trong quá trình duyệt các box
- **isEmpty:** thể hiện group này có phần tử boxes nào hay không
- **X:** toạ độ X của góc trên trái
- **Y:** toạ độ Y của góc trên trái
- **Height:** chiều cao của groupBox
- **Width:** chiều dài của GroupBox

- **Alignment:** căn lề của GroupBox

❖ Gói “Core.Helper”

Thiết kế lớp “ImageHelper”

Các thuộc tính và phương thức:

Class ImageHelper
+ SliceImage (image, x, y, width, height): image + DrawRectangle (image, predictedObject): image + Boxes (): BoundingBox [] + Colors (): Dictionary <Tag, Color> + ScaleByPercent (image, percent): image + OptimizeImage (image, scale, quality): image

Trong đó:

- **SliceImage (image, x, y, width, height):** cắt ảnh theo kích thước và tọa độ trên ảnh gốc
- **DrawRectangle (image, predictedObject):** Vẽ khung bao quanh của một đối tượng dựa trên vị trí và tọa độ
- **Boxes ():** danh sách boundingBox được phân tích dựa trên bức ảnh
- **Colors ():** lấy ra từ điển màu tương ứng với tên các phần tử trong HTML nhằm mục đích phân loại
- **ScaleByPercent (image, percent):** phóng to hoặc thu nhỏ kích thước bức ảnh theo tỉ lệ
- **OptimizeImage (image, scale, quality):** tối ưu kích thước lưu trữ của bức ảnh dựa trên tỉ lệ và chất lượng của bức ảnh

❖ Gói “Core.Model”

Thiết kế lớp “BoundingBox”

Các thuộc tính và phương thức:

Class BoundingBox
+ boxColor: color + Top: double + Left: double + Height: double + Width: double + TopLeft: <double, double> + BottomLeft: <double, double> + TopRight: <double, double> + BottomRight: <double, double> + PredictedObject: PredictedObject

Trong đó:

- **boxColor:** màu sử dụng để dựng khung bao quanh box
- **Top:** tọa độ theo trục Y của góc trên-trái
- **Left:** tọa độ theo trục X của góc trên-trái
- **TopLeft:** tọa độ đỉnh trên-trái
- **BottomLeft:** tọa độ đỉnh dưới-trái
- **TopRight:** tọa độ đỉnh top-phải
- **BottomRight:** tọa độ đỉnh dưới-phải
- **Height:** chiều cao của box
- **Width:** chiều dài của box
- **PredictedObject:** đối tượng dự đoán nằm bên trong box

Thiết kế lớp “PredictionObject”

Các thuộc tính và phương thức:

Class PredictedObject
+ Text: string + Probability: double + isSelected: boolean + isWrong: boolean + SlicedImage: image + FileName: string + ClassName: string

Trong đó:

- **Text:** Dòng chữ xuất hiện nội tại trong box
- **Probability:** Xác suất mô hình dự đoán ra đối tượng nằm trong box
- **isSelected:** sử dụng để đánh dấu các đối tượng được sử dụng trong quá trình sinh mã code
- **isWrong:** đánh dấu đối tượng bị nhận dạng sai so hình ảnh
- **SlicedImage:** Ảnh cắt đối tượng xuất hiện trong box từ ảnh thiết kế gốc
- **FileName:** tên của bức ảnh
- **ClassName:** tên của đối tượng dự đoán phục vụ trong việc sinh mã code

Thiết kế lớp “PredictionDetail”

Các thuộc tính và phương thức:

Class PredictedDetail
+ OriginalImage: string + PredictionImage: string + GroupBox: GroupBox + SlicedImage: image [] + AllClasses: string []

Trong đó:

- **OriginalImage**: link đường dẫn tới file ảnh gốc
- **PredictionImage**: link dẫn đến ảnh chụp màn hình view thể hiện bởi mã code sinh tự động
- **GroupBox**: GroupBox được dựng nên tương ứng với bức ảnh
- **SlidedImage**: danh sách đường dẫn tới các ảnh cắt các phần tử xuất hiện trong bức ảnh gốc
- **AllClasses**: Mảng danh sách className tương ứng với các phần tử và thông số của chúng dùng để xây dựng mã code

❖ Gói “WebServer.User.Controller”

Thiết kế lớp “AppController”

Các thuộc tính và phương thức:

Class AppController
+ Upload (): void + SelectPredictedObect (): void + GenerateCode (): file + SaveResult (): void + TakeSnapshot (): void + Details (): void + SaveFile (): file

Trong đó:

- + **Upload ()**: tải ảnh lên hệ thống và lưu trữ lên Azure Blob Storage
- + **SelectPredictedObect ()**: Lựa chọn các PredictedObject hợp lệ để tiến hành sinh mã code
- + **GenerateCode ()**: thực hiện sinh mã code tự động dựa trên đối tượng đã chọn
- + **SaveResult ()**: lưu trữ file lên hệ thống Azure Blob Storage
- + **TakeSnapshot ()**: chụp ảnh giao diện được render từ mã code sinh tự động
- + **Details ()**: Xem chi tiết kết quả dự đoán đối tượng
- + **SaveFile ()**: đóng gói và lưu trữ file vào bộ nhớ máy tính

Thiết kế lớp “LayoutController”

Các thuộc tính và phương thức:

Class LayoutController
+ HtmlResult (): void + ReactResult (): void + AtomicReactResult (): void

+ ReactNativeResult () : void

Trong đó:

+ **HtmlResult ()**: render ra file “common.html” là kết quả sinh mã code HTML

+ **ReactResult ()**: render ra file “react.html” là kết quả sinh mã code ReactJs và cắm trực tiếp vào HTML bằng thẻ <script> </script>

+ **AtomicReactResult ()**: sinh ra mã code Reactjs được triển khai theo nguyên lý thiết kế Atomic design khởi tạo bằng create-react-app và đóng gói zip

+ **ReactNativeResult ()**: sinh ra mã code ReactNative được đóng gói zip cùng mã code cơ sở của ReactNative

❖ Gói “WebServer.User.Helper”

Thiết kế lớp “ControllerHelper”

Các thuộc tính và phương thức:

Class ControllerHelper
+ RenderViewToHTML () : void + RenderViewToReact () : void + RenderViewToReactNative () : file + SaveResult () : void

Trong đó:

+ **RenderViewToHTML ()**: sinh mã code HTML theo GroupBox kết hợp với framework Bootstrap4 cho phần hiển thị style.

+ **RenderViewToReact ()**: sinh mã code Reactjs theo GroupBox, tên lớp được đặt theo cú pháp Bootstrap4 cho phần hiển thị style, cuối cùng được nhúng vào file HTML thông qua thẻ <script/>.

+ **RenderViewToAtomicReact ()**: Phân tích các phần tử xuất hiện trong Group Box, tiến hành tạo ra các file Atom thể hiện cho các phần tử đó, từ GroupBox tạo ra “Page” theo cú pháp JSX thể hiện cấu trúc của thiết kế.

+ **RenderViewToReactNative ()**: Phân tích các phần tử xuất hiện trong Group Box, tiến hành tạo ra các component tương ứng thể hiện cho các phần tử đó, từ cây GroupBox tạo file “index.js” thể hiện cấu trúc của thiết kế.

❖ Gói “WebServer.Admin.Controller”

Thiết kế lớp “AppController”

Các thuộc tính và phương thức:

Class AppController
+ Details () : void + DownloadHtmlCode () : void

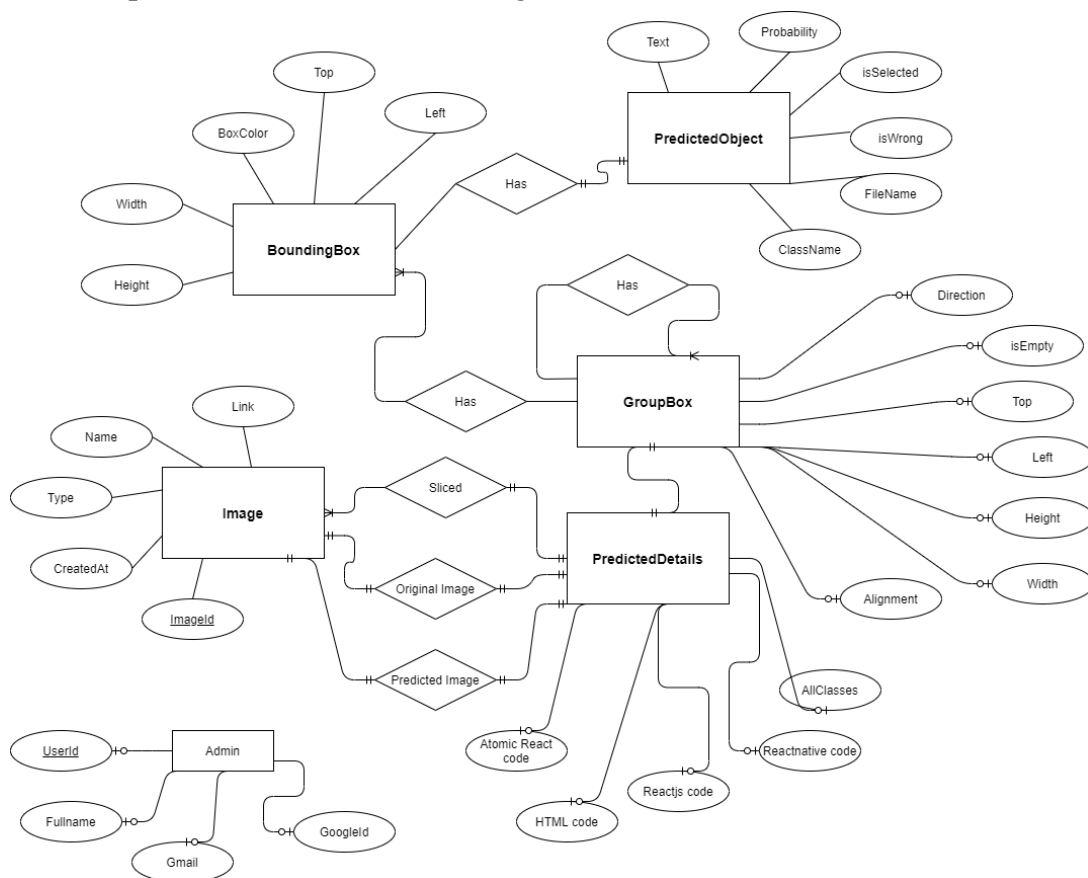
- + DownloadReactCode (): void
- + DownloadAtomicReactCode (): void
- + DownloadReactNativeCode (): void
- + RemoveSketchData (): void
- + UpdatePredictedObject (): void
- + AddTrainData (): void
- + RetrainModel (): void

Trong đó:

- + **Details ()**: Xem chi tiết kết dữ liệu sinh mã code
- + **DownloadHtmlCode ()**: tải về code HTML tương ứng
- + **DownloadReactCode ()**: tải về code React tương ứng
- + **DownloadAtomicReactCode ()**: tải về file zip của code Reactjs theo thiết kết Atomic tương ứng
- + **DownloadReactNativeCode ()**: tải về file zip của code ReactNative tương ứng
- + **RemoveSketchData ()**: xoá dữ liệu bản ghi sinh mã code
- + **AddTrainData ()**: thêm kết quả dự đoán vào trong tập dữ liệu huấn luyện
- + **RetrainModel ()**: Yêu cầu huấn luyện lại mô hình

4.5 Thiết kế cơ sở dữ liệu

Biểu đồ phân tử liên kết của hệ thống:



Hình 4.14 Biểu đồ phân tử liên kết

Từ biểu đồ phân tử liên kết trên, dưới đây là các bảng thiết kế CSDL cho hệ thống như sau:

Bảng PredictedDetails: Lưu thông tin chi tiết của một quá trình sinh mã code tự động

Tên trường	Kiểu dữ liệu	Mô tả
id	string	Khóa chính, id của đối tượng
original_image_id	string	Id của ảnh nguyên bản
predicted_image_id	string	Id của ảnh chụp giao diện thiết kế sinh tự động
group_box_id	string	Id của groupBox
all_classes	array	Mảng danh sách các tên class được sinh ra
html_code_url	string	URL đến file code html
react_code_url	string	URL đến file code react
Atomic_react_code_url	string	URL đến file code atomic react
React_native_code_url	string	URL đến file code reactnative

Bảng 4.3 Cấu trúc bảng PredictedDetails

Bảng Image: Lưu thông tin ảnh

Tên trường	Kiểu dữ liệu	Mô tả
id	string	Khóa chính, id của ảnh
name	string	Tên của bức ảnh
type	string	Loại của bức ảnh
created_at	timestamp	Thời gian tạo
Link	string	Link của bức ảnh

Bảng 4.4 Cấu trúc bảng Image

Bảng Slide_Image: Lưu thông tin ảnh

Tên trường	Kiểu dữ liệu	Mô tả
predicted_details_id	string	Khóa chính, id của predicted details
image_id	string	Khoá chính, id của bức ảnh

Bảng 4.5 Cấu trúc bảng Slide_Image

Bảng GroupBox: Lưu thông tin cây cấu trúc

Tên trường	Kiểu dữ liệu	Mô tả
id	string	Khóa chính, id của GroupBox
direction	string	Chiều duyệt phần tử của GroupBox
top	double	Toạ độ Y đỉnh trên-trái của GroupBox
left	double	Toạ độ X đỉnh trên-trái của GroupBox
width	string	Chiều dài của GroupBox
height	string	Chiều cao của GroupBox
alignment	string	Căn lề của GroupBox
child_group_box_id	string	Id của GroupBox con

Bảng 4.6 Cấu trúc bảng GroupBox

Bảng BoundingBox: Lưu thông tin vị trí của đối tượng định dạng

Tên trường	Kiểu dữ liệu	Mô tả
id	string	Khóa chính, id của BoudingBox
group_box_id	string	Id của GroupBox cha
color_box	string	Màu đánh dấu của BoudingBox
top	double	Toạ độ Y đỉnh trên-trái của BoudingBox
left	double	Toạ độ X đỉnh trên-trái của BoudingBox
width	string	Chiều dài của BoudingBox
height	string	Chiều cao của BoudingBox

Bảng 4.7 Cấu trúc bảng BoundingBox

Bảng PredictedObject: Lưu thông tin chi tiết của đối tượng định dạng

Tên trường	Kiểu dữ liệu	Mô tả
bounding_box_id	string	Khoá chính, là Id của BoundingBox tương ứng
text	string	Text nội tại của đối tượng

is_selected	boolean	Đánh dấu đối tượng có được chọn để sinh mã code hay không
is_wrong	boolean	Đánh dấu đối tượng có bị nhận dạng sai không
probability	double	Xác suất dự đoán ra đối tượng
className	string	ClassName được hệ thống sinh ra cho sử dụng cho quá trình sinh mã code

Bảng 4.8 Cấu trúc bảng PredictedObject

Bảng Admin: Lưu thông tin của các quản trị viên của hệ thống

Tên trường	Kiểu dữ liệu	Mô tả
Google_id	string	Khoá chính, là do Google tạo tương ứng với gmail
gmail	string	Gmail của quản trị viên
full_name	string	Tên đầy đủ của quản trị viên
state	string	Đánh dấu trạng thái của quản trị viên

Bảng 4.9 Cấu trúc bảng Admin

4.6 Xây dựng ứng dụng

4.6.1 Thư viện và công nghệ sử dụng

Công cụ	Mục đích	Địa chỉ URL
+ ASP.NET core 3.1 kết hợp với cú pháp Razor	+ Xây dựng trang web sinh mã code tự động + Chuyển đổi data thành mã code tương ứng theo cú pháp Razor	https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1
Reactjs 17.0	+ Xây dựng giao diện cho trang quản lý + Sinh tự động data thành mã code Reactjs	https://reactjs.org/
React Native 0.63	+ Sinh mã code tự động Reactnative và kiểm thử trên môi trường Android	https://reactnative.dev/
Microsoft Custom Vision	+ Mô hình huấn luyện Nhận dạng vật thể	https://www.customvision.ai/

	+ Dịch vụ Trích xuất văn bản	
Visual Studio 2019	+ IDE	https://visualstudio.microsoft.com/vs/
Android Emulator	+ Máy ảo android cho kiểm thử ứng dụng android	https://developer.android.com/studio/run/emulator

4.6.2 Kết quả đạt được

Hệ thống gồm có trang web sinh mã code tự động từ bản vẽ thiết kế tay và trang web quản lý dữ liệu sinh mã code tự động.

Sau khi hoàn thành hệ thống và so sánh với các hệ thống sinh mã code tự động khác thì có những ưu điểm mà hệ thống khác chưa có:

- Hệ thống đã có thể sinh mã code tự động dựa trên bản vẽ thiết kế có độ tin cậy thấp, hay bản vẽ tay đơn giản.
- Có chức năng sinh mã code Reactjs hỗ trợ lập trình viên phát triển GUI trên nền tảng web.
- Có chức năng sinh mã code ReactNative hỗ trợ lập trình viên phát triển các ứng dụng di động nền tảng Android và IOS.
- Hệ thống có số lượng phần tử được sinh ra đa dạng hơn so với hệ thống Sketch2code.
- Hệ thống hỗ trợ chức năng phản hồi kết quả dự đoán sai, giúp nâng cao hiệu năng của mô hình.
- Hệ thống cung cấp giao diện dễ dùng, đơn giản.

Với hệ thống quản lý dữ liệu sinh mã code tự động, hệ thống được thiết kế đơn giản quản lý các quy trình sinh mã code và quản lý mô hình huấn luyện.

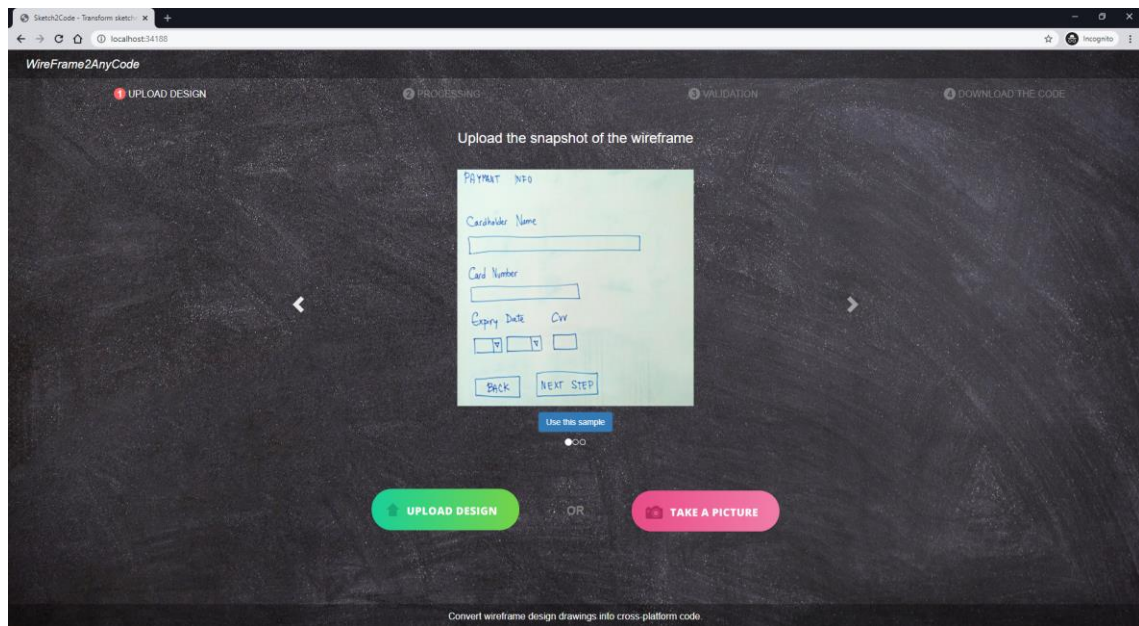
Minh họa các chức năng chính:

Trang chủ:

Trang chủ là trang chính của hệ thống, tại đây hỗ trợ người dùng tải ảnh lên, chụp ảnh, chọn ảnh ví dụ từ đó tiến hành nhận dạng và sinh mã code.

Quy trình sinh mã code gồm 4 bước:

- Bước 1: Upload design: người dùng tải lên ảnh chụp bản vẽ wireframe theo 3 cách mà hệ thống cung cấp: sử dụng ảnh mẫu, chụp ảnh trực tiếp, tải ảnh lên.
- Bước 2: Processing: Hệ thống tiến hành phân tích hình ảnh
- Bước 3. Validation: Người dùng xác thực lại các đối tượng do hệ thống nhận diện.
- Bước 4: Download the code: Người dùng chọn và tải về mã code tương ứng.

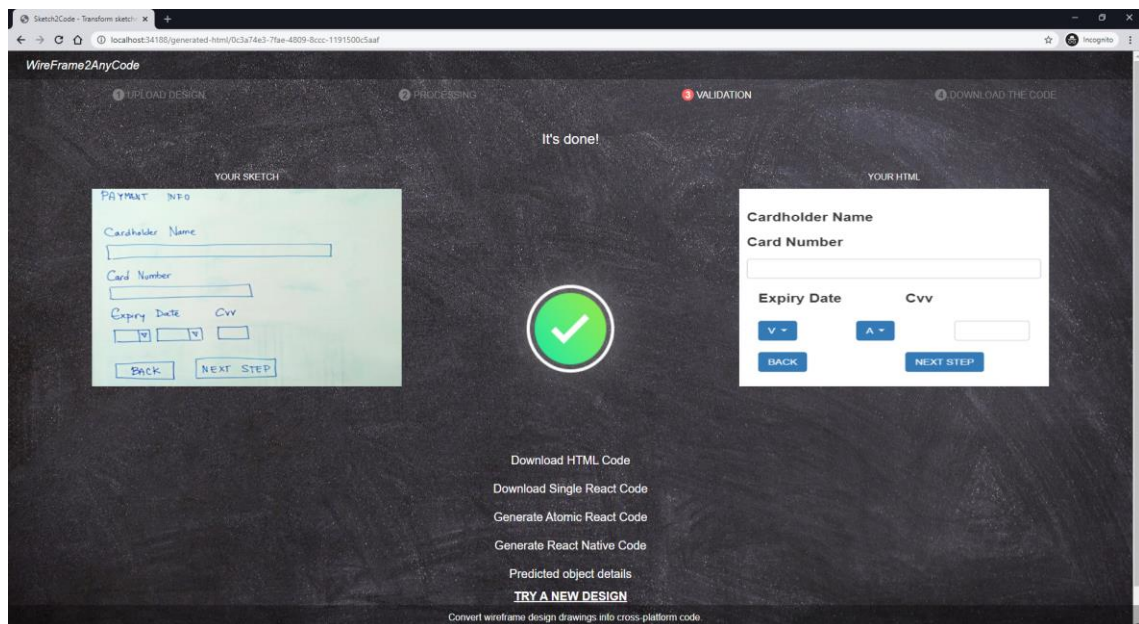


Hình 4.15 Giao diện trang chủ

Chức năng phản hồi kết quả dự đoán sai

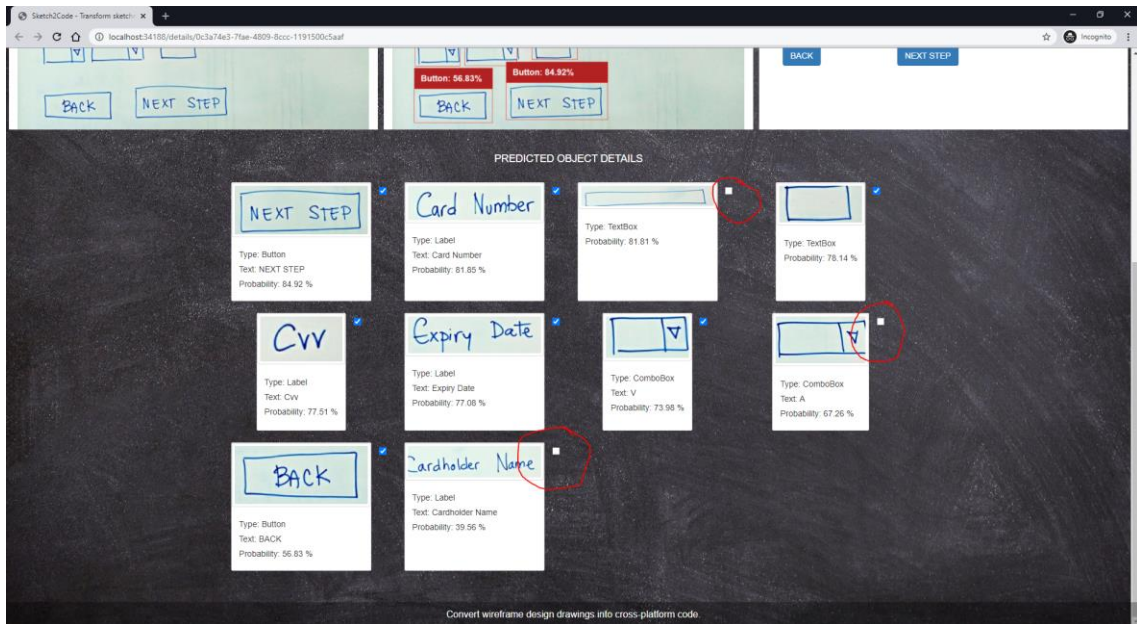
Hệ thống sẽ tự động lựa chọn các phần tử có độ chính xác trên 25%, tuy nhiên các phần tử có độ chính xác cao hơn vẫn có thể dự đoán sai hay các phần tử có độ chính xác thấp hơn có thể là dự đoán đúng.

Người dùng sẽ lựa chọn lại các phần tử chính xác đúng với thiết kế ban đầu và xác nhận.



Hình 4.16 Giao diện preview kết quả sinh mã code

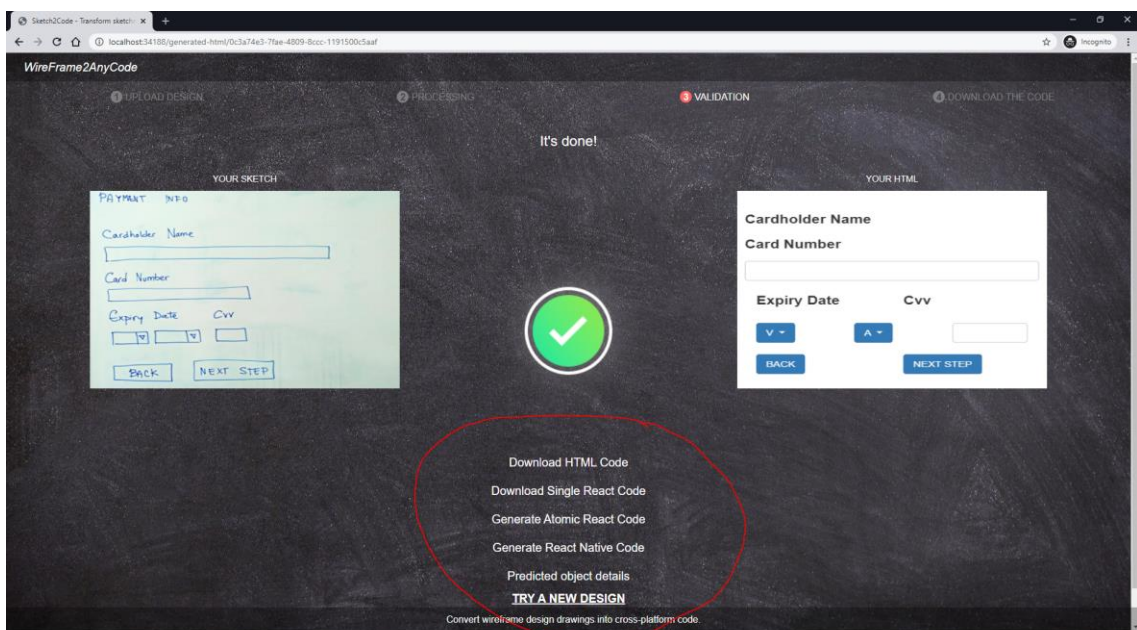
Người dùng ấn vào nút CheckBox để lựa chọn/ bỏ chọn các phần tử dùng để sinh mã code



Hình 4.17 Giao diện cập nhật dữ liệu

Chức năng sinh mã code tự động

Hệ thống cung cấp các lựa chọn sinh mã code cho người dùng, sau khi lựa chọn, hệ thống sẽ trích xuất ra file tương ứng cho người dùng tải về.



Hình 4.18 Giao diện các lựa chọn sinh mã code

❖ Kết quả sinh mã code tự động

Với đầu vào là bức ảnh chụp bản vẽ thiết kế đơn giản sau.

Reactjs nhúng trong file HTML:

Mã code HTML và giao diện tương ứng

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
```

```

<title>React Result</title>
<link
  rel="stylesheet"
  href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
  integrity="sha384-
WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMIId/iRN3ewGhXQFZCSftd1LZCfmhktB"
  crossorigin="anonymous"
/>
</head>

<body>
  <div id="root" dir="auto"></div>
  <script type="text/babel">
    function App() {
      return (
        <div class="container body-content">
          <div class="container" id="generatedHTML" style={{ padding: 25 }}>
            <div class="row justify-content-start" style={{ paddingTop: 10 }}>
              <label>Cardholder Name</label>
            </div>
            <div class="row justify-content-start" style={{ paddingTop: 10 }}>
              <label>Card Number</label>
            </div>
            <div class="row justify-content-start" style={{ paddingTop: 10 }}>
              <input class="form-control"></input>
            </div>
            <div class="row justify-content-start" style={{ paddingTop: 10 }}>
              <div class="col" style={{ paddingTop: 10 }}>
                <label>Expiry Date</label>
              </div>
              <div class="col" style={{ paddingTop: 10 }}>
                <label>Cvv</label>
              </div>
            </div>
            <div class="row justify-content-start" style={{ paddingTop: 10 }}>
              <div class="col" style={{ paddingTop: 10 }}>
                <div class="dropdown">
                  <button
                    type="button"
                    class="btn btn-primary dropdown-toggle"
                    data-toggle="dropdown"
                  >
                    V
                  </button>
                  <div class="dropdown-menu">
                    <a class="dropdown-item" href="#"></a>
                  </div>
                </div>
              </div>
              <div class="col" style={{ paddingTop: 10 }}>
                <div class="dropdown">
                  <button
                    type="button"
                    class="btn btn-primary dropdown-toggle"
                    data-toggle="dropdown"
                  >
                    A

```

```

        </button>
        <div class="dropdown-menu">
          <a class="dropdown-item" href="#"></a>
        </div>
      </div>
    </div>
    <div class="col" style={{ paddingTop: 10 }}>
      <input class="form-control"></input>
    </div>
  </div>
  <div class="row justify-content-start" style={{ paddingTop: 10 }}>
    <div class="col" style={{ paddingTop: 10 }}>
      <button class="btn btn-primary">BACK</button>
    </div>
    <div class="col" style={{ paddingTop: 10 }}>
      <button class="btn btn-primary">NEXT STEP</button>
    </div>
  </div>
</div>
</div>
);
}

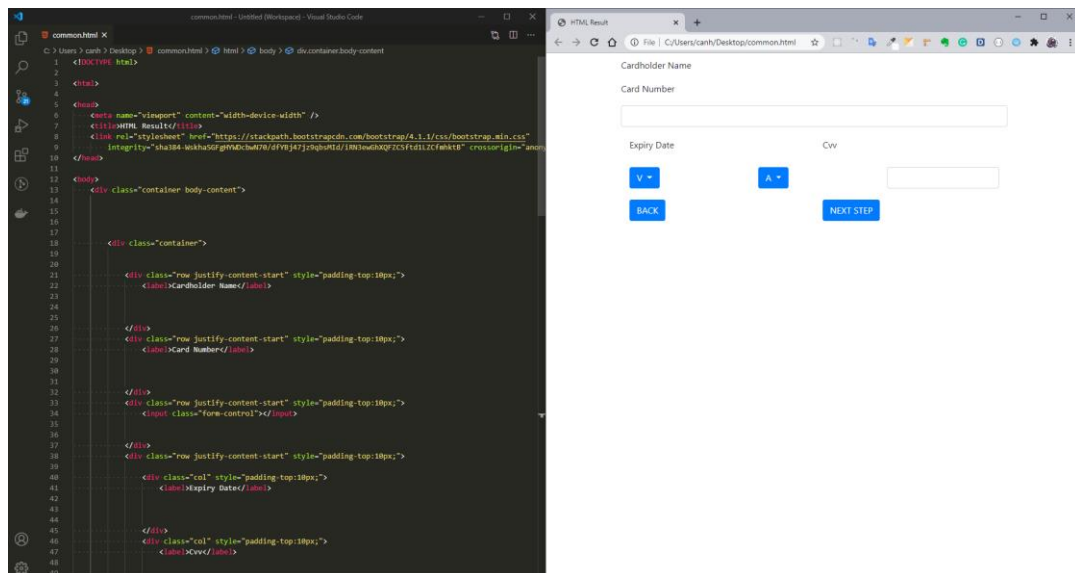
ReactDOM.render(<App />, document.getElementById('root'));
</script>

<script
  src="https://unpkg.com/react@16.13.1/umd/react.production.min.js"
  crossorigin="anonymous"
></script>
<script
  src="https://unpkg.com/react-dom@16.13.1/umd/react-dom.production.min.js"
  crossorigin="anonymous"
></script>

<script
src="https://cdn.jsdelivr.net/npm/dataformsjs@4.0.1/js/react/jsxLoader.min.js"></script>
</body>
</html>

```

Hình 4.19 Mã code Reactjs nhúng trong file HTML



Hình 4.20 Mã code HTML và kết quả hiển thị tương ứng

Mã code HTML được nhúng sẵn Bootstrap cho việc styling.

ReactJs theo mẫu thiết kế Atomic

Mã code sinh ra gồm các component atom, mỗi atom là 1 loại đối tượng phần tử dạng lá của cây cấu trúc HTML, ví dụ button, image, text, ...

Các đối tượng dạng nhánh như form, header, table, ... sẽ được xếp vào component molecules tái sử dụng các component con.

```
import React, { Fragment } from 'react';
import PageTemplate from '../templates/PageTemplate';
import Button from '../atoms/Button';
import CheckBox from '../atoms/CheckBox';
import ComboBox from '../atoms/ComboBox';
import Heading from '../atoms/Heading';
import Image from '../atoms/Image';
import Label from '../atoms/Label';
import Link from '../atoms/Link';
import Paragraph from '../atoms/Paragraph';
import RadioButton from '../atoms/RadioButton';
import TextBox from '../atoms/TextBox';

const GeneratePage = () => {
  return (
    <PageTemplate>
      <div className="container" id="generatedHTML" style={{ padding: 25 }}>
        <div className="row justify-content-start" style={{ padding: 10 }}>
          <Label text='Cardholder Name' />
        </div>
        <div className="row justify-content-start" style={{ padding: 10 }}>
          <Label text='Card Number' />
        </div>
        <div className="row justify-content-start" style={{ padding: 10 }}>
          <TextBox />
        </div>
        <div className="row justify-content-start" style={{ padding: 10 }}>
          <div className="col" style={{ padding: 10 }}>
            <Label text='Expiry Date' />
          </div>
          <div className="col" style={{ padding: 10 }}>
            <Label text='Cvc' />
          </div>
        </div>
      </div>
    </PageTemplate>
  );
};
```

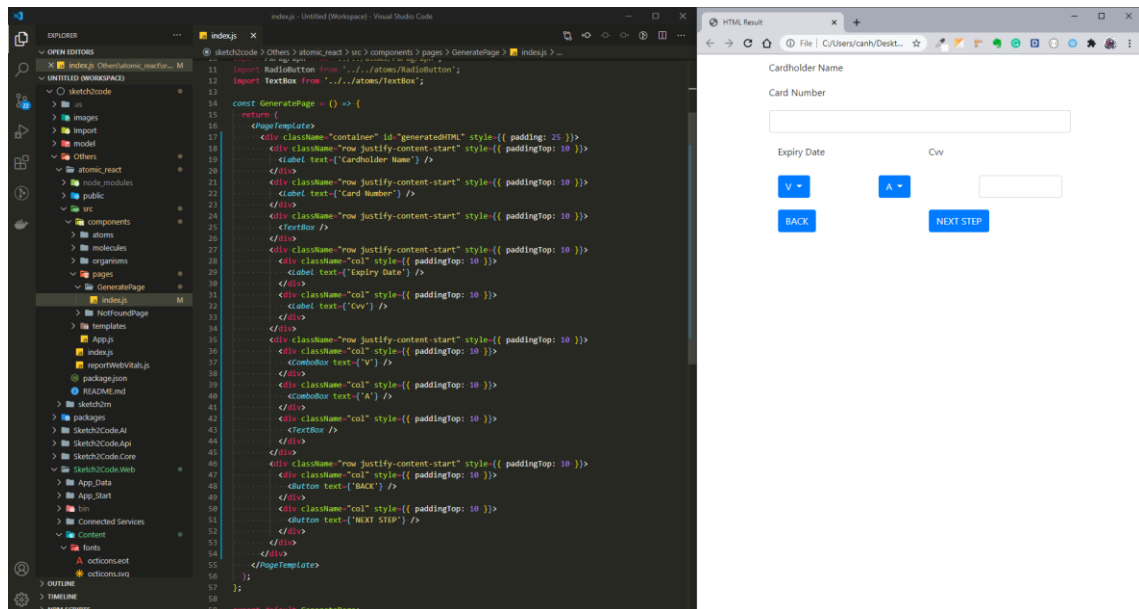


```

</div>
<div className="col" style={{ paddingTop: 10 }}>
  <Label text={'Cvv'} />
</div>
</div>
<div className="row justify-content-start" style={{ paddingTop: 10 }}>
  <div className="col" style={{ paddingTop: 10 }}>
    <ComboBox text={'V'} />
  </div>
  <div className="col" style={{ paddingTop: 10 }}>
    <ComboBox text={'A'} />
  </div>
  <div className="col" style={{ paddingTop: 10 }}>
    <TextBox />
  </div>
</div>
<div className="row justify-content-start" style={{ paddingTop: 10 }}>
  <div className="col" style={{ paddingTop: 10 }}>
    <Button text={'BACK'} />
  </div>
  <div className="col" style={{ paddingTop: 10 }}>
    <Button text={'NEXT STEP'} />
  </div>
</div>
</PageTemplate>
);
};
export default GeneratePage;

```

Hình 4.21 Mã code Reactjs thiết kế theo module Atomic



Hình 4.22 Mã code JSX và giao diện tương ứng

Trang web hiện tại đang hiển thị là sự kết hợp của các atom, molecules, ... tạo thành một page giao diện.

ReactNative

Mỗi loại đối tượng phần tử sẽ được xây dựng tương đương là một component.

Mã code sinh là tương ứng là một View trong ứng dụng Android/iOS là kết hợp các component con và các thông số khác.

```
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';
import Button from '../components/Button';
import CheckBox from '../components/CheckBox';
import ComboBox from '../components/ComboBox';
import Heading from '../components/Heading';
import Image from '../components/Image';
import Label from '../components/Label';
import Link from '../components/Link';
import Paragraph from '../components/Paragraph';
import RadioButton from '../components/RadioButton';
import TextBox from '../components/TextBox';
const GeneratedPage = () => {
  return (
    <View>
      <div className="container" id="generatedHTML" style={{padding: 25}}>
        <div style={styles.flexStart}>
          <Label text={'Cardholder Name'} />
        </div>
        <div style={styles.flexStart}>
          <Label text={'Card Number'} />
        </div>
        <div style={styles.flexStart}>
          <TextBox />
        </div>
        <div style={styles.flexStart}>
          <div style={noFlex}>
            <Label text={'Expiry Date'} />
          </div>
          <div style={noFlex}>
            <Label text={'Cvv'} />
          </div>
        </div>
        <div style={styles.flexStart}>
          <div style={noFlex}>
            <ComboBox text={'V'} width={131} height={66} />
          </div>
          <div style={noFlex}>
            <ComboBox text={'A'} width={153} height={79} />
          </div>
          <div style={noFlex}>
            <TextBox />
          </div>
        </div>
        <div style={styles.flexStart}>
          <div style={noFlex}>
            <Button text={'BACK'} />
          </div>
          <div style={noFlex}>
            <Button text={'NEXT STEP'} />
          </div>
        </div>
      </div>
    </View>
  );
};
```

```

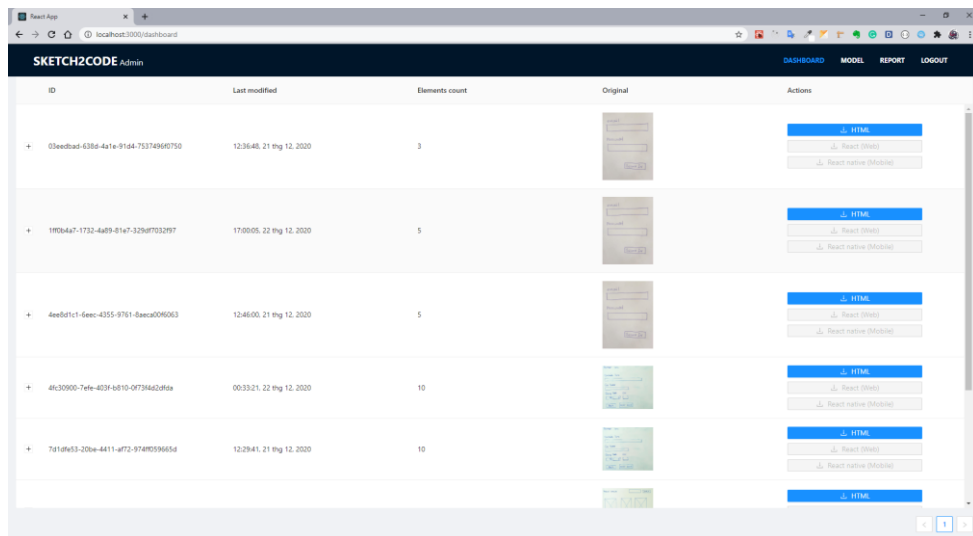
</View>
);
};
const styles = StyleSheet.create({
  noFlex: {},
  flexStart: {
    flexDirection: 'row',
    justifyContent: 'flex-start',
  },
  flexEnd: {
    flexDirection: 'row',
    justifyContent: 'flex-start',
  },
  flexCenter: {
    flexDirection: 'row',
    justifyContent: 'center',
  },
});
export default GeneratedPage;






```

Hình 4.23 File View chính trong ReactNative được sinh tự động

Hệ thống quản lý dữ liệu sinh mã code tự động

Hệ thống sẽ hiển thị danh sách các quá trình sinh mã code tự động. Nếu mã code đã được sinh ra, quản trị viên có thể tải về và chạy thử nghiệm.

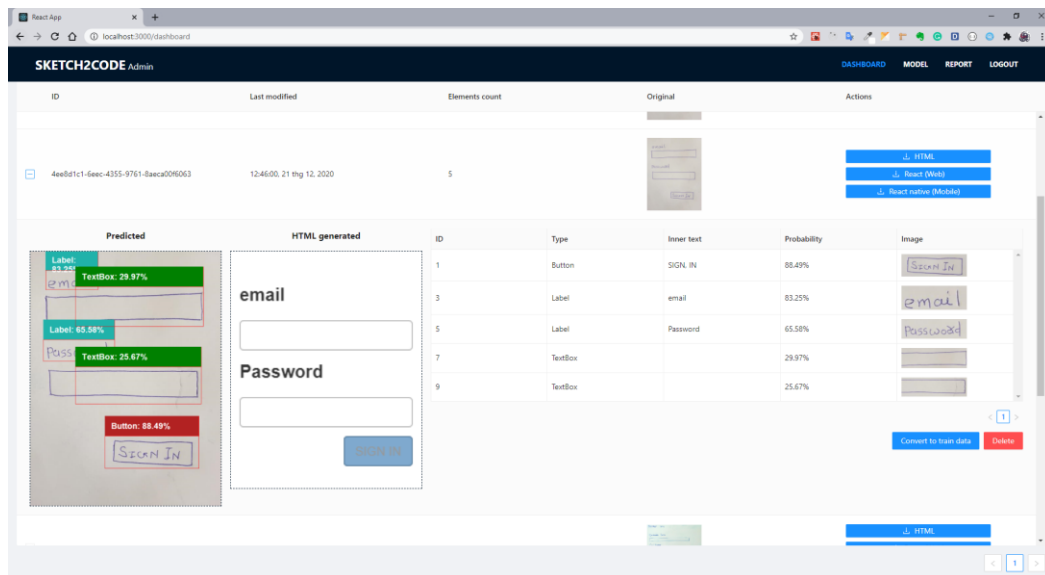


ID	Last modified	Elements count	Original	Actions
+ 03eedbad-638d-4a1e-91a4-7537496d750	12:36:40, 21 thg 12, 2020	3		HTML React (Web) React native (Mobile)
+ 1f0cb4e7-1732-4a09-81a7-329d70329f7	17:00:05, 22 thg 12, 2020	5		HTML React (Web) React native (Mobile)
+ 4eebd1c1-6ee0-4355-9761-6aeca096063	12:46:00, 21 thg 12, 2020	5		HTML React (Web) React native (Mobile)
+ 4fc30900-7efe-403f-b010-07386a2d8fa	00:33:21, 22 thg 12, 2020	10		HTML React (Web) React native (Mobile)
+ 7d1dfe53-20ba-4411-a772-9740599665d	12:29:41, 21 thg 12, 2020	10		HTML React (Web) React native (Mobile)

Hình 4.24 Giao diện danh sách lịch sử sinh mã code

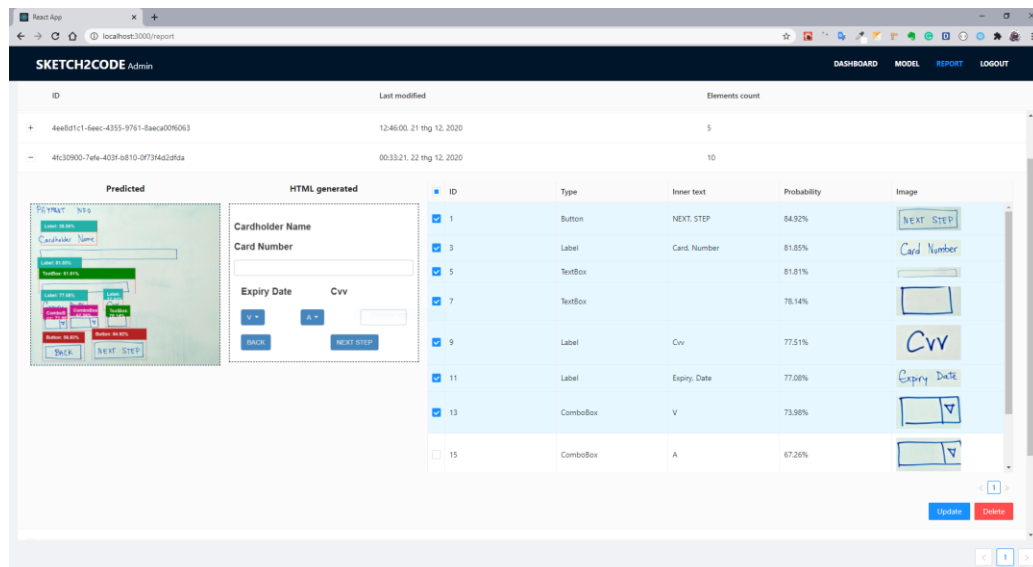
Quản trị viên ấn vào dấu “+” đầu mỗi cột để xem chi tiết dữ liệu phân tích và danh sách đối tượng của quá trình sinh code tự động.

Quản trị viên có thể xóa bản ghi hoặc chuyển bản ghi thành dữ liệu huấn luyện.



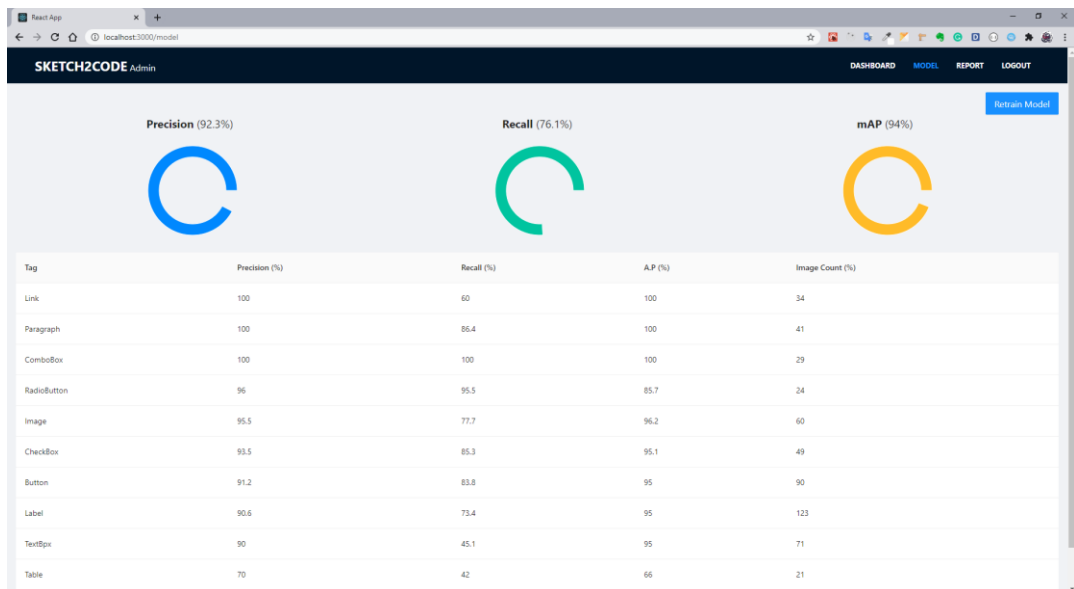
Hình 4.25 Giao diện chi tiết lịch sử sinh mã code

Quản lý phản hồi: Các đối tượng do người dùng chọn sẽ được sáng lên trong bảng đối tượng, người dùng lựa chọn có thể chưa chính xác. Quản trị viên có thể lựa chọn lại các object hợp lệ và cập nhật lại dữ liệu. Nếu không cần thiết, quản trị viên có thể xóa bản ghi này.



Hình 4.26 Giao diện quản lý phản hồi người dùng

Quản lý mô hình huấn luyện: Hiển thị dữ liệu và thông số liên quan đến hiệu năng của mô hình dự đoán. Quản trị viên có thể kích hoạt tái huấn luyện mô hình.



Hình 4.27 Giao diện quản lý mô hình dự đoán

CHƯƠNG 5. CÁC ĐÓNG GÓP NỔI BẬT

Chương này sẽ giới thiệu về các chức năng chính và đóng góp nổi bật của hệ thống. Thông qua việc áp dụng thành quả của kỹ thuật học máy vào bài toán sinh mã code cho ứng dụng web và ứng dụng di động. Đồng thời chương này nêu lên những đóng góp mở rộng từ bài toán Sketch2code.

5.1 Xây dựng cấu trúc cây HTML

Khi chúng ta thiết kế một bản thiết kế lên giấy, công việc của chúng ta là bắt đầu xây dựng layout chung của trang web, chia nhỏ trang web thành các thành phần chính rồi mới tiến hành đi vào thiết kế cho từng thành phần con. Tương tự như cách con người làm việc, để máy tính có thể hiểu được bản vẽ, chúng ta phải cung cấp cho nó một tài liệu hướng dẫn đọc để nó có thể làm theo. Tài liệu ở đây chính là cây cấu trúc HTML. Khi trình duyệt tiến hành duyệt cây, chính là việc nó truy cập qua tất cả các nút của cây và sinh ra mã code cho các nút này. Để có thể duyệt cây bao phủ toàn bộ các nút cũng như không có nút nào bị lặp lại, việc tiến hành duyệt cây từ nút gốc sẽ dễ dàng hơn rất nhiều so với duyệt từ vị trí bất kì.

Từ lớp đối tượng do mô hình huấn luyện trả về, các đối tượng được đặt tại các vị trí ngẫu nhiên, dựa trên thiết kế của bản vẽ. Vì thế để dễ dàng trong tính toán, chúng ta sẽ sử dụng đối tượng gốc trên-trái làm nút gốc trong quá trình duyệt cây.

Ta có thuật toán đệ quy sau:

Data: Các phần tử được sắp xếp theo vùng tăng dần: $E = (x_0, y_0, w_0, h_0, []), \dots, (x_n, y_n, w_n, h_n, [])$,

$sectioned \leftarrow []$;

for $element_1$ **in** E **do**

$group \leftarrow []$;

for $element_2$ **in** $sectioned$ **do**

if $element_1$ contains $element_2$ **then**

$group.push(element_2)$;

$sectioned.remove(element_2)$;

end

$element_1.group \leftarrow group$;

end

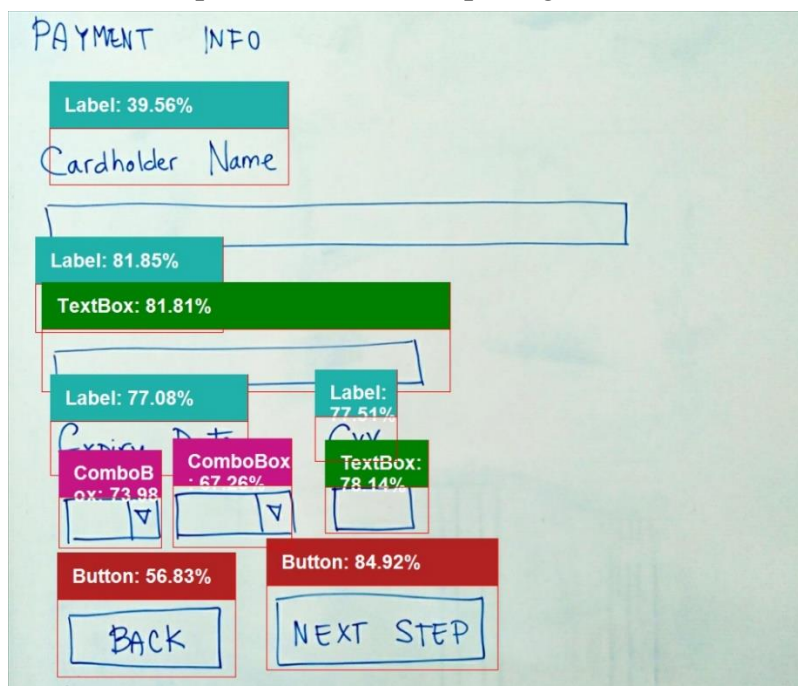
$sectioned.append \leftarrow element_1$;

end

Trong đó: + (x_x, y_x) là tọa độ đỉnh trên-trái của boundingBox của đối tượng X
 + w_x là chiều dài của boundingBox đối tượng X
 + h_x là chiều cao của boundingBox đối tượng X

Thuật toán đệ quy trên xây dựng cây HTML dựa trên tọa độ và giới hạn của các boundingBox bằng cách nhóm đệ quy các đối tượng lồng nhau bên trong một đối tượng khác thành một nhóm.

Dưới đây là ví dụ về kết quả của thuật toán áp dụng cho hình ảnh thực tế:



Hình 5.1 Tọa độ và kích thước boundingBox của các đối tượng



Hình 5.2 Biểu đồ dữ liệu xuất phát từ nút gốc và từ Group [6]

Sau quá trình đệ quy, kết quả của thuật toán đưa ra được cây cấu trúc bao gồm:

- + Các đối tượng nằm trên cùng một hàng ngang sẽ được nhóm vào 1 nhóm.
- + Các đối tượng con được tiếp tục xử lý đệ quy để tạo thành group bao gồm các đối tượng có bậc thấp hơn và nằm trong đối tượng con.
- + Quá trình đệ quy kết thúc khi tất cả các đối tượng được thêm vào cây cấu trúc.

Với hệ thống sinh mã code dựa trên bản vẽ thiết kế có độ tin cậy cao như mô hình pix2code hay sketchcode, hệ thống thường lý tưởng hoá các đối tượng xuất hiện trong bản thiết kế. Các đối tượng cùng cấp luôn có kích thước tương tự nhau, các thuộc tính như chiều cao và chiều rộng bị bỏ qua. Vì thế các group luôn được xếp song song nhau nếu duyệt đối tượng theo chiều từ trên xuống dưới.



Hình 5.3 Bản vẽ thiết kế huấn luyện cho mô hình sketchcode và pix2code

Ở bản vẽ bên phải, dữ liệu huấn luyện mô hình pix2code, các đối tượng được sắp xếp đều đặn, việc áp dụng thuật toán đệ quy để chia nhóm các đối tượng tương đối dễ dàng

Đối với bản vẽ bên trái, dữ liệu huấn luyện mô hình sketchcode, các đối tượng có vẻ giống như bản vẽ thiết kế có độ tin cậy thấp hơn. Bản vẽ bên trái là kết quả của quy trình tạo dữ liệu huấn luyện của tác giả dựa trên tập dữ liệu pix2code (ví dụ bên phải) sử dụng các kỹ thuật như:

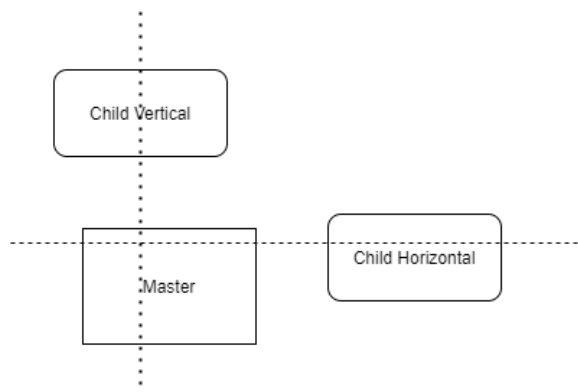
- Thay đổi bán kính đường viền của các phần tử để làm cong các nút và các thẻ div
- Điều chỉnh độ dày của đường viền bất chước các bản phác thảo vẽ bằng tay
- Thay đổi phông chữ giống như chữ viết tay.

Vì thế, việc xác định vị trí và kích thước và vị trí của các boundingBox vẫn được bỏ qua, dữ liệu vẫn rất khác so với bài toán thực tế.

Đối với bài toán áp dụng trên bản vẽ thiết kế có độ tin cậy thấp, kích thước và vị trí của boundingBox khi so sánh các đối tượng với nhau là rất khác, các đối tượng có thể xen kẽ nhau theo một chiều duyệt dẫn đến nhập nhằng trong việc chọn nhóm. Vì thế, chúng ta sẽ xây dựng các phép so sánh tương đối, phục vụ cho quá trình phân chia nhóm.

❖ Các thuật toán liên quan hỗ trợ so sánh các đối tượng

- So sánh vị trí của các đối tượng theo các chiều



Hình 5.4 So sánh các vị trí các đối tượng theo chiều ngang và chiều dọc

Các phần tử con sẽ thuộc phạm vi của phần tử cha nếu đường thẳng trung vị của nó cắt qua phần tử trung vị:

Xét theo chiều ngang: child thuộc phạm vi của master nếu:

$\text{child.MiddleHeight} > \text{master.Top} \ \&\& \ \text{child.MiddleHeight} < (\text{master.Top} + \text{master.Height})$

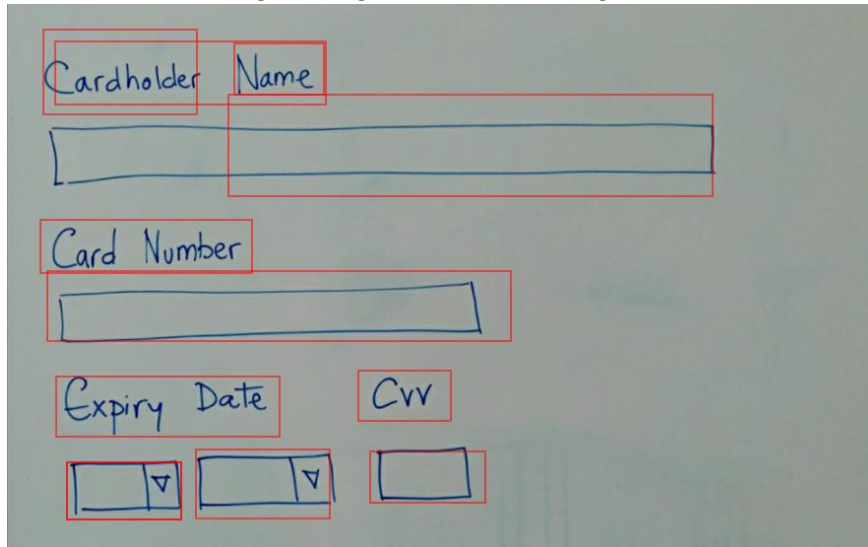
Trong đó: $\text{MiddleHeight} = | \text{child.Top} - \text{child.Height} | / 2$

Xét theo chiều dọc: child thuộc phạm vi của master nếu:

$\text{child.MiddleWidth} > \text{master.Left} \ \&\& \ \text{child.MiddleWidth} < (\text{master.Left} + \text{master.Width});$

Trong đó: $\text{MiddleWidth} = | \text{child.Left} + \text{child.Width} | / 2$

- Xác định độ chồng chéo giữa các đối tượng

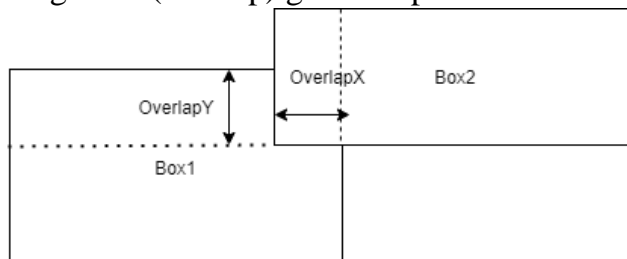


Hình 5.5 Các đối tượng chồng chéo lẫn nhau

Trong quá trình xác định các đối tượng, mô hình có thể đưa ra dự đoán sai về boundingBox của các đối tượng cũng như vị trí của boundingBox dẫn đến tình trạng, tại mỗi đối tượng trong bản vẽ, mô hình dự đoán ra nhiều đối tượng khác nhau có xác suất khác nhau. Như hình trên, nhãn “Cardholder Name” được xác định bởi 2 đối tượng có boundingBox khác nhau. Chúng ta sẽ loại bỏ đi những đối tượng có xác suất dự đoán thấp hơn.

Mặt khác, do xác định sai kích thước boundingBox, hệ thống có thể đưa ra các đối tượng có boundingBox chồng chéo một phần lên nhau, trong trường hợp này chúng ta không thể xóa bất kỳ đối tượng nào, thay vào đó sẽ dịch chuyển đối tượng chồng chéo này ra xa nhau.

- a. Xác định độ chồng chéo (overlap) giữa các phần tử với nhau



Hình 5.6 Độ chồng chéo của các đối tượng

- OverlapX: tỉ lệ chồng chéo theo trục X

$$\text{OverlapX} = \text{Max}(0, \text{Min}(b1_left + b1_width, b2_left + b2_width) - \text{Max}(b1_left, b2_left))$$

- OverlapY: độ chồng chéo theo trục Y

$$\text{OverlapY} = \text{Max}(0, \text{Min}(b1_top + b1_height, b2_top + b2_height) - \text{Max}(b1_top, b2_top))$$

- OverlapArea: độ chồng chéo dựa trên diện tích

$$\text{OverlapArea} = \text{OverlapX} * \text{OverlapY}$$

Loại bỏ các đối tượng ghi đè lên nhau:

- Với 2 đối tượng có diện tích ghi đè lớn hơn 75% diện tích đối tượng nhỏ hơn, chúng ta sẽ loại bỏ đi đối tượng có xác suất dự đoán thấp hơn.
- Nếu 2 đối tượng có độ ghi đè nhỏ hơn 25% so với diện tích đối tượng nhỏ hơn thì sẽ giữ nguyên 2 đối tượng
- Ngược lại, dịch chuyển hai đối tượng ra xa nhau theo trục Ox một đoạn $\text{OverlapX} / 2$ và theo trục Y một đoạn $\text{OverlapY} / 2$

Từ đó, chúng ta giải quyết được vật đè ghi đè giữa các đối tượng dự đoán với nhau.

5.2 Sinh mã code dựa trên cấu trúc cây HTML

5.2.1 Quy trình sinh mã code từ cấu trúc HTML

Từ dữ liệu phân chia các đối tượng thành các group lồng nhau, chúng ta sẽ sử dụng thuật toán đệ quy cho việc triển khai sinh mã code như sau

Data: Object thể hiện nút gốc R (R là nút gốc khởi tạo ban đầu trong quá trình tạo cây cấu trúc HTML)

Xét tại nút gốc R:

code = “”;

If R.isEmpty **then End**;

If R.Groups.length > 0 **then**

//Tạo layout cho group dựa trên R.direction (1);

layout = <div></div>;

For g **in** R.Groups **do**:

child = Gọi đệ quy với nút gốc là g; (2)

layout.child.append ← child;

End;

code.append ← layout

End;

Else do

element = <R.boxes.type></R.boxes.type>; (3)

```

element.className ← R.boxes.Alignment;
element.style.width ← R.boxes.width;
element.style.height ← R.boxes.height;
element.innerText ← R.boxes.text;
code.append ← element;

```

End

Trong đó:

- Nút R chỉ có thể là element dạng lá hoặc là element dạng nút để thể hiện layout. Nếu R.boxes tồn tại thì R sẽ là nút lá, ngược lại R sẽ là nút và hiển thị layout
- Tại bước (1) dựa trên chiều duyệt đối tượng sinh mã code chúng ta sẽ tiến hành tạo khung layout tương ứng cho chiều hiển thị nội dung bên trong, Ví dụ: kết hợp tên class tương ứng với cú pháp CSS:
 - “*display: flex; flex-direction: column;*” để hiển thị nội dung theo chiều dọc
 - “*display: flex; flex-direction: row;*” để hiển thị nội dung theo chiều ngang
- Tại bước (2) chúng ta sẽ gọi đệ quy tới chính phương trình này với nút gốc là nút đang xét tới, sau khi xây dựng xong cấu trúc cho nhánh này, chúng ta sẽ gán nó vào trong layout xây dựng ở bước 1.
- Tại bước (3), khi đó R sẽ là một nút lá, dựa trên các thuộc tính mà mô hình trả về, chúng ta sẽ xây dựng nên element tương ứng.

❖ Một số phương thức hỗ trợ trong quá trình sinh thuật toán.

Trong bản thiết kế thực tế, các đối tượng cùng loại trong thiết kế khi đặt trong cùng một phạm vi ngữ nghĩa thường có những tham số cấu hình giống nhau: như padding, margin, width, height, ... Việc đồng bộ hoá này là bắt buộc đối với các hệ thống thực tế. Nó đem lại cảm giác đồng bộ, đơn giản và nhất thống cho hệ thống giúp mang lại trải nghiệm tốt hơn.

Tuy nhiên, theo quan điểm cá nhân, dù quá trình đồng bộ có thể dẫn đến tính nhất quán, tuy nhiên chúng ta chỉ nên dừng lại ở mức độ trong cùng một nhóm. Đối với các phân tử thuộc nhóm khác, việc đồng bộ có thể gây sai lệch so với ý định thiết kế ban đầu.

AlignTop(boxes): căn lề trên cho các đối tượng truyền vào.

AlignLeft(boxes): căn lề trái cho các đối tượng truyền vào.

MakeSameSize(boxes): cập nhật kích thước của các đối tượng cùng loại được truyền vào dựa trên giá trị trung bình của các đối tượng này.

5.2.2 Sử dụng cú pháp Razor cho khởi tạo file

Sau khi chuẩn hoá các element và layout, giai đoạn cuối của hệ thống sinh mã code là sinh ra file code tương ứng theo nền tảng và ngôn ngữ lựa chọn.

- Đối với các element, chúng ta sẽ khởi tạo sẵn bộ từ điển tương ứng với các element đó.

- Đối với layout, đa phần chúng ta sẽ chỉ hiển thị trên một giao diện nhất định, vì thế chúng ta sẽ tạo ra 1 file layout kết hợp cùng với các element tương ứng

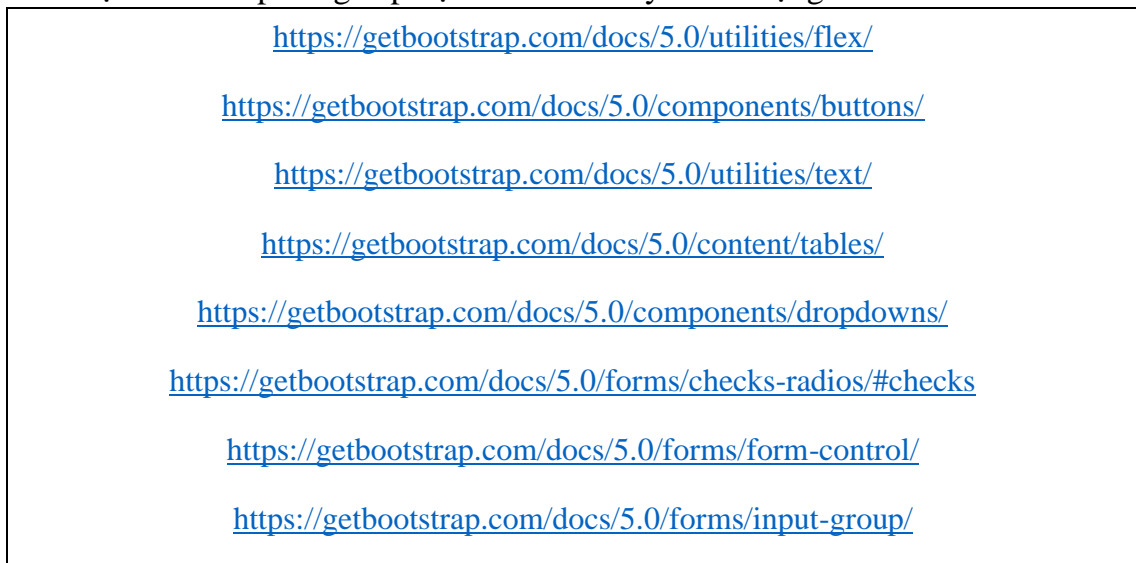
Dưới đây, chúng ta sẽ tiến hành tạo file cho các ngôn ngữ và nền tảng khác nhau.

5.2.3 Tạo code HTML và code Reactjs nhúng vào code HTML

Đối với code HTML thuần, tuy sử dụng các element có sẵn của HTML, nhưng để có thể thay đổi các thuộc tính của chúng, chúng ta cũng sẽ xây dựng bộ từ điển sinh element tương ứng.

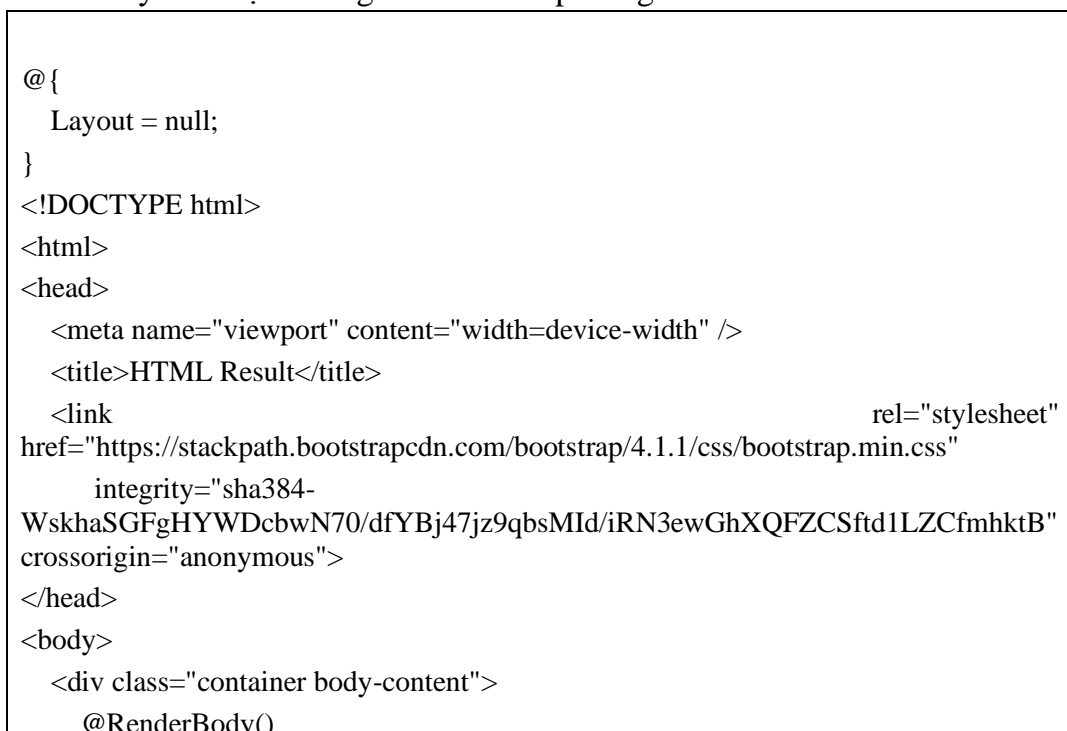
Giao diện sinh ra sử dụng thư viện Bootstrap thay vì tự triển khai mã CSS cho từng đối tượng.

Thư viện Bootstrap cung cấp bộ element và layout đa dạng:



Hình 5.7 Hệ thống element do Bootstrap cung cấp

❖ Layout được nhúng sẵn Bootstrap trong thẻ <head/>



```

</div>
</body>
</html>

```

Hình 5.8 Layout sinh mã code HTML

❖ Tương ứng với mỗi element là bộ sinh element tương ứng

```

@model Sketch2Code.Core.Entities.PredictedObject
<div class="dropdown">
  <button type="button" class="btn btn-primary dropdown-toggle" data-
toggle="dropdown">
    @String.Join(" ", Model.Text)
  </button>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="#"></a>
  </div>
</div>

```

Hình 5.9 Model tạo element Dropdown

Đối với mã code Reactjs được nhúng trực tiếp tiếp vào file HTML. React hỗ trợ sử dụng thư viện thông qua hệ thống CDN. Mã code React sinh ra sẽ được nhúng trực tiếp HTML thông qua thẻ script. Khi hệ thống rendering, mã code react sẽ được bundle thành html/js/css và được gán vào element “root” thông qua DOM.

```

@{
  Layout = null;
}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width" />
  <title>React Result</title>
  <link
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
    integrity="sha384-
WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMIId/iRN3ewGhXQFZCSftd1LZCfmhktB"
    crossorigin="anonymous">
  </head>
<body>
  <div id="root" dir="auto"></div>

  <script type="text/babel">
    function App() {
      return (
        <div class="container body-content">
          @RenderBody()
        </div>
      )
    }
  </script>

```

```

    }
    ReactDOM.render(
      <App />,
      document.getElementById('root')
    );
  </script>
  <script      src="https://unpkg.com/react@16.13.1/umd/react.production.min.js"
crossorigin="anonymous"></script>
  <script      src="https://unpkg.com/react-dom@16.13.1/umd/react-dom.production.min.js"
crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/dataformsjs@4.0.1/js/react/jsxLoader.min.js"></script>
</body>
</html>

```

Hình 5.10 Layout file HTML nhúng ReactJs

5.2.4 Tạo code Reactjs theo kiến trúc Atomic

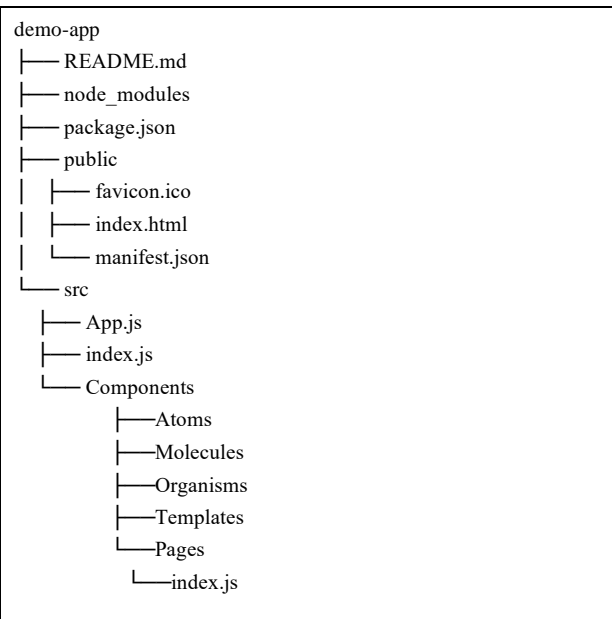
Trong kiến trúc Atomic, Atoms là thành phần nhỏ nhất, những block cơ bản nhất và không thể nhỏ hơn nữa (ví dụ: buttons, input fields, checkboxes, links). Chúng cũng có thể trừ tượng như color, font, ... Vì thế, tương ứng với các element dạng lá, chúng ta sẽ xây dựng các component atom tương ứng.

Molecules: Gồm các atom kết hợp vs nhau là các phần tử bên ngoài như đơn vị (ví dụ: một input field và một button có thể kết hợp thành một khung tìm kiếm). Sự kết hợp giữa các atom với nhau có thể tạo ra cấp số nhân các component molecules. Vì thế chúng ta sẽ chỉ hướng tới xây dựng những đối tượng phổ biến như: khung tìm kiếm.

Với các thành phần như Organisms và Templates, việc phân chia các component này phụ thuộc vào chủ đích người xây dựng phần mềm, mỗi cá nhân sẽ có một quan điểm xây dựng khác nhau, vì thế chúng ta sẽ không tập trung triển khai các module này.

Module page là module chúng ta sử dụng để hiển thị giao diện chính. Đây là nơi chúng ta kết hợp các component con lại với nhau tạo thành 1 thiết kế hoàn chỉnh

Kiến trúc mã nguồn code được sinh ra dựa trên cơ sở của tiện ích create-react-app



Hình 5.11 Kiến trúc mã nguồn Reactjs theo thiết kế Atomic

```

@{
  Layout = null;
}
import React,{Fragment} from 'react';
import PageTemplate from '../templates/PageTemplate';
import Button from '../atoms/Button';
import CheckBox from '../atoms/CheckBox';
import ComboBox from '../atoms/ComboBox';
import Heading from '../atoms/Heading';
import Image from '../atoms/Image';
import Label from '../atoms/Label';
import Link from '../atoms/Link';
import Paragraph from '../atoms/Paragraph';
import RadioButton from '../atoms/RadioButton';
import TextBox from '../atoms/TextBox';
import Heading from '../molecules/SearchBox';

const GeneratePage = () => {
return (
  <PageTemplate>
    @RenderBody()
  </PageTemplate>
);
};
export default GeneratePage;

```

Hình 5.12 Layout file Page/index.js trong Reactjs Atomic

```

import React from 'react';

const ComboBox = ({
  text = 'Dropdown button',
  items = [
    { key: 'Action', value: 'Action' },
    { key: 'Another action', value: 'Another action' },
    { key: 'Something else here', value: 'Something else here' },
  ],

```

```

    }) => {
      return (
        <div class="dropdown">
          <button
            class="btn btn-secondary dropdown-toggle"
            type="button"
            id="dropdownMenuButton"
            data-toggle="dropdown"
            aria-haspopup="true"
            aria-expanded="false"
          >
            {text}
          </button>
          <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
            {items.map((item) => (
              <a class="dropdown-item" key={item.value} href="#">
                {item.key}
              </a>
            ))}
          </div>
        </div>
      );
    };
  };
export default ComboBox;

```

Hình 5.13 Component Atom cho Dropdown

```

@model Sketch2Code.Core.Entities.PredictedObject
<ComboBox text={"@String.Join(" ", Model.Text)"} />

```

Hình 5.14 Model render Dropdown tương ứng

Sau khi mã code tạo ra, hệ thống sẽ sử dụng plugin Eslint kết hợp với Prettier để xóa bỏ code dư thừa và format code.

Để kiểm nghiệm ứng dụng, người dùng sẽ tải source code và cài đặt theo hướng dẫn tại file README.md

5.2.5 Tạo code Reactnative cho nền tảng di động

So với Reactjs, việc xử lý và sinh mã code mã code React Native không có quá nhiều khác biệt. Tuy nhiên, trong React Native, các element chưa đa dạng như Reactjs, để triển khai các element này, chúng ta sẽ sử dụng các thư viện nhỏ cung cấp element tương tự.

Ngoài ra, ReactNative nhằm mục đích xây dựng ứng dụng di động cũng như không hỗ trợ Bootstrap. Styling ứng dụng sẽ được thiết kế bằng tay thông qua tên class và mã CSS

```

@{
  Layout = null;
}
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';
import Button from '../components/Button';
import CheckBox from '../components/CheckBox';

```

```

import ComboBox from '../components/ComboBox';
import Heading from '../components/Heading';
import Image from '../components/Image';
import Label from '../components/Label';
import Link from '../components/Link';
import Paragraph from '../components/Paragraph';
import RadioButton from '../components/RadioButton';
import TextBox from '../components/TextBox';
const GeneratedPage = () => {
  return (
    <View>
      @RenderBody()
    </View>
  );
};
const styles = StyleSheet.create({
  noFlex: {},
  flexStart: {
    flexDirection: 'row',
    justifyContent: 'flex-start',
  },
  flexEnd: {
    flexDirection: 'row',
    justifyContent: 'flex-start',
  },
  flexCenter: {
    flexDirection: 'row',
    justifyContent: 'center',
  },
});
export default GeneratedPage;

```

Hình 5.15 Layout file View/index.js trong ReactNative

```

import React, {useEffect, useState} from 'react';
import {Picker} from '@react-native-picker/picker';

const ComboBoxComponent = ({text, width = 200, height = 50}) => {
  const [value, setValue] = useState("Option 1");
  useEffect(() => {
    if (text) {
      setValue(text);
    }
  }, [text]);
  return (
    <Picker
      selectedValue={value}
      style={{height: height, width: width}}
      onValueChange={(itemValue) => setValue(itemValue)}>
      <Picker.Item label="text" value={'text'} />
      <Picker.Item label="Option 1" value={'Option 1'} />
      <Picker.Item label="Option 2" value={'Option 2'} />
    </Picker>
  );
};
export default ComboBoxComponent;

```

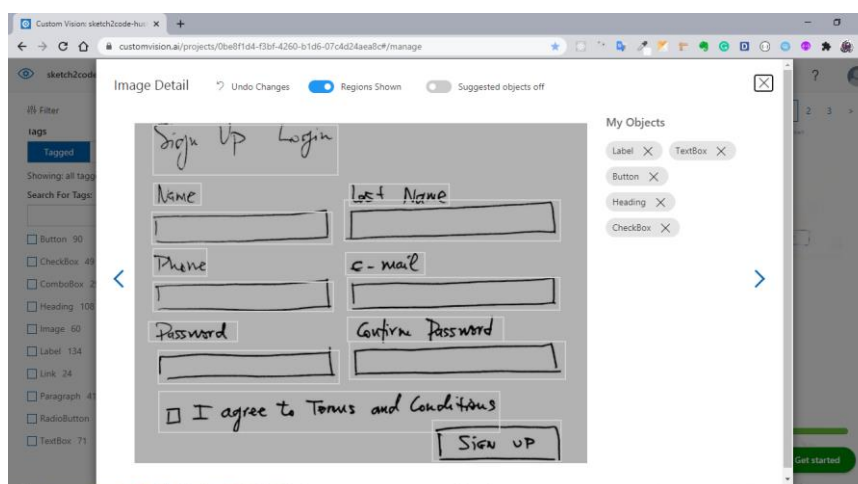
5.3 Mở rộng đối tượng nhận dạng

❖ Mở rộng dữ liệu huấn luyện

Hiện tại hệ thống sketch2code đang hỗ trợ nhận dạng và sinh mã code HTML dựa trên 5 element dạng lá cơ bản: title, image, button, input, paragraph. Để tiếp tục mở rộng bài toán này, chúng ta triển khai nhận dạng và sinh mã code cho các đối tượng sau:

- + Link
- + Textarea
- + Carousel
- + Table
- + Pagination
- + Form

Các đối tượng mở rộng được em đánh nhãn trực tiếp thông qua dịch vụ Azure Computer vision.



Hình 5.17 Công cụ đánh nhãn đối tượng của Azure

Mô hình ban đầu được Microsoft cung cấp bao gồm 195 bức ảnh chụp wireframe, để mở rộng dữ liệu huấn luyện mô hình, đồng nghĩa với việc nâng cao hiệu năng của mô hình, em đã áp dụng một số phương pháp sau:

❖ Xây dựng ứng dụng quản lý lịch sử sinh mã code

Kết quả dự đoán của mô hình sau khi được xác thực lại sẽ chuyển vào trong tập dữ liệu huấn luyện.

Hệ thống này đã được thảo luận và xây dựng ở các phần trên

❖ Áp dụng các kỹ thuật Tăng cường dữ liệu (Data augmentation) từ đó làm tăng dữ liệu huấn luyện cho mô hình như: xoay, cắt góc ngẫu nhiên, thay đổi màu.

Hình ảnh sau khi chỉnh sửa được đẩy vào mô hình dự đoán trực tiếp, sau đó sẽ tiến hành kiểm tra và đánh nhãn lại bằng tay.

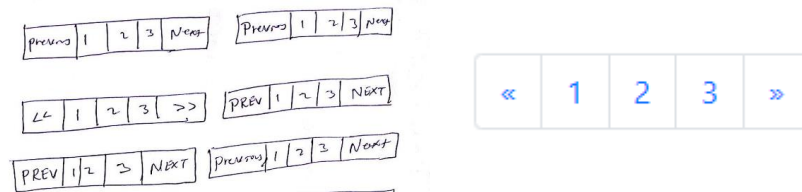
❖ Tự thiết kế các bản vẽ bằng tay

Phương pháp này không đem lại năng suất cao theo số lượng, tuy nhiên, chúng ta có thể kiểm tra thực nghiệm kết quả nhận dạng của mô hình.

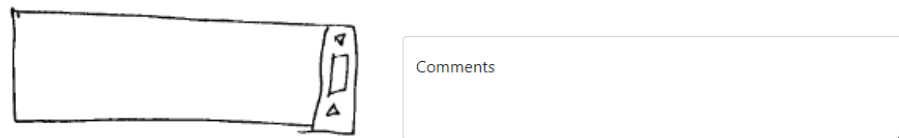
Sau các phương pháp trên, hiện tại mô hình đang được huấn luyện trên tập dữ liệu hơn 450 ảnh. Tuy nhiên chất lượng nhãn dán chưa cao.

❖ Mở rộng phương pháp sinh mã code cho các phần tử mới

Đối với các phần tử đơn giản dạng lá như: Link, RadioButton, CheckBox, TextArea, Carousel, Pagination cách triển khai được áp dụng như các element có sẵn.



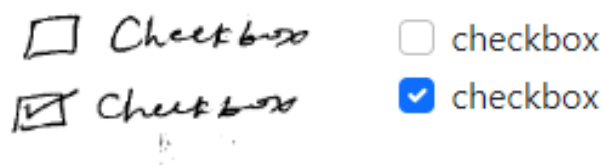
Hình 5.18 So sánh wireframe và mã code của pagination



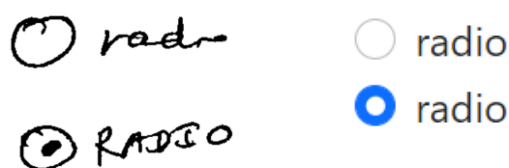
Hình 5.19 So sánh wireframe và mã code của textarea



Hình 5.20 So sánh wireframe và mã code của carousel



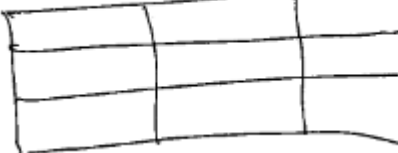
Hình 5.21 So sánh wireframe và mã code của checkbox



Hình 5.22 So sánh wireframe và mã code của radio button

Tuy nhiên khi triển khai sinh mã code cho các đối tượng dạng nút như table hay form, chúng ta không thể áp dụng như các element dạng lá được. Đối với các element này, khi triển khai cây HTML, 1 table bản thân nó vừa là một boxes đồng thời cũng là 1 group chứa các phần tử bên trong nó. Điều này dẫn đến mâu thuẫn trong quy trình xây dựng cây cấu trúc HTML.

Subtotal	2 items	\$109.00
Promo Code <u>Edit</u>	10% OFF	-\$8.00
TOTAL		\$101.00



Hình 5.23 Hai dạng hiển thị table trong wireframe

Với dạng hiển thị bên phải, chúng ta không quan tâm tới nội dung bên trong của bảng mà chỉ tập trung vào nhận dạng đối tượng bảng. Đối tượng bảng ở đây được coi như là một nút lá.

Với dạng hiển thị bên phải, nội dung bên trong các cột không thể bỏ qua, vì thế bảng ở đây được coi là một nút, bên trong nút có các cột chứa label.

Với cái thuật toán hỗ trợ trong việc phân nhóm các element, các element này đều được đẩy vào luồng ngoại lệ. Ví dụ như bài toán chống ghi đè, xét trên một element table, các thành phần dữ liệu sẽ bị mất hoàn toàn sau khi áp dụng bài toán này.

5.4 Kết quả đạt được

Trong chương này, chúng ta đã đạt được một số thành quả nhất định cũng như phát hiện ra một số điểm nghẽn khi tiếp tục mở rộng bài toán:

- Dữ liệu huấn luyện mô hình có thể tăng theo nhiều phương pháp khác nhau, từ đó làm tăng hiệu năng của mô hình
- Mô hình đã có thể sinh ra mã code cho các nền tảng khác nhau như ứng dụng web viết bằng Reactjs hay ứng dụng di động viết bằng Reactnative.
- Vấn đề sinh mã code cho các element dạng nút đang là trở ngại lớn.

CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận

Trong quá trình làm đồ án tốt nghiệp, em đã kết thừa hệ thống Sketch2code để xây dựng hệ thống sinh mã code riêng với nhiều tính năng hơn và chất lượng hơn. Hệ thống có giao diện đơn giản dễ sử dụng có tính thực tế cao hơn so với phiên bản gốc. Hệ thống có các tính năng nổi bật:

- Hỗ trợ xác định và sinh mã code cho nhiều đối tượng HTML hơn.
- Hỗ trợ sinh mã code ReactJs theo thiết kế Atomic giúp giảm thời gian và chi phí phát triển ứng dụng trên nền tảng web
- Hỗ trợ sinh mã code ReactNative giúp xây dựng ứng dụng trên nền tảng di động bao gồm Android và IOS
- Hệ thống có hiệu năng dự đoán tốt hơn nhờ chức năng quản lý dữ liệu sinh mã code, giúp hệ thống ngày càng tốt lên.

Bên cạnh những điểm nổi bật trên, em cũng nhận thấy hệ thống có nhiều hạn chế:

- Hệ thống sử dụng sẵn mô hình huấn luyện bên thứ 3 thay vì tự phát triển mô hình
- Dữ liệu huấn luyện chưa được kiểm soát hoàn toàn, do số lượng các element tăng lên, đồng nghĩa với việc các dữ liệu huấn luyện cũ sẽ bị sai sót hoặc thiếu gây ảnh hưởng tới kết quả dự đoán.
- Hệ thống element vẫn đang còn hạn chế so với thực tế, đặc biệt vấn đề về các element dạng lá chưa được giải quyết.

Đồng thời qua đồ án, em đã học hỏi thêm được rất nhiều kinh nghiệm cho bản thân:

- Thiết kế giao diện vô cùng quan trọng, thiết kế đẹp và có sự thông nhất toàn cục trong trang web sẽ gây ấn tượng và giúp người dùng trải nghiệm tốt hơn.
- Các mô hình trí tuệ nhân tạo chỉ đem lại kết quả khi được áp dụng vào việc giải quyết các vấn đề thực tế
- Phương pháp quản lý thời gian tốt giúp tối ưu hiệu suất học và làm việc.

6.2 Giải pháp và hướng phát triển

Để hoàn thành đồ án này, em đã tìm hiểu và học hỏi được rất nhiều kiến thức mới như:

- Nắm vững các công nghệ phát triển ứng dụng web và ứng dụng di động như: APS.NET core, Reactjs, Nodejs, React native, ...
- Kiến thức cơ bản về học máy, đặc biệt là bài toán Xác định vật thể và làm quen với một số mô hình phổ biến như ANN, CNN, LSTM
- Áp dụng trí tuệ nhân tạo vào giải quyết vấn đề thực tế như thế nào, gặp phải những khó khăn gì trong việc phát triển sản phẩm.
- Nâng cao kỹ năng học hỏi từ các bài nghiên cứu nổi tiếng trên thế giới.

Trong phạm vi đồ án, do thời gian có hạn và kiến thức còn tương đối hạn chế nên em xin phép đề xuất hướng phát triển tiếp theo cho hệ thống như sau:

- Tiếp tục nghiên cứu bài toán sinh mã code tự động cho các phần tử HTML mới, đặc biệt là các phần tử dạng nút
- Thêm tính năng sinh mã code cho các nền tảng ứng dụng cũng như công nghệ khác
- Xây dựng hệ thống quản lý người dùng từ đó làm tăng hiệu quả xác thực dữ liệu huấn luyện
- Tự xây dựng mô hình huấn luyện riêng

Thông qua quá trình làm đồ án, dưới sự hướng dẫn của PGS.TS Cao Tuấn Dũng, em đã xây dựng được hệ thống sinh mã code tự động cho các nền tảng ứng dụng khác nhau. Đồng thời em học thêm được nhiều kiến thức mới, tăng khả năng phân tích, tìm hiểu, nghiên cứu và giải quyết vấn đề. Dưới vai trò là một lập trình viên Full-stack, em mong rằng hệ thống này sẽ tiếp tục phát huy hơn nữa từ đó giúp ích cho chính bản thân mình cũng như người trong ngành Công nghệ thông tin nói chung.

TÀI LIỆU THAM KHẢO

- [1] A. Robinson, sketch2code: Generating a website from a paper, 2018.
- [2] T. Beltramelli, pix2code: Generating Code from a Graphical User Interface Screenshot, 2017.
- [3] A. Kumar, "Automated front-end development using deep learning," 2018.
- [4] B. Frost, "Atomic design," <https://bradfrost.com/blog/post/atomic-web-design/>.
- [5] "Reactjs," Facebook, [Online]. Available: <https://reactjs.org/>.
- [6] "React Native," Facebook, [Online]. Available: <https://reactnative.dev/>.
- [7] "ASP.NET," Microsoft, [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1>.
- [8] "Microsoft Azure," Microsoft, [Online]. Available: <https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/>.
- [9] "Machine learning cơ bản," [Trực tuyến]. Available: <https://machinelearningcoban.com>.
- [10] "Deep Learning cơ bản," [Online]. Available: <https://nttuan8.com/>.