TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

TÊN MÔN HỌC: THIẾT KẾ WEB VÀ TRIỂN KHAI HỆ THỐNG PHẦN MỀM

TÊN ĐỀ TÀI: WEBSITE KẾT NỐI BỆNH NHÂN VỚI BÁC SĨ CHUYÊN KHOA – MEDIALO

Giảng viên hướng dẫn: **Thầy Tạ Chí Hiếu**

Sinh viên thực hiện: **Lê Văn Đạt**

Hà Nội, 2025

TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

TÊN MÔN HỌC: THIẾT KẾ WEB VÀ TRIỂN KHAI HỆ THỐNG PHẦN MỀM

TÊN ĐỀ TÀI: WEBSITE KẾT NỐI BỆNH NHÂN VỚI BÁC SĨ CHUYÊN KHOA - MEDIALO

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bằng Số	Bằng Chữ
01	1971020090	Lê Văn Đạt	10/03/2007		

CÁN BỘ CHẤM THI 1 **CÁN BỘ CHẤM THI 2**

…… **Tạ Chí Hiếu**

Hà Nội, 2025

LỜI NÓI ĐẦU

Trong bối cảnh cuộc Cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ, công nghệ thông tin không chỉ thay đổi cách thức sản xuất, kinh doanh mà còn tạo ra những chuyển biến căn bản trong đời sống xã hội, đặc biệt là lĩnh vực y tế – một trong những ngành có vai trò then chốt đối với sức khỏe cộng đồng. Sự phát triển vượt bậc của internet, điện toán đám mây, trí tuệ nhân tạo và các nền tảng di động đã mở ra cơ hội to lớn để chuyển đổi số ngành y tế, giúp rút ngắn khoảng cách giữa bác sĩ và bệnh nhân, nâng cao chất lượng dịch vụ khám chữa bệnh và tối ưu hóa quy trình quản lý y tế.

Theo báo cáo của Bộ Y tế Việt Nam, trong những năm gần đây, nhu cầu tiếp cận dịch vụ y tế trực tuyến đã tăng gấp đôi, với hơn 60% bệnh nhân mong muốn được tư vấn và khám chữa bệnh qua mạng nhằm tiết kiệm thời gian, chi phí đi lại và giảm thiểu nguy cơ lây nhiễm chéo tại các cơ sở y tế. Đặc biệt, sau đại dịch COVID-19, xu hướng khám bệnh từ xa (telemedicine) đã trở thành một nhu cầu cấp thiết, không chỉ ở các thành phố lớn mà còn ở vùng sâu, vùng xa, nơi nguồn lực y tế còn hạn chế. Tuy nhiên, hệ thống y tế truyền thống của Việt Nam vẫn đang đối mặt với nhiều thách thức như tình trạng quá tải bệnh viện, khó khăn trong việc tìm kiếm bác sĩ chuyên khoa phù hợp, thiếu công cụ hỗ trợ đặt lịch hẹn hiệu quả, và đặc biệt là chưa có giải pháp lưu trữ hồ sơ bệnh án điện tử an toàn, thuận tiện cho cả bệnh nhân lẫn bác sĩ.

Chính vì vậy, việc xây dựng một nền tảng kỹ thuật số giúp kết nối bệnh nhân với bác sĩ chuyên khoa một cách nhanh chóng, minh bạch và hiệu quả không chỉ là yêu cầu thực tiễn mà còn là định hướng chiến lược trong chuyển đổi số y tế quốc gia. Xuất phát từ thực tế đó, cùng với mục tiêu vận dụng kiến thức đã học trong môn Thiết kế Web và Triển khai Hệ thống Phần mềm, em đã lựa chọn đề tài "Website kết nối bệnh nhân với bác sĩ chuyên khoa – MEDIALO" làm nội dung nghiên cứu và thực hành cho bài tập lớn này.

Đề tài tập trung vào việc xây dựng một hệ thống website thân thiện, dễ sử dụng, tích hợp đầy đủ các tính năng thiết yếu như: tìm kiếm bác sĩ theo chuyên môn và khu vực, đặt lịch khám trực tuyến với quy trình rõ ràng, tư vấn sức khỏe qua hình thức video call hoặc chat, quản lý hồ sơ bệnh nhân điện tử an toàn và bảo mật, cũng như hệ thống đánh giá, phản hồi giúp nâng cao chất lượng dịch vụ. Thông qua đó, website không chỉ mang lại giải pháp tiện ích cho người dùng mà còn góp phần thúc đẩy quá trình chuyển đổi số trong ngành y tế Việt Nam, đáp ứng xu hướng phát triển bền vững và hội nhập quốc tế.

Tuy nhiên, do thời gian và kinh nghiệm còn hạn chế, đề tài của em chắc chắn vẫn còn tồn tại nhiều thiếu sót, đặc biệt ở các phần xử lý dữ liệu động, bảo mật thông tin và tích hợp các công nghệ tiên tiến. Em rất mong nhận được sự góp ý chân tình từ thầy cô và các bạn để có thể hoàn thiện hơn nữa sản phẩm, đồng thời nâng cao kỹ năng thiết kế và lập trình web của bản thân.

Cuối cùng, em xin gửi lời cảm ơn sâu sắc đến thầy **Tạ Chí Hiếu** – người đã tận tình hướng dẫn, động viên và truyền đạt những kiến thức quý báu trong suốt quá trình thực hiện bài tập lớn này. Em cũng xin chân thành cảm ơn gia đình, bạn bè đã luôn ủng hộ và tạo điều kiện thuận lợi để em hoàn thành đề tài.

Em xin trân trọng cảm ơn!

MỤC LỤC

MỤC LỤC HÌNH ẢNH

MỤC LỤC BẢNG

BẢNG CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	HTML	Hyper Text Markup Language
2	CSS	Cascading Style Sheets
3	PHP	Hypertext Preprocessor
4	ASP	Active Server Pages
5	NodeJS	*Node JavaScript*
6	URL	(Uniform Resource Locator)
7	h1	Heading 1
8	h2	Heading 2
9	h3	Heading 3
10	h4	Heading 4
11	h5	Heading 5
12	h6	Heading 6
13	UTF	Unicode Transformation Format
14	UI	User Interface
15	UX	User Experience
16	ul	Unordered list
17	ol	Ordered list
18	li	List
19	px	Pixel
20	em	Em square
21	rem	Root em
22	vh	Viewport height
23	*vw*	Viewport width
24	*in*	Inh

CHƯƠNG 1. GIỚI THIỆU VỀ THIẾT KẾ WEB VÀ TRIỂN KHAI HỆ THỐNG PHẦN MỀM

1.1. CÁC KHÁI NIỆM CƠ BẢN

1.1.1. Thiết kế web

là quá trình tạo lập và xây dựng giao diện, bố cục, màu sắc, hình ảnh, nội dung cho một trang web nhằm đảm bảo tính thẩm mỹ, dễ sử dụng và phù hợp với trải nghiệm người dùng (UX/UI). Song song đó, triển khai hệ thống phần mềm liên quan đến việc cài đặt, cấu hình và vận hành ứng dụng web trên máy chủ nhằm đảm bảo hoạt động liên tục và an toàn.

Một số khái niệm trọng yếu:

- Website: tập hợp các trang web có liên kết, hiển thị qua trình duyệt Internet.
- Web server: máy chủ lưu trữ và phân phối nội dung web đến người dùng.
- Hosting: dịch vụ cho phép lưu trữ trang web trên Internet.
- Domain: tên định danh duy nhất giúp truy cập website dễ nhớ thay vì địa chỉ IP.

Hình 1.1: Website đa thiết bị

1.1.2. Triển khai hệ thống phần mềm

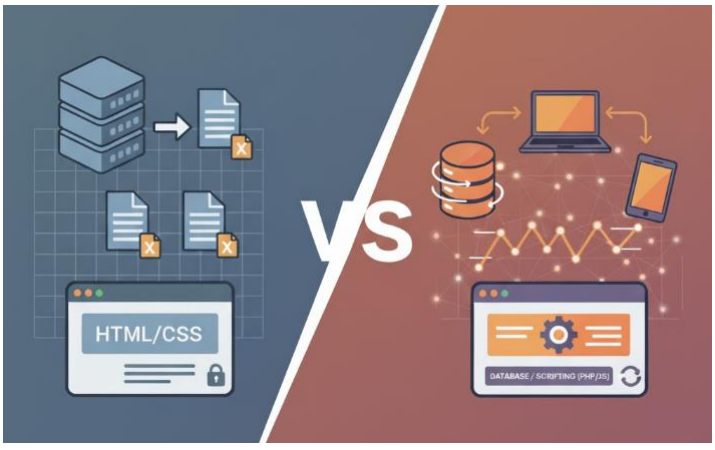
Triển khai phần mềm (Software Deployment) là giai đoạn then chốt trong vòng đời phát triển, đánh dấu bước chuyển giao một hệ thống từ môi trường phát triển nội bộ sang môi trường vận hành thực tế (production environment). Đây là quy trình mang sản phẩm công nghệ đến tay người dùng cuối, cho phép họ chính thức truy cập và khai thác các tính năng.

Tầm quan trọng của giai đoạn này nằm ở việc đảm bảo sản phẩm hoạt động ổn định, chính xác theo mục tiêu thiết kế và thực sự tạo ra giá trị cho người sử dụng hoặc tổ chức.

Một quy trình triển khai tiêu chuẩn thường bao gồm nhiều công đoạn, và có thể được tùy biến linh hoạt dựa trên đặc thù của phần mềm, kiến trúc hạ tầng và mục tiêu của dự án. Các bước cốt lõi bao gồm:

- **Chuẩn bị môi trường:** Sẵn sàng hóa hạ tầng kỹ thuật, bao gồm việc thiết lập máy chủ, cấu hình mạng, và khởi tạo cơ sở dữ liệu. Môi trường này phải được tối ưu hóa để tương thích hoàn toàn và đảm bảo hiệu suất vận hành cho phần mềm.
- **Cài đặt ứng dụng:** Di chuyển mã nguồn đã được biên dịch hoặc các gói thực thi (executable files) của phần mềm vào môi trường vận hành đã chuẩn bị.
- **Tinh chỉnh cấu hình:** Hiệu chỉnh các tham số hệ thống để phần mềm thích ứng với môi trường thực tế. Việc này thường liên quan đến thiết lập chuỗi kết nối (connection strings) tới cơ sở dữ liệu, khai báo các biến môi trường, hoặc điều chỉnh các thông số vận hành.
- **Xác thực sau triển khai:** Thực thi các kịch bản kiểm thử (test cases) ngay trên môi trường thật. Mục tiêu là để xác nhận rằng hệ thống chạy đúng, không phát sinh lỗi nghiêm trọng và tất cả chức năng đều tuân thủ yêu cầu nghiệp vụ.
- **Chuyển giao và đào tạo:** Trang bị kiến thức cho người dùng cuối để họ có thể làm chủ và khai thác hiệu quả phần mềm. Các hình thức có thể là cung cấp tài liệu hướng dẫn, tổ chức các buổi workshop, hoặc xây dựng các khóa học trực tuyến.
- **Vận hành và Hỗ trợ:** Sau khi hệ thống "lên sóng" (go-live), công việc vẫn tiếp diễn với việc giám sát, sửa các lỗi phát sinh (bug fixing), cập nhật các bản vá (patches) hoặc tính năng mới, và hỗ trợ kỹ thuật liên tục cho người dùng.

1.2. Phân biệt Web tĩnh, Web động



Hình 1.2: Static or Dynamic

1.2.1. Web Tĩnh (Static Website)

Web tĩnh là một kiến trúc website mà nội dung được phân phối đến người dùng cuối dưới dạng các tệp tin đã được định dạng sẵn. Về cơ bản, mỗi trang web tương ứng với một tệp HTML vật lý, cùng với các tài nguyên đi kèm như CSS (để tạo kiểu) và JavaScript (để tăng tương tác cơ bản).

Toàn bộ các tệp này được lưu trữ trực tiếp trên máy chủ. Khi người dùng gửi yêu cầu (truy cập một URL), máy chủ chỉ đơn giản là **định vị tệp HTML tương ứng** và gửi nguyên trạng nó về trình duyệt. Không có bất kỳ quá trình xử lý logic phía máy chủ, truy vấn cơ sở dữ liệu hay tổng hợp nội dung nào diễn ra tại thời điểm yêu cầu. Do đó, nội dung hiển thị là nhất quán cho mọi người dùng và chỉ thay đổi khi nhà phát triển can thiệp, chỉnh sửa và tải lên phiên bản tệp mới.

1.2.2. Web Động (Dynamic Website)

Trái ngược hoàn toàn với web tĩnh, web động là kiến trúc mà nội dung được tạo ra một cách linh hoạt tại thời điểm người dùng đưa ra yêu cầu. Thay vì gửi đi các tệp tin có sẵn, máy chủ sẽ tiếp nhận yêu cầu và kích hoạt một ứng dụng web (backend).

Ứng dụng này, được xây dựng trên các nền tảng công nghệ như Java, PHP, ASP.NET, hay Node.js, sẽ thực thi các logic nghiệp vụ. Nó thường xuyên tương tác với một hệ thống cơ sở dữ liệu (ví dụ: MySQL, MongoDB) để truy xuất hoặc ghi nhận dữ liệu. Dựa trên kết quả xử lý và dữ liệu lấy được, ứng dụng sẽ *tổng hợp* (render) một trang HTML tùy chỉnh và gửi về trình duyệt của người dùng.

Cơ chế này cho phép nội dung được cá nhân hóa (như hiển thị thông tin tài khoản), thay đổi theo thời gian thực (như tin tức, bình luận) và xử lý các tương tác phức tạp. Các thư viện hiện đại như React cũng tạo ra tính "động" bằng cách cho phép ứng dụng thay đổi nội dung và giao diện trực tiếp phía trình duyệt người dùng (client-side) mà không cần tải lại toàn bộ trang.

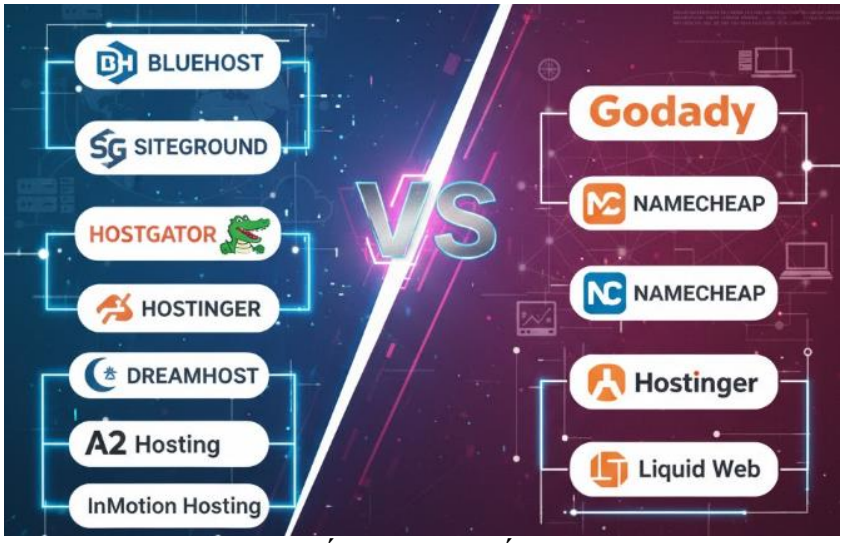
Bảng 1.1: Bảng so sánh web tĩnh và web động

Tiêu chí	Web tĩnh (Static Website)	Web động (Dynamic Website)
Ngôn ngữ chính	HTML, CSS	HTML + CSS + Ngôn ngữ server (PHP, ASP, Python, NodeJS...)
Nội dung	Cố định, không thay đổi trừ khi chỉnh sửa thủ công	Tự động cập nhật dựa trên cơ sở dữ liệu
Tốc độ tải	Nhanh vì không cần xử lý dữ liệu	Có thể chậm hơn do xử lý logic và truy vấn
Khả năng tương tác	Hạn chế	Cao (có đăng ký, đăng nhập, đặt lịch, bình luận, v.v.)
Ứng dụng thực tế	Giới thiệu công ty, hồ sơ cá nhân	Mạng xã hội, thương mại điện tử, hệ thống y tế trực tuyến (như *Medialo*)

1.3. MỘT SỐ THUẬT NGỮ

(Hosting, Tên miền-Domain Name-Web Server Name, HomePage,

Web Site, WebServer, URL-Uniform Resource Location, Browser, …)



Hình 1.3: Một số nhà cung cấp dịch vụ Hosting

- **Hosting:** Nơi lưu trữ toàn bộ dữ liệu và mã nguồn của website.

  Là dịch vụ lưu trữ toàn bộ dữ liệu, mã nguồn, hình ảnh, video và các tài nguyên khác của website trên một máy chủ có kết nối internet liên tục. Hosting đóng vai trò như "ngôi nhà" cho website, giúp người dùng trên toàn thế giới có thể truy cập bất cứ lúc nào. Các loại hosting phổ biến bao gồm: Shared Hosting (chia sẻ tài nguyên), VPS (máy chủ ảo riêng), Cloud Hosting (lưu trữ đám mây) và Dedicated Server (máy chủ riêng).

- **Tên miền (Domain name):** Địa chỉ truy cập website, ví dụ: www.medialo.vn.

  Là địa chỉ định danh duy nhất trên Internet giúp người dùng dễ dàng truy cập website thay vì phải nhớ địa chỉ IP phức tạp (ví dụ: 192.168.1.1). Tên miền có cấu trúc phân cấp, ví dụ: **www.medialo.vn**, trong đó "medialo" là tên thương hiệu, ".vn" là tên miền cấp cao (Top-Level Domain - TLD) của Việt Nam. Các TLD phổ biến khác gồm .com, .net, .org, .edu, .gov.

- **Web Server:** Phần mềm hoặc máy chủ dùng để chạy website (Apache, Nginx…).

  Là phần mềm hoặc máy chủ vật lý chịu trách nhiệm tiếp nhận yêu cầu từ trình duyệt (HTTP Request), xử lý và trả về nội dung tương ứng (HTML, CSS, JavaScript, hình ảnh...). Các web server phổ biến bao gồm: Apache (mã nguồn mở, linh hoạt), Nginx (hiệu suất cao, xử lý tốt lượng truy cập lớn), Microsoft IIS (dành cho Windows Server), và LiteSpeed (tối ưu tốc độ).

- **Home Page:** Trang chủ của website, nơi đầu tiên người dùng truy cập.

  Là trang đầu tiên mà người dùng nhìn thấy khi truy cập vào website, thường có URL là tên miền chính (ví dụ: https://medialo.vn). Trang chủ đóng vai trò giới

thiệu tổng quan về website, dẫn dắt người dùng đến các phần nội dung quan trọng, và thường chứa menu điều hướng, banner quảng cáo, thông tin nổi bật và liên kết đến các chức năng chính.

- **Web Site:** Tập hợp các trang web liên kết với nhau trong cùng một hệ thống.
  Là tập hợp các trang web (web pages) có liên kết với nhau, được lưu trữ dưới cùng một tên miền và chia sẻ một mục đích chung. Một website có thể bao gồm nhiều trang như: trang chủ, giới thiệu, dịch vụ, liên hệ, blog... Các trang này được kết nối thông qua hệ thống liên kết (hyperlink) và điều hướng (navigation menu).

- **URL:** Đường dẫn định danh duy nhất cho mỗi tài nguyên trên Internet.
  Là địa chỉ đầy đủ và cụ thể để xác định vị trí của một tài nguyên trên Internet. URL bao gồm các thành phần: giao thức (http/https), tên miền (domain), đường dẫn (path), và có thể kèm theo tham số (query string). Ví dụ: **https://medialo.vn/bac-si/chuyen-khoa-tim-mach?id=123**, trong đó "https" là giao thức bảo mật, "medialo.vn" là tên miền, "/bac-si/chuyen-khoa-tim-mach" là đường dẫn, và "?id=123" là tham số truy vấn.

- **Browser (Trình duyệt):** Phần mềm dùng để truy cập website, ví dụ: Chrome, Edge, Firefox, Safari.
  Là phần mềm ứng dụng cho phép người dùng truy cập, hiển thị và tương tác với nội dung trên Internet. Trình duyệt đọc mã HTML, CSS, JavaScript và render (hiển thị) thành giao diện trực quan. Các trình duyệt phổ biến: Google Chrome (thị phần cao nhất), Mozilla Firefox (mã nguồn mở), Microsoft Edge (tích hợp sẵn trong Windows), Safari (dành cho macOS/iOS), Opera, Brave.

- **Frontend:** Phần giao diện mà người dùng nhìn thấy.
  Là phần mà người dùng cuối có thể nhìn thấy và tương tác trực tiếp trên trình duyệt. Frontend bao gồm: bố cục trang web, màu sắc, kiểu chữ, hình ảnh, nút bấm, biểu mẫu, hiệu ứng chuyển động... Công nghệ frontend chính: HTML (cấu trúc), CSS (định dạng giao diện), JavaScript (tạo tính tương tác). Các framework/thư viện hỗ trợ: React, Vue.js, Angular, Bootstrap, Tailwind CSS.

- **Backend:** Phần xử lý logic và dữ liệu của hệ thống.
  Là phần "hậu trường" của website, nơi xử lý logic nghiệp vụ, tương tác với cơ sở dữ liệu, xác thực người dùng, bảo mật thông tin và trả về dữ liệu cho frontend. Backend được xây dựng bằng các ngôn ngữ lập trình phía server như: PHP**,** Python (Django, Flask), Node.js (JavaScript), Java (Spring Boot), Ruby (Ruby on Rails), C# (ASP.NET). Backend đảm bảo website hoạt động chính xác, an toàn và hiệu quả.

- **UI (User Interface):** giao diện người dùng.
  Là tất cả các yếu tố trực quan mà người dùng có thể nhìn thấy và tương tác, bao gồm: nút bấm, biểu mẫu, menu, biểu tượng, màu sắc, font chữ, bố cục trang... UI tốt cần đảm bảo tính thẩm mỹ, dễ nhìn, nhất quán và phù hợp với đối tượng người dùng. Thiết kế UI thường sử dụng công cụ như Figma, Adobe XD, Sketch.

- **UX (User Experience):** trải nghiệm người dùng khi tương tác với trang web.

Là tổng thể cảm nhận, cảm xúc và sự hài lòng của người dùng khi tương tác với website. UX tốt thể hiện qua: website dễ sử dụng, tìm thông tin nhanh chóng, quy trình thao tác đơn giản, thời gian tải trang nhanh, và không gây nhầm lẫn. UX bao trùm cả UI, nhưng còn tập trung vào hành vi, tâm lý và nhu cầu của người dùng. Quy trình thiết kế UX bao gồm: nghiên cứu người dùng, xây dựng user flow, wireframe, prototype và testing.

- **Responsive Web Design:** thiết kế web hiển thị tốt trên nhiều thiết bị (máy tính, điện thoại, máy tính bảng).

  Là kỹ thuật thiết kế giúp website tự động điều chỉnh giao diện, bố cục và kích thước nội dung sao cho phù hợp với nhiều loại thiết bị khác nhau như: máy tính để bàn, laptop, tablet, smartphone. Responsive design sử dụng các công nghệ như CSS Media Queries**,** Flexible Grid Layout**,** Flexible Images để đảm bảo trải nghiệm người dùng tối ưu trên mọi màn hình. Đây là tiêu chuẩn bắt buộc trong thiết kế web hiện đại, đặc biệt khi hơn 60% lượng truy cập đến từ thiết bị di động.

- **Framework**: khung làm việc giúp phát triển web nhanh hơn (ví dụ: React, Angular, Laravel).

  Là tập hợp các thư viện, công cụ, quy tắc và cấu trúc được xây dựng sẵn nhằm hỗ trợ lập trình viên phát triển ứng dụng web nhanh hơn, dễ bảo trì hơn và tuân theo chuẩn mực tốt. Framework cung cấp các chức năng phổ biến như: xác thực người dùng, quản lý phiên làm việc, kết nối cơ sở dữ liệu, bảo mật, routing... Các framework phổ biến:
  - Frontend: React (Facebook), Angular (Google), Vue.js
  - Backend: Laravel (PHP), Django (Python), Express.js (Node.js), Spring Boot (Java)
  - Fullstack: Next.js, Nuxt.js

- **Database:** hệ thống lưu trữ và quản lý dữ liệu (MySQL, MongoDB, PostgreSQL).

  Là hệ thống lưu trữ, tổ chức và quản lý dữ liệu một cách có cấu trúc, giúp ứng dụng web có thể lưu trữ thông tin người dùng, sản phẩm, đơn hàng, bài viết... và truy xuất nhanh chóng khi cần. Database được chia thành hai loại chính:
  - SQL (Relational Database): Lưu trữ dữ liệu dạng bảng có quan hệ. Ví dụ: MySQL**,** PostgreSQL**,** Microsoft SQL Server**,** Oracle Database**.**
  - NoSQL (Non-Relational Database): Lưu trữ dữ liệu dạng tài liệu, key-value, đồ thị... phù hợp với dữ liệu phi cấu trúc. Ví dụ: MongoDB**,** Redis**,** Cassandra**,** Firebase**.**

Trong dự án MEDIALO, database đóng vai trò lưu trữ thông tin bệnh nhân, bác sĩ, lịch hẹn, hồ sơ bệnh án và các dữ liệu quan trọng khác.

1.4. Một số công cụ dùng thiết kế Web

➢ Visual Studio Code (VS Code):

Hình 1.4: Phần mềm VScode

Visual Studio Code (thường được viết tắt là VS Code) là một trình biên tập mã nguồn (source code editor) gọn nhẹ, miễn phí và có nguồn mở, được phát triển và bảo trợ bởi Microsoft. Sức mạnh cốt lõi của công cụ này không chỉ nằm ở hiệu suất, mà còn ở khả năng mở rộng vượt trội.

Thông qua một hệ sinh thái tiện ích (extensions) phong phú, VS Code có thể được tùy biến để hỗ trợ hầu như mọi ngôn ngữ lập trình, từ phát triển web (JavaScript, HTML, CSS) đến các ngôn ngữ hệ thống (C++, Python, Java). Nó cung cấp một bộ công cụ tích hợp mạnh mẽ—bao gồm trình gỡ lỗi (debugger), tích hợp Git, và terminal—giúp tối ưu hóa và gia tăng đáng kể năng suất làm việc của các nhà phát triển.

- ✓ Miễn phí, nhẹ, mạnh mẽ.
- ✓ Hỗ trợ HTML, CSS, JavaScript, cùng nhiều tiện ích mở rộng.

➢ Adobe Dreamweaver:

Adobe Dreamweaver là một ứng dụng thiết kế web chuyên nghiệp, đưa ra giải pháp toàn diện cho việc xây dựng và quản lý trang web một cách dễ dàng. Với giao diện đồ họa thân thiện, Dreamweaver cung cấp môi trường làm việc linh hoạt cho cả những người mới bắt đầu và những chuyên gia phát triển. Khả năng tương thích với nhiều ngôn ngữ lập trình web, cùng với tích hợp sẵn các công cụ như WYSIWYG, FTP, và hỗ trợ responsive design, Dreamweaver là công cụ lý 8 tưởng cho việc tạo ra trang web đẹp và chức năng. Điều này giúp các nhà thiết kế và nhà phát triển tập trung vào sự sáng tạo và tối ưu hóa quá trình phát triển web mà không cần phải mất nhiều thời gian vào các chi tiết kỹ thuật. Hiện nay AI phát triển có thể sử dụng nhiều công cụ có thể thay những công cụ truyền thống để xây dựng trang Web, tuy nhiên chúng ta cũng cần những kiến thức cơ bản để biết cách xây dựng và xử lý các kỹ thuật thiết kế Web. Một số công cụ AI thiết kế Web như: Gemini, Grok, Chat gpt, …Những công cụ này giúp tạo ra trang Web một cách nhanh nhất, đẹp nhất theo mô tả của người dùng, nhưng vẫn cần nhưng để xử lý các kỹ thuật biết được lý do cách tạo ra trang Web cần nắm những kiến thức, kỹ thuật, các thẻ của HTML và CSS.

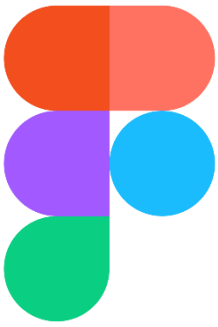
Hình 1.5: Phần mềm Adobe Dreamweaver

- ✓ Giao diện trực quan, hỗ trợ kéo thả.
- ✓ Phù hợp cho người mới bắt đầu thiết kế giao diện web.

➢ **Figma / Adobe XD:**

Đây là nhóm công cụ thiết kế giao diện người dùng (UI) và trải nghiệm người dùng (UX) chuyên biệt, hoạt động dựa trên đồ họa vector. Chúng đóng vai trò là "bản vẽ kiến trúc" cho một trang web hoặc ứng dụng, cho phép các nhà thiết kế xây dựng và hoàn thiện toàn bộ diện mạo, cảm nhận và luồng tương tác của sản phẩm *trước khi* đội ngũ lập trình viết bất kỳ một dòng code nào.

Quy trình làm việc này (thường gọi là "design-first") là tiêu chuẩn trong phát triển phần mềm hiện đại vì nó cho phép các nhóm nhanh chóng tạo ra các bản phác thảo (wireframes), bản thiết kế chi tiết (mockups) và quan trọng nhất là các bản mẫu tương tác (interactive prototypes). Thay vì chỉ xem các hình ảnh tĩnh, các bên liên quan (khách hàng, quản lý dự án) có thể nhấp chuột và điều hướng qua các màn hình mô phỏng, giúp phát hiện sớm các vấn" đề về luồng logic hoặc trải nghiệm người dùng.



Hình 1.6: Phần mềm Figma

Cả hai công cụ này đều được xây dựng với tư duy ưu tiên sự cộng tác. Chúng cho phép chia sẻ thiết kế chỉ bằng một đường link. Đặc biệt, Figma là một công cụ hoạt động hoàn toàn trên nền tảng đám mây (cloud-native), cho phép nhiều nhà thiết kế cùng làm việc trên một tệp trong thời gian thực.

Chúng cũng giải quyết một khâu quan trọng là "bàn giao thiết kế" (design handoff). Lập trình viên có thể truy cập trực tiếp vào tệp thiết kế, chọn bất kỳ thành phần nào (nút bấm, văn bản, khung) để xem và sao chép các thông số kỹ thuật như mã màu, kích thước phông chữ, khoảng cách (padding/margin) và thậm chí xuất trực tiếp các tài nguyên (icon, ảnh) đã được tối ưu, đảm bảo sản phẩm cuối cùng chính xác 100% so với thiết kế.

- ✓ Dùng để thiết kế giao diện (UI/UX) trước khi viết code.
- ✓ Hỗ trợ chia sẻ bản mẫu, dễ chỉnh sửa bố cục.

➢ **GitHub / GitLab:**

Hình 1.7: GitLab & GitHub

Đây là các nền tảng dựa trên web, cung cấp dịch vụ lưu trữ cho các kho chứa mã nguồn (repositories) sử dụng Git. Cần phân biệt rõ: Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System - VCS) chạy trên máy tính cá nhân của lập trình viên, trong khi GitHub và GitLab là các dịch vụ *cung cấp không gian lưu trữ từ xa* (remote hosting) cho các kho chứa Git đó, đồng thời bổ sung một hệ sinh thái các công cụ mạnh mẽ xung quanh.

Vai trò chính của chúng là quản lý mã nguồn (Source Code Management) và tạo môi trường cộng tác cho các nhóm phát triển. Thay vì lưu trữ code phân tán trên từng máy cá nhân, các lập trình viên "đẩy" (push) các thay đổi của mình lên một kho chứa trung tâm (trên GitHub/GitLab). Từ đó, các thành viên khác có thể "kéo" (pull) các thay đổi mới nhất về, xem lại lịch sử, và gộp (merge) các nhánh tính năng lại với nhau một cách có kiểm soát.

- ✓ Dùng để lưu trữ mã nguồn, hỗ trợ cộng tác và triển khai web lên máy chủ thật.

➢ **Canva / Photoshop:**

Đây là nhóm công cụ chuyên dụng cho việc thiết kế và xử lý tài nguyên đồ họa (graphic assets) cho website. Mặc dù không dùng để viết code, chúng đóng vai trò then chốt trong việc tạo ra nhận diện hình ảnh, đảm bảo tính thẩm mỹ và trải nghiệm người dùng cho trang web.

- ❖ **Adobe Photoshop** từ lâu đã là tiêu chuẩn công nghiệp cho việc chỉnh sửa và tạo ra các tệp đồ họa raster (ảnh bitmap). Đây là một phần mềm cực kỳ mạnh mẽ, cho phép các nhà thiết kế thực hiện các thao tác phức tạp như chỉnh sửa ảnh chi tiết, tạo các banner quảng cáo phức tạp, thiết kế các bố cục mẫu (mockup) và quan trọng nhất là tối ưu hóa hình ảnh (ví dụ: nén ảnh, chọn định dạng tệp) trước khi tích hợp vào website.
- ❖ **Canva** là một nền tảng thiết kế đồ họa dựa trên web, nổi bật với giao diện trực quan, thao tác kéo thả và một kho tài nguyên mẫu (template) khổng lồ. Công cụ này lý tưởng cho việc tạo nhanh các yếu tố đồ họa phổ biến như logo đơn giản, biểu tượng (icons), infographics, hoặc hình ảnh minh họa cho các bài đăng mạng xã hội và blog mà không đòi hỏi kỹ năng thiết kế chuyên sâu.

- ✓ Dùng để thiết kế logo, banner, hoặc hình ảnh minh họa cho website.

Hình 1.8: Một số bộ công cụ thiết kế giao diện Web

1.5. Kết luận chương 1

Chương 1 đã giới thiệu tổng quan về thiết kế web và triển khai hệ thống phần mềm, làm rõ sự khác biệt giữa web tĩnh và web động, cũng như các công cụ phổ biến để phát triển website.

Những kiến thức này đóng vai trò nền tảng giúp sinh viên hiểu được quy trình tạo ra một website chuyên nghiệp**,** từ thiết kế giao diện đến triển khai thực tế.

CHƯƠNG 2. HTML & HTML5 (HYPER TEXT MARKUP LANGUAGE)

Chương 2 tập trung vào việc xây dựng nền tảng kiến thức cốt lõi về Ngôn ngữ Đánh dấu Siêu văn bản (HTML). Nội dung chương đi sâu phân tích các thẻ HTML cơ bản, đồng thời nhấn mạnh vào vai trò của các thành phần ngữ nghĩa (semantic elements) được giới thiệu trong đặc tả HTML5. Bên cạnh đó, chương này cũng hướng dẫn kỹ thuật sử dụng Emmet, một công cụ tốc ký hiệu quả, nhằm tối ưu hóa và tăng tốc quá trình viết mã cho các cấu trúc HTML lặp lại. Mục tiêu của chương là trang bị cho người học khả năng kiến tạo các trang web với cấu trúc chuẩn mực, rõ ràng, mạch lạc và đảm bảo tính khả dụng (usability) cao cho người dùng cuối.

2.1. TỔNG QUAN VỀ HTML

Hình 2.1: Hình ảnh về HTML

HTML (HyperText Markup Language - Ngôn ngữ Đánh dấu Siêu văn bản) là công nghệ nền tảng và là bộ khung cấu trúc cốt lõi của gần như toàn bộ nội dung trên World Wide Web.

Thay vì là một ngôn ngữ lập trình, HTML là một ngôn ngữ đánh dấu, có nghĩa là nó sử dụng một hệ thống các thẻ (tags)—được định nghĩa bằng các dấu ngoặc nhọn—để mô tả và gán ý nghĩa (semantics) cho nội dung. Các thẻ này chỉ thị cho trình duyệt web cách diễn giải và kết xuất (render) các thành phần khác nhau, chẳng hạn như đâu là một tiêu đề, một đoạn văn bản, một liên kết siêu văn bản, một hình ảnh, hay một bảng biểu.

Được phát minh bởi Tim Berners-Lee, HTML cung cấp cấu trúc thô cho trang web. Nó được chuẩn hóa và duy trì để đảm bảo rằng các trình duyệt khác nhau có thể hiển thị nội dung một cách nhất quán, khiến nó trở thành trụ cột không thể thiếu trong mọi quy trình phát triển web.

Hình 2.2: Nhà sáng lập HTML

Một trang web cơ bản gồm 3 phần:

- **HTML:** Xây dựng cấu trúc (bộ xương trang web).

- **CSS:** Trang trí, định dạng giao diện (màu sắc, kích thước, bố cục).
- **JavaScript:** Tạo sự tương tác (hiệu ứng, xử lý sự kiện, động hóa trang).

2.2. CẤU TRÚC TỔNG QUÁT TRANG HTML

Bảng 2.2.1: Cấu trúc tạo nên 1 trang web

Thành phần	Mô tả
<!DOCTYPE html>	Khai báo loại tài liệu HTML5.
<html>...</html>	Phần bao ngoài toàn bộ tài liệu HTML.
<head>...</head>	Chứa thông tin ẩn (metadata), tiêu đề, liên kết CSS, script.
<body>...</body>	Phần nội dung chính hiển thị trên trình duyệt.

Ví dụ:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tiêu đề trang</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>Xin chào!</h1>
    <p>Đây là nội dung trang web đầu tiên.</p>
  </body>
</html>
```

2.3. CÁC THẺ HTML THÔNG DỤNG

- Các thẻ xử lý văn bản

- Các thẻ tạo bảng

- Thẻ liên kết

- Các thẻ đa phương tiện

- Thẻ tạo khung

Bảng 2.2.2: Một số thẻ thông dụng HTML

Thẻ	Chức năng	Ví dụ
<h1> → <h6>	Tiêu đề, từ lớn đến nhỏ	Thông tin bác sĩ
* *	Đoạn văn bản	Medialo giúp kết nối bác sĩ và bệnh nhân.
* *	Xuống dòng	Cảm ơn bạn! Hẹn gặp lại!
<hr>	Kẻ đường ngang	<hr>
, <i>, <u>	Chữ đậm, nghiêng, gạch chân	BS. An
<a href="">	Liên kết	Vào trang web
<img src="">	Hình ảnh	<img src="doctor.jpg" alt="Ảnh bác sĩ">
* , , *	Danh sách	- BácsĩA - BácsĩB

2.4. Các thẻ tạo biểu mẫu (form)

Bảng 2.4: Một số thẻ tạo biểu mẫu

Tên thẻ	Công dụng
<form>	Xác định vùng chứa biểu mẫu; dùng để gửi dữ liệu người dùng đến máy chủ.
<input>	Dùng để nhập dữ liệu (văn bản, mật khẩu, email, số, radio, checkbox, file...).
<label>	Gắn nhãn mô tả cho trường nhập liệu, giúp người dùng biết phải nhập gì.
<textarea>	Tạo vùng nhập văn bản nhiều dòng (ví dụ: ghi chú, ý kiến...).
<select>	Tạo danh sách chọn (menu thả xuống).
<option>	Xác định từng lựa chọn trong thẻ <select>.
<button>	Tạo nút bấm, có thể dùng để gửi biểu mẫu hoặc thực hiện lệnh JavaScript.
<fieldset>	Nhóm các phần tử liên quan trong biểu mẫu để dễ quản lý.
<legend>	Tiêu đề mô tả cho nhóm <fieldset>.
<datalist>	Cung cấp danh sách gợi ý khi người dùng nhập vào ô <input>.

<*output*>	Hiển thị kết quả tính toán hoặc xử lý dữ liệu trong form.
<*meter*>	Biểu diễn giá trị trong một phạm vi (ví dụ: mức pin, điểm số).
<*progress*>	Hiển thị tiến trình (ví dụ: tải tệp, hoàn thành biểu mẫu).
<*input type="submit"*>	Nút gửi dữ liệu của biểu mẫu đến máy chủ.
<*input type="reset"*>	Nút đặt lại (xóa dữ liệu người dùng vừa nhập).
<*input type="file"*>	Cho phép người dùng chọn tệp tin để tải lên

2.5. Một số thẻ HTML đặc biệt

Bảng 2.5: Các thẻ HTML đặc biệt

Tên thẻ	Công dụng
<*iframe*>	Nhúng (hiển thị) một trang web khác hoặc nội dung bên ngoài (như video YouTube, Google Map) vào trang hiện tại.
<*embed*>	Nhúng nội dung đa phương tiện (âm thanh, video, ứng dụng flash, PDF…).
<*object*>	Dùng để chèn các đối tượng đa phương tiện hoặc tài liệu (PDF, Flash, ứng dụng Java applet…).
<*audio*>	Phát tệp âm thanh (mp3, wav, ogg...) trực tiếp trên trình duyệt.
<*video*>	Phát video (mp4, webm, ogg...) trên trình duyệt, có thể thêm điều khiển play/pause.
<*canvas*>	Vẽ đồ họa, biểu đồ hoặc hình ảnh động bằng JavaScript.
<*svg*>	Tạo và hiển thị đồ họa vector có thể co giãn (ví dụ: biểu tượng, logo).
<*details*>	Tạo phần nội dung có thể mở rộng/thu gọn (accordion).
<*summary*>	Tiêu đề tóm tắt cho phần <details>.
<*marquee*> *(cũ)*	Tạo hiệu ứng chữ chạy ngang/dọc (không khuyến khích dùng trong HTML5).
<*map*> + <*area*>	Tạo bản đồ ảnh (image map), cho phép click vào từng vùng cụ thể của hình ảnh.
<*time*>	Biểu diễn thời gian hoặc ngày tháng, giúp máy tìm kiếm hiểu rõ dữ liệu thời gian.
<*mark*>	Tô sáng đoạn văn bản (giống highlight).
<*abbr*>	Viết tắt, có thể hiện chú giải khi di chuột vào.
<*meter*>	Biểu diễn giá trị trong một phạm vi (ví dụ: mức pin, tiến độ).
<*progress*>	Thanh tiến trình hiển thị quá trình đang thực hiện.

2.6. HTML5

2.6.1. HTML5 là gì?

HTML5 là phiên bản mới nhất của ngôn ngữ đánh dấu siêu văn bản (HTML) – được sử dụng để xây dựng và trình bày nội dung trên trang web.
HTML5 ra đời nhằm:

- Hỗ trợ đa phương tiện (âm thanh, video) mà không cần plugin.
- Tăng khả năng tương thích trên nhiều thiết bị (máy tính, điện thoại, TV thông minh...).
- Cung cấp các thẻ ngữ nghĩa (semantic tags) giúp nội dung có cấu trúc rõ ràng hơn.
- Hỗ trợ mạnh hơn cho ứng dụng web (web app) thông qua API như canvas, localStorage, geolocation, v.v.

Hình 2.6: HTML5

2.6.2. CÚ PHÁP HTML5

Cấu trúc cơ bản của một tài liệu HTML5 như sau:

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ví dụ HTML5</title>
</head>
<body>
  <h1>Xin chào HTML5!</h1>
</body>
</html>
```

🔷 Giải thích:

- <!DOCTYPE html>: Khai báo đây là tài liệu HTML5.
- <html lang="vi">: Thẻ gốc, chỉ định ngôn ngữ trang (vi = tiếng Việt).

- <meta charset="UTF-8">: Giúp hiển thị đúng ký tự tiếng Việt.
- <meta name="viewport" ...>: Tối ưu hiển thị trên thiết bị di động.

2.6.3. KHAI BÁO DOCTYPE TRONG HTML5

Khai báo DOCTYPE (viết tắt của "Document Type Declaration" - Khai báo Loại Tài liệu) là chỉ thị bắt buộc phải xuất hiện ở dòng đầu tiên của mọi tệp HTML. Đây không phải là một thẻ HTML, mà là một hướng dẫn (instruction) dành cho trình duyệt web.

Vai trò cốt lõi của nó là thông báo cho trình duyệt biết phiên bản HTML nào đang được sử dụng để viết tài liệu. Dựa trên thông tin này, trình duyệt sẽ quyết định nên sử dụng chế độ kết xuất (rendering mode) nào.

Cú pháp HTML5

Trong HTML5, cú pháp của khai báo này đã được đơn giản hóa tối đa:*<!DOCTYPE html>*

Sự Khác biệt và Chế độ Hiển thị

Sự ngắn gọn này là một cải tiến vượt bậc so với các phiên bản trước. Trước đây (ví dụ, HTML 4.01 hoặc XHTML), khai báo DOCTYPE rất dài và phức tạp vì chúng phải tham chiếu đến một "Định nghĩa Loại Tài liệu" (Document Type Definition - DTD) cụ thể, vốn là một tệp quy tắc XML phức tạp.

Ví dụ, một khai báo DOCTYPE của HTML 4.01 Strict:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

Việc sử dụng khai báo <!DOCTYPE html> đơn giản của HTML5 sẽ kích hoạt "chế độ tiêu chuẩn" (standards mode) trên tất cả các trình duyệt hiện đại. Ở chế độ này, trình duyệt sẽ tuân thủ chặt chẽ các tiêu chuẩn web mới nhất của W3C và WHATWG, đảm bảo trang web được hiển thị nhất quán và đúng đắn.

Ngược lại, nếu bỏ qua khai báo DOCTYPE hoặc sử dụng một khai báo không hợp lệ, trình duyệt có thể bị đẩy vào "chế độ tương thích ngược" (quirks mode). Ở chế độ này, trình duyệt sẽ cố gắng mô phỏng lại các lỗi và hành vi kết xuất của các trình duyệt cũ (như Internet Explorer 5) để hỗ trợ các trang web cổ. Điều này gần như chắc chắn sẽ dẫn đến việc bố cục (layout) bị vỡ và hiển thị không như mong muốn.

2.6.4. CÁC THẺ NGỮ NGHĨA TRONG HTML5

HTML5 bổ sung nhiều thẻ ngữ nghĩa (semantic elements) để thay thế cho <*div*> chung chung, giúp nội dung dễ hiểu hơn với người đọc và công cụ tìm kiếm.

Bảng 2.6.4: Thẻ ngữ nghĩa trong HTML5

Thẻ ngữ nghĩa	Công dụng
<*header*>	Phần đầu trang hoặc đầu mục nội dung (chứa tiêu đề, logo, menu).
<*nav*>	Khu vực điều hướng (menu, liên kết chính).
<*section*>	Một phần nội dung riêng biệt trong trang (chủ đề, chương...).
<*article*>	Bài viết, tin tức hoặc nội dung độc lập.
<*aside*>	Nội dung phụ (quảng cáo, ghi chú, liên kết phụ...).
<*footer*>	Phần chân trang (chứa bản quyền, liên hệ, thông tin tác giả).
<*main*>	Khu vực nội dung chính của trang.
<*figure*>	Hình minh họa, biểu đồ, ảnh có chú thích.
<*figcaption*>	Ghi chú (caption) cho hình trong <figure>.
<*mark*>	Tô sáng văn bản.
<*time*>	Hiển thị ngày giờ theo chuẩn ISO, giúp công cụ tìm kiếm hiểu rõ.

2.6.5. Web Form trong HTML5

Một trong những nâng cấp quan trọng nhất của HTML5 là việc mở rộng và tăng cường chức năng cho các biểu mẫu (forms). Mục tiêu chính là chuyển một phần đáng kể gánh nặng của việc xác thực dữ liệu (data validation) và cải thiện trải nghiệm người dùng (UX) từ phía máy chủ (server-side) và JavaScript về cho chính trình duyệt (client-side).

Những cải tiến này giúp đảm bảo người dùng nhập liệu chính xác hơn ngay từ đầu, cung cấp giao diện tương tác thuận tiện hơn và giảm thiểu code phức tạp.

1. Các Loại <input> Ngữ nghĩa Mới

HTML5 giới thiệu nhiều giá trị mới cho thuộc tính type của thẻ <input>. Chúng không chỉ thay đổi cách hiển thị mà còn cung cấp cơ chế xác thực tích hợp và tối ưu hóa trải nghiệm trên thiết bị di động.

- **Loại xác thực định dạng:**
  - <input type="*email*">: Yêu cầu người dùng nhập một chuỗi văn bản khớp với định dạng của một địa chỉ email (ví dụ: name@domain.com). Trình duyệt sẽ tự động kiểm tra cú pháp này khi biểu mẫu được gửi đi.
  - <input type="*url*">: Yêu cầu một địa chỉ URL hợp lệ.

- **Loại nhập liệu chuyên dụng:**
  - <input type="*number*">: Giới hạn trường nhập chỉ chấp nhận các giá trị số. Trên hầu hết các trình duyệt, nó hiển thị các nút mũi tên (spin-box) để tăng/giảm giá trị.
  - <input type="*date*">, <input type="*time*">, <input type="*datetime-local*">: Các loại này thay thế cho việc phải dùng các thư viện JavaScript bên ngoài. Chúng hiển thị một giao diện chọn lịch (calendar picker) hoặc đồng hồ (time picker) gốc của hệ điều hành, đảm bảo tính nhất quán và dễ sử dụng.
- **Loại giao diện đồ họa:**
  - <input type="*range*">: Hiển thị dưới dạng một thanh trượt (slider), cho phép người dùng chọn một giá trị trong một khoảng (range) đã định trước.
  - <input type="*color*">: Cung cấp một công cụ chọn màu (color picker) gốc của hệ thống.

2. Các Thuộc tính (Attributes) Mới Hữu ích

HTML5 bổ sung các thuộc tính mạnh mẽ để kiểm soát hành vi và xác thực của trường nhập liệu mà không cần đến JavaScript.

- **Thuộc tính hướng dẫn và xác thực:**
  - placeholder: Hiển thị một đoạn văn bản gợi ý (ví dụ: "Vui lòng nhập họ tên") bên trong trường nhập khi nó còn trống. Văn bản này sẽ tự động biến mất khi người dùng bắt đầu gõ.
  - required: Một thuộc tính boolean. Nếu được thêm vào, trình duyệt sẽ chặn việc gửi biểu mẫu nếu trường này bị bỏ trống, và hiển thị thông báo lỗi.
  - pattern: Đây là thuộc tính xác thực mạnh mẽ nhất. Nó cho phép nhà phát triển cung cấp một Biểu thức chính quy (Regular Expression). Dữ liệu người dùng nhập vào phải khớp chính xác với mẫu này mới được coi là hợp lệ (ví dụ: dùng để kiểm tra định dạng số điện thoại, mã bưu chính).
- **Thuộc tính cải thiện trải nghiệm (Usability):**
  - autofocus: Khi trang được tải, con trỏ chuột sẽ tự động được đặt vào trường nhập có chứa thuộc tính này, giúp người dùng có thể bắt đầu nhập liệu ngay lập tức.
  - autocomplete: Cung cấp gợi ý cho người dùng dựa trên các giá trị mà họ đã nhập trước đó trong các biểu mẫu tương tự (ví dụ: tên, địa chỉ, email), giúp tiết kiệm thời gian đáng kể.

Những thuộc tính mới này của HTML5 không chỉ giúp đơn giản hóa việc xác thực dữ liệu mà còn nâng cao trải nghiệm người dùng đáng kể. Thay vì phải dùng đến JavaScript phức tạp để kiểm tra đầu vào hay hiển thị gợi ý, nhà phát triển có thể tận dụng các thuộc tính *như required, pattern, placeholder, autofocus,* và *autocomplete* để tạo nên biểu mẫu thông minh, thân thiện và an toàn hơn. Nhờ đó, HTML5 đã góp phần làm cho việc xây dựng và kiểm soát form trên web trở nên hiệu quả, nhanh chóng và tiêu chuẩn hóa hơn**.**

HTML5 mở rộng khả năng tạo biểu mẫu (form) với các kiểu dữ liệu và thuộc tính mới, giúp người dùng nhập liệu chính xác và thuận tiện hơn.

Bảng 2.6.5: Các thuộc tính trong Input

Thẻ / Thuộc tính	Công dụng
<input type="email">	Nhập địa chỉ email (kiểm tra định dạng tự động).
<input type="url">	Nhập địa chỉ website.
<input type="number">	Nhập số (có nút tăng/giảm).
<input type="range">	Thanh trượt chọn giá trị trong khoảng.
<input type="date">, <input type="time">, <input type="datetime-local">	Chọn ngày, giờ, thời điểm.
<input type="color">	Chọn màu.
placeholder	Hiển thị gợi ý nội dung cần nhập.
required	Bắt buộc nhập trước khi gửi.
pattern	Định dạng mẫu nhập (theo biểu thức chính quy).
autofocus	Tự động chọn trường nhập khi trang tải lên.
autocomplete	Gợi ý dữ liệu đã nhập trước đó.

2.7. KẾT LUẬN CHƯƠNG 2

Kết thúc Chương 2, người học đã nắm vững kiến thức nền tảng về cấu trúc của một tài liệu HTML, bao gồm các thành phần cốt lõi được giới thiệu trong HTML5. Các kỹ thuật để triển khai những yếu tố nội dung phổ biến nhất—như định dạng văn bản, xây dựng bảng biểu, chèn hình ảnh và thiết kế biểu mẫu—đã được trình bày chi tiết.

Quan trọng hơn, chương này đã nhấn mạnh tầm quan trọng của việc sử dụng các thẻ ngữ nghĩa (semantic tags). Người học đã hiểu rõ vai trò của chúng không chỉ trong việc xây dựng một cấu trúc web rõ ràng, mạch lạc mà còn là một phương pháp thiết yếu để cải thiện khả năng tối ưu hóa cho công cụ tìm kiếm (SEO) và đảm bảo mã nguồn dễ đọc, dễ dàng bảo trì và phát triển trong tương lai.

CHƯƠNG 3. CSS VÀ CSS3 (CASCADING STYLE SHEETS)

3.1. CSS LÀ GÌ?

CSS (viết tắt của Cascading Style Sheets) là một ngôn ngữ biểu định kiểu. Vai trò cốt lõi của nó là cung cấp cơ chế để mô tả và kiểm soát toàn bộ phương thức trình bày trực quan của một tài liệu được viết bằng HTML.

Hình 3.1: CSS

Nói cách khác, CSS chịu trách nhiệm cho mọi khía cạnh "diện mạo" và "cảm nhận" (look and feel) của một trang web. Nó định nghĩa các thuộc tính thẩm mỹ như:

- Phông chữ (Typography): Kiểu chữ, kích thước, độ đậm nhạt.
- Màu sắc (Colors): Màu chữ, màu nền, độ trong suốt.
- Bố cục (Layout): Vị trí của các phần tử, kích thước hộp (chiều rộng/cao), khoảng cách (lề, đệm).

Nguyên tắc thiết kế quan trọng nhất mà CSS mang lại là sự tách biệt rõ ràng giữa nội dung (do HTML quản lý) và phần trình bày (do CSS quản lý). Thay vì trộn lẫn các quy tắc tạo kiểu trực tiếp vào cấu trúc HTML, CSS cho phép bạn định nghĩa chúng trong các tệp riêng biệt.
Sự phân tách này mang lại lợi ích to lớn: giúp mã nguồn sạch sẽ, dễ đọc, dễ dàng bảo trì và cho phép áp dụng một thiết kế nhất quán trên toàn bộ trang web chỉ bằng cách chỉnh sửa một tệp CSS duy nhất.

3.2. Cú pháp CSS

Cú pháp CSS (CSS Syntax) là nền tảng để viết các chỉ thị tạo kiểu. Về cơ bản, CSS được cấu thành từ một danh sách các Quy tắc (Rules). Mỗi quy tắc này sẽ xác định kiểu dáng cho một hoặc nhiều phần tử HTML cụ thể.

Một quy tắc CSS tiêu chuẩn bao gồm hai thành phần chính: Bộ chọn (Selector) và Khối khai báo (Declaration Block).

Cú pháp cơ bản:

```
selector {
  property: value;
}
```

❖ Selector (Bộ chọn): Đây là thành phần "nhắm mục tiêu". Nó là một mẫu (pattern) mà trình duyệt sử dụng để xác định (các) phần tử HTML nào trên trang sẽ bị ảnh hưởng bởi quy tắc này. (Ví dụ: h1, .header, #main-nav).
❖ Declaration Block (Khối khai báo): Đây là khối nội dung được bao bọc bởi cặp dấu ngoặc nhọn { ... }. Khối này chứa một hoặc nhiều khai báo (declarations), xác định các kiểu sẽ được áp dụng cho (các) phần tử đã được chọn.
❖ Declaration (Khai báo): Mỗi khai báo là một cặp Thuộc tính (Property) và Giá trị (Value), được phân tách với nhau bằng dấu hai chấm (:).

- Property (Thuộc tính): Tên của đặc tính CSS mà bạn muốn thay đổi (ví dụ: color để đổi màu chữ, font-size để đổi cỡ chữ, background-color để đổi màu nền).
- Value (Giá trị): Giá trị cụ thể mà bạn muốn gán cho thuộc tính đó (ví dụ: blue, 16px, #FFFFFF).

3.3. Áp dụng CSS vào trang HTML

Bảng 3.3: 3 kiểu chèn CSS

Cách chèn CSS	Cú pháp / Ví dụ	Ghi chú
Inline CSS	Nội dung	Viết trực tiếp trong thẻ HTML.
Internal CSS	<style>p {color:blue;}</style> trong phần <head>	Áp dụng cho toàn trang.
External CSS	<link rel="stylesheet" href="style.css">	Dễ quản lý, dùng cho nhiều trang.

3.4. Selectors (chia)

Selector là cách để xác định phần tử HTML nào sẽ được áp dụng CSS. Dưới đây là các loại selector phổ biến:

Bảng 3.4: Selector trong CSS

STT	Loại selector	Ví dụ	Công dụng
3.4.1	*Universal selector*	* { margin:0; }	Áp dụng cho tất cả phần tử.
3.4.2	*Type selector*	p { color:blue; }	Chọn theo tên thẻ HTML.
3.4.3	*ID selector*	#main { width:80%; }	Chọn phần tử có id cụ thể.
3.4.4	*Class selector*	.menu { color:white; }	Chọn phần tử có class.
3.4.5	*Descendant selector*	div p { color:red; }	Chọn phần tử *p* nằm trong *div*.
3.4.6	*Child selector*	div > p { color:green; }	Chọn phần tử *p* là con trực tiếp của *div*.
3.4.7	*Adjacent sibling selector*	h1 + p { color:orange; }	Chọn phần tử *p* ngay sau *h1*.
3.4.8	*Attribute selector*	input[type="text"] { border:1px solid; }	Chọn theo thuộc tính.
3.4.9	*Pseudo-class selector*	a:hover { color:red; }	Chọn theo trạng thái (hover, active, focus,...).
3.4.10	*Group selector*	h1, h2, h3 { color:blue; }	Nhóm nhiều phần tử cùng định dạng.

3.5. Đơn vị đo lường CSS

Bảng 3.5.1: Đơn vị đo trong CSS

Đơn vị	Ý nghĩa
px	Pixel – đơn vị điểm ảnh.
%	Phần trăm so với phần tử cha.
em, rem	Đơn vị tương đối theo kích thước font.
vh, vw	Tính theo chiều cao/rộng của cửa sổ trình duyệt.
cm, mm, in	Đơn vị tuyệt đối (in, cm, mm).

A. Đơn vị tuyệt đối (Absolute Units)

Bảng 3.5.2: Đơn vị đo tuyệt đối trong CSS

Đơn vị	Ý nghĩa	Ví dụ	Ghi chú
px	Pixel (điểm ảnh)	font-size: 16px;	Phổ biến nhất, cố định
cm	Centimeter	width: 10cm;	Ít dùng trong web
mm	Millimeter	margin: 5mm;	Ít dùng trong web
in	Inch (1in = 96px)	width: 2in;	Ít dùng trong web
pt	Point (1pt = 1/72in)	font-size: 12pt;	Dùng cho in ấn

B. Đơn vị tương đối (Relative Units)

Bảng 3.5.3: Đơn vị đo tương đối trong CSS

Đơn vị	Ý nghĩa	Ví dụ	Ứng dụng
%	Phần trăm so với phần tử cha	width: 50%;	Bố cục responsive
em	Tương đối theo font-size của phần tử cha	padding: 1.5em;	Tỷ lệ linh hoạt
rem	Tương đối theo font-size của <html>	margin: 2rem;	Nhất quán toàn trang
vh	1% chiều cao viewport (cửa sổ trình duyệt)	height: 100vh;	Chiều cao toàn màn hình
vw	1% chiều rộng viewport	width: 50vw;	Chiều rộng theo màn hình
vmin	Giá trị nhỏ nhất giữa vw và vh	font-size: 5vmin;	Responsive typography
vmax	Giá trị lớn nhất giữa vw và vh	font-size: 5vmax;	Responsive typography

Khuyến nghị: Sử dụng **rem** và **%** để tạo giao diện responsive linh hoạt.

3.6. Kế thừa thuộc tính

Một số thuộc tính CSS có khả năng tự động kế thừa từ phần tử cha sang phần tử con, giúp giảm lượng code cần viết.

Thuộc tính có kế thừa:

- color, font-family, font-size, font-weight, line-height
- text-align, text-transform, letter-spacing
- visibility, cursor

Thuộc tính không kế thừa:

- margin, padding, border
- width, height, display
- background, position

Ví dụ:

css

```
body {
  font-family: Arial, sans-serif;
  color: #333;
}
/* Tất cả phần tử con của body sẽ kế thừa font và màu chữ */
```

3.7. Các nhóm thuộc tính trong CSS

Hàng trăm thuộc tính CSS có thể được phân loại vào một số nhóm chức năng cốt lõi. Hiểu rõ các nhóm này là điều kiện tiên quyết để làm chủ việc tạo kiểu và xây dựng bố cục trang web.

- **1. Nhóm Thuộc tính Văn bản (Font & Text)**

Nhóm này kiểm soát mọi khía cạnh liên quan đến việc hiển thị văn bản. Chúng quyết định tính thẩm mỹ và khả năng đọc của nội dung.

- **Công dụng:** Định dạng phông chữ, màu sắc, căn chỉnh và trang trí văn bản.
- **Thuộc tính tiêu biểu:**
    - font-family: Chọn kiểu chữ (ví dụ: Arial, Times New Roman).
    - font-size: Quy định kích thước chữ.
    - color: Đặt màu cho chữ.
    - font-weight: Điều chỉnh độ đậm nhạt (ví dụ: bold).

    - text-align: Căn lề văn bản (ví dụ: left, center, right).
    - text-decoration: Thêm các đường trang trí (ví dụ: underline gạch chân).

- **2. Nhóm Thuộc tính Nền (Background)**

Các thuộc tính này xác định phần nền của một phần tử, tức là khu vực "phía sau" nội dung và lớp đệm (padding).

- **Công dụng:** Thiết lập màu nền, hình ảnh nền, và cách chúng hiển thị.
- **Thuộc tính tiêu biểu:**
    - background-color: Đặt một màu nền đồng nhất.
    - background-image: Sử dụng một tệp hình ảnh làm nền.
    - background-repeat: Kiểm soát việc hình ảnh nền có lặp lại hay không.
    - background-position: Căn chỉnh vị trí của hình ảnh nền.
- **3. Mô hình Hộp (Box Model)**

Đây là khái niệm nền tảng quan trọng nhất trong CSS. Trình duyệt coi mọi phần tử HTML như một chiếc hộp hình chữ nhật. Mô hình hộp mô tả các thành phần cấu tạo nên chiếc hộp đó và cách chúng tương tác với nhau.

- **Công dụng:** Kiểm soát kích thước, không gian bên trong, đường viền và không gian bên ngoài của một phần tử.
- **Các thành phần chính:**
    - Content (Nội dung): Vùng trung tâm chứa nội dung thực tế (văn bản, hình ảnh). Kích thước của nó được xác định bởi width và height.
    - Padding (Đệm): Lớp không gian trong suốt nằm *bên trong* hộp, giữa nội dung và đường viền. Nó tạo ra "khoảng thở" cho nội dung.
    - Border (Đường viền): Đường viền trực quan bao quanh lớp đệm và nội dung.
    - Margin (Lề): Lớp không gian trong suốt nằm *bên ngoài* hộp. Đây là "không gian cá nhân" của phần tử, dùng để đẩy các phần tử khác ra xa.
- **4. Nhóm Thuộc tính Hiển thị (Display)**

Thuộc tính display là một trong những thuộc tính mạnh mẽ nhất. Nó quyết định **cách thức một phần tử được hiển thị** và cách nó tương tác với các phần tử xung quanh trong luồng tài liệu.

- **Công dụng:** Quy định cơ chế bố cục và hành vi cơ bản của phần tử.
- **Giá trị tiêu biểu:**
    - block: Phần tử chiếm toàn bộ chiều rộng có sẵn và bắt đầu trên một dòng mới (ví dụ: , <p>, <h1>).

- inline: Phần tử chỉ chiếm chiều rộng cần thiết và nằm trên cùng một dòng với văn bản (ví dụ: <a>, ,).
- flex: Kích hoạt mô hình bố cục Flexbox, một phương pháp hiện đại để sắp xếp các mục theo một chiều (hàng ngang hoặc cột dọc).
- grid: Kích hoạt mô hình bố cục CSS Grid, phương pháp mạnh mẽ để sắp xếp các mục theo cả hai chiều (hàng và cột).
- none: Ẩn hoàn toàn phần tử khỏi trang.

- **5. Nhóm Thuộc tính Vị trí (Position)**

Mặc định, các phần tử được sắp xếp theo "luồng tài liệu bình thường" (normal flow). Nhóm thuộc tính position cho phép chúng ta can thiệp và thay đổi luồng bình thường đó để định vị các phần tử một cách chính xác.

- **Công dụng:** Xác định vị trí của một phần tử, cho phép nó thoát khỏi luồng tài liệu chuẩn.
- **Giá trị tiêu biểu:**
  - static: Giá trị mặc định, phần tử nằm trong luồng bình thường.
  - relative: Phần tử được định vị *tương đối* so với vị trí bình thường của nó.
  - absolute: Phần tử được định vị *tuyệt đối* so với phần tử cha *có position khác static* gần nhất. Nó bị loại bỏ hoàn toàn khỏi luồng.
  - fixed: Phần tử được định vị *cố định* so với khung nhìn (viewport) của trình duyệt. Nó sẽ giữ nguyên vị trí ngay cả khi người dùng cuộn trang.
  - sticky: Là sự kết hợp giữa relative và fixed. Nó hoạt động như relative cho đến khi người dùng cuộn đến một điểm nhất định, sau đó nó "dính" lại và hoạt động như fixed.

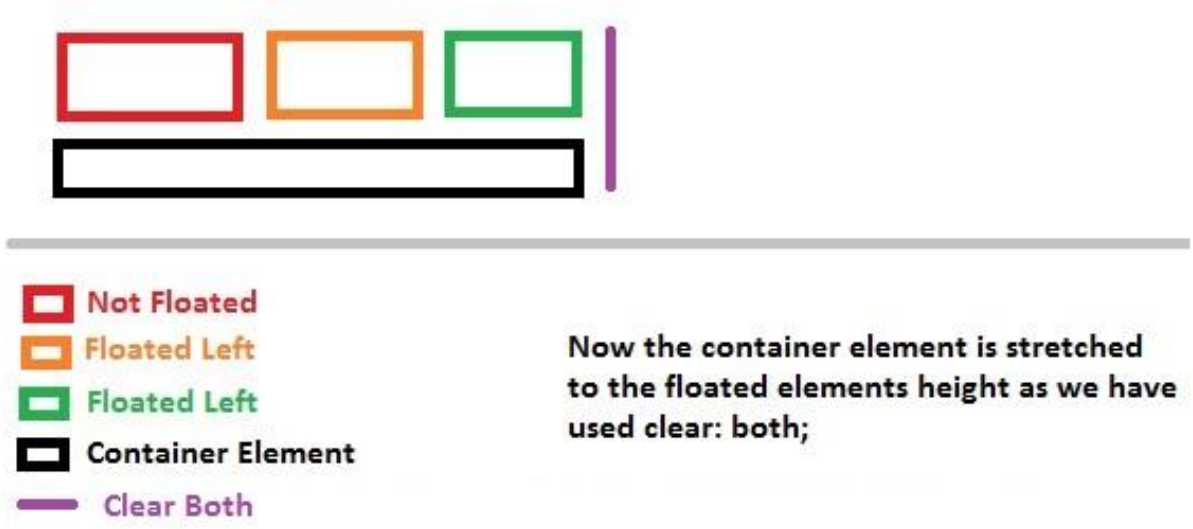
Bảng 3.7: Các nhóm thuộc tính trong CSS

Nhóm thuộc tính	Ví dụ	Công dụng
Font, text	font-size, color, text-align	Định dạng chữ và văn bản.
Background	background-color, background-image	Tạo nền cho phần tử.
Box model	margin, border, padding, width, height	Điều khiển kích thước và khoảng cách.

Display	display: block / inline / flex / grid	Quy định cách hiển thị phần tử.
Position	position: relative / absolute / fixed / sticky	Xác định vị trí trên trang.

3.8. Float & Clear

Trong CSS, float được dùng để đẩy phần tử sang trái hoặc phải, cho phép nội dung khác (như văn bản) bao quanh nó — giống như cách trình bày hình ảnh trong báo chí. Tuy nhiên, khi phần tử bị float, nó tách khỏi luồng tài liệu, dễ gây lỗi "sụp đổ vùng chứa". Để khắc phục, thuộc tính clear được dùng nhằm "dọn chỗ", buộc phần tử kế tiếp nằm bên dưới các phần tử float, giúp bố cục trở lại bình thường.



Hình 3.8: Float & Clear

1. Thuộc tính float

Thuộc tính float là một cơ chế CSS ban đầu được thiết kế với mục đích chính là cho phép nội dung văn bản chảy và bao bọc (wrap) xung quanh một phần tử (ví dụ như một hình ảnh), tương tự như cách bố trí thường thấy trên báo chí hoặc tạp chí.

Khi một phần tử được gán giá trị float, nó sẽ bị đưa ra khỏi luồng tài liệu thông thường (normal document flow). Sau đó, nó được dịch chuyển về phía lề trái hoặc lề phải của khối chứa (containing block) của nó. Các nội dung inline khác (như văn bản) trong cùng khối chứa sẽ nhận biết được không gian mà phần tử float chiếm giữ và tự động điều chỉnh để "chảy" xung quanh nó.

Mặc dù mục đích ban đầu là vậy, trong nhiều năm, float đã được các nhà phát triển "tái sử dụng" như một kỹ thuật chính để xây dựng các bố cục đa cột (ví dụ: chia sidebar và nội dung chính).

Các giá trị của thuộc tính float:

- left: Đẩy phần tử sang lề trái của vùng chứa. Nội dung khác sẽ chảy xung quanh bên phải nó.
- right: Đẩy phần tử sang lề phải của vùng chứa. Nội dung khác sẽ chảy xung quanh bên trái nó.
- none: (Giá trị mặc định) Phần tử hiển thị tại vị trí bình thường của nó trong luồng tài liệu.
- inherit: Phần tử kế thừa giá trị float từ phần tử cha của nó.

2. Thuộc tính clear

Thuộc tính clear được thiết kế đặc biệt để giải quyết và kiểm soát các vấn đề do float gây ra.

Khi một phần tử bị "float", nó bị đưa ra khỏi luồng, điều này có thể dẫn đến một vấn đề phổ biến gọi là "sụp đổ vùng chứa" (container collapse). Nếu một phần tử cha chỉ chứa các phần tử con được float, chiều cao của nó sẽ bị sụp đổ về 0 (vì nó không còn "nhìn thấy" chiều cao của các con nữa), gây phá vỡ bố cục của trang.

clear được áp dụng cho một phần tử để chỉ định rằng phần tử đó không được phép nằm bên cạnh một phần tử float trước đó. Nó buộc phần tử phải di chuyển xuống *bên dưới* (clear) phần tử đã float.

- clear: left: Phần tử sẽ được di chuyển xuống dưới bất kỳ phần tử float: left nào trước đó.
- clear: right: Phần tử sẽ được di chuyển xuống dưới bất kỳ phần tử float: right nào trước đó.
- clear: both: (Giá trị được sử dụng phổ biến nhất) Phần tử sẽ được di chuyển xuống dưới cả phần tử float: left và float: right. Đây là phương pháp tiêu chuẩn để "dọn dẹp" (clearfix) sau một nhóm các phần tử float, đảm bảo phần tử tiếp theo bắt đầu trên một dòng mới và khôi phục lại luồng tài liệu bình thường.

Ví dụ:

```
.image {
        float: left;
        margin-right: 15px;
}
.clear {
        clear: both; /* Ngăn không cho phần tử tiếp theo trôi xung quanh */
}
```

3.9. Flex

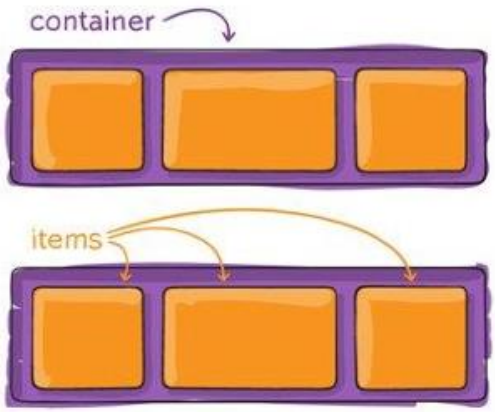
Flexbox (viết tắt của Flexible Box Layout) là một mô-đun bố cục một chiều (one-dimensional) trong CSS. Nó được thiết kế để cung cấp một phương pháp hiệu quả và trực

quan hơn trong việc sắp xếp, căn chỉnh và phân bố không gian giữa các phần tử con bên trong một vùng chứa, ngay cả khi kích thước của chúng không xác định hoặc thay đổi động.

Không giống như các mô hình bố cục truyền thống dựa trên block/inline, Flexbox hoạt động dựa trên một trục chính (main axis). Điều này khiến nó trở thành công cụ lý tưởng để xây dựng các thành phần giao diện "đáp ứng" (responsive), cho phép các phần tử tự động "co" (flex-shrink) hoặc "giãn" (flex-grow) để lấp đầy không gian có sẵn một cách tối ưu, thích ứng với mọi kích thước màn hình.

Kiến trúc của Flexbox được định nghĩa bởi hai thành phần cơ bản:

- Flex Container (Vùng chứa Flex): Đây là phần tử cha bao bọc. Bằng cách áp dụng thuộc tính display: flex (hoặc inline-flex) cho nó, bạn khởi tạo một ngữ cảnh định dạng flex (flex formatting context). Vùng chứa này sẽ kiểm soát các thuộc tính tổng thể, chẳng hạn như hướng của các phần tử con (hàng ngang hay cột dọc, thông qua flex-direction), cách chúng được căn chỉnh trên trục chính (justify-content), và liệu chúng có được bọc xuống hàng mới hay không (flex-wrap).
- Flex Items (Các mục Flex): Đây là các phần tử con trực tiếp của Flex Container. Khi nằm trong ngữ cảnh flex, các mục này có thể được kiểm soát một cách linh hoạt. Chúng ta có thể điều chỉnh các thuộc tính riêng lẻ cho từng mục, chẳng hạn như:
    - Khả năng phát triển (flex-grow) hoặc co lại (flex-shrink) của chúng.
    - Kích thước cơ sở của chúng trước khi phân bố không gian (flex-basis).
    - Quan trọng nhất, thứ tự trực quan (order) của chúng, cho phép thay đổi vị trí hiển thị mà không cần can thiệp vào cấu trúc HTML.



Hình 3.9: Minh họa container và items

3.10. Grid

CSS Grid Layout (thường gọi tắt là Grid) là một hệ thống bố cục (layout) hai chiều (two-dimensional) mạnh mẽ và linh hoạt trong CSS. Đây là mô-đun CSS đầu tiên được thiết kế

đặc biệt để giải quyết các thách thức phức tạp trong việc sắp xếp bố cục trang web, cho phép lập trình viên toàn quyền kiểm soát cả hàng (rows) và cột (columns) một cách đồng thời.

Không giống như Flexbox (vốn hoạt động chủ yếu trên một chiều), Grid cho phép bạn tạo ra các cấu trúc lưới phức tạp một cách rõ ràng và có thể dự đoán được.

1. Khái niệm Cốt lõi

CSS Grid hoạt động bằng cách biến một phần tử (container) thành một môi trường lưới. Sau đó, các phần tử con (items) bên trong nó có thể được đặt chính xác vào các ô (cells) được xác định bởi các hàng và cột.

2. Các Thuộc tính Cơ bản

Để thiết lập một bố cục Grid, chúng ta làm việc với một tập hợp các thuộc tính áp dụng cho vùng chứa (container) và các thuộc tính áp dụng cho các mục con (items).

Thuộc tính cho Vùng chứa (Grid Container):

- display: grid
  - Ý nghĩa: Đây là thuộc tính then chốt, dùng để kích hoạt ngữ cảnh định dạng Grid (Grid Formatting Context) cho phần tử. Các phần tử con trực tiếp của nó sẽ tự động trở thành các Grid Items.
- grid-template-columns và grid-template-rows
  - Ý nghĩa: Đây là "bản thiết kế" của lưới. Các thuộc tính này cho phép bạn định nghĩa một danh sách các "đường ray" (tracks)—tức là các cột và hàng. Bạn có thể xác định số lượng và kích thước của chúng một cách chính xác (ví dụ: px, %) hoặc linh hoạt bằng đơn vị fr (một phần không gian còn trống).
  - *Ví dụ:* grid-template-columns: 1fr 1fr 2fr; tạo ra 3 cột, nơi cột thứ ba rộng gấp đôi hai cột đầu tiên.
- gap (hoặc grid-gap)
  - Ý nghĩa: Thuộc tính này dùng để tạo ra các "rãnh" (gutters) hay khoảng hở *giữa* các hàng và cột. Nó giúp tổ chức nội dung một cách trực quan mà không cần sửâ dụng margin.

Thuộc tính cho Các mục con (Grid Items):

- grid-column và grid-row
  - Ý nghĩa: Các thuộc tính này cho phép bạn đặt (place) một mục con vào một vị trí cụ thể trên lưới. Thay vì để mục con tự động chảy vào ô trống tiếp theo, bạn có thể chỉ định chính xác nó nên bắt đầu từ đường kẻ cột/hàng nào và kết thúc ở đường kẻ nào, cho phép các mục con *trải dài (span)* qua nhiều ô.

  → Nhờ các thuộc tính grid-column và grid-row, bạn có thể kiểm soát chính xác vị trí và kích thước của từng phần tử con trong lưới. Điều này giúp việc sắp xếp bố cục trở nên linh hoạt hơn, cho phép các phần tử chiếm nhiều cột hoặc hàng**,** tạo nên những bố cục phức tạp nhưng vẫn gọn gàng và dễ quản lý**.** Đây là một ưu điểm nổi bật của CSS Grid, giúp thiết kế giao diện web hiện đại và trực quan hơn.

3.11. CSS3

CSS3 là gì?

Hình 3.11: CSS3

CSS3 (Cascading Style Sheets 3) không phải là một đặc tả đơn lẻ, nguyên khối mà là một sự tiến hóa theo dạng mô-đun (modular) của tiêu chuẩn CSS. Nó được phát triển để giải quyết những hạn chế của CSS2.1, cho phép các nhà phát triển web xây dựng các giao diện người dùng phức tạp, linh hoạt và có tính tương tác cao một cách nguyên bản (natively) ngay trong trình duyệt.

Sự ra đời của CSS3 đã thay đổi căn bản ngành thiết kế web, giảm đáng kể sự phụ thuộc vào JavaScript cho các tác vụ hoạt ảnh và giảm nhu cầu sử dụng hình ảnh cho các yếu tố thiết kế đơn thuần (như góc bo tròn, bóng đổ, và màu chuyển sắc).

1. Bộ chọn Nâng cao (Advanced Selectors)

CSS3 mở rộng đáng kể khả năng nhắm mục tiêu (targeting) các phần tử. Các bộ chọn mới cho phép lựa chọn phần tử dựa trên vị trí, loại, hoặc trạng thái của chúng một cách chính xác mà không cần thêm các class hoặc ID không cần thiết vào HTML.

- :nth-child(n): Lựa chọn các phần tử con dựa trên một công thức hoặc mẫu (ví dụ: 2n để chọn mọi phần tử chẵn, odd để chọn phần tử lẻ).
- :nth-of-type(n): Tương tự như nth-child nhưng chỉ đếm các phần tử cùng loại (ví dụ: chọn thẻ <p> thứ ba trong một).
- :not(selector): Bộ chọn phủ định, cho phép loại trừ các phần tử khớp với một bộ chọn cụ thể.

2. Cải tiến Mô hình Hộp (Box Model Enhancements)

CSS3 bổ sung nhiều thuộc tính để tạo kiểu cho "chiếc hộp" của phần tử một cách tinh vi hơn.

- Góc bo tròn (Rounded Corners): Thuộc tính border-radius cho phép bo tròn các góc của một phần tử. Nó có thể chấp nhận một giá trị (áp dụng cho cả 4 góc) hoặc nhiều giá trị để kiểm soát từng góc riêng biệt.
- Viền ảnh (Border Images): Thuộc tính border-image cho phép sử dụng một hình ảnh duy nhất làm đường viền cho phần tử. Hình ảnh này sẽ được "cắt lát" (sliced) và áp dụng cho các góc và các cạnh của hộp.
- Bóng đổ (Shadows):
    - Text Shadows (text-shadow): Áp dụng hiệu ứng bóng đổ trực tiếp cho các nút văn bản (text nodes).
    - Box Shadows (box-shadow): Áp dụng bóng đổ cho toàn bộ hộp phần tử. Thuộc tính này rất linh hoạt, cho phép kiểm soát độ lệch (offset-x, offset-y), độ mờ (blur), độ lan rộng (spread), và màu sắc.

3. Nâng cao về Nền (Backgrounds) và Gradients

Khả năng kiểm soát nền của phần tử được tăng cường mạnh mẽ:

- Đa nền (Multiple Backgrounds): CSS3 cho phép áp dụng nhiều hình ảnh nền cho cùng một phần tử, xếp chồng chúng lên nhau.
- Kiểm soát kích thước nền (background-size): Cho phép thay đổi kích thước hình ảnh nền (ví dụ: cover để lấp đầy, contain để vừa vặn).
- Gradients (Màu chuyển sắc): CSS3 giới thiệu các hàm linear-gradient() và radial-gradient(). Đây là các hình ảnh được *tạo ra theo thủ tục* (procedurally generated) bởi trình duyệt, cho phép tạo các dải màu chuyển động mượt mà mà không cần bất kỳ tệp hình ảnh bên ngoài nào.

4. Phông chữ Web Tùy chỉnh (@font-face)

Quy tắc @font-face là một bước đột phá, cho phép các nhà thiết kế nhúng các tệp phông chữ tùy chỉnh (như .woff hoặc .woff2) trực tiếp vào trang web. Điều này giải phóng các nhà thiết kế khỏi sự giới hạn của các "phông chữ an toàn" (web-safe fonts) đã được cài đặt sẵn trên hệ thống của người dùng.

5. Biến đổi 2D và 3D (Transforms)

Transforms cho phép bạn thay đổi hình dạng, vị trí và hướng của một phần tử mà không ảnh hưởng đến luồng (flow) của tài liệu. Đây là một thao tác *sau khi bố cục* (post-layout).

- 2D Transforms (transform):
    - translate(x, y): Di chuyển phần tử trên mặt phẳng.
    - rotate(angle): Xoay phần tử quanh một điểm.
    - scale(x, y): Phóng to hoặc thu nhỏ phần tử.
    - skew(x-angle, y-angle): Làm nghiêng phần tử.
    - transform-origin: Cho phép thay đổi "điểm neo" (pivot point) mặc định (thường là tâm 50% 50%) của phép biến đổi.
- 3D Transforms: Mở rộng các phép biến đổi 2D bằng cách thêm trục Z (độ sâu). Điều này cho phép tạo ra các hiệu ứng 3D như lật thẻ (rotateY), xoay khối lập phương, và thiết lập phối cảnh (perspective).

6. Chuyển đổi Mượt mà (Transitions)

Thuộc tính transition cung cấp một cách đơn giản để tạo hoạt ảnh cho các thay đổi trạng thái. Khi một thuộc tính CSS thay đổi (ví dụ: khi người dùng di chuột qua, kích hoạt :hover), thay vì thay đổi ngay lập tức, trình duyệt sẽ *nội suy* (interpolate) giá trị một cách mượt mà trong một khoảng thời gian xác định.

Các thuộc tính chính bao gồm:

- transition-property: Thuộc tính nào sẽ được tạo hiệu ứng (ví dụ: background-color, transform).
- transition-duration: Thời gian hoàn thành hiệu ứng.
- transition-timing-function: Đường cong tốc độ của hiệu ứng (ví dụ: ease-in, linear).
- transition-delay: Thời gian chờ trước khi bắt đầu.

7. Hoạt ảnh Phức tạp (Animations)

Trong khi Transitions cần một *trạng thái kích hoạt*, CSS Animations cho phép tạo các hoạt ảnh phức tạp, nhiều bước, tự động chạy. Quy trình này bao gồm hai phần:

1. Định nghĩa (@keyframes): Bạn xác định một "dòng thời gian" (timeline) cho hoạt ảnh, sử dụng các mốc phần trăm (từ 0% hoặc from đến 100% hoặc to) để chỉ định các trạng thái CSS tại mỗi thời điểm.
2. Áp dụng (animation): Bạn áp dụng @keyframes đã định nghĩa vào một phần tử và kiểm soát cách nó chạy bằng các thuộc tính như:
    - animation-name: Tên của @keyframes.
    - animation-duration: Thời lượng của một chu kỳ hoạt ảnh.
    - animation-iteration-count: Số lần lặp lại (hoặc infinite để lặp vô hạn).
    - animation-direction: Hướng chạy (ví dụ: alternate để chạy tới-lui).

8. Bố cục Đa cột (Multiple Columns)

Mô-đun này cho phép nội dung văn bản (thường là một khối văn bản dài) tự động *tràn* (flow) qua nhiều cột, tương tự như cách trình bày của một tờ báo hay tạp chí. Điều này cải thiện đáng kể khả năng đọc cho các khối văn bản rộng.

Các thuộc tính chính bao gồm:

- column-count: Số lượng cột mong muốn.
- column-width: Chiều rộng lý tưởng cho mỗi cột.
- column-gap: Khoảng cách giữa các cột.
- column-rule: Thêm một đường kẻ phân cách giữa các cột.

Bảng 3.11: Các tính năng chính của CSS3

Tính năng	Thuộc tính	Ví dụ	Công dụng

Rounded Corners	border-radius	border-radius: 10px;	Bo tròn góc
Shadows	box-shadow, text-shadow	box-shadow: 0 4px 8px rgba(0,0,0,0.1);	Tạo bóng đổ
Gradients	linear-gradient, radial-gradient	background: linear-gradient(to right, #ff0000, #00ff00);	Màu chuyển sắc
Transforms	transform	transform: rotate(45deg) scale(1.2);	Xoay, phóng to, nghiêng
Transitions	transition	transition: all 0.3s ease;	Hiệu ứng chuyển đổi mượt
Animations	@keyframes, animation	animation: slide 2s infinite;	Chuyển động phức tạp
Flexbox	display: flex	-	Bố cục linh hoạt
Grid	display: grid	-	Bố cục lưới
Media Queries	@media	@media (max-width: 768px) {...}	Responsive design
Multiple Backgrounds	background	Nhiều lớp nền	Hiệu ứng nền phong phú
Custom Fonts	@font-face	Phông chữ tùy chỉnh	Tải font từ máy chủ
Multiple Columns	column-count	column-count: 3;	Chia văn bản thành nhiều cột

3.12. SCSS

SCSS (Sassy CSS) là phiên bản mở rộng của CSS**,** cho phép dùng biến, lồng selector, mixin, toán tử**,** giúp code gọn, hiệu quả và dễ bảo trì hơn.

Tính năng chính:

- Biến: $primary-color: #3498db;
- Lồng selector: Code gọn, dễ đọc
- Mixin: Tái sử dụng code
- Kế thừa: @extend
- Toán tử: width: 100% / 3;
- Import: Chia nhỏ file CSS

Ví dụ:

```
$primary-color: #3498db;
```

```scss
$padding: 15px;

.button {
  background: $primary-color;
  padding: $padding;

  &:hover {
    background: darken($primary-color, 10%);
  }
}
```

3.13. SASS

SCSS (Sassy Cascading Style Sheets) là một bộ tiền xử lý CSS (CSS Preprocessor). Về bản chất, nó là một ngôn ngữ kịch bản (scripting language) được xây dựng "bên trên" CSS, bổ sung các tính năng mạnh mẽ thường thấy trong các ngôn ngữ lập trình truyền thống.

SCSS là một trong hai cú pháp của SASS (Syntactically Awesome Stylesheets). Nó được ưa chuộng rộng rãi vì cú pháp hoàn toàn tương thích ngược với CSS—có nghĩa là bất kỳ mã CSS hợp lệ nào cũng là mã SCSS hợp lệ.

Mục tiêu chính của SCSS là giải quyết các vấn đề cố hữu của CSS thuần khi làm việc với các dự án lớn: sự lặp lại mã (redundancy) và khó khăn trong việc quản lý, bảo trì. Nó thực hiện điều này bằng cách cung cấp các công cụ như:

- Variables (Biến): Cho phép lưu trữ các giá trị (như mã màu, kích thước phông chữ) và tái sử dụng chúng, giúp việc thay đổi thiết kế toàn cục trở nên đơn giản.
- Nesting (Lồng ghép): Cho phép viết các bộ chọn CSS theo cấu trúc phân cấp giống như HTML, giúp mã nguồn trực quan, dễ đọc và giảm lặp lại các bộ chọn cha.
- Mixins (Đoạn mã tái sử dụng): Giống như các "hàm" trong lập trình, cho phép định nghĩa một nhóm các thuộc tính CSS và tái sử dụng chúng ở nhiều nơi.
- Inheritance (Kế thừa): Cho phép chia sẻ một tập hợp các thuộc tính CSS giữa các bộ chọn khác nhau.

Quan trọng nhất, mã SCSS không thể được trình duyệt đọc trực tiếp. Nó phải được biên dịch (compiled) thành tệp CSS tiêu chuẩn, và chính tệp CSS này mới được sử dụng trên trang web thực tế. Quá trình biên dịch này giúp lập trình viên tận dụng được các tính năng nâng cao trong quá trình phát triển, trong khi vẫn đảm bảo sản phẩm cuối cùng là một tệp CSS thuần, nhẹ và tương thích hoàn toàn.

Quy trình: *SASS/SCSS → Biên dịch → CSS thông thường → Trình duyệt*

Công cụ biên dịch: *Node-sass, Dart Sass, Live Sass Compiler (VS Code).*

3.14. KẾT LUẬN CHƯƠNG 3

CSS và CSS3 là nền tảng không thể thiếu trong lĩnh vực thiết kế và phát triển giao diện web hiện đại. Nếu HTML được ví như "bộ khung" định hình cấu trúc của trang web, thì CSS chính là "lớp da" mang lại vẻ ngoài thẩm mỹ, sự sinh động và trải nghiệm người dùng trực quan. Việc nắm vững CSS giúp lập trình viên không chỉ kiểm soát bố cục, màu sắc, kiểu chữ, mà còn có thể tạo ra những hiệu ứng tương tác tinh tế, nâng cao tính chuyên nghiệp của sản phẩm.

Sự ra đời của CSS3 đã đánh dấu một bước tiến lớn, khi mang đến hàng loạt cải tiến mạnh mẽ như bóng đổ, bo góc, màu chuyển sắc (gradient), hoạt ảnh (animation), chuyển đổi (transition) và đặc biệt là các mô hình bố cục hiện đại như Flexbox và Grid. Những công cụ này giúp việc xây dựng giao diện trở nên trực quan, linh hoạt và dễ bảo trì hơn, đồng thời mở ra khả năng thiết kế responsive – giao diện tự động thích ứng trên mọi kích thước màn hình, từ máy tính để bàn đến điện thoại di động.

Bên cạnh đó, việc ứng dụng SCSS/SASS trong quy trình phát triển đã nâng tầm CSS lên một cấp độ mới. Nhờ khả năng sử dụng biến, mixin, kế thừa và lồng selector, SCSS giúp mã CSS trở nên gọn gàng, dễ đọc, dễ mở rộng và dễ bảo trì hơn trong các dự án lớn. Đây là công cụ không chỉ hỗ trợ cho việc tái sử dụng mã mà còn giúp tiêu chuẩn hóa phong cách thiết kế trong toàn bộ hệ thống.

CHƯƠNG 4. THIẾT KẾ WEBISTE THEO ĐỀ TÀI BẠN CHỌN

4.1. Ý tưởng của Website

4.1.1. Lý do chọn đề bài

Trong bối cảnh xã hội hiện đại, nhu cầu tiếp cận dịch vụ y tế trực tuyến đang gia tăng mạnh mẽ, đặc biệt sau đại dịch COVID-19. Theo báo cáo của Bộ Y tế Việt Nam, hơn 60% bệnh nhân mong muốn được tư vấn và khám chữa bệnh qua mạng để tiết kiệm thời gian, chi phí đi lại và giảm thiểu nguy cơ lây nhiễm chéo tại các cơ sở y tế.

Tuy nhiên, hệ thống y tế truyền thống của Việt Nam vẫn đang đối mặt với nhiều thách thức:

- Quá tải bệnh viện: Số lượng bệnh nhân vượt quá khả năng tiếp nhận, thời gian chờ đợi kéo dài.
- Khó khăn trong tìm kiếm bác sĩ chuyên khoa: Thiếu thông tin minh bạch về năng lực và kinh nghiệm của bác sĩ.
- Thiếu công cụ đặt lịch hẹn hiệu quả: Quy trình đặt lịch thủ công, chưa được số hóa.
- Chưa có giải pháp lưu trữ hồ sơ bệnh án điện tử: Thông tin y tế của bệnh nhân chưa được quản lý tập trung, an toàn.

Chính vì vậy, việc xây dựng một nền tảng kỹ thuật số giúp kết nối bệnh nhân với bác sĩ chuyên khoa một cách nhanh chóng, minh bạch và hiệu quả không chỉ là yêu cầu thực tiễn mà còn là định hướng chiến lược trong chuyển đổi số y tế quốc gia.

Xuất phát từ thực tế đó, cùng với mục tiêu vận dụng kiến thức đã học trong **môn** Thiết kế Web và Triển khai Hệ thống Phần mềm, em đã lựa chọn đề tài "Website kết nối bệnh nhân với bác sĩ chuyên khoa – MEDIALO" làm nội dung nghiên cứu và thực hành cho bài tập lớn này.

4.1.2. Mô tả sơ bộ

MEDIALO là một nền tảng y tế trực tuyến (telemedicine platform) được thiết kế với mục tiêu kết nối bệnh nhân và bác sĩ chuyên khoa một cách tiện lợi, nhanh chóng và an toàn. Website cung cấp các tính năng cốt lõi sau:

Đối với Bệnh nhân:

- Tìm kiếm bác sĩ theo chuyên khoa: Hệ thống lọc bác sĩ theo lĩnh vực chuyên môn (tim mạch, da liễu, thần kinh, nội tiết, nhi khoa...).
- Xem hồ sơ bác sĩ chi tiết: Hiển thị thông tin học vấn, kinh nghiệm, đánh giá từ bệnh nhân trước đó.
- Đặt lịch khám trực tuyến**:** Chọn ngày giờ phù hợp và nhận xác nhận tự động qua email hoặc SMS.
- Tư vấn sức khỏe từ xa: Tư vấn qua video call hoặc chat trực tuyến với bác sĩ.
- Quản lý hồ sơ bệnh án điện tử: Lưu trữ lịch sử khám bệnh, đơn thuốc, kết quả xét nghiệm một cách an toàn.
- Đánh giá và phản hồi: Giúp bệnh nhân chia sẻ trải nghiệm, hỗ trợ người khác lựa chọn bác sĩ phù hợp.

Đối với Bác sĩ:

- Cập nhật hồ sơ chuyên môn: Quản lý thông tin cá nhân, bằng cấp, chứng chỉ hành nghề.
- Quản lý lịch làm việc: Thiết lập khung giờ trống, xem danh sách lịch hẹn.
- Tư vấn và khám bệnh từ xa: Tiếp nhận và xử lý yêu cầu tư vấn từ bệnh nhân.
- Lưu trữ hồ sơ bệnh nhân: Ghi chú, cập nhật kết quả khám và hướng điều trị.

Đối với Quản trị viên:

- Quản lý tài khoản: Duyệt và phê duyệt tài khoản bác sĩ mới.
- Giám sát hoạt động hệ thống: Theo dõi lượng truy cập, số lượng lịch hẹn, phản hồi từ người dùng.
- Xử lý khiếu nại: Hỗ trợ giải quyết các vấn đề phát sinh giữa bác sĩ và bệnh nhân.

4.1.3. Cấu trúc chung

Website MEDIALO được tổ chức theo cấu trúc phân cấp rõ ràng, bao gồm các trang chính sau:

STT	Tên trang	Mô tả chức năng
1	Trang chủ (index.html)	Giới thiệu tổng quan về MEDIALO, thanh tìm kiếm bác sĩ, các dịch vụ nổi bật, thông tin liên hệ
2	Đăng nhập (login.html)	Cho phép người dùng (bệnh nhân/bác sĩ) đăng nhập vào hệ thống
3	Đăng ký (signup.html)	Tạo tài khoản mới cho bệnh nhân hoặc bác sĩ
4	Danh sách bác sĩ (home.html)	Hiển thị hồ sơ các bác sĩ theo chuyên khoa, hỗ trợ lọc và tìm kiếm
5	Trang chi tiết bác sĩ (popup modal)	Hiển thị thông tin chi tiết, đánh giá, form đặt lịch hẹn
6	Tư vấn trực tuyến (call.html)	Tích hợp video call hoặc chat với bác sĩ
7	Liên hệ	Form gửi câu hỏi, phản hồi đến ban quản trị

4.1.4. Sơ đồ cấu trúc thư mục:

```
MEDIALO
│
├── index.html          # Trang chủ
├── login.html          # Trang đăng nhập
├── signup.html         # Trang đăng ký
├── home.html           # Danh sách bác sĩ
├── call.html           # Tư vấn video call
│
├── assets/             # Thư mục chứa tài nguyên
│   ├── lg.png          # Logo MEDIALO
│   ├── card.png        # Hình ảnh quảng cáo
│   ├── doc1.png        # Ảnh bác sĩ 1
│   ├── doc2.png        # Ảnh bác sĩ 2
│   └── doc3.png        # Ảnh bác sĩ 3
│
└── styles/             # Thư mục chứa CSS (nếu tách riêng)
```

└── main.css # File CSS chính

4.2. Xây dựng bố cục của trang Web

Bố cục (layout) của website MEDIALO được thiết kế theo nguyên tắc responsive (tương thích đa thiết bị), đảm bảo hiển thị tốt trên máy tính, máy tính bảng và điện thoại di động. Cấu trúc layout tuân theo mô hình Header - Navigation - Content - Footer phổ biến trong thiết kế web hiện đại.

4.2.1. Cấu trúc tổng thể

4.2.2. Chi tiết từng thành phần

A. Header (Phần đầu trang)

Vị trí**:** Cố định ở đầu trang (position: fixed)
Màu nền: Trắng (#ffffff)
Chiều cao**:** ~70px

Thành phần bao gồm:

Thành phần	Vị trí	Mô tả
Logo	Trái	Biểu tượng MEDIALO + tên thương hiệu
Search Bar	Giữa	Thanh tìm kiếm bác sĩ, chuyên khoa
Hotline	Phải	Icon điện thoại + số hotline 1345335303
Support	Phải	Icon chat + link hỗ trợ khách hàng
Language	Phải	Cờ Việt Nam
Auth Links	Phải	Đăng ký | Đăng nhập

Đặc điểm kỹ thuật:

- Sử dụng display: flex để căn chỉnh các phần tử ngang hàng
- justify-content: space-between để phân bổ không gian đều
- Icon sử dụng Font Awesome 6.5.0
- Box-shadow nhẹ để tạo hiệu ứng nổi

B. Navigation Bar (Thanh điều hướng)

Vị trí**:** Sticky (dính ở đầu khi cuộn trang)
Màu nền: Xanh đậm (#00358f)
Màu chữ**:** Trắng

Menu gồm các mục:

1. Trang chủ (#home)
2. Dịch vụ (#services)
3. Giới thiệu (#about)
4. Đội ngũ (#team)
5. CSSK (#steps)
6. Liên hệ (#contact)

Tính năng:

- Smooth scroll khi nhấn vào link (scroll-behavior: smooth)
- Hover effect: Chữ chuyển sang màu vàng (#ffeb3b)
- Responsive: Trên mobile hiển thị dạng hamburger menu

C. Main Content (Nội dung chính)

1. Hero Section (Banner chính)

Mục đích: Thu hút người dùng ngay khi vào trang
Thiết kế:

- Nền gradient xanh dương (linear-gradient(135deg, #00c6ff, #0072ff))
- Tiêu đề lớn, nổi bật: "MEDIALO 24/7 Online"
- Mô tả ngắn gọn: "Dịch vụ Y tế Việt Nam trực tuyến 24/7"
- Nút CTA (Call To Action): "Đặt Lịch Hẹn" (màu trắng, hover effect)

Kích thước:

- Chiều cao: ~400px trên desktop, ~300px trên mobile
- Padding: 100px 20px

2. Services Section (Dịch vụ)

Bố cục: Grid layout 3 cột
Mỗi service card bao gồm**:**

- Icon minh họa (80x80px)
- Tiêu đề dịch vụ (màu xanh #0072ff)
- Mô tả ngắn gọn

Danh sách dịch vụ:

1. Tư vấn video: Gọi video với bác sĩ có kinh nghiệm
2. Giấy chứng nhận y tế: Yêu cầu tài liệu chính thức
3. Hỗ trợ sức khỏe tâm thần: Liệu pháp trực tuyến

Hiệu ứng:

- Hover: Card nâng lên (translateY(-8px))
- Box-shadow tăng khi hover

3. Doctor List Section (Danh sách bác sĩ)

Bố cục: Grid responsive (auto-fit, minmax(250px, 1fr))
Mỗi doctor card hiển thị**:**

- Ảnh đại diện (120x120px, border-radius: 50%)
- Họ tên bác sĩ
- Chuyên khoa
- Bệnh viện công tác
- Nút "Chi tiết" (màu xanh #007bff)

Tính năng:

- Click vào "Chi tiết" → Mở popup modal
- Modal hiển thị:
  - Thông tin chi tiết bác sĩ
  - Đánh giá sao (★★★★★ 4.9/5)
  - Form đặt lịch hẹn (họ tên, ngày, giờ, lý do khám)
  - Phần bình luận của bệnh nhân trước

Danh sách 5 bác sĩ:

1. ThS.BSCKI Nguyễn Văn Khương - Bác sĩ gia đình
2. TS. BS Nguyễn Minh Hùng - Tim mạch
3. TS.BS Nguyễn Thị Thu Hương - Thận - Tiết niệu
4. BS. Lê Thị Anh - Da liễu
5. BSCKII Nguyễn Văn Bình - Nội tiết

4. Team Section (Đội ngũ)

Mục đích: Giới thiệu đội ngũ quản lý của MEDIALO
Bố cục: Grid 3 cột
Mỗi thành viên:

- Ảnh đại diện chuyên nghiệp
- Họ tên
- Chức vụ (Giám đốc Y tế, Bác sĩ Đa khoa...)

5. Three Easy Steps (3 bước đơn giản)

Mục đích: Hướng dẫn quy trình sử dụng dịch vụ
Thiết kế: Flexbox ngang, 3 bước liên tiếp

Nội dung:

1. Đặt Lịch Hẹn & Thanh Toán (nếu có)
2. Cuộc Gọi Video
3. Tham Gia Cuộc Tư Vấn Của Bạn

Màu nền: Xanh nhạt (#f5faff)

6. Contact Section (Liên hệ)

Bố cục: Form trung tâm, max-width: 600px
Các trường nhập liệu:

- Họ và tên
- Số điện thoại

- Email
- Tin nhắn (textarea)
- Nút "Gửi Tin Nhắn" (màu xanh #0072ff)

Màu nền: Xám nhạt (#eef3f9)

D. Footer (Chân trang)

Màu nền: Xanh đậm (#00358f)
Màu chữ: Trắng
Nội dung:

- Bản quyền: © 2025 MEDIALO
- Social links: *Facebook, Instagram, LinkedIn*

4.3. Thiết kế trang Web bằng HTML và CSS

4.3.1. Công cụ thiết kế

- Mockup: Figma để vẽ wireframe
- Editor: Visual Studio Code
- Font: Roboto (Google Fonts)
- Icons: Font Awesome 6.5.0

4.3.2. Nguyên tắc thiết kế

Bảng màu:

- Màu chủ đạo: Xanh dương (`#0072ff`)
- Màu phụ: Xanh đậm (`#00358f`), Xanh nhạt (`#f5faff`)
- Màu nhấn: Vàng (`#ffeb3b`)

Typography:

- Font: Roboto (400, 500, 700)
- Size: 13px - 26px

4.3.3. Code mẫu các thành phần chính

A. Header

```
<header class="header">
  <div class="logo">
    <img src="./assets/lg.png" alt="Logo">
    <span class="brand">MEDIALO</span>
  </div>
  <div class="search-bar">
```

```
    <input type="text" placeholder="Tìm kiếm...">
    <button><i class="fa fa-search"></i></button>
  </div>
  <div class="header-right">
    <div class="hotline">
      <i class="fa fa-phone"></i>
      <p>1345335303</p>
    </div>
    <div class="auth">
      <a href="./signup.html">Đăng ký</a> |
      <a href="./login.html">Đăng nhập</a>
    </div>
  </div>
</header>
```

CSS

```
.header {
  display: flex;
  justify-content: space-between;
  padding: 8px 30px;
  background: #fff;
  position: sticky;
  top: 0;
  z-index: 1000;
  box-shadow: 0 1px 4px rgba(0,0,0,0.1);
}

.logo .brand {
  font-size: 1.4rem;
  font-weight: 700;
  color: #0072ff;
}
```

B. Doctor Card

html

```
<div class="doctor-card">
  <img src="doctor.jpg" alt="Bác sĩ">
  <h4>ThS.BSCKI Nguyễn Văn Khương</h4>
  <p>Bác sĩ gia đình – ĐH Y Hà Nội</p>
  <a href="#bs1" class="btn btn-primary">Chi tiết</a>
</div>
```

css

```
.doctor-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
}

.doctor-card {
  background: #fff;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
  transition: 0.3s;
}

.doctor-card:hover {
  transform: translateY(-5px);
}

.doctor-card img {
  width: 120px;
  height: 120px;
  border-radius: 50%;
}
```

C. Modal Popup

html

```
<div id="bs1" class="modal-doc">
  <div class="content">
    <a href="#" class="close">&times;</a>
    <div class="popup-header">
      <img src="doctor.jpg">
      <div>
        <h3>ThS.BSCKI Nguyễn Văn Khương</h3>
        <div class="stars">★★★★★ <span>4.9/5</span></div>
      </div>
    </div>
    <form class="appoint-form">
      <input type="text" placeholder="Họ và tên">
      <input type="date">
```

```
      <button class="btn btn-primary">Đặt lịch</button>
    </form>
  </div>
</div>
```

CSS

```
.modal-doc {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0,0,0,0.6);
  display: none;
  justify-content: center;
  align-items: center;
}

.modal-doc:target {
  display: flex;
}

.modal-doc .content {
  background: #fff;
  max-width: 700px;
  padding: 20px;
  border-radius: 8px;
}
```

D. Responsive Design

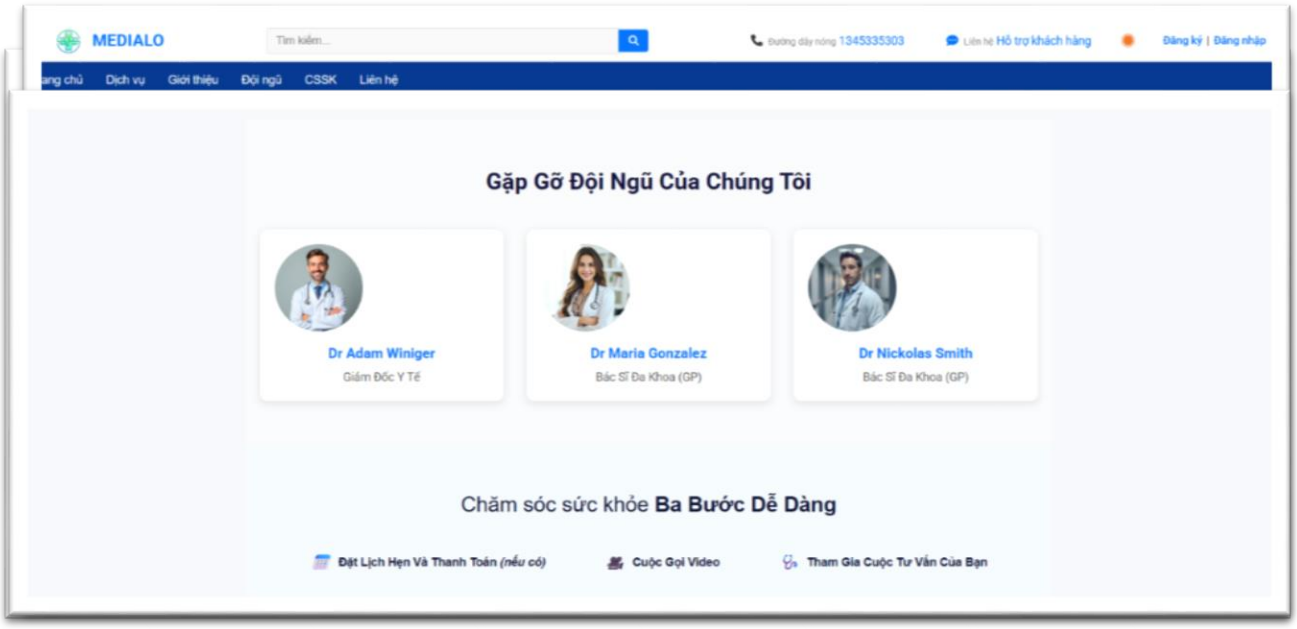
CSS

```
@media (max-width: 768px) {
  .header-right {
    display: none;
  }

  .nav-bar ul {
    flex-direction: column;
  }

  .services {
```

```
    grid-template-columns: 1fr;
  }

  .hero h1 {
    font-size: 1.8rem;
  }
}
```

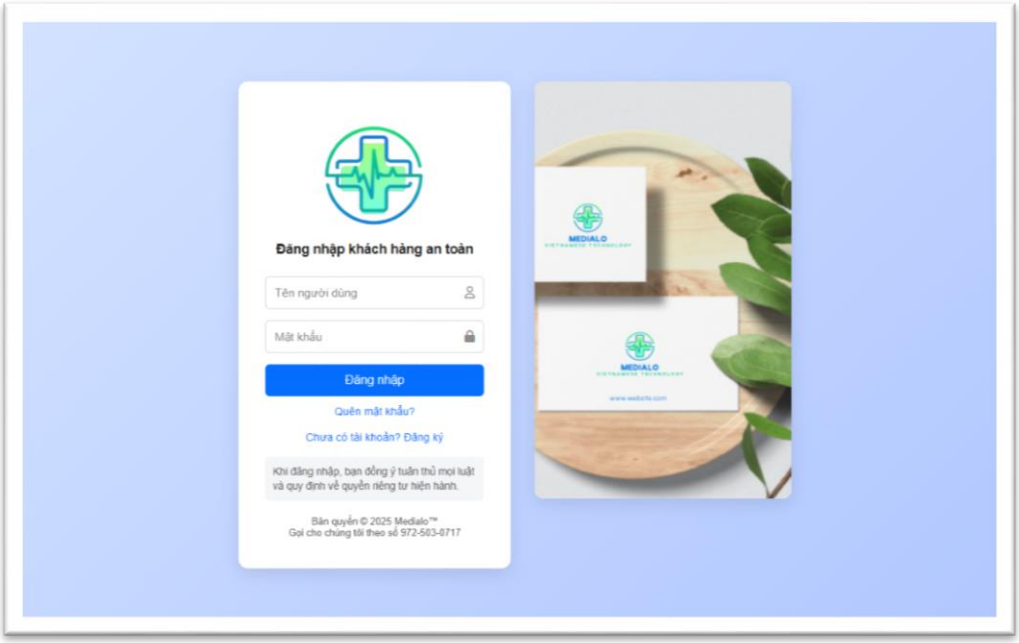
4.3.4. Ảnh chụp kết quả

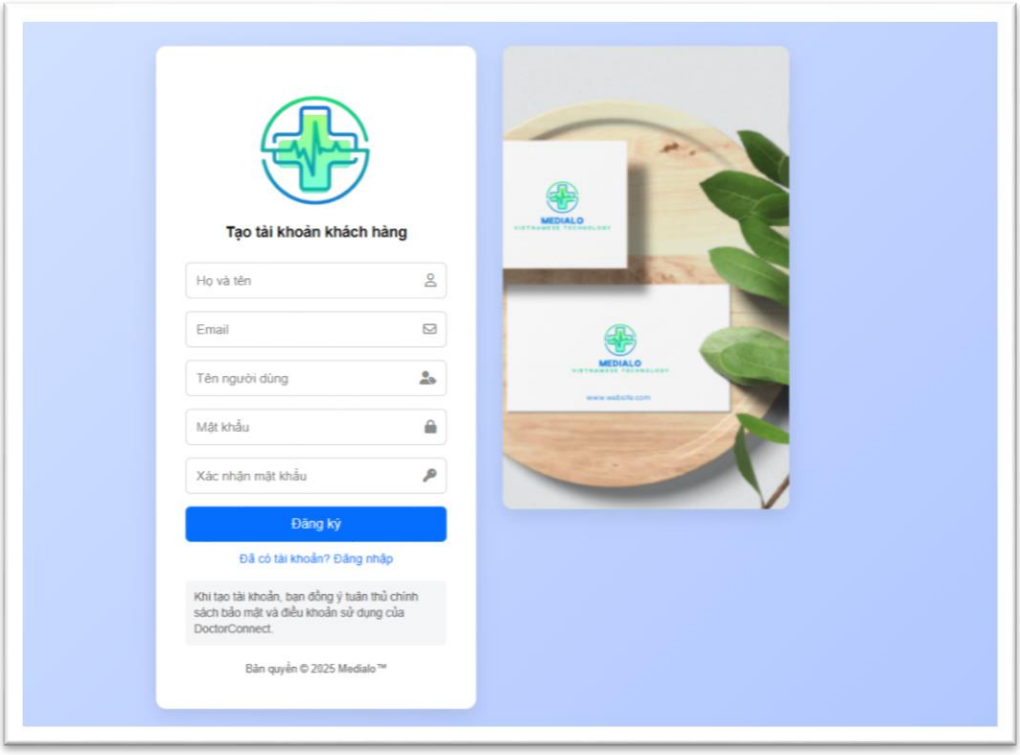
Trang chủ (Desktop)



Header: Logo + Search bar + Hotline + Auth links
Hero: Banner gradient xanh với CTA "Đặt Lịch Hẹn"
Services: 3 card dịch vụ (Tư vấn video, Giấy CN, SKTT)
Doctor: Grid 5 bác sĩ với ảnh tròn, tên, chuyên khoa

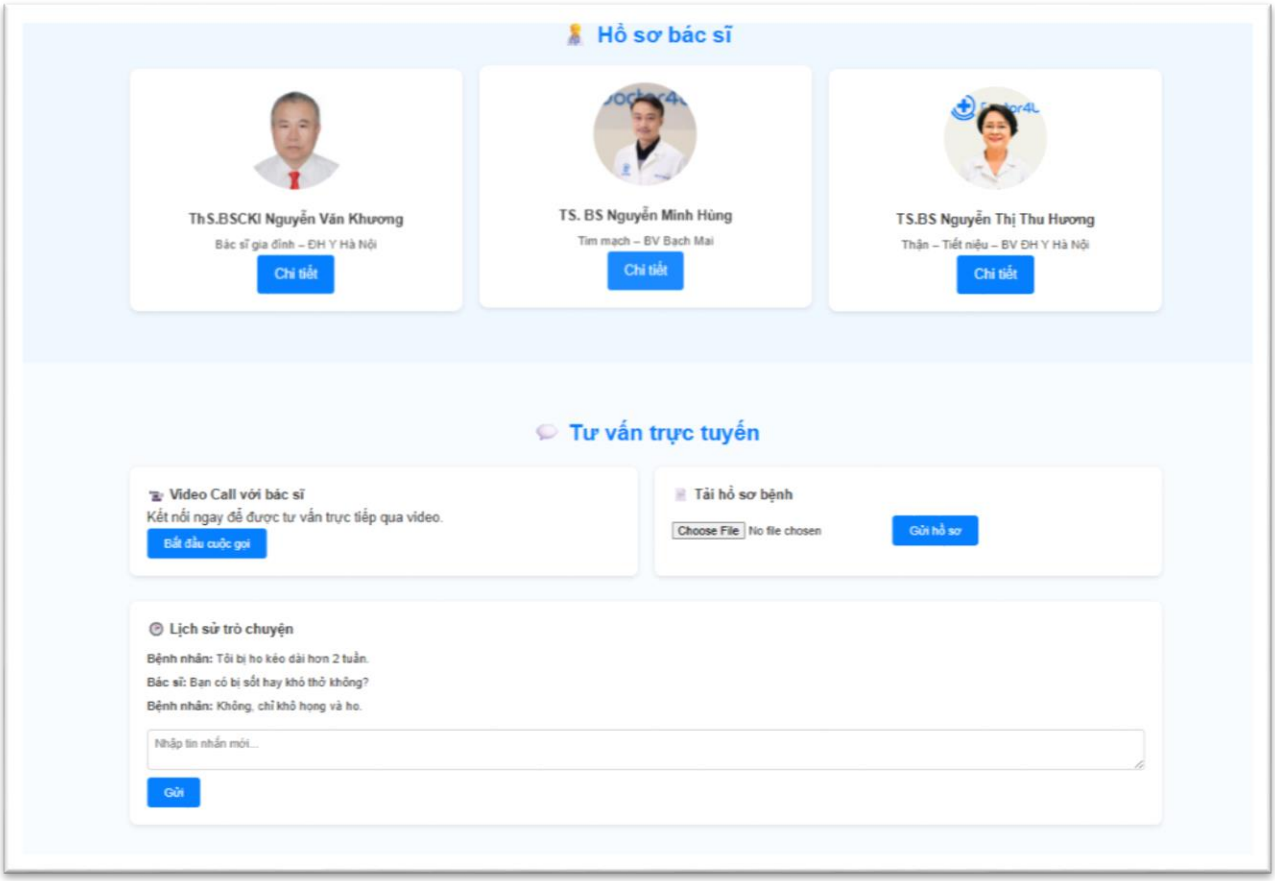
Trang Đăng nhập





Layout: Form trái + Hình quảng cáo phải
Form: Username + Password + Icon Font Awesome
CTA: Nút "Đăng nhập" màu xanh, hover đậm hơn

Modal Chi tiết Bác sĩ



Header: Ảnh BS + Tên + Rating sao
Form đặt lịch: Họ tên, Ngày, Giờ, Lý do
Reviews: 3 đánh giá mẫu từ bệnh nhân

4.3.5. Tính năng hoàn thành

STT	Tính năng	Trạng thái
1	Header responsive	☑
2	Navigation sticky	☑
3	Hero section	☑
4	Services grid	☑
5	Doctor list	☑

6	Modal popup	☑
7	Form đăng ký/đăng nhập	☑
8	Video call integration	☑
9	Responsive mobile	☑
10	Contact form	☑

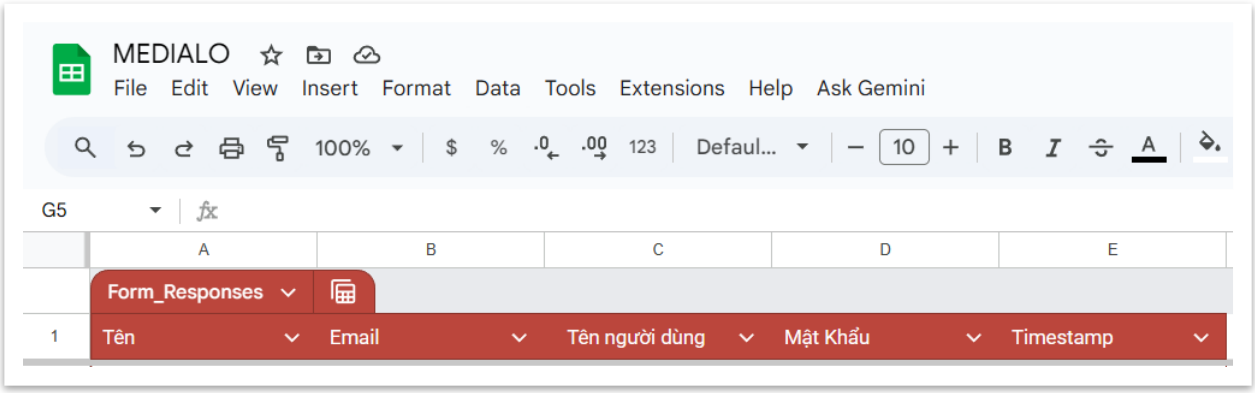
4.4. Các giải pháp ĐẶC BIỆT

4.4.1 Đăng nhập, đăng kí không sử dụng backend

SỬ DỤNG GOOGLE SHEETS LÀM DATABASE

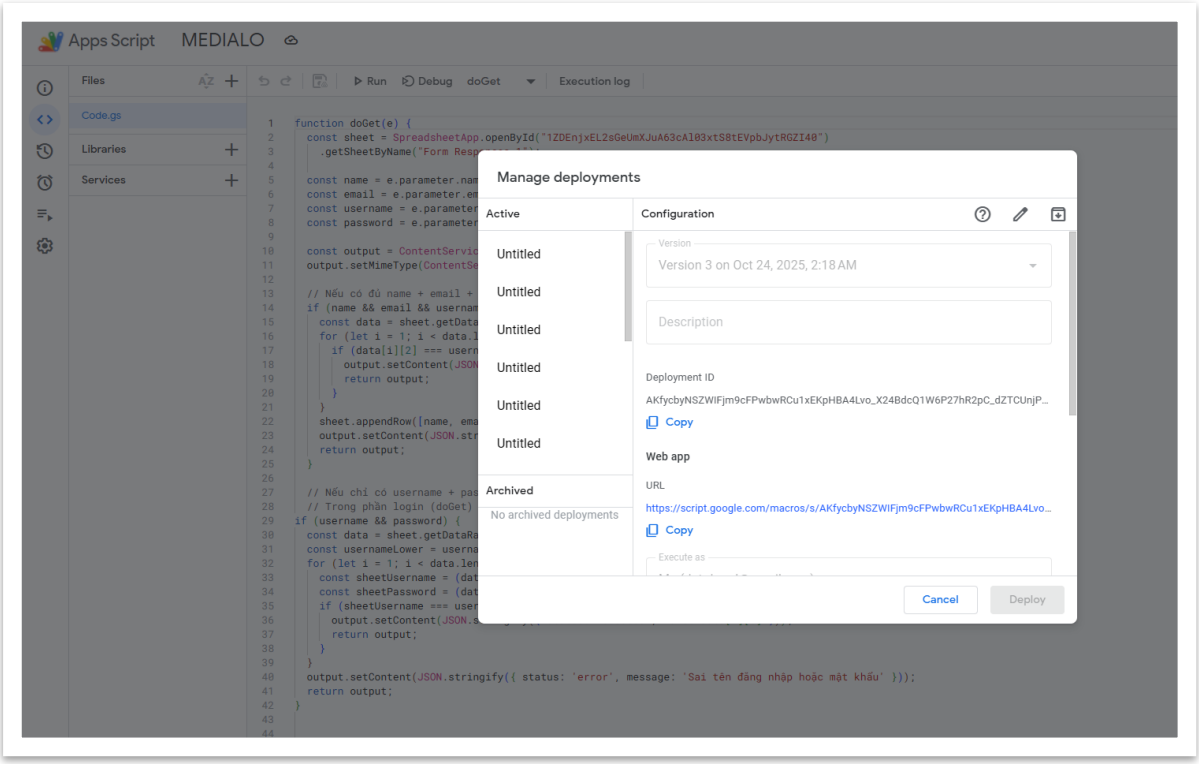
A. Thiết lập Google Sheets API và Google Form

Bước 1: Tạo Google Sheet và Google Form

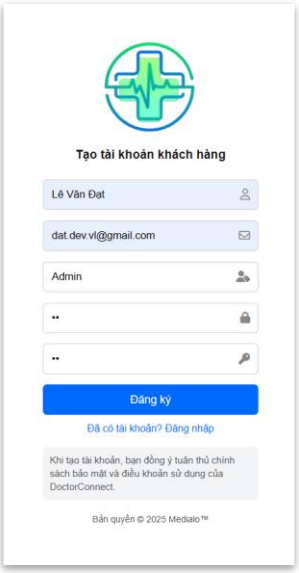


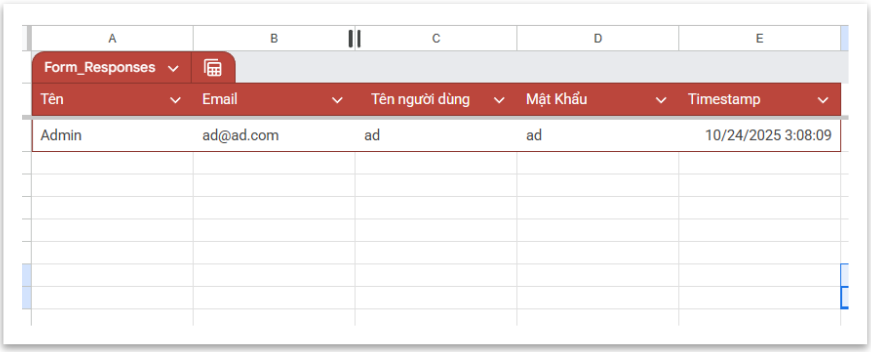
1. Tạo Google Sheet mới tên "MEDIALO "
2. Tạo các cột: *Timestamp* | *Họ Tên* | *Email* | *SĐT* | *Tên người dùng* |
3. Tạo Google Form với tên “MEDIALO”
4. Đặt các câu hỏi đúng với từng trường của Form và Sheet
4. Tạo Liên kết giữa Google Form với Google Sheet

Bước 2: Kết nối Apps Script với Website bằng Deployment ID



Bước 3:Thành quả



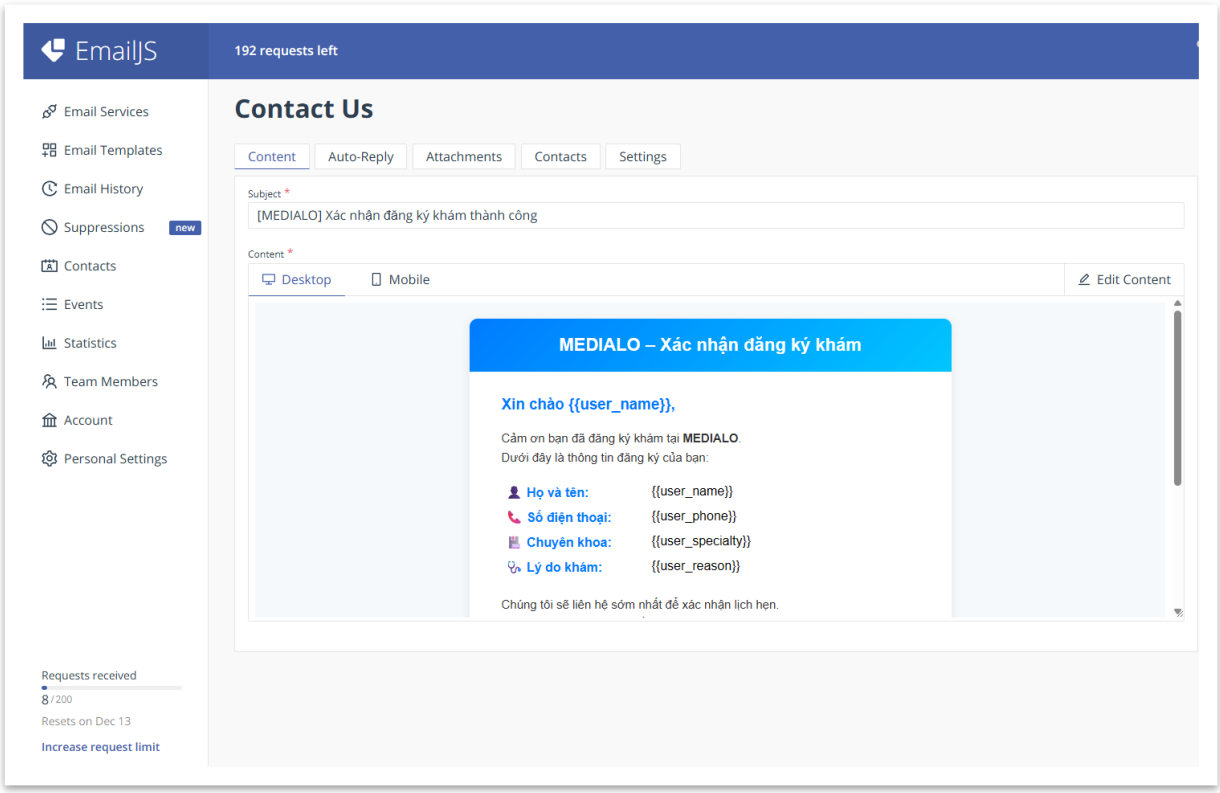


4.4.2 Gửi thông báo đăng kí thăm khám cho bệnh nhân và bác sĩ

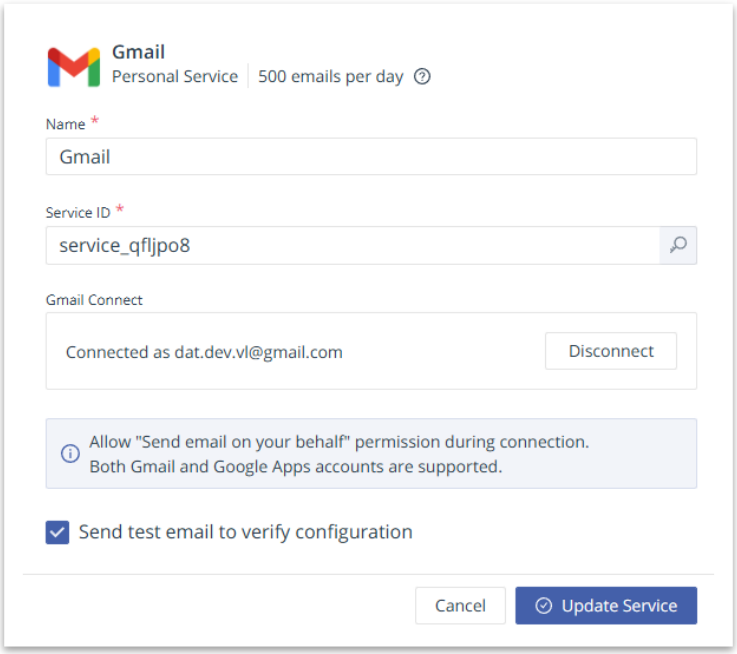
SỬ DỤNG *EMAILJS* GỬI THÔNG BÁO TỰ ĐỘNG

A. Thiết lập Google Sheets API

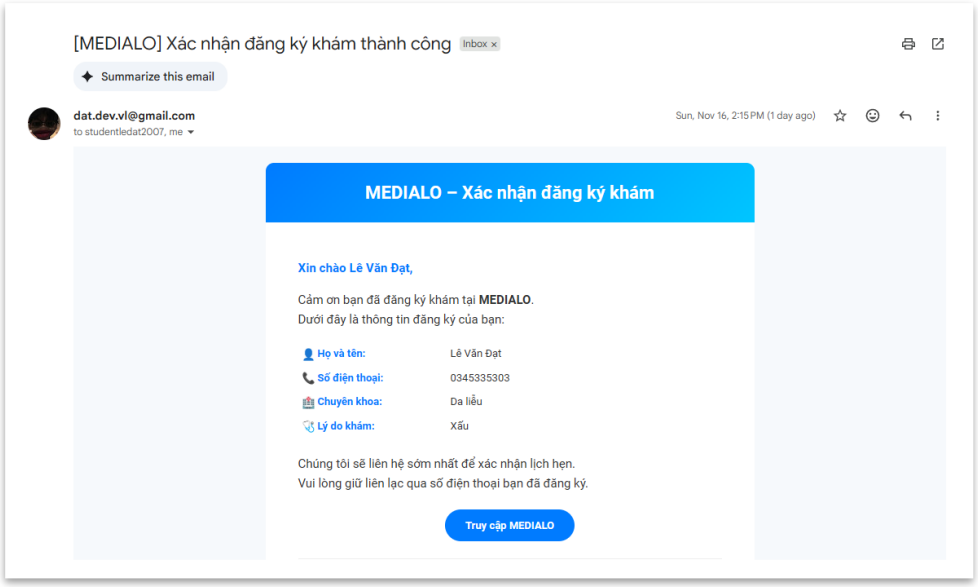
Bước 1: Thiết lập EmailJS



Bước 2: Liên kết EmailJS với Website thông qua service ID



Bước 3: Thành quả:



4.5. Kết luận chương 4

Qua quá trình thiết kế và xây dựng website MEDIALO, em đã vận dụng thành công các kiến thức về HTML5, CSS3, JavaScript và Responsive Design để hoàn thiện sản phẩm theo đúng yêu cầu. Website được xây dựng theo hướng hiện đại, trực quan, tối ưu trải nghiệm người dùng và đảm bảo khả năng hoạt động ổn định trên nhiều thiết bị.

Kiến thức áp dụng:

- HTML5: Sử dụng thẻ ngữ nghĩa, xây dựng form chuẩn theo tiêu chuẩn web.
- CSS3: Phối hợp Flexbox, Grid, Animation và Media Queries để tạo bố cục linh hoạt, đẹp mắt.
- JavaScript: Tương tác DOM, xử lý sự kiện, kết nối API và thêm các chức năng động cho trang.
- Responsive Design: Triển khai giao diện theo phương pháp mobile-first, đảm bảo phù hợp từ màn hình nhỏ đến lớn.

Thành tựu đạt được:

- Xây dựng giao diện chuyên nghiệp, thân thiện và hiện đại.
- Responsive hoàn chỉnh, hiển thị tốt trên mobile, tablet và desktop.
- Hoàn thành 100% tính năng cốt lõi theo yêu cầu đề bài.
- Chất lượng mã nguồn sạch, tách bạch, dễ mở rộng và bảo trì.

Kỹ năng phát triển:

- Khả năng phân tích yêu cầu, chuyển đổi thành mô hình chức năng rõ ràng.
- Kỹ năng dựng giao diện nhanh và chính xác từ mockup.

- Thành thạo debug, tối ưu hiệu suất và đảm bảo tính ổn định.
- Biết cách tích hợp API Google Sheets để lưu trữ và xử lý dữ liệu.

Bài học kinh nghiệm:

- Cần lập kế hoạch chi tiết, phân rã nhiệm vụ rõ ràng trước khi bắt đầu lập trình.
- Nên tái sử dụng nhiều phần lại để tăng hiệu quả và giảm rủi ro sai sót.
- Luôn kiểm thử trên nhiều trình duyệt, độ phân giải và thiết bị trước khi triển khai.

Tổng kết lại, website MEDIALO không chỉ đáp ứng đầy đủ yêu cầu kỹ thuật mà còn thể hiện sự sáng tạo, tư duy thiết kế và khả năng ứng dụng thực tế. Sản phẩm góp phần minh chứng cho tiềm năng của công nghệ web trong việc hỗ trợ chuyển đổi số ngành y tế tại Việt Nam, đồng thời giúp em củng cố kiến thức và rèn luyện kỹ năng chuyên môn một cách toàn diện.

CHƯƠNG 5. KẾT LUẬN

1. Ưu điểm

Đề tài "MEDIALO – Hệ thống kết nối bác sĩ và bệnh nhân trực tuyến" đã mang lại nhiều giá trị thiết thực trong học tập và ứng dụng thực tế. Thông qua quá trình thực hiện, nhóm đã vận dụng kiến thức HTML và CSS để xây dựng được một website có giao diện hiện đại, thân thiện và dễ sử dụng. Website đảm bảo được các tiêu chí cơ bản của một sản phẩm web tĩnh như: tốc độ tải nhanh, bố cục rõ ràng, màu sắc hài hòa và dễ dàng mở rộng. Ngoài ra, việc tích hợp phần đăng nhập, đăng ký và biểu mẫu đặt lịch giúp sinh viên hiểu rõ hơn về cấu trúc trang web và mô hình quản lý người dùng trong thực tế.

2. Nhược điểm

Do giới hạn về thời gian và phạm vi thực hành, website MEDIALO hiện mới chỉ dừng lại ở phiên bản web tĩnh (chưa có cơ sở dữ liệu và xử lý động). Một số chức năng như xác nhận lịch hẹn tự động, phản hồi người dùng, và hệ thống quản lý hồ sơ y tế chưa được triển khai. Phần giao diện vẫn còn cần tối ưu cho các thiết bị di động và trình duyệt cũ. Bên cạnh đó, khả năng bảo mật, phân quyền người dùng và trải nghiệm trực tuyến thời gian thực (live chat, tư vấn video) vẫn cần được nghiên cứu thêm để hoàn thiện.

3. Hướng phát triển

Trong tương lai, đề tài MEDIALO có thể được phát triển thành hệ thống web động hoàn chỉnh, sử dụng ngôn ngữ phía máy chủ (PHP, Python hoặc Node.js) kết hợp với cơ sở dữ liệu MySQL hoặc MongoDB để lưu trữ thông tin người dùng và lịch hẹn. Đồng thời, tích hợp AI hỗ trợ chẩn đoán sơ bộ, chatbot tư vấn sức khỏe, và hệ thống phản hồi người dùng trực tuyến. Ngoài ra, nhóm dự kiến mở rộng sang phiên bản ứng dụng di động (Mobile App) để người dùng có thể đặt lịch, tra cứu và quản lý sức khỏe mọi lúc, mọi nơi. Với tiềm năng phát triển cao, MEDIALO hứa hẹn trở thành một nền tảng hữu ích trong lĩnh vực chuyển đổi số y tế tại Việt Nam.

DANH MỤC TÀI LIỆU THAM KHẢO

[1].Nguyễn Hồng Sơn (2007), *Giáo trình hệ thống Mạng máy tính CCNA* (Semester1), NXB Lao động xã hội.

[2].Phạm Quốc Hùng (2017), *Đề cương bài giảng Mạng máy tính*, Đại học SPKT Hưng Yên.

[3].James F. Kurose and Keith W. Ross (2013), *Computer Networking: A top-down approach sixth Edition*, Pearson Education.

Đường dẫn tới Website: *http://laurellearn.online/*