

**TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN
TỐT NGHIỆP CÔNG NGHỆ THÔNG TIN**

Đề tài

HỆ THỐNG QUẢN LÝ PHÒNG KHÁM TƯ NHÂN

**Giảng viên hướng dẫn:
TS: Thái Minh Tuấn**

**Sinh viên thực hiện:
Họ và tên: Đặng Văn Phúc
MSSV: B1910432
Khóa: K45 CNTT**

Cần Thơ, 11/2023

Lời cảm ơn

Lời đầu tiên em xin chân thành gửi lời cảm ơn đến những kiến thức quý báu mà thầy Thái Minh Tuấn đã hỗ trợ cho em trong suốt học kỳ vừa qua. Đó không chỉ là kiến thức đơn thuần mà còn là những kinh nghiệm quý báu của thầy. Nhờ đó giúp em dễ dàng hơn trong việc áp dụng kiến thức vào trong thực tế, tiết kiệm rất nhiều thời gian so với tự tìm hiểu công nghệ. Một lần nữa chân trọng cảm ơn thầy rất nhiều!

Mục lục

Mục lục.....	3
Chương I. TỔNG QUAN ĐỀ TÀI.....	6
1. Đặt Vấn Đề.....	6
2. Mục Tiêu.....	6
3. Đối Tượng Và Phạm Vi Nghiên Cứu	6
4. Phương Pháp Nghiên Cứu.....	6
Chương 2: CƠ SỞ LÝ THUYẾT.....	7
1. Nodejs Và Expressjs.....	7
1.1. Tổng quan về nodejs.....	7
1.2. Lịch sử hình thành và phát triển.....	7
1.3. Lý do nên sử dụng.....	8
1.4. Phân loại ứng dụng.....	9
1.5. Tổng quan về Expressjs.....	9
2. Single-Page Application.....	11
2.1. Giới thiệu về single-page application.....	11
2.2. Nguyên lý hoạt động.....	13
2.3. Các khó khăn.....	13
2.4. Ứng dụng.....	15
3. ReactJs.....	16
3.1. Giới Thiệu ReactJs.....	14
3.2. Component.....	14
3.3. VirtualDom.....	15
3.4. TSX.....	16
3.5. State.....	16
3.6. Props.....	16
3.7. Lifecycle.....	16
3.8. Unidirectional Data Flow.....	16
3.9. Event Handling.....	16
3.10.....	Tổng quan 16

4. Prisma.....	17
4.1. Tổng Quan Về Mysql.....	17
Chương 3. ĐẶC TẢ CÁC YÊU CẦU CỦA HỆ THỐNG.....	17
1. Đặc Tả Yêu Cầu Về Chức Năng.....	17
2. Đặc Tả Yêu Cầu Về Cơ Sở Dữ Liệu.....	17
Chương 4. THIẾT KẾ GIAO DIỆN HỆ THỐNG.....	25
1. Giới Thiệu Về Ứng Dụng Web.....	25
2. Thiết Kế Giao Diện Theo Người Dùng.....	25
2.1. Người dùng quản trị.....	25
2.2. Người dùng khách hàng.....	27
CHƯƠNG 5. KẾT LUẬN.....	34
1. Kết Quả Đạt Được.....	34
2. Hạn Chế.....	34
3. Hướng Phát Triển Đề Tài.....	34
3.1. Các dự định sau khi hoàn thành đề tài.....	34
3.2. Các tính năng nghiên cứu phát triển trong tương lai.....	34
Link github	34
Tài liệu tham thảo.....	35

Chương I. TỔNG QUAN ĐỀ TÀI

1. ĐẶT VẤN ĐỀ

Ngày nay cùng với sự phát triển của khoa học công nghệ, việc áp dụng công nghệ mới vào trong sản xuất cũng như mua bán trao đổi hàng hóa và quản lý được thúc đẩy rất nhanh chóng. Bởi việc áp dụng công nghệ phù hợp với quy mô doanh nghiệp không chỉ góp phần tăng năng suất lao động, giảm thiểu sai sót mà về lâu dài còn tiết kiệm chi phí nhân công, đảm bảo uy tín và tăng khả năng cạnh tranh với các doanh nghiệp khác.

Ngày nay, với sự phát triển của ngành y tế và công nghệ, cùng với nhu cầu tăng cao về dịch vụ chăm sóc sức khỏe, việc xây dựng hệ thống quản lý phòng khám tư nhân trở nên ngày càng quan trọng. Hệ thống này giúp cung cấp thông tin chi tiết về các dịch vụ y tế, lịch hẹn, và hồ sơ bệnh nhân một cách dễ dàng và thuận tiện. Giao diện trực quan và dễ sử dụng sẽ giúp bệnh nhân tiếp cận thông tin và sử dụng các dịch vụ y tế một cách thuận lợi, tương tự như trải nghiệm mua sắm online.

Bên cạnh đó em đang học về cách lập trình web fullstack với bộ công nghệ *SERN Stack* với *typescript* (*MongoDB*, *ExpressJs*, *ReactJs* và *NodeJs*) nên việc áp dụng các kiến thức vào dự án có thể giúp em rất nhiều trong việc củng cố kiến thức của mình. Cả quá trình thiết kế và phát triển dự án đã giúp em có cái nhìn tổng quan về phát triển một ứng dụng web là như thế nào. Các vấn đề cần lưu ý khi phát triển ứng dụng web theo hướng trang đơn (SPA),... Và rất nhiều vấn đề khác mà qua đó giúp em học thêm nhiều điều.

2. MỤC TIÊU

Các vấn đề cần tìm hiểu và xử lý:

- Tìm hiểu về công nghệ lập trình MEVN Stack với TypeScript. Các yêu cầu cho việc phát triển ứng dụng web trang đơn (*Single Page Application*)
- Thiết kế các chức năng xoay quanh đề tài, viết kế hoạch các việc cần làm.
- Xây dựng backend cho ứng dụng. Thiết kế các API (*Application Protocol Interface*) cần cung cấp. Phân quyền người dùng cho hệ thống.
- Thiết kế giao diện phía client, xây dựng giao diện phù hợp cho từng loại người dùng.
- Kết nối client và server. Xử lý các vấn đề logic trong quá trình tương tác với hệ thống.

3. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

- Đối tượng nghiên cứu:
 - + Phương pháp xây dựng ứng dụng web fullstack với bộ công nghệ MEVN Stack với typescript
 - + Xây dựng website quản lý phòng khám phù hợp với các phòng khám có quy mô vừa và nhỏ.
- Phạm vi nghiên cứu:
 - + Với ứng dụng web: Cách sử dụng bộ công nghệ NodeJS, ExpressJs, Mysql, ReactJS, TypeScript... Và các hỗ trợ để có thể áp dụng xây dựng ứng dụng.

4. PHƯƠNG PHÁP NGHIÊN CỨU

- **Trên lớp:** Nghe bài giảng của giảng viên, đọc tài liệu được cung cấp và chạy thử các demo code của giảng viên (Học phần “*Phát triển ứng dụng web*” – CT449”)
- **Tại nhà:** Tìm hiểu trên Google và đọc các bài viết về công nghệ (Viblo asia,...), xem các demo code (Youtube,...), tìm lỗi trên Stack Overflow, Github,... Ghi chú các thông tin cần nhớ. Code demo thử lại ví dụ và chú thích lại các phần quan trọng.

Chương 2: CƠ SỞ LÝ THUYẾT

1. NODEJS VÀ EXPRESSJS

1.1. Tổng quan về Nodejs

Nodejs là một nền tảng phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome giúp xây dựng các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng. NodeJs là mã nguồn mở và được hỗ trợ nâng cấp bởi hàng triệu lập trình viên khắp nơi trên thế giới.

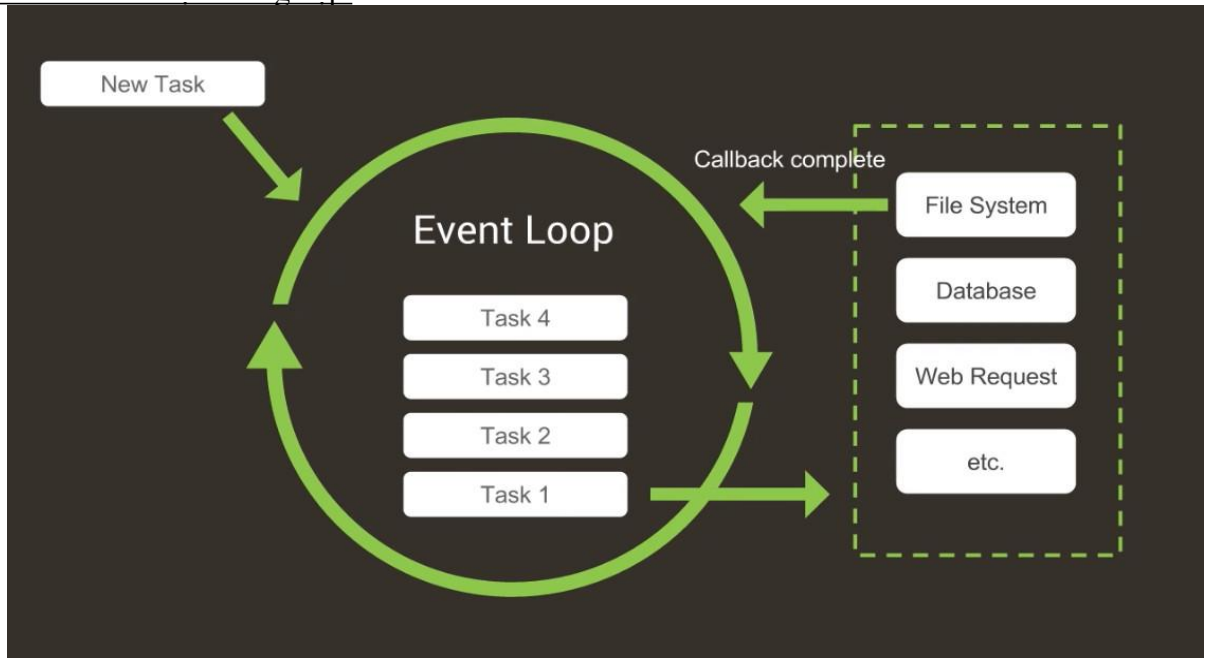
1.2. Lịch sử hình thành và phát triển

Nodejs được xây dựng và phát triển từ năm 2009 bởi Ryan Dahl và những lập trình viên khác làm việc tại Joyent. Phiên bản đầu tiên của NodeJS chỉ hỗ trợ hệ điều hành Linux và MacOS X. Sau đó được bảo trợ bởi công ty Joyent, trụ sở tại California, Hoa Kỳ. Phần core bên dưới của Nodejs được viết hầu hết bằng C++ nên cho tốc độ xử lý và hiệu năng khá cao.

Vào tháng 1 năm 2010, NPM - một trình quản lý của NodeJS đã được giới thiệu để giúp các lập trình viên dễ dàng xuất bản và chia sẻ mã nguồn của các gói NodeJS, đồng thời đơn giản hóa việc cài đặt, gỡ cài đặt và cập nhật các phiên bản.

Vào năm 2011, Microsoft và Joyent đã cùng nhau phát triển phiên bản Windows của NodeJS, mở rộng số lượng hệ điều hành mà công cụ này có thể hỗ trợ và cung cấp thêm nhiều lựa chọn cho các nhà phát triển.

Nền tảng NodeJS đã được hợp nhất với JS Foundation vào năm 2019 để tạo thành OpenJS Foundation gồm những nền tảng giúp quản lý dự án phát triển mã nguồn mở, phân tán của NodeJS.



Kiến trúc xử lý bất đồng bộ của NodeJs

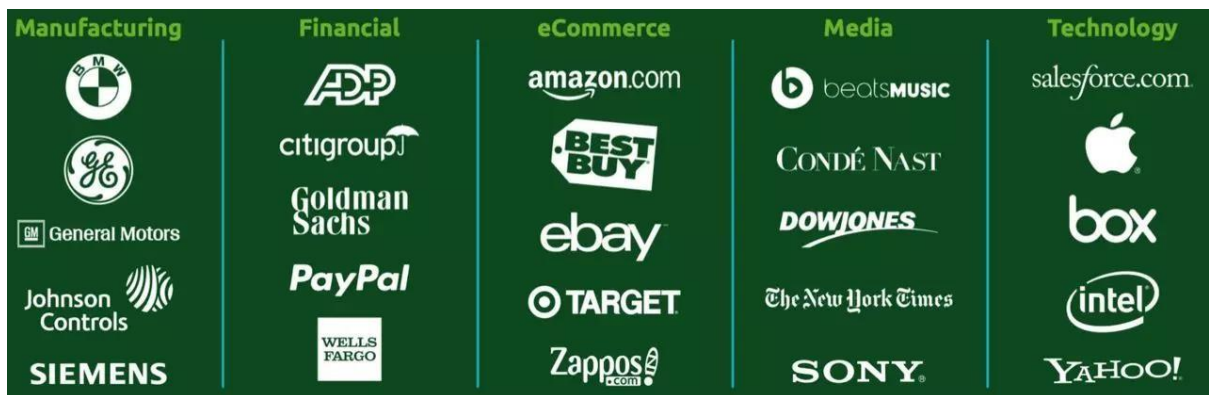
1.3. Lí do nên sử dụng NodeJs

- Miễn phí và liên tục được nâng cấp: Nền tảng mã nguồn mở nên có thể sử dụng miễn phí và đồng thời được liên tục được chỉnh sửa, cải tiến bởi cộng đồng cá nhân phát triển toàn cầu.
- Javascript Stack: Có thể phát triển website ở cả frontend và backend chỉ với ngôn ngữ Javascript với NodeJs. Từ đó ra đời bộ các công nghệ Javascript như MEAN Stack, MERN Stack, MERN Stack,... Tất cả đều phát triển ứng dụng web theo hướng SPA và sử dụng ngôn ngữ chính là Javascript.
- Khả năng mở rộng nhanh: Với kiến trúc Event Loop, bộ nhớ để xử lý cho từng request nhỏ lại giúp đáp ứng giúp đáp ứng nhiều yêu cầu xử lý hơn.
- Hiệu năng cao: Node Js sử dụng engine V8 của Google (một trình thông dịch JavaScript chạy cực nhanh trên trình duyệt Chrome), sử dụng kiến trúc hướng sự kiện Event-driven cùng với cơ chế Non-blocking I/O đẩy nhanh tốc độ xử lý của trang web.

- Cộng đồng lập trình viên phát triển lớn: Các thách thức hầu như đều có thể giải đáp trên các diễn đàn, các webmaster về Javascript.

1.4. Phân loại ứng dụng

- Fast File Upload: Những chương trình upload file với tốc độ cao.
- Websocket server: Các máy chủ của web socket gồm các dạng như: Game server, online chat,...
- Restful API: gồm các ứng dụng được dùng cho những ứng dụng khác thông qua API.
- Any Real-time Data Application: Những ứng dụng có yêu cầu cao về tốc độ thực hiện trong thời gian thực.
- Ad server: Các máy chủ quảng cáo.



Các công ty lớn đang sử dụng NodeJs

1.5. Tổng quan về Expressjs

1.5.1 Giới thiệu về ExpressJs

ExpressJS là một web framework mã nguồn mở miễn phí được xây dựng trên nền tảng NodeJs. Expressjs cung cấp các hàm HTTP và middleware để tạo ra API đơn giản và dễ sử dụng. Chúng ta có thể tạo server với NodeJs bằng cách can thiệp vào cái gói tin TCP ở cấp độ thấp, tuy nhiên để phát triển ứng dụng nhanh chóng thì cần tập trung vào chức năng, vì thế ExpressJs là một lựa chọn hàng đầu cho việc xây dựng web server.

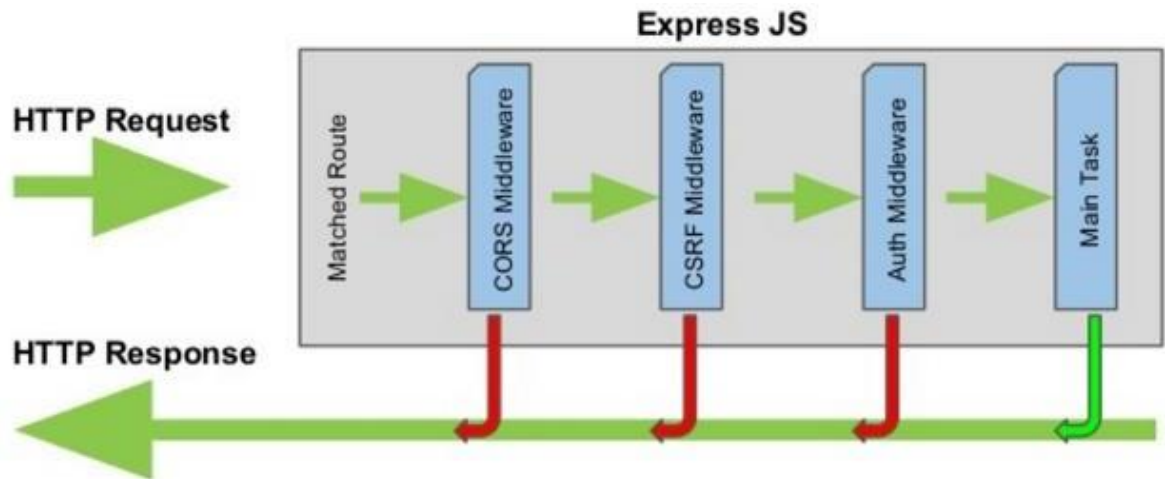
1.5.2 Các tính năng chính

- Phát triển máy chủ nhanh chóng: Expressjs cung cấp nhiều tính năng dưới dạng

Báo cáo tiểu luận tốt nghiệp

các hàm để dễ dàng sử dụng ở bất kỳ đâu trong chương trình. Điều này đã loại bỏ nhu cầu viết mã từ đó tiết kiệm được thời gian.

- Phần mềm trung gian Middleware: Đây là phần mềm trung gian có quyền truy cập vào cơ sở dữ liệu, yêu cầu của khách hàng và những phần mềm trung gian khác. Phần mềm Middleware này chịu trách nhiệm chính cho việc tổ chức có hệ thống các chức năng của Express.js. Các hàm Middleware cho phép chúng ta thực hiện hành động trước request bất kì và điều chỉnh nó trước khi gửi lại response.



- Định tuyến - Routing: Express js cung cấp cơ chế định tuyến giúp duy trì trạng thái của website với sự trợ giúp của URL.
- Tạo mẫu - Templating: Các công cụ tạo khuôn mẫu được Express.js cung cấp cho phép các nhà xây dựng nội dung động trên các website bằng cách tạo dựng các mẫu HTML ở phía máy chủ.
- Gỡ lỗi - Debugging: Để phát triển thành công các ứng dụng web không thể thiết đi việc gỡ lỗi. Giờ đây với Expressjs việc gỡ lỗi đã trở nên dễ dàng hơn nhờ khả năng xác định chính xác các phản ứng dụng web có lỗi.

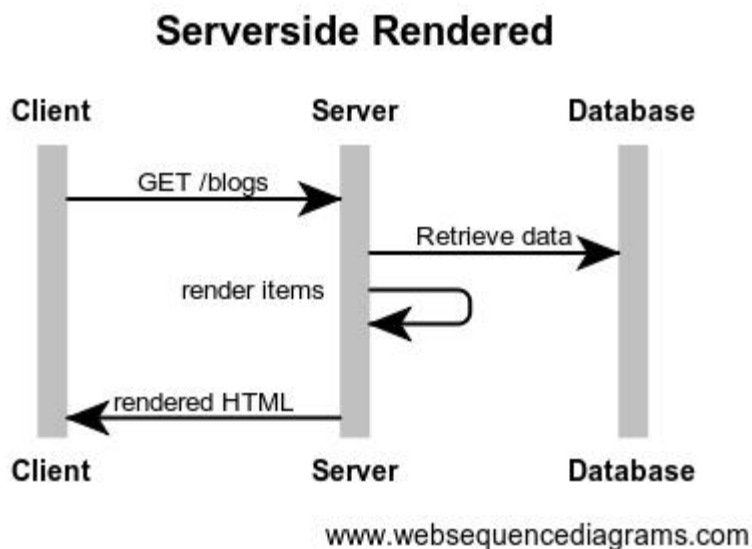
2. SINGLE-PAGE APPLICATION

2.1. Giới thiệu về single-page application

SPA (*Single Page Application* - ứng dụng trang đơn) là tên gọi chung cho một kiểu lập trình web.

Có 2 đặc điểm lớn tạo nên sự khác biệt về cách hoạt động của web SPA so với web truyền thống là:

- Web SPA có backend và frontend rõ ràng.
Web SPA sẽ đẩy mạnh xử lý ở frontend



Ngoài ra, khác biệt lớn khi trải nghiệm là SPA không hiển thị tải trang khi người dùng yêu cầu tài nguyên nào đó, còn ở website truyền thống có tải lại toàn bộ trang để hiển thị yêu cầu.

Website code theo kiểu truyền thống có thể chỉ cần 1 dự án, nhưng website xây dựng theo kiểu SPA cần 2 dự án: 1 dự án cho backend và 1 dự án cho frontend, theo đó thì backend và frontend được coi là 2 dự án khác nhau, việc trao đổi dữ liệu giữa backend và frontend thường qua các Restful API, định dạng dữ liệu thường là JSON.



So sánh vòng đời của hai kiểu phát triển website

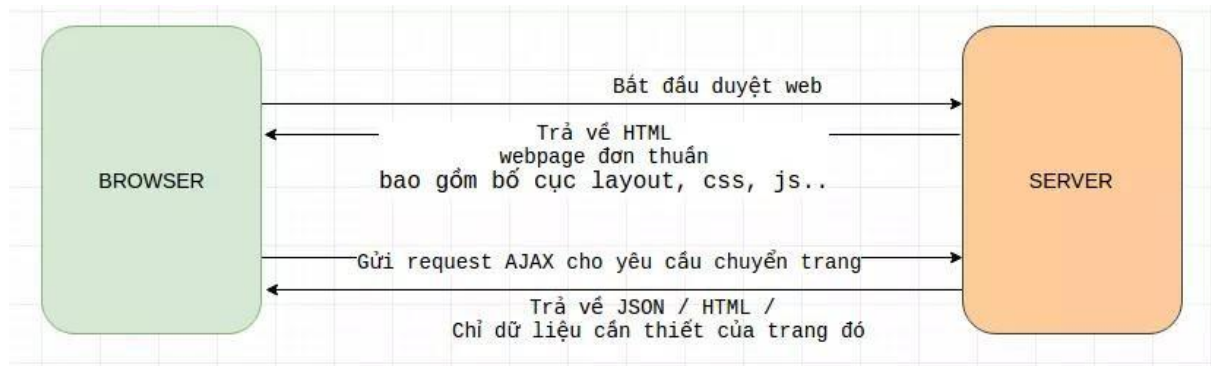
Cách tương tác giữa frontend và backend của website kiểu SPA được mô tả như sau:

- Khi người dùng truy cập vào các trang web, frontend sẽ là thành phần tiếp nhận request chứ không phải backend – hay có thể nói là web SPA thực hiện routing ở frontend thay vì backend như kiểu lập trình truyền thống.
- Sau khi tiếp nhận request, frontend sẽ biết được người dùng muốn sử dụng tính năng nào, tính năng này cần những dữ liệu gì, sau đó mới gửi request tới backend, yêu cầu backend trả về dữ liệu mong muốn.
- Frontend nhận dữ liệu từ backend (thường là dạng json), và dựa vào dữ liệu này để render ra nội dung trang web hoàn chỉnh.

Nếu như ở backend có kiến trúc MVC thì ở frontend cũng có thể áp dụng MVC để xử lý các yêu cầu phức tạp. Điển hình ở các framework SPA cũng hoạt động theo một kiến trúc cụ thể, ví dụ MVC (Angular), MV-VM (VueJS, ReactJS),...

2.2. Nguyên lý hoạt động

Khi truy cập vào một trang web kiểu SPA bất kỳ, server sẽ trả về toàn bộ code giao diện cho client. Tại client, SPA sẽ xử lý code nhận được và hiển thị ra một trang HTML đơn (**Client Side Rendering**), sau đó dựa trên yêu cầu của người dùng, SPA sẽ tiếp tục tải các HTML khác trong cùng một trang đó, các yêu cầu dữ liệu được thực hiện ngầm thông qua các Ajax, fetch API (cũng như các thư viện liên quan), kết quả trả về được lưu vào kho lưu trữ chung hoặc là gắn trực tiếp vào phần tử HTML cần hiển thị.



So sánh cụ thể hơn giữa hai kiểu lập trình

2.3. Các khó khăn

Mô hình này kém phù hợp với các thiết bị có hiệu năng từ trung bình trở xuống do mọi thao tác đều phải được xử lý trên cùng một trang. Đồng thời, trình duyệt cũng phải được kích hoạt Javascript.

Việc phát triển SPA không hề đơn giản, đòi hỏi đội ngũ lập trình viên phải nắm chắc cách thức sử dụng ngôn ngữ lập trình JavaScript cũng như các framework liên quan như VueJs, ReactJS,...

Tốc độ chạy trang lần đầu sẽ kém hơn so với web truyền thống.

Các kỹ thuật SEO nâng cao như tạo cấu trúc Silo sẽ không áp dụng được với SPA.

Nội dung không có độ chi tiết và cụ thể cao do bị giới hạn nội dung trên trang.

2.4. Ứng dụng

Một trang web khi xử lý hiển thị theo kiểu SPA thường sẽ đem lại trải nghiệm mượt mà, khiến người dùng có cảm giác như đang sử dụng một ứng dụng mobile chứ không

phải một trang web. Mặc khác có thể tiết kiệm về băng thông. Có thể tận dụng code để phát triển ứng dụng mobile. Vì thế mà lập trình theo hướng SPA đang là một xu hướng cho các dự án website hiện nay, mang lại sự cạnh tranh so với các thiết kế website truyền thống.

Các website lớn như *facebook.com*, *google.com*, *youtube.com*, *twitter.com*,... đều đang sử dụng kiểu lập trình SPA cho sản phẩm của mình.

Một số thư viện cũng như framework nổi tiếng giúp phát triển trang web theo hướng SPA có thể kể đến như: ReactJs, VueJs, AngularJs,...

3. ReactJs

3.1. Giới Thiệu React

ReactJs là một thư viện JavaScript phát triển bởi Facebook, đã trở thành một công cụ quan trọng trong phát triển ứng dụng web hiện đại. Được biết đến với sự linh hoạt và hiệu suất, ReactJS đưa ra nhiều khái niệm quan trọng mà nhà phát triển cần hiểu để tận dụng hết khả năng của nó. Dưới đây là một số cơ sở lý thuyết quan trọng về ReactJS:

3.2. Component

Trong thiết kế ứng dụng, đặc biệt là trong thiết kế ứng dụng SPA có những thành phần được lặp đi lặp lại, ví dụ: Header, Navbar, Footer, Button,... Việc copy sử dụng lại các đoạn code này gây khó khăn cho việc debug, code dài dòng khó theo dõi. Vì vậy component ra đời để lưu trữ các đoạn code chức năng thành một file riêng, sử dụng như một module gọi lại. Chia nhỏ như vậy ban đầu sẽ tốn thời gian, nhưng về lâu dài sẽ đạt hiệu quả cao bởi khả năng tận dụng lại code là rất lớn. Các file .tsx đóng vai trò component này được đặt trong thư mục Components.

Component là một trong những tiện ích mạnh mẽ nhất của ReactJs, là một thành phần quan trọng để tạo nên một ứng dụng SPA với hệ thống react (phản ứng) mạnh mẽ của ReactJs.

```
3 import App from './App.tsx'
4 import './index.css'
5 import { BrowserRouter } from 'react-router-dom'
6 import { AppProvider } from './contexts/app.context'
7 import { QueryClient, QueryClientProvider } from '@tanstack/react-query'
8 const queryClient = new QueryClient({
9   defaultOptions: {
10     queries: {
11       refetchOnWindowFocus: false,
12       retry: 0
13     }
14   }
15 })
16
17 ReactDOM.createRoot(document.getElementById('root')!).render(
18   <React.StrictMode>
19     <BrowserRouter>
20       <QueryClientProvider client={queryClient}>
21         <AppProvider>
22           <App />
23         </AppProvider>
24       </QueryClientProvider>
25     </BrowserRouter>
26   </React.StrictMode>
27 )
```

Kiến trúc Single File Component (SFC)

3.3. Virtual Dom(Dom ảo)

React sử dụng mô hình DOM ảo để cải thiện hiệu suất. Thay vì cập nhật trực tiếp DOM mỗi khi có thay đổi, React tạo một DOM ảo, so sánh với DOM thực tế và chỉ cập nhật những phần cần thiết. Điều này giúp giảm độ trễ và tối ưu hóa quá trình render.

- Hủy instance (destroying)

3.4. TSX

TSX là một phần không thể thiếu của React, cho phép mô tả giao diện người dùng trong JavaScript một cách dễ đọc và dễ hiểu hơn. TSX kết hợp cú pháp của XML với sức mạnh của JavaScript, tạo ra một cách tiếp cận mạnh mẽ cho việc xây dựng giao diện người dùng.

3.5. State

Khái niệm về "state" cho phép React theo dõi trạng thái của một component. Khi trạng thái thay đổi, React tự động cập nhật giao diện người dùng, tạo ra một trải nghiệm linh hoạt và đáp ứng.

3.6. Props

Props là cách chúng ta truyền dữ liệu giữa các component. Điều này tạo ra một mô hình dữ liệu một chiều, từ component cha đến component con, giúp quản lý và tái sử dụng mã nguồn một cách hiệu quả.

3.7. Lifecycle

React cung cấp một loạt các phương thức lifecycle cho các component, cho phép nhà phát triển can thiệp vào quá trình khởi tạo, cập nhật và hủy bỏ một component. Điều này làm cho việc quản lý vòng đời trở nên dễ dàng và linh hoạt.

3.8. Unidirectional Data Flow

Trong React, dữ liệu di chuyển một chiều từ component cha đến component con. Điều này giúp dễ dàng theo dõi và duy trì ứng dụng, đồng thời giảm nguy cơ xung đột dữ liệu.

3.9. Event Handling

React sử dụng mô hình xử lý sự kiện tương tự như HTML, nhưng cung cấp cải tiến để làm cho xử lý sự kiện trở nên thuận tiện hơn trong môi trường React.

3.10. Tổng quan

Điều này chỉ là một cái nhìn tổng quan về những khái niệm cơ bản của ReactJS. Trong bài tiểu luận này, chúng ta sẽ đàm phán chi tiết hơn về mỗi khía cạnh và cách chúng tương tác để xây dựng ứng dụng web hiệu quả và dễ bảo trì. Hãy khám phá sâu hơn vào thế giới của ReactJS và khám phá cách nó thay đổi cách chúng ta phát triển giao diện người dùng ngày nay.

4. Prisma

4.1. Tổng Quan Về Mysql

Prisma là một Object Relational Mapping (ORMs) được dùng để xây dựng các máy chủ như GraphQL Server, RESTful APIs, microservice, ... Prisma đơn giản là 1 layer nằm giữa Webserver và Database. Prisma giúp chúng ta giao tiếp với db một cách dễ dàng hơn.

Prisma bao gồm ba phần chính:

- + Prisma Client : Trình tạo truy vấn an toàn và được tạo tự động cho Node.js và TypeScript.
- + Prisma Migrate : Một hệ thống di chuyển và mô hình hóa dữ liệu khai báo.
- + Prisma Studio : Một GUI để xem và chỉnh sửa dữ liệu trong cơ sở dữ liệu của bạn.

Cách truyền thống mà Webserver giao tiếp với Database là thông qua các câu lệnh SQL Query như SELECT, UPDATE hay DELETE. Giờ đây, nhờ vào các công cụ ORMs nói chung và Prisma nói riêng. Chúng tạo ra một tầng abstraction giữa Webserver và Database. Điều này giúp cho lập trình viên dễ dàng trong việc thao tác với Database. Thay vì viết những câu lệnh SQL khô khan, có thể sai bất cứ lúc nào thì chúng ta có thể viết các hàm tương ứng.

Chương 3. ĐẶC TẢ CÁC YÊU CẦU CỦA HỆ THỐNG

1. ĐẶC TẢ YÊU CẦU VỀ CHỨC NĂNG


- Trong hệ thống có 4 loại người dùng:
 - + **Người dùng vắng lai (Public-User):** Người dùng không đăng kí tài khoản, vô trang web xem thông tin bác sĩ và tin tức.
 - + **Người dùng bình thường(User):** Thành viên có tài khoản trong hệ thống, người dùng có thể đặt lịch hẹn.
 - + **Người dùng quản trị (Admin):** tài khoản có quyền lực cao nhất trong hệ thống, sử dụng toàn bộ các chức năng quản lý.
 - + **Người dùng bác sĩ (doctor):** tài khoản có quyền thao tác xem lịch làm việc , khám bệnh cho bệnh nhân và lưu hồ sơ bệnh nhân, xuất bill
 - + **Người dùng dược sĩ (pharmarsist):** tài khoản có quyền thao tác quản lý thuốc
 - + **Người dùng hỗ trợ (supporter):** tài khoản có quyền duyệt lịch khám bệnh và tạo bệnh nhân và lịch hẹn

2. ĐẶC TẢ YÊU CẦU VỀ CƠ SỞ DỮ LIỆU CÁC TABEL

```
CREATE TABLE `roles` (  
  `id` varchar(255) PRIMARY KEY,  
  `name` varchar(255),  
  `created_at` DateTime,  
  `updated_at` DateTime  
);  
  
CREATE TABLE `users` (  
  `id` varchar(255) PRIMARY KEY,  
  `name` varchar(255),  
  `gender` varchar(255),  
  `avatar` varchar(255),  
  `address` varchar(255),  
  `phone` varchar(255),  
  `email` varchar(255) UNIQUE,  
  `password` varchar(255),  
  `is_patient` boolean,  
  `forgot_password_token` varchar(255),  
  `date_of_birth` date,  
  `role_id` varchar(255),  
  `created_at` DateTime,  
  `updated_at` DateTime  
);
```



-  CREATE TABLE `refresh_tokens` (
 `id` varchar(255) PRIMARY KEY,
 `token` text,
 `user_id` varchar(255),
 `exp` DateTime,
 `iat` DateTime,
 `created_at` DateTime,
 `updated_at` DateTime
);

-  CREATE TABLE `news` (
 `id` varchar(255) PRIMARY KEY,
 `title` varchar(255),
 `images` varchar(255),
 `description` varchar(255),
 `user_id` varchar(255),
 `category_id` varchar(255),
 `status` boolean,
 `created_at` DateTime,
 `updated_at` DateTime
);

-  CREATE TABLE `services` (
 `id` varchar(255) PRIMARY KEY,
 `name` varchar(255),
 `price` varchar(255),
 `created_at` DateTime,
 `updated_at` DateTime
);

-  CREATE TABLE `categories` (
 `id` varchar(255) PRIMARY KEY,
 `name` varchar(255),
 `created_at` DateTime,
 `updated_at` DateTime
);

```
64
65 • ⊖ CREATE TABLE `caterogies` (
66     `id` varchar(255) PRIMARY KEY,
67     `name` varchar(255),
68     `created_at` DateTime,
69     `updated_at` DateTime
70 ) ;
71
72 • ⊖ CREATE TABLE `staffSchedules` (
73     `id` varchar(255) PRIMARY KEY,
74     `date` DateTime,
75     `staff_id` varchar(255),
76     `shift` varchar(255),
77     `note` varchar(255),
78     `created_at` DateTime,
79     `updated_at` DateTime
80 ) ;
81
```

-  CREATE TABLE `medicines` (
 `id` varchar(255) PRIMARY KEY,
 `name` varchar(255),
 `manufacturer` varchar(255),
 `usage` varchar(255),
 `quantity` varchar(255),
 `price` varchar(255),
 `purchase_price` varchar(255),
 `created_at` DateTime,
 `updated_at` DateTime
);
-  CREATE TABLE `medicalRecord` (
 `id` varchar(255) PRIMARY KEY,
 `diagnosis` varchar(255),
 `note` varchar(255),
 `appointment_id` varchar(255),
 `created_at` DateTime,
 `updated_at` DateTime
);
-  CREATE TABLE `prescriptions` (
 `id` varchar(255) PRIMARY KEY,
 `appointment_id` varchar(255)
);
-  CREATE TABLE `bills` (
 `id` varchar(255) PRIMARY KEY,
 `medical_record_id` varchar(255),
 `status` varchar(255),
 `amount` varchar(255)
);

```
CREATE TABLE `doctorProfile` (  
  `doctor_id` varchar(255) PRIMARY KEY,  
  `certification` varchar(255),  
  `expricence` varchar(255),  
  `education` varchar(255),  
  `created_at` DateTime,  
  `updated_at` DateTime  
);
```

```
CREATE TABLE `appointments` (  
  `id` varchar(255) PRIMARY KEY,  
  `date` DateTime,  
  `doctor_id` varchar(255),  
  `patient_id` varchar(255),  
  `status` varchar(255),  
  `created_at` DateTime,  
  `updated_at` DateTime  
);
```

```
CREATE TABLE `medicalOnServices` (  
  `medical_record_id` varchar(255),  
  `service_id` varchar(255),  
  `created_at` DateTime,  
  `updated_at` DateTime  
);
```

```
CREATE TABLE `prescriptionDetail` (  
  `prescription_id` varchar(255),  
  `medicine_id` varchar(255),  
  `dosage` varchar(255),  
  `quantity` varchar(255)  
);
```

3. Mô tả chi tiết các table

Roles(Vai trò): id (Khóa chính), name, created_at, updated_at

Báo cáo tiểu luận tốt nghiệp

Users(Người dùng): id (Khóa chính), name, gender, avatar, address, phone, email, password, is_patient, forgot_password_token, date_of_birth, role_id (FK - roles), created_at, updated_at

refresh_tokens(Chứa các refresh token): id (Khóa chính), token, user_id (FK - users), exp, iat, created_at, updated_at

News(tin tức): id (Khóa chính), title, images, description, user_id (FK - users), category_id (FK - categories), status, created_at, updated_at

Services(dịch vụ trong phòng khám): id (Khóa chính), name, price, created_at, updated_at

Categories(danh mục bài viết): id (Khóa chính), name, created_at, updated_at

staffSchedules(lich làm việc của nhân viên): id (Khóa chính), date, staff_id (FK - users), shift, note, created_at, updated_at

Medicines(thuốc): id (Khóa chính), name, manufacturer, usage, quantity, price, purchase_price, created_at, updated_at

medicalRecord(hồ sơ bệnh án): id (Khóa chính), diagnosis, note, appointment_id (FK - appointments), created_at, updated_at

Prescriptions(đơn thuốc chi tiết): id (Khóa chính), appointment_id (FK - appointments)

Bills(hóa đơn thuốc): id (Khóa chính), medical_record_id (FK - medicalRecord), status, amount

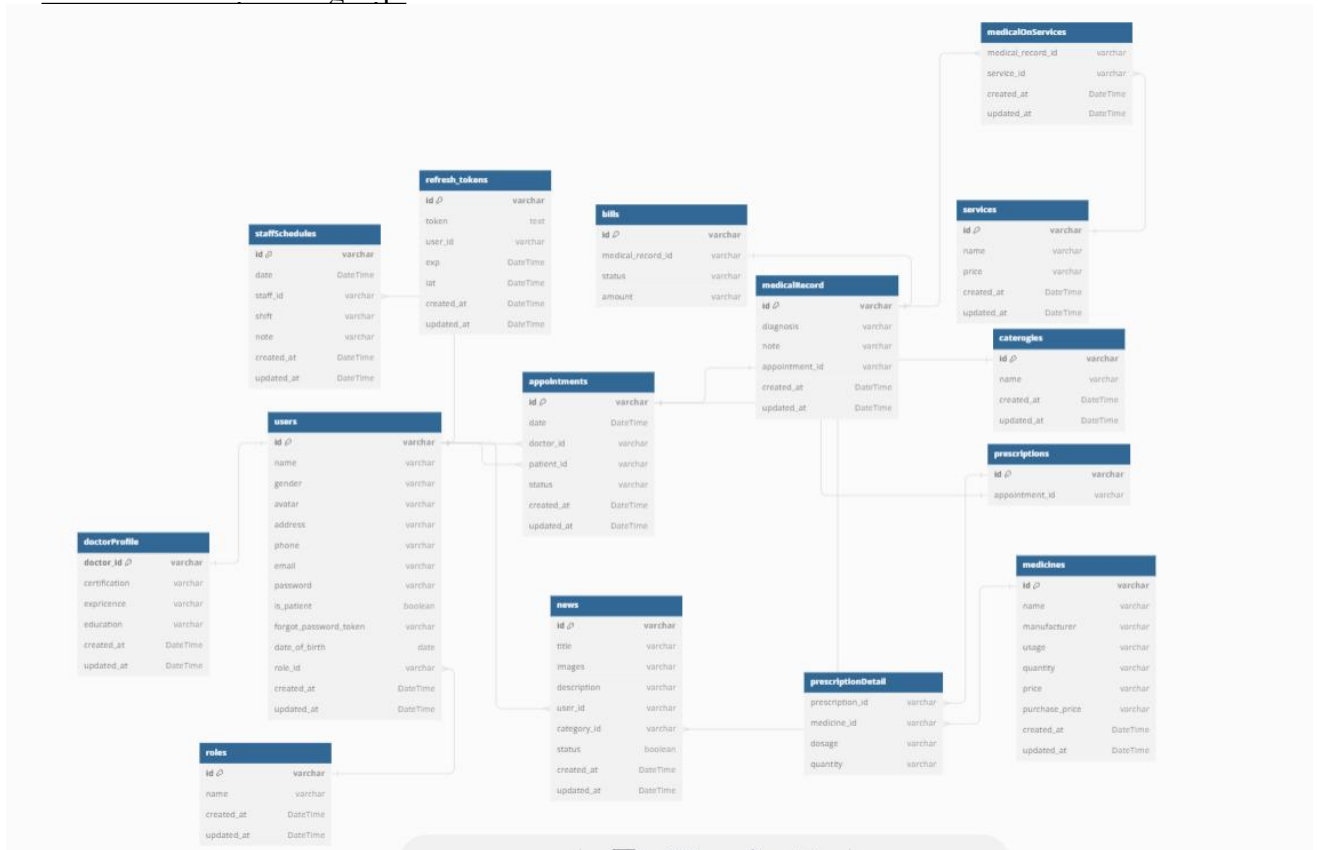
doctorProfile(hồ sơ bác sĩ): doctor_id (FK - users), certification, experience, education, created_at, updated_at

Appointments(Lịch hẹn): id (Khóa chính), date, doctor_id (FK - users), patient_id (FK - users), status, created_at, updated_at

medicalOnServices(Liên kết nhiều nhiều giữa hồ sơ và dịch vụ): medical_record_id (FK - medicalRecord), service_id (FK - services), created_at, updated_at

prescriptionDetail(đơn thuốc chi tiết): prescription_id (FK - prescriptions), medicine_id (FK - medicines), dosage, quantity

4. SƠ ĐỒ CDM



Chương 4. THIẾT KẾ GIAO DIỆN HỆ THỐNG


1. GIỚI THIỆU VỀ ỨNG DỤNG WEB

- **Tên gọi:** Shine smile
- **Phiên bản:** 1.0
- **Ngày hoàn thành:** 22/11/2023
- **Công nghệ sử dụng:**
 - Backend: *NodeJs, ExpressJs, Prisma, Typescript* .
 - Frontend: *ReactJs*
 - Database: *Mysql*
- **Kỹ thuật sử dụng:**
 - Xác thực người dùng: *JWT (Json Web Token)*

2. THIẾT KẾ GIAO DIỆN THEO NGƯỜI DÙNG

2.1. Người dùng quản trị

2.1.1. Giao diện login:



- Mô tả: Trang này gồm có 1 form để điền thông tin đăng nhập và click nút đăng nhập để bắt đầu trải nghiệm vai trò quản trị nếu bạn là người dùng quản trị

2.1.2. Giao diện làm việc của trang dashboard



- Quản lý danh mục
- Quản lý bài viết
- Quản lý nhân viên
- Quản lý lịch làm việc
- Quản lý bác sĩ
- Quản lý dịch vụ
- Quản lý lịch làm việc
- Quản lý lịch hẹn
- Quản lý thuốc
- Đăng xuất

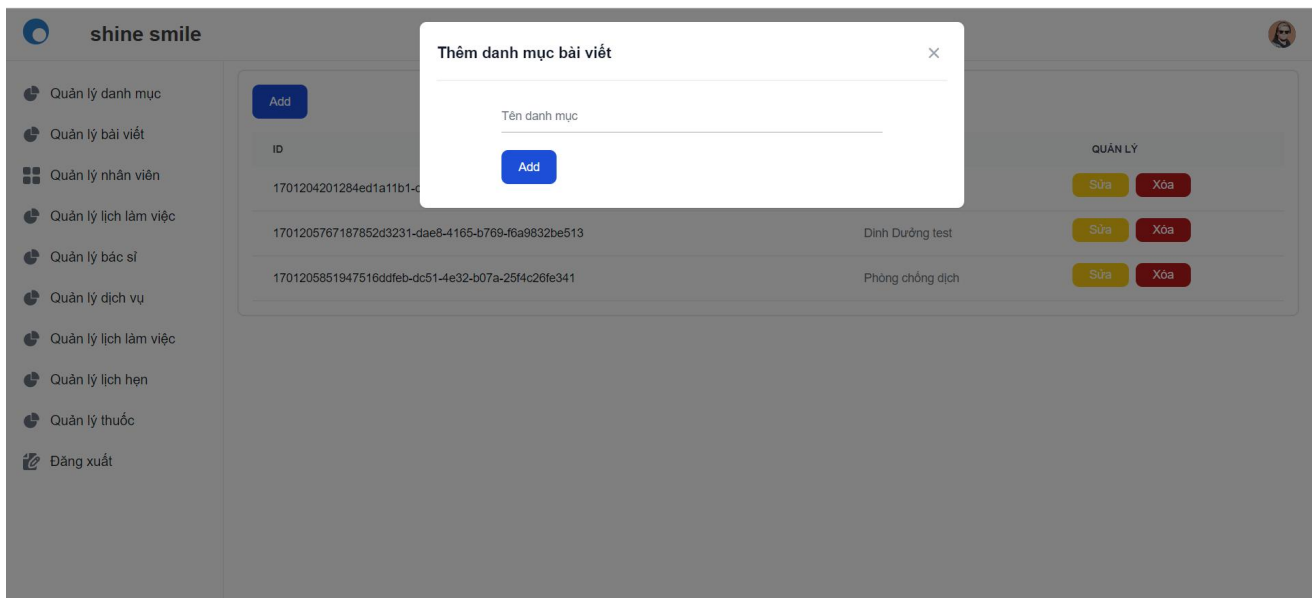
Add

ID	TÊN	QUẢN LÝ
1701204201284ed1a11b1-cb08-4594-a1d9-5aa276b5b81e	Cảm cúm	Sửa Xóa
1701205767187852d3231-dae8-4165-b769-f6a9832be513	Dinh Dưỡng test	Sửa Xóa
1701205851947516ddfeb-dc51-4e32-b07a-25f4c26fe341	Phòng chống dịch	Sửa Xóa

- Mô tả: Trang này hiển thị các danh mục sản phẩm trong hiện có trong hệ thống

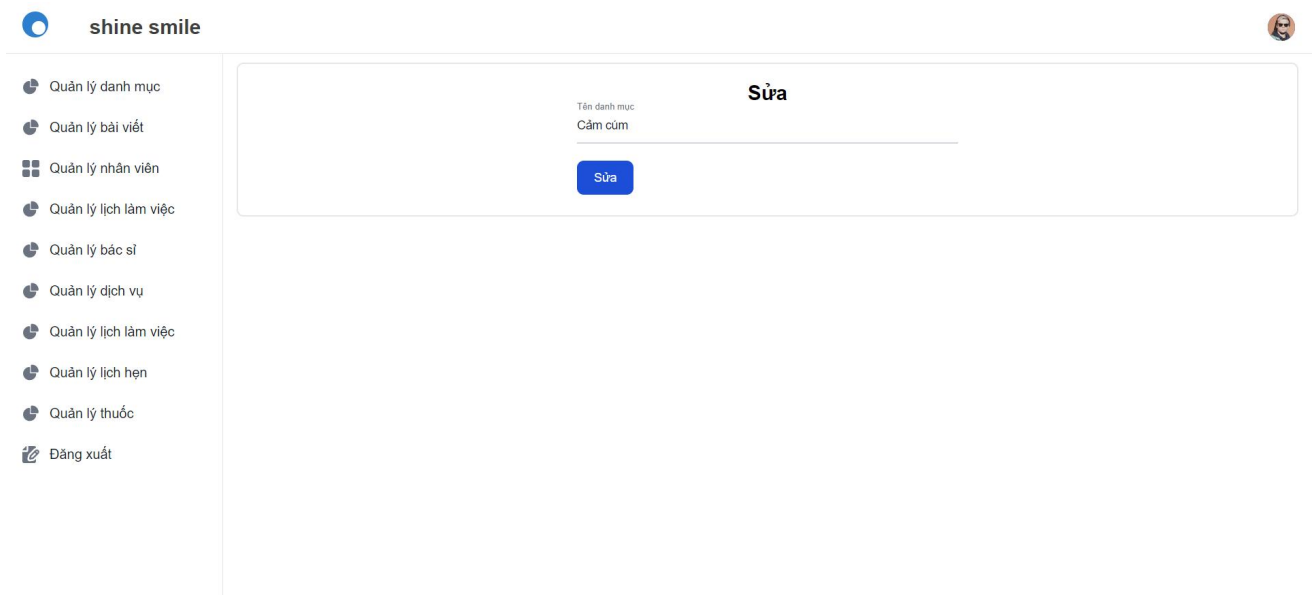
Báo cáo tiểu luận tốt nghiệp

2.1.3. Giao diện trang thêm danh mục:



- Mô tả: Trang này chứa thông tin trường tên danh mục sau khi nhập tên danh mục cần thêm và ấn submit thì tên danh mục sẽ thêm vào database

2.1.4. Giao diện trang sửa danh mục:

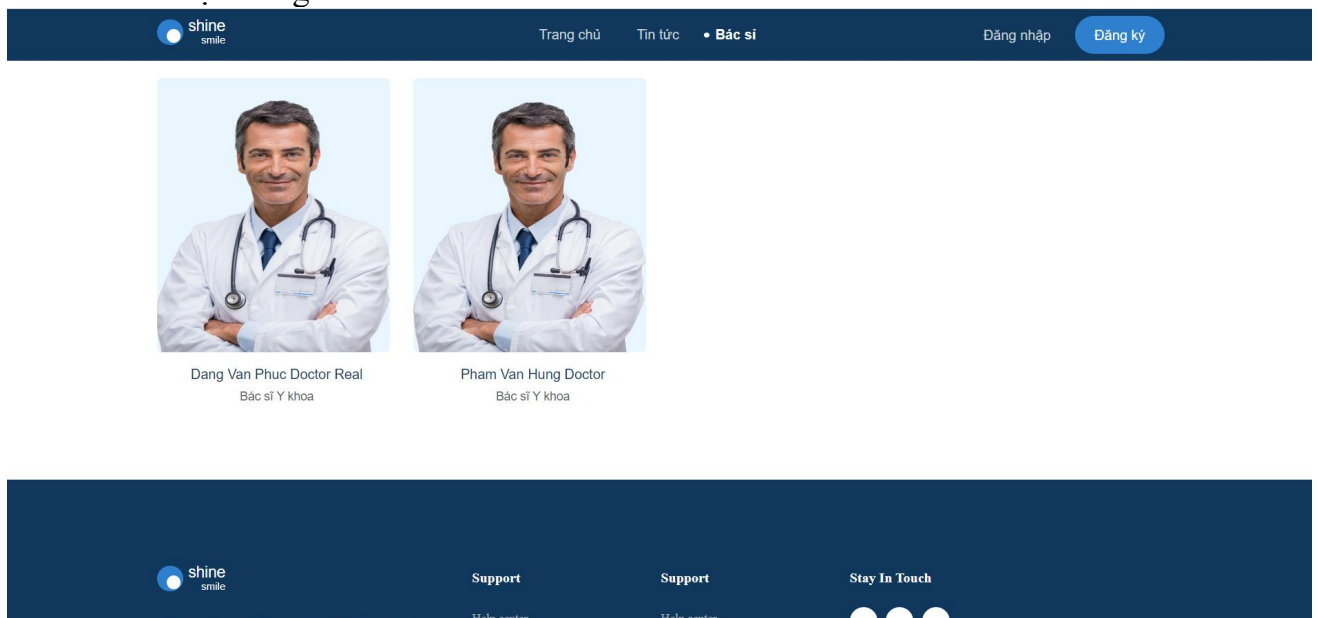


2.2. Người dùng khách hàng

2.2.1. Giao diện trang chủ:



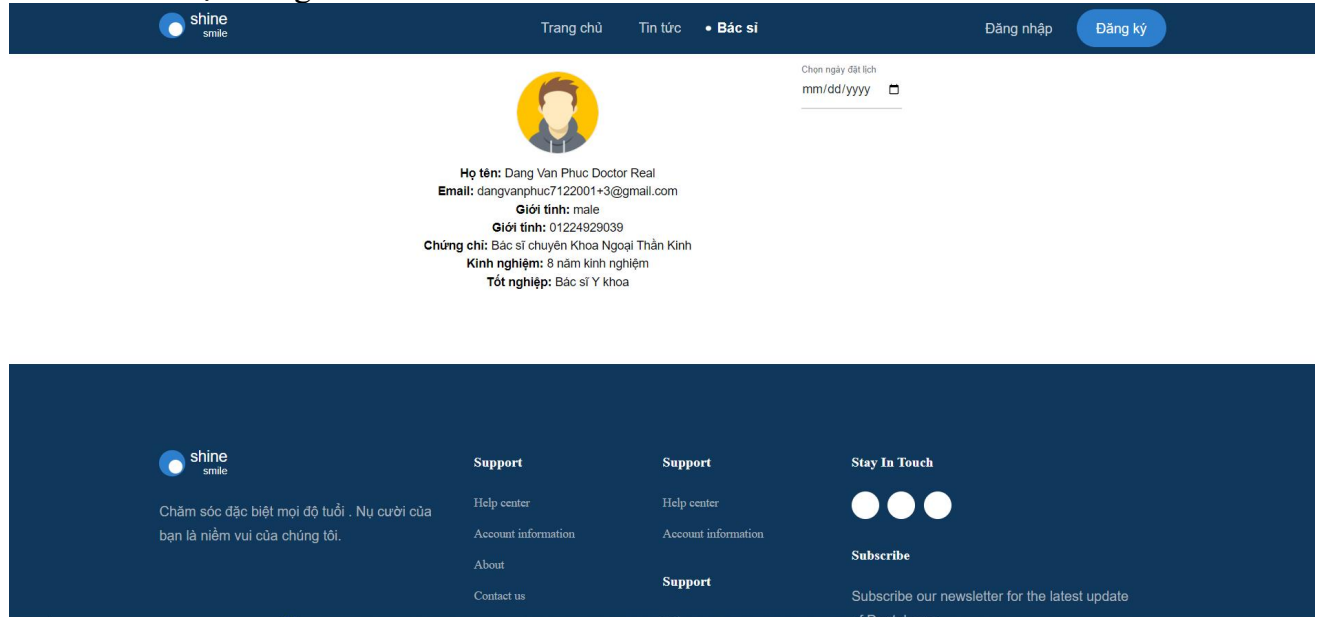
- Mô tả: Trang chủ là trang chứa các thông tin sơ lược về phòng khám .
- 2.2.2. Giao diện trang bác sĩ



- Mô tả: Trang chứa các bác sĩ của phòng khám

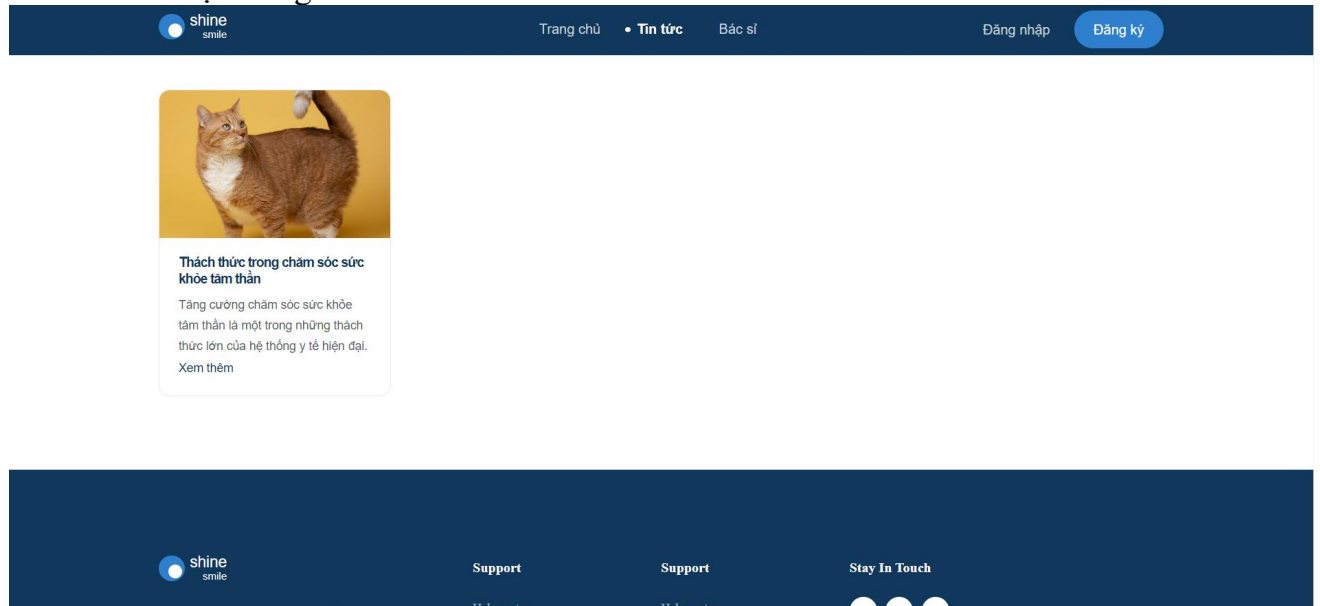
Báo cáo tiểu luận tốt nghiệp

2.2.3. Giao diện trang bác sĩ



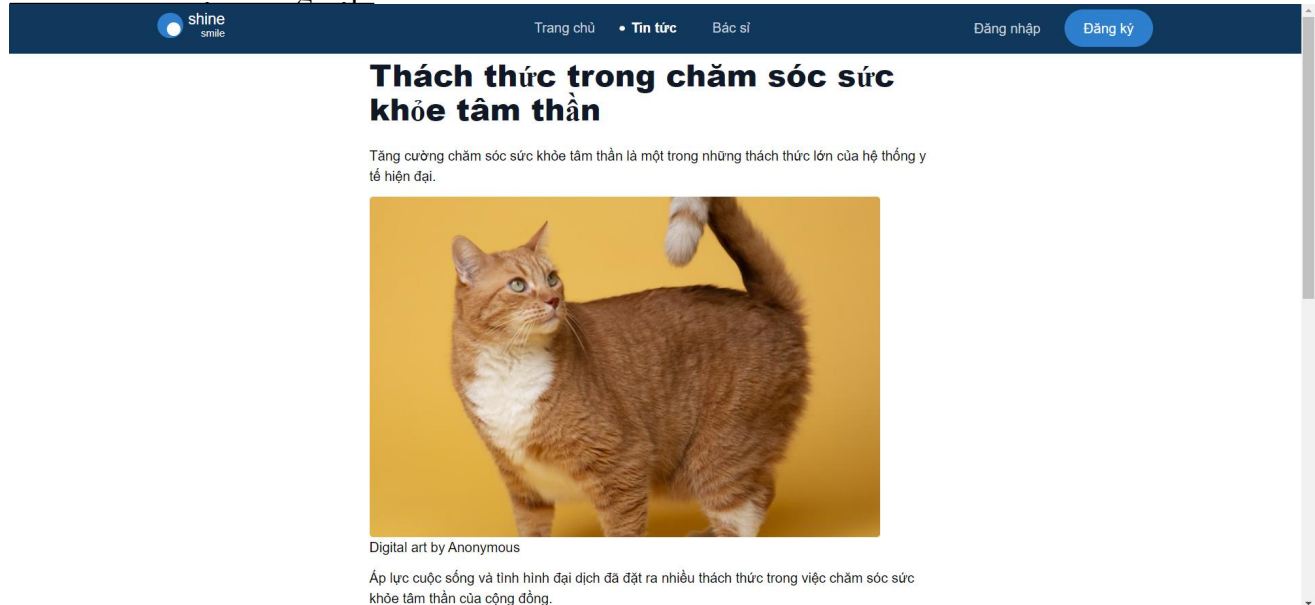
- Mô tả: Trang thông tin chi tiết của bác sĩ nếu người dùng đăng nhập thì có thể đặt lịch

2.2.4. Giao diện trang tin tức

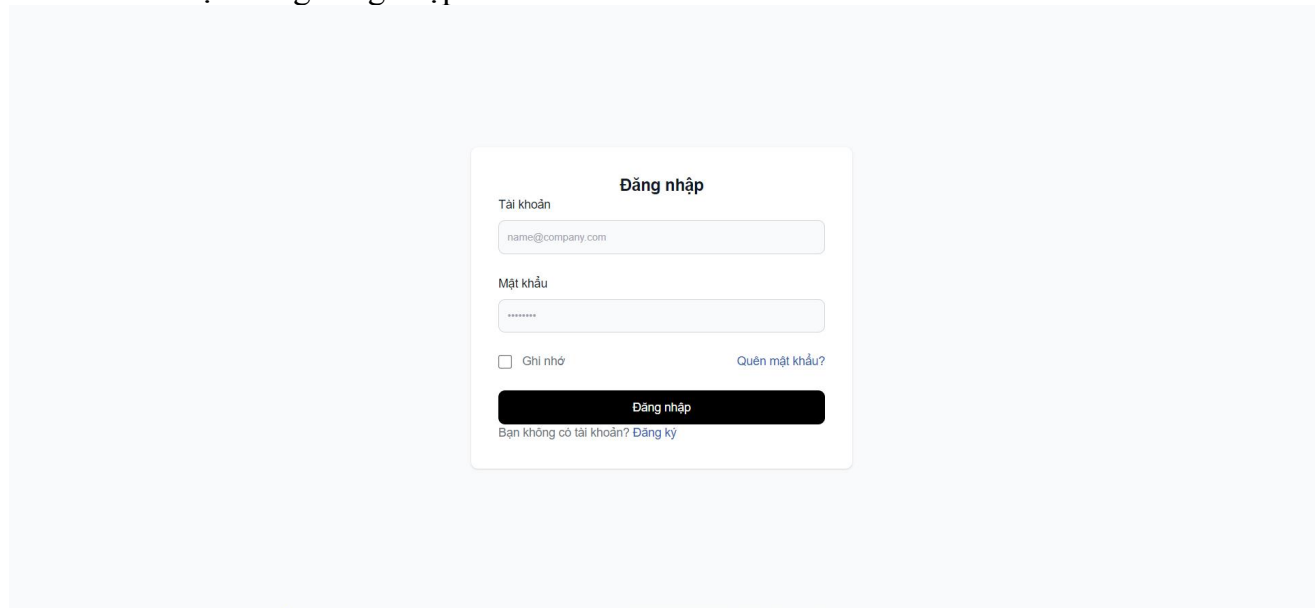


- Mô tả: Trang chứa các bài báo.

2.2.5. Giao diện trang tin tức chi tiết.

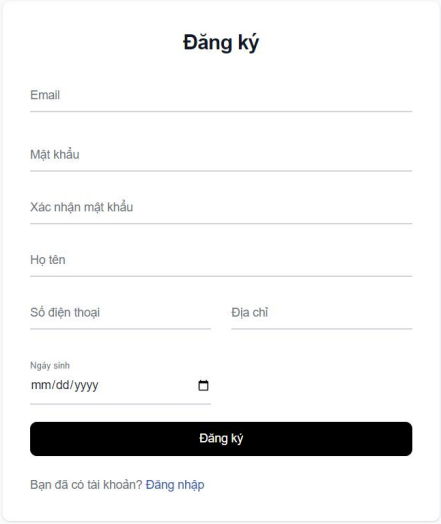


2.2.6. Giao diện trang đăng nhập:



- Mô tả: Trang đăng nhập là 1 form để người dùng nhập vào đó để đăng nhập vào tài khoản của họ

2.2.7. Giao diện đăng ký



The image shows a registration form titled "Đăng ký" (Register) centered on a light gray background. The form is a white card with the following fields: "Email", "Mật khẩu" (Password), "Xác nhận mật khẩu" (Confirm password), "Họ tên" (Full name), "Số điện thoại" (Phone number), "Địa chỉ" (Address), and "Ngày sinh" (Date of birth) with a calendar icon. Below the fields is a black "Đăng ký" button. At the bottom, there is a link: "Bạn đã có tài khoản? Đăng nhập" (Do you have an account? Log in).



- Mô tả: Giao diện này giành cho bệnh nhân đăng ký tài khoản


2.2.8. Giao diện hồ sơ cá nhân

The screenshot shows a web interface for a user profile. At the top is a dark blue header with the 'shine smile' logo and navigation links: 'Trang chủ', 'Tin tức', and 'Bác sĩ'. Below the header, on the left, is a circular profile picture placeholder with the 'aws' logo. To its right, the user's name 'Họ tên: Dang Van Phuc 3' and email 'Email: dangvanphuc7122001+7@gmail.com' are displayed. A black button labeled 'Đăng xuất' is positioned below this information. On the right side of the page, there are three buttons: 'Lịch sử khám', 'Hồ sơ', and 'Đổi mật khẩu'. The 'Hồ sơ' button is active, showing a form titled 'Hồ sơ'. The form contains the following fields: 'Họ tên' (Dang Van Phuc 3), 'Số' (01224929039), 'Email' (dangvanphuc7122001+7@gmail.com), 'Địa chỉ' (Dong Thap), 'Ngày sinh' (11/20/2023) with a calendar icon, 'Chọn giới tính' (Other) with a dropdown arrow, and 'Upload file' with a 'Choose File' button and 'No file chosen' text. At the bottom of the form is a blue button labeled 'Cập nhật hồ sơ'.

- Mô tả: Đây là trang hồ sơ cá nhân có thể update và đăng xuất

2.2.9. Giao diện trang lịch sử các cuộc hẹn

Trang chủ Tin tức Bác sĩ 




Họ tên: Dang Van Phuc 3
Email: dangvanphuc7122001+7@gmail.com

[Lịch sử khám](#) [Hồ sơ](#) [Đổi mật khẩu](#)

NGÀY	HỌ TÊN BÁC SĨ	HỌ TÊN BỆNH NHÂN	TRẠNG THÁI LỊCH HẸN
2023-12-16	Dang Van Phuc Doctor Real	Dang Van Phuc 3	unconfirmed



Previous **1** 2 Next


[Đăng xuất](#)

Support Support Stay In Touch

Đây là trang chi tiết các cuộc hẹn ngày nào , trạng thái thể nào

2.2.10. Giao diện trang đổi mật khẩu

Trang chủ Tin tức Bác sĩ 



Họ tên: Dang Van Phuc 3
Email: dangvanphuc7122001+7@gmail.com

[Lịch sử khám](#) [Hồ sơ](#) [Đổi mật khẩu](#)

Đổi mật khẩu


Mật khẩu cũ

Mật khẩu mới

Xác nhận mật khẩu mới

[Đổi mật khẩu](#)

[Đăng xuất](#)

Support Support Stay In Touch

CHƯƠNG 5. KẾT LUẬN

1. Kết Quả Đạt Được

Sau một thời gian tìm hiểu và học hỏi thì nay em đã có thêm nhiều kinh nghiệm trong thiết kế website như:

- Hiểu được quy trình thiết kế một trang web theo hướng SPA với MEVN Stack, nắm được cách giao tiếp trao đổi dữ liệu giữa frontend và backend cũng như các xử lý liên quan.
- Học hỏi thêm kinh nghiệm về phân tích, thiết kế hệ thống.
- Có thêm kinh nghiệm trong thiết kế giao diện với Bootstrap Grid.

2. Hạn Chế

Ứng dụng được viết ra chưa được bảo mật tốt khẩu bảo mật còn kém , chạy trên máy tính cá nhân, chưa đưa ứng dụng lên host để sử dụng thực tế nên có thể còn nhiều lỗi logic. Chỉ được 1 phần frontend, chưa được nhiều

Do mới tìm hiểu công nghệ SERN Stack nên các xử lý còn ở mức cơ bản, chưa thể xây dựng theo hướng chuyên sâu.

3. Hướng Phát Triển Đề Tài

3.1. Các dự định sau khi hoàn thành đề tài

- Đưa ứng dụng lên Hosing để triển khai đến với bạn bè.
- Lấy ý kiến bạn bè để sửa chữa các lỗi logic, cũng như sửa chữa các điểm không hợp lý trong giao diện.
- Học thêm 1 số thư viện và làm thêm 1 số tính năng để website hoàn thiện hơn (gửi mail , openssl, passportJs)

3.2. Các tính năng nghiên cứu phát triển trong tương lai

- Nghiên cứu thêm thư viện TessaractJs để sử dụng hiệu quả hơn.
- Phát triển thêm tính năng cho thành viên đăng bài viết . Tính năng bình luận cũng như trò chuyện trực tuyến để tăng sự tương tác của người dùng với nhau.

Link github: dẫn đến mã nguồn

<https://github.com/vanphuc7122001/private-clinic-management>

Tài Liệu Tham Khảo

- [1]. <https://react.dev/>
- [2]. <https://www.typescriptlang.org/>
- [3] . <https://nodejs.org/en>
- [4] <https://www.prisma.io/docs/concepts/components/prisma-client/crud>