

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN GIỮA KÌ MÔN
PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS

REAL-TIME CHAT APPLICATION

Người hướng dẫn: **THẦY TRƯỞNG HỒNG PHÁT**

Người thực hiện: **NGUYỄN TẤN LỰC – 52000078**

NGUYỄN TIẾN ĐẠT – 52000026

Lớp : **20050201**

Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN
PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS**

REAL-TIME CHAT APPLICATION

Người hướng dẫn: **THẦY TRƯỞNG HỒNG PHÁT**
Người thực hiện: **NGUYỄN TẤN LỰC - 52000078**
NGUYỄN TIẾN ĐẠT – 52000026
Lớp : **20050201**
Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn sâu sắc đến thầy Trương Hồng Phát đã nhiệt tình giảng dạy chúng em trong môn học Phát triển ứng dụng web với Nodejs. Qua môn học này, chúng em đã học được nhiều kiến thức mới như cách tạo trang web bằng NodeJS, HTML, CSS, tăng tương tác với người dùng bằng ngôn ngữ Javascript, xây dựng trang web động bằng cơ sở dữ liệu NodeJS và MongoDB. Vấn đề bảo mật trong trang web..

Em cũng muốn gửi lời cảm ơn đến các bạn trong nhóm của em vì đã cùng nhau hoàn thành dự án thực tế trong môn học này. Chúng em đã học được nhiều từ nhau và phát triển các kỹ năng làm việc nhóm cũng như trao đổi thông tin một cách hiệu quả. Cuối cùng, chúng em muốn gửi lời cảm ơn đến Ban giám hiệu và các giảng viên khác của trường đã cung cấp cho chúng em môi trường học tập và nghiên cứu tốt nhất. Chúng em tin rằng những kiến thức và kỹ năng học được từ môn học này sẽ giúp chúng em phát triển trong sự nghiệp của mình trong tương lai.

Xin cảm ơn.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của thầy Dương Hữu Phúc. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 27 tháng 04 năm

Tác giả

(ký tên và ghi rõ họ tên)

Lực

Nguyễn Tấn Lực

Đạt

Nguyễn Tiến Đạt

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Phần đề tài, chúng em có tìm hiểu về Web Socket, bao gồm các hoạt động sau:

Sử dụng web socket để tạo kết nối liên tục giữa các client và server. HTML và CSS được dùng để xây dựng giao diện người dùng. Khi server nhận được tin nhắn từ client này, nó sẽ dùng web socket để gửi tin nhắn đến tất cả các client khác để cập nhật.

Client sẽ gửi tin nhắn đến server bằng JavaScript. Server sẽ duy trì danh sách các client kết nối và lưu trữ tin nhắn. Khi server nhận được tin nhắn, nó sẽ gửi đến tất cả client khác để cập nhật cuộc trò chuyện.

Việc sử dụng web socket cho phép tạo ứng dụng chat thời gian thực với khả năng cập nhật nhanh chóng. HTML, CSS và JavaScript được dùng để xây dựng giao diện và chức năng client.

Phần lý thuyết chúng em có tìm hiểu, nêu khái niệm về front-end, back-end, công nghệ Node.js và Express.js.

MỤC LỤC

LỜI CẢM ƠN	i
TÓM TẮT	iv
MỤC LỤC	1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ	3
CHƯƠNG 1 – PHẦN ĐỀ TÀI	4
1.1 Lý thuyết và các công nghệ được sử dụng trong đề tài	4
1.1.1 WebSocket và Socket.io	4
1.1.2 NodeJS	6
1.1.3 ReactJS	6
1.1.4 MongoDB	7
1.1.5 HTML, CSS và JavaScript	7
1.2 Điểm mạnh, điểm yếu của các công nghệ	8
1.2.1 WebSocket và Socket.io	8
1.2.2 NodeJS	9
1.2.3 ReactJS	10
1.2.4 MongoDB	11
1.3 Kiến trúc tổng quan trong hệ thống:	12
1.4 Kết quả sản phẩm:	13
1.5 Hạn chế và hướng phát triển:	15
1.5.1 Hạn chế	15
1.5.2 Hướng phát triển	15
CHƯƠNG 2 – PHẦN LÝ THUYẾT	16
2.1 Câu 1:	16
2.1.1: Câu hỏi:	16
2.1.2: Trả lời:	16
2.2 Câu 2:	19

2.2.1: Câu hỏi.....	19
2.2.2: Trả lời.....	19
2.3 Câu 3:	22
2.3.1: Câu hỏi.....	22
2.3.2: Trả lời.....	23
2.4 Câu 4:	25
2.4.1: Câu hỏi.....	25
2.4.2: Trả lời.....	25

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 1.1 Sự khác nhau giữa giao thức WebSocket và HTTP truyền thống (nguồn tham khảo).....	5
Hình 1.2 Cách thức giao tiếp giữa Client và Server của socket.io (nguồn tham khảo) ...	6
Hình 1.3 Trang chủ của trang web	13
Hình 1.4 Trang đăng nhập của trang web	13
Hình 1.5 Trang đăng ký của trang web	14
Hình 1.6 Trang chat và nội dung chat (chat riêng tư)	14
Hình 1.7 Trang chat và nội dung chat (chat trong nhóm công khai)	15
Hình 1.8 Sơ đồ đánh front-end và back-end (nguồn tham khảo).....	18
Hình 1.9 Sự tương tác giữa front-end và back-end trong hệ thống Node.JS (nguồn tham khảo).....	21

DANH MỤC BẢNG

Bảng 1.1 Bảng đánh giá front-end và back-end.....	19
Bảng 1.2 Bảng đánh giá front-end và back-end trong Node.JS	22
Bảng 1.3 Bảng đánh giá front-end và back-end trong Express.js	25

CHƯƠNG 1 – PHẦN ĐỀ TÀI

1.1 Lý thuyết và các công nghệ được sử dụng trong đề tài

Ứng dụng chat trực tuyến là một ứng dụng thời gian thực, yêu cầu việc truyền tải dữ liệu giữa các máy khách và máy chủ một cách nhanh chóng và tin cậy. Những công nghệ, kỹ thuật được sử dụng trong đề tài này là:

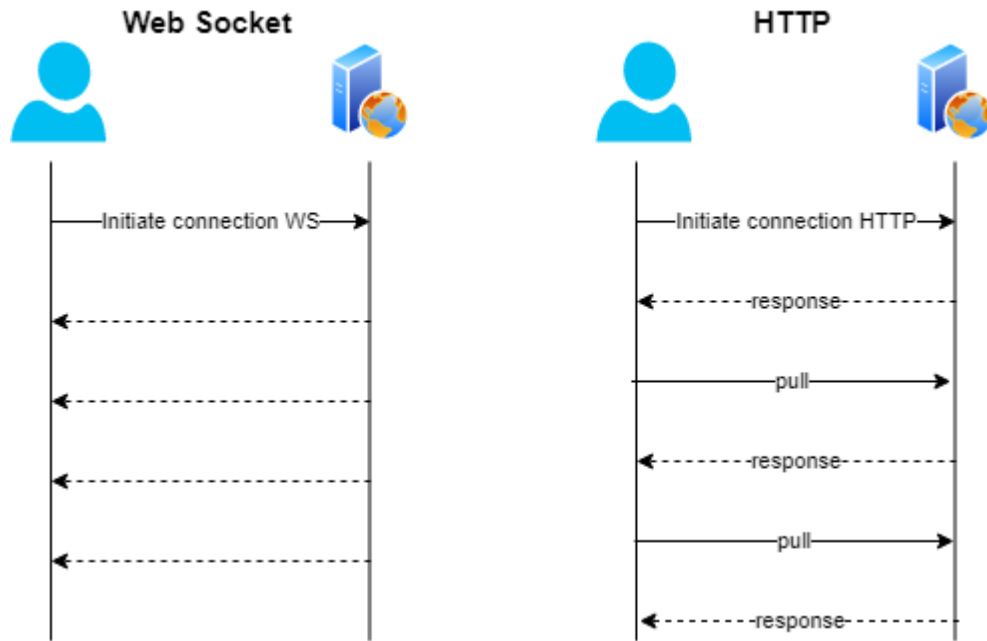
1.1.1 *WebSocket và Socket.io*

- WebSocket:

WebSocket là một giao thức truyền tải dữ liệu hai chiều giữa Client và Server cho phép truyền tải dữ liệu theo thời gian thực bằng cách sử dụng một TCP socket để tạo một kết nối hiệu quả và ít tốn kém. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.

Ví dụ, trong một ứng dụng chat trực tuyến, các thông điệp được gửi từ máy khách đến máy chủ hoặc ngược lại. Với giao thức HTTP truyền thống, mỗi lần máy khách muốn gửi một thông điệp đến máy chủ, nó phải tạo ra một kết nối mới với máy chủ. Điều này tốn thời gian và tài nguyên, và có thể làm chậm quá trình truyền tải thông điệp.

Do đó, với WebSocket, khi kết nối ban đầu được thiết lập giữa máy khách và máy chủ, các thông điệp có thể được truyền tải giữa hai bên mà không cần tạo ra các kết nối mới. Điều này giúp cho việc truyền tải thông điệp trong ứng dụng chat trực tuyến trở nên nhanh chóng và hiệu quả hơn.

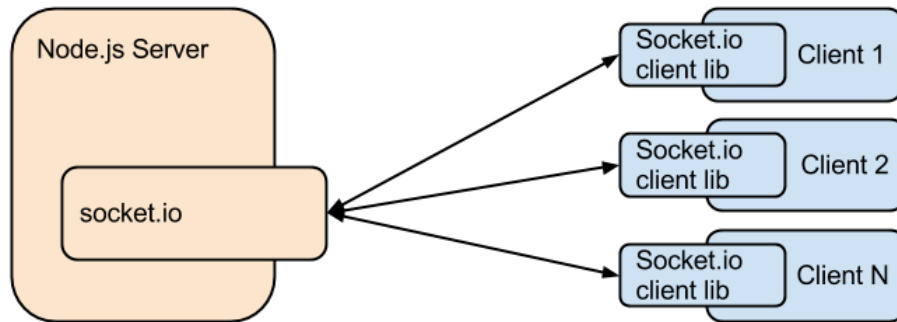


Hình 1.1 Sự khác nhau giữa giao thức WebSocket và HTTP truyền thống (nguồn tham khảo)

- Socket.io:

Socket.IO là một thư viện JavaScript được dùng để xây dựng ứng dụng realtime trên nền tảng WebSocket. Socket.IO cho phép thiết lập các kết nối socket giữa client và server, giúp truyền thông tin giữa hai bên một cách nhanh chóng và liên tục.

Trong ứng dụng chat realtime, Socket.IO được sử dụng để thiết lập các kết nối socket giữa các client và server, để các tin nhắn được gửi và nhận một cách nhanh chóng và liên tục giữa các người dùng. Khi một người dùng gửi tin nhắn, Socket.IO sẽ gửi tin nhắn đó đến server, sau đó server sẽ truyền tin nhắn đó đến tất cả các client khác đang kết nối đến ứng dụng chat.



Hình 1.2 Cách thức giao tiếp giữa Client và Server của socket.io (nguồn tham khảo)

1.1.2 NodeJS

NodeJS là một môi trường runtime chạy JavaScript đa nền tảng và có mã nguồn mở, được sử dụng để chạy các ứng dụng web bên ngoài trình duyệt của client.

Node.js cũng có một số thư viện hỗ trợ để xây dựng ứng dụng Chat Realtime như Socket.io để truyền thông tin trong thời gian thực giữa các client và server thông qua WebSockets. Thư viện này cũng cung cấp nhiều tính năng hữu ích cho việc xây dựng các ứng dụng chat như xác thực người dùng, gửi tin nhắn, tạo room, phát âm thanh, video,...

Ngoài ra, Node.js còn được tích hợp với các cơ sở dữ liệu phổ biến như MongoDB, MySQL, Postgres, ... giúp dễ dàng lưu trữ và quản lý dữ liệu liên quan đến chương trình Chat Realtime

1.1.3 ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, ra đời vào năm 2013. Mục đích của việc tạo ReactJS là tạo ra các ứng dụng web nhanh, hiệu quả và hấp dẫn với nỗ lực viết mã tối thiểu. Mục tiêu chính của ReactJS là bất kỳ trang web nào sử dụng ReactJS phải mượt mà, nhanh chóng, khả năng mở rộng cao và dễ thực hiện.

Một trong những điểm nổi bật nhất của ReactJS đó là việc render dữ liệu không chỉ thực hiện được trên tầng Server mà còn có thể thực hiện ở dưới Client.

Các thành phần chính của ReactJS:

- Redux: là một công cụ dự đoán phép tính trong ứng dụng Javascript. Redux giúp các ứng dụng được viết hoạt động nhất quán và có thể hoạt động trong các môi trường khác nhau như client, server và native
- Virtual DOM (Virtual Document Object Model): là một định dạng dữ liệu JavaScript nhẹ được dùng để thể hiện nội dung của DOM tại một thời điểm nhất định nào đó. Nó có tất cả các thuộc tính giống như DOM nhưng không có khả năng tương tác lên màn hình như DOM. Virtual DOM không được tạo ra bởi React tuy nhiên nó được React sử dụng và cung cấp miễn phí.

1.1.4 MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL), được phát triển bởi công ty MongoDB, Inc. MongoDB được phát triển dựa trên ngôn ngữ lập trình C++, và hỗ trợ các ngôn ngữ lập trình phổ biến như Java, Python, Ruby, và nền tảng Node.js.

Đặc điểm nổi bật của MongoDB:

- Dữ liệu được lưu trữ dưới dạng tài liệu JSON
- Hỗ trợ các thao tác CRUD (Create, Read, Update và Delete) để quản lý dữ liệu
- Có thể tích hợp với các ngôn ngữ lập trình như Java, Python, Ruby, và nền tảng Node.js

1.1.5 HTML, CSS và JavaScript

HTML (HyperText Markup Language) là ngôn ngữ đánh dấu được sử dụng để xây dựng cấu trúc và nội dung của một trang web. Trong ứng dụng chat, HTML được sử dụng để tạo các thành phần giao diện người dùng như các nút bấm, hộp văn bản, khu vực hiển thị tin nhắn, v.v.

CSS (Cascading Style Sheets) là ngôn ngữ để thiết kế và trình bày các thành phần giao diện người dùng trên trang web. CSS được sử dụng để định dạng các thành phần

HTML, tạo ra các kiểu hiển thị khác nhau cho các thành phần và giúp tạo ra giao diện ứng dụng chat hấp dẫn và dễ sử dụng.

JavaScript là một ngôn ngữ lập trình được sử dụng để tạo ra các tương tác trên trang web, bao gồm các ứng dụng realtime như chat. JavaScript cho phép tương tác với các thành phần HTML và CSS, thực hiện các thao tác đọc và ghi dữ liệu từ server, và hiển thị các tin nhắn trên trang web ngay lập tức khi người dùng gửi chúng.

1.2 Điểm mạnh, điểm yếu của các công nghệ

1.2.1 WebSocket và Socket.io

- WebSocket:

Ưu điểm

- WebSockets cung cấp khả năng giao tiếp hai chiều mạnh mẽ, có độ trễ thấp và dễ xử lý lỗi. Không cần phải có nhiều kết nối.
- API dễ sử dụng trực tiếp mà không cần bất kỳ các tầng bổ sung nào

Nhược điểm

- Chưa được tất cả các trình duyệt hỗ trợ
- Không có phạm vi yêu cầu nào. Khi bắt đầu một request, nó sẽ thiết lập một contest được ràng buộc với luồng request. Khi request đó kết thúc, bộ lọc hủy bỏ contest này

=> WebSocket phù hợp với:

- Phát triển ứng dụng real-time
- Phát triển ứng dụng IoT
- Phát triển các ứng dụng đa nền tảng

- Socket.io:

Ưu điểm

- Socket.IO hỗ trợ nhiều giao thức truyền thông, giúp tăng tính linh hoạt và khả năng tương thích của Socket.IO với các trình duyệt và thiết bị khác nhau.
- Cho phép các kết nối bị mất hoặc bị gián đoạn do những vấn đề về mạng trở lại và tiếp tục hoạt động một cách bình thường mà không cần phải thực hiện các thao tác phức tạp.
- Cho phép lập trình viên tạo ra các sự kiện tùy chỉnh để truyền thông tin giữa client và server, giúp cho việc phát triển ứng dụng trở nên dễ dàng hơn

Nhược điểm

- Có thể tạo ra độ trễ khi truyền thông tin giữa client và server, ảnh hưởng đến tính năng thời gian thực của ứng dụng
- Không có tính năng bảo mật mặc định
- Khó phát hiện lỗi để xử lý

=> Socket.io phù hợp với:

- Các ứng dụng web hoặc di động có tính tương tác cao và cần cập nhật dữ liệu ngay lập tức.
- Các ứng dụng chat, trò chơi trực tuyến, ứng dụng đồng bộ hóa dữ liệu, và các ứng dụng cần giao tiếp nhanh giữa client và server

1.2.2 NodeJS

Ưu điểm

- Input/Output hướng đến các sự kiện không đồng bộ, cho phép xử lý nhiều yêu cầu đồng thời
- Sử dụng JavaScript – một ngôn ngữ lập trình dễ học.
- NPM(Node Package Manager) và module Node đang được phát triển tốt hơn

- Có một cộng đồng lớn, dễ dàng chia sẻ, hỗ trợ nhau

Nhược điểm

- Khó thao tác với cơ sở dữ liệu quan hệ.
- Cần có kiến thức tốt về JavaScript
- Mỗi callback sẽ đi kèm với rất nhiều callback lồng nhau khác
- Không phù hợp với các tác vụ đòi hỏi nhiều CPU.

=> NodeJS phù hợp với:

- Phát triển ứng dụng web real-time nhờ vào cơ chế xử lý sự kiện không đồng bộ của nó, NodeJS cho phép xử lý hàng nghìn kết nối đồng thời mà không gây ra tình trạng block hoặc tắc nghẽn
- Xây dựng ứng dụng yêu cầu dữ liệu lớn: NodeJS cho phép xử lý dữ liệu lớn một cách hiệu quả nhờ vào việc sử dụng thư viện xử lý bất đồng bộ.
- Xây dựng ứng dụng API: NodeJS được sử dụng rộng rãi để xây dựng các ứng dụng API bởi vì nó cho phép xử lý các request HTTP một cách nhanh chóng và hiệu quả.

1.2.3 ReactJS

Ưu điểm

- Hiệu năng cao: ReactJS sử dụng Virtual để tối ưu hóa quá trình render trên trình duyệt. Virtual DOM giúp cập nhật chỉ những thành phần thay đổi, giảm thiểu số lần render của DOM thực tế, từ đó giảm thiểu thời gian render và tăng hiệu năng của ứng dụng.
- Sử dụng JSX, giúp ReactJS kết hợp HTML và JavaScript để làm cho code dễ đọc, dễ hiểu hơn và giảm thiểu lỗi trong quá trình phát triển.

Nhược điểm

- Khó học hơn so với các framework khác

- Phải sử dụng nhiều công nghệ kết hợp để phát triển ứng dụng hoàn chỉnh, khiến cho quá trình phát triển trở nên phức tạp hơn
- ReactJS chỉ là một thư viện, không phải là một framework hoàn chỉnh

=> **ReactJS phù hợp với:**

- Xây dựng các ứng dụng web có nhiều tương tác qua lại giữa các người dùng
- Xây dựng các ứng dụng web động và đơn trang (single-page applications) với hiệu suất cao
- Xây dựng các ứng dụng desktop và các ứng dụng realtime khi kết hợp với Socket.io

1.2.4 MongoDB

Ưu điểm

- Xử lý dữ liệu nhanh chóng, phù hợp cho các ứng dụng có yêu cầu xử lý dữ liệu lớn và phức tạp
- Có khả năng mở rộng tốt, cho phép mở rộng hệ thống để đáp ứng nhu cầu sử dụng của mình
- Dữ liệu trong MongoDB được lưu trữ dưới dạng tài liệu JSON, cho phép các developer dễ dàng tạo, truy xuất và hiển thị dữ liệu

Nhược điểm

- Không cung cấp các tính năng quản lý đầy đủ như một số hệ quản trị cơ sở dữ liệu khác. Điều này có thể gây khó khăn cho những người mới sử dụng MongoDB hoặc những người không có kinh nghiệm về quản trị cơ sở dữ liệu
- Gặp khó khăn trong việc truy vấn dữ liệu phức tạp
- Chi phí tăng cao khi tăng độ phức tạp của hệ thống
- Tài liệu hướng dẫn bị hạn chế

=> MongoDB phù hợp với:

- Các ứng dụng yêu cầu sử dụng ngôn ngữ lập trình JavaScript.
- Các ứng dụng này không yêu cầu tính toán vụn vặt.
- Các ứng dụng có nhu cầu mở rộng trong tương lai và muốn có tốc độ truy vấn nhanh
- Các ứng dụng yêu cầu dữ liệu phi cấu trúc hoặc có cấu trúc phức tạp

1.3 Kiến trúc tổng quan trong hệ thống:

Đối với hệ thống Realtime Chat, ứng dụng sử dụng các công nghệ và kỹ thuật như Node.js, React.js, MongoDB và Socket.IO thường được xây dựng theo kiểu kiến trúc phân tầng, bao gồm các thành phần sau:

- **Client-side:** Đây là phần giao diện người dùng của sản phẩm, được xây dựng bằng React.js và HTML/CSS/JS để hiển thị các tin nhắn và tương tác với người dùng. Client-side cũng có nhiệm vụ thiết lập kết nối với server-side thông qua Socket.IO và nhận và gửi dữ liệu giữa client và server.
- **Server-side:** Đây là phần xử lý logic của sản phẩm, được xây dựng bằng Node.js và sử dụng Socket.IO để thiết lập kết nối với client-side. Server-side có nhiệm vụ xử lý các yêu cầu từ client-side, truy vấn và lưu trữ dữ liệu vào MongoDB, và gửi các thông tin mới nhất đến client-side thông qua Socket.IO.
- **Database:** Hệ thống sử dụng MongoDB làm cơ sở dữ liệu để lưu trữ các tin nhắn và thông tin người dùng. MongoDB là một cơ sở dữ liệu phi quan hệ (NoSQL) mang lại tính linh hoạt và khả năng mở rộng cho sản phẩm.
- **Socket.IO:** Là thư viện JavaScript cho phép truyền thông tin song song và thời gian thực giữa client và server. Socket.IO được sử dụng để thiết lập kết nối giữa client-side và server-side, và gửi và nhận các sự kiện tùy chỉnh giữa hai bên.

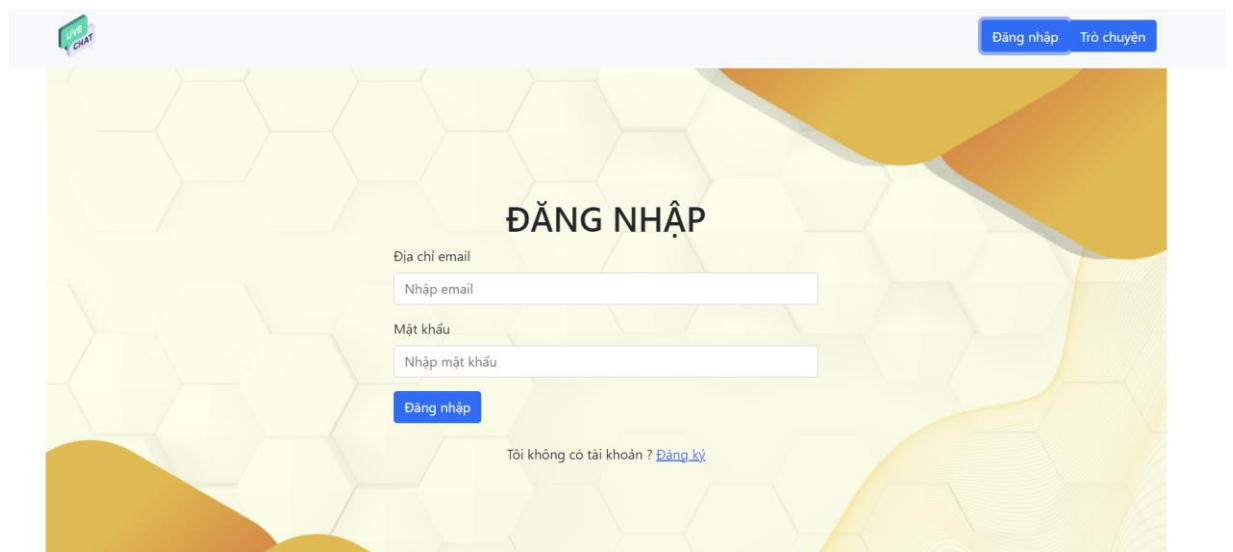
1.4 Kết quả sản phẩm:

- Giao diện trang chủ:

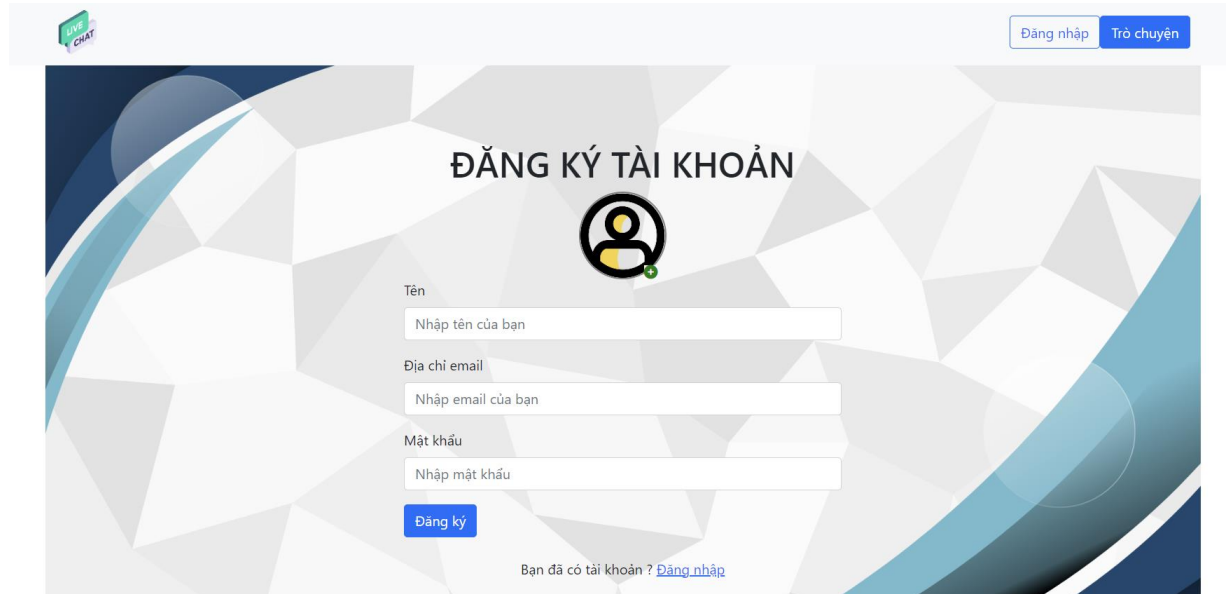


Hình 1.3 Trang chủ của trang web

- Giao diện đăng nhập, đăng ký tài khoản chat realtime:



Hình 1.4 Trang đăng nhập của trang web



ĐĂNG KÝ TÀI KHOẢN

Tên

Địa chỉ email

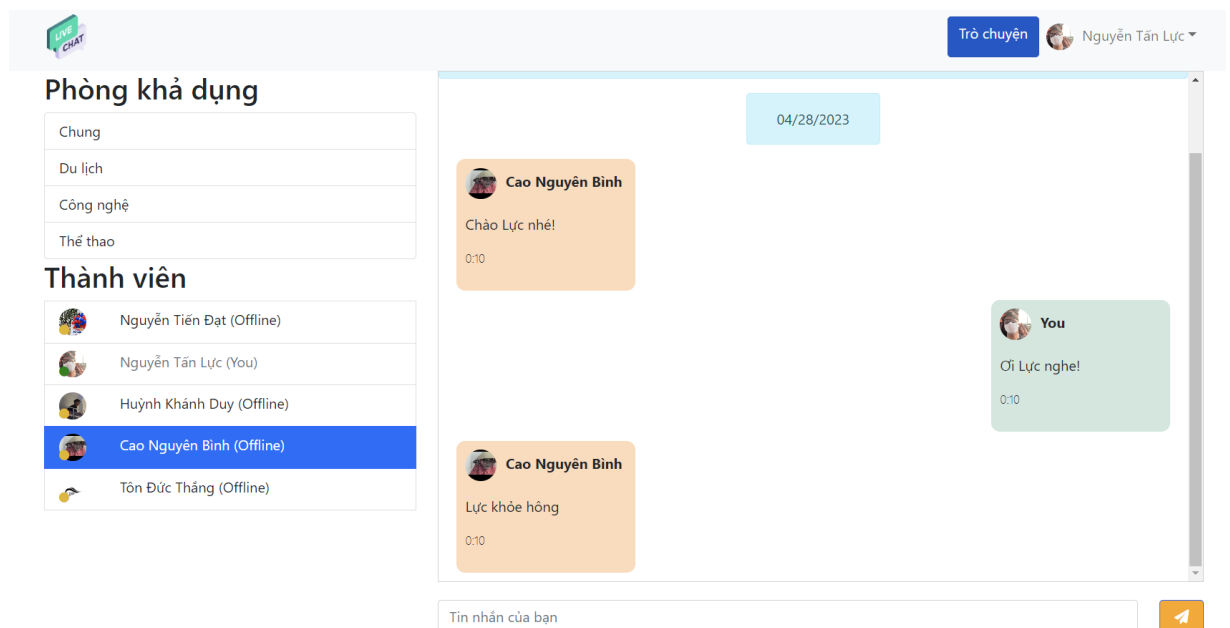
Mật khẩu

Đăng ký

Bạn đã có tài khoản? [Đăng nhập](#)

Hình 1.5 Trang đăng ký của trang web

- Giao diện chat, đoạn chat riêng và đoạn chat nhóm:



Phòng khả dụng

- Chung
- Du lịch
- Công nghệ
- Thể thao

Thành viên

- Nguyễn Tiến Đạt (Offline)
- Nguyễn Tấn Lực (You)
- Huỳnh Khánh Duy (Offline)
- Cao Nguyên Bình (Offline)**
- Tôn Đức Thắng (Offline)

Chat Conversation:

04/28/2023

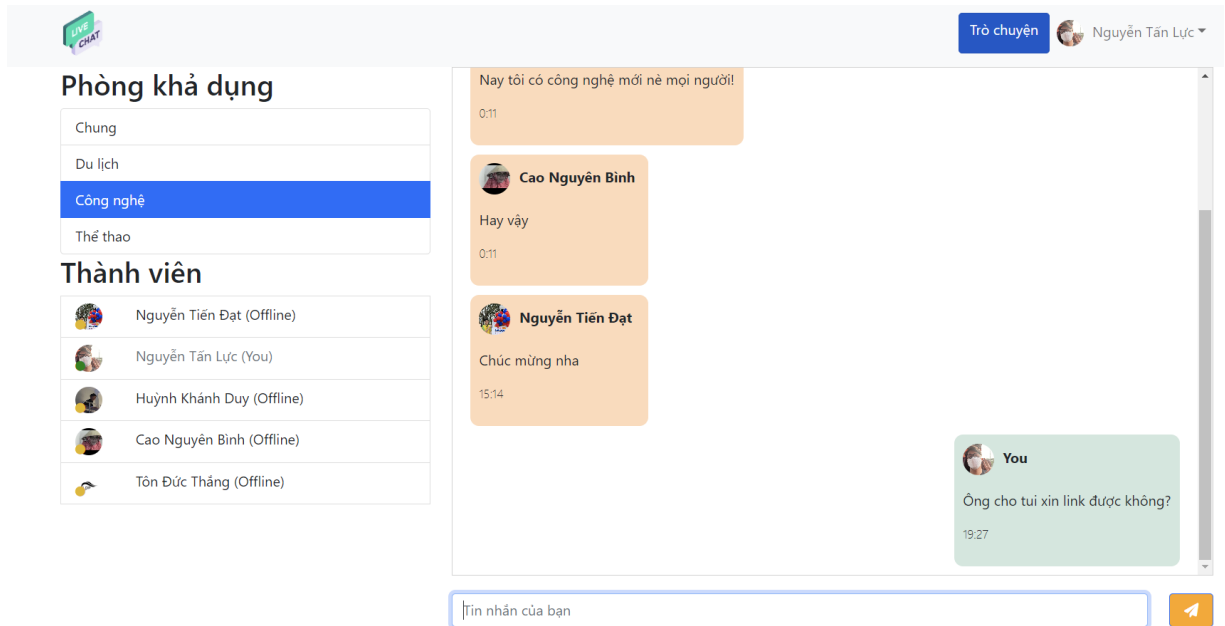
Cao Nguyên Bình
 Chào Lực nhé!
 0:10

You
 Ơi Lực nghe!
 0:10

Cao Nguyên Bình
 Lực khỏe hông
 0:10

Tin nhắn của bạn

Hình 1.6 Trang chat và nội dung chat (chat riêng tư)



Hình 1.7 Trang chat và nội dung chat (chat trong nhóm công khai)

1.5 Hạn chế và hướng phát triển:

1.5.1 Hạn chế

- Ứng dụng hiện tại chỉ có thể gửi cho nhau những văn bản thuần
- Thời gian khởi động project vẫn còn chậm và mắc phải một số lỗi
- Chưa tích hợp các hình thức đăng nhập auth của bên thứ 3
- Các phòng chat là giá trị mặt định, không thể tạo và thay đổi bên phía client
- Khi có người dùng online thì thanh ký hiệu online chỉ đổi màu, không hiện thông báo rõ ràng.

1.5.2 Hướng phát triển

- Thêm các tính năng gửi file, gửi video,...
- Tạo một cuộc gọi thoại/video riêng hoặc nhóm
- Thêm chức năng xóa, thu hồi, chỉnh sửa tin nhắn
- Thêm tính năng gắn thẻ tin nhắn khi cần ghi nhớ
- Thêm chức năng tạo phòng chat và thêm thành viên vào phòng chat đó

- Thêm tính năng hiển thị đã xem, nếu là trong phòng chat chung thì hiển thị thông tin của những người đã xem.

CHƯƠNG 2 – PHẦN LÝ THUYẾT

2.1 Câu 1:

2.1.1: Câu hỏi:

Anh/chị hãy cho biết front-end, back-end là gì? Cho ví dụ.

2.1.2: Trả lời:

Front-end là một phần của một website ở đó người dùng có thể tương tác để sử dụng, là tất cả những gì mà bạn nhìn thấy trên một website bao gồm: font chữ, màu sắc, danh mục sản phẩm, menu, thanh trượt,.. đều là front-end. Các kỹ thuật front-end thường được sử dụng để tạo ra trang web động, có thể tương tác với người dùng và hiển thị các thông tin động. Các công nghệ phổ biến được sử dụng để phát triển front-end bao gồm:

- HTML - cung cấp cấu trúc và nội dung cho website
- CSS - định dạng và cải thiện giao diện người dùng
- JavaScript - làm website tương tác và động

Ví dụ về front-end:

- Một ví dụ cụ thể về front-end là giao diện người dùng trên trang web. Ví dụ, khi bạn truy cập vào trang Facebook, bạn sẽ thấy giao diện người dùng bao gồm các nút bấm, hộp văn bản, hình ảnh và các phần khác để bạn có thể tương tác với trang web. Tất cả các phần này đều là phần của front-end.
- Ví dụ khác, các trang web thương mại điện tử cũng có các phần của front-end như giỏ hàng, thanh toán và các nút bấm "Mua ngay". Các phần tử này được tạo ra bằng HTML và CSS, và JavaScript được sử dụng để xử lý các sự kiện tương tác như nhấp chuột và kéo thả.

Back-end là tất cả những phần hỗ trợ hoạt động của website hoặc ứng dụng mà người dùng không thể nhìn thấy được. Có thể cho rằng Backend giống như bộ não của con người. Nó xử lý những yêu cầu, câu lệnh và lựa chọn thông tin chính xác để hiển thị lên màn hình.

Phần back end của một trang web bao gồm:

- Ngôn ngữ lập trình back-end: PHP, Python, Java, v.v. Để xử lý logic nghiệp vụ và giao tiếp với cơ sở dữ liệu.
- Cơ sở dữ liệu: MySQL, MongoDB, v.v. Để lưu trữ dữ liệu một cách có cấu trúc.
- Server: Apache, Nginx, v.v. Để lưu trữ các tập tin back-end và trung gian giữa người dùng và ứng dụng.

Ví dụ về back-end:

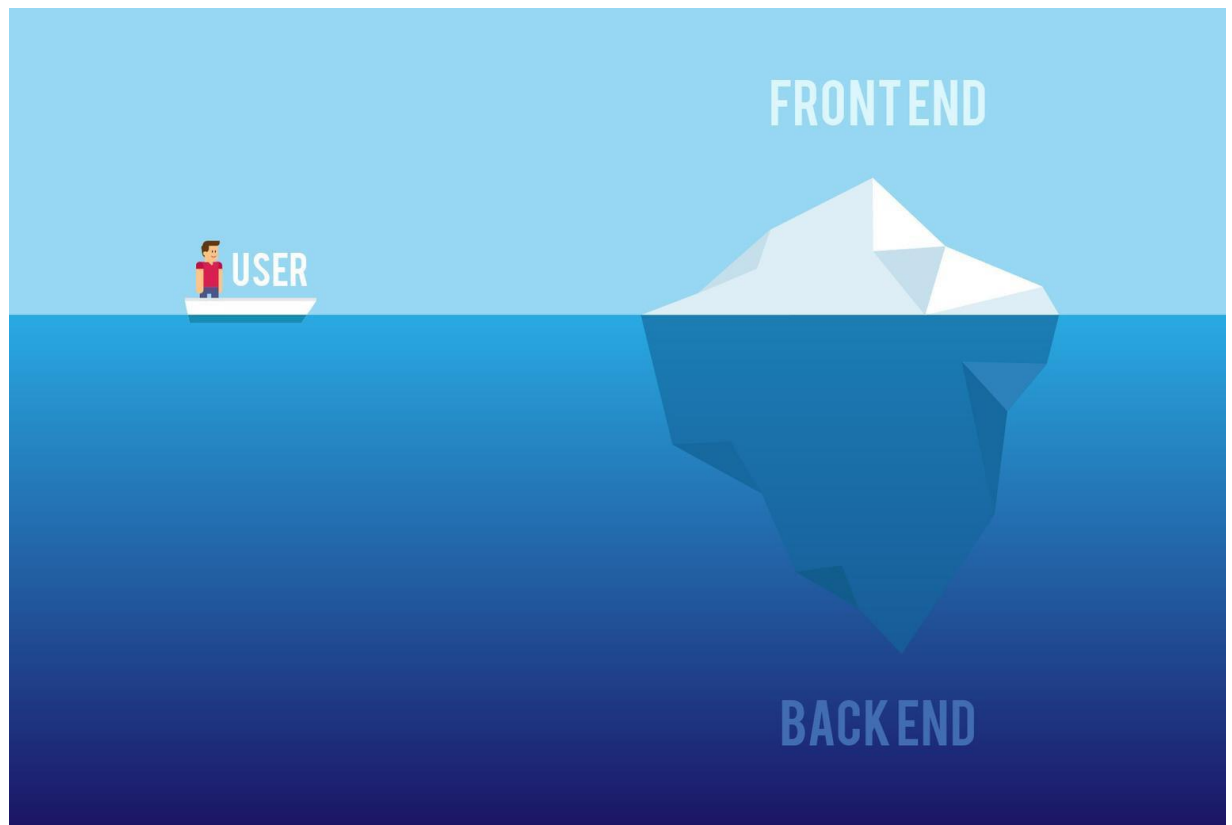
- Hệ thống quản lý đơn hàng của một trang thương mại điện tử: Back-end sẽ xử lý các yêu cầu của khách hàng, như thêm sản phẩm vào giỏ hàng, thanh toán đơn hàng, cập nhật trạng thái đơn hàng, và lưu trữ thông tin đơn hàng trong cơ sở dữ liệu.
- Ứng dụng xem phim trực tuyến: Back-end sẽ quản lý cơ sở dữ liệu phim, xử lý yêu cầu phát video từ phía người dùng, tổ chức các thông tin phim, và cập nhật dữ liệu khi có thay đổi.
- Hệ thống quản lý nhân viên của một công ty: Back-end sẽ quản lý thông tin nhân viên, bao gồm tên, địa chỉ, số điện thoại, lương, v.v., và cho phép quản lý viên thực hiện các thao tác như thêm, sửa và xóa thông tin nhân viên.

* So sánh front-end và back-end:

- Front end và Back end là 2 khái niệm tương phản với nhau. Nếu Backend là phần chìm thì FrontEnd chính là phần nổi của tảng băng trôi. Cụ thể hơn, FrontEnd là phần hiển thị ra bên ngoài giao diện và tương tác với

người dùng. Vì vậy, nó chú trọng vào mặt trực quan, thẩm mỹ và bố cục dễ sử dụng. Còn BackEnd là những công việc liên quan đến phần cơ sở dữ liệu lưu trữ bên trong để hiển thị cho máy chủ. Chức năng chính là lưu trữ dữ liệu, truy xuất thông tin nhanh và chính xác với từng lệnh được đưa ra.

- Front end: Giao tiếp với người dùng
- Back end: Giao tiếp với máy chủ



Hình 1.8 So sánh front-end và back-end (nguồn tham khảo)

* Đánh giá front-end và back-end:

	Ưu điểm	Nhược điểm
Front-end	<ul style="list-style-type: none"> - Tương tác trực tiếp với người dùng, quan tâm đến trải nghiệm người dùng. - Công nghệ thay đổi nhanh, luôn phải cập nhật kiến thức mới. - Không cần quan tâm đến cơ sở dữ liệu và server. 	<ul style="list-style-type: none"> - Phụ thuộc vào back-end để lấy dữ liệu. - Nhiều người dùng truy cập cùng lúc có thể ảnh hưởng hiệu năng.
Back-end	<ul style="list-style-type: none"> - Quan tâm đến dữ liệu và logic nghiệp vụ - là trái tim của một ứng dụng. - Ít bị ảnh hưởng bởi số lượng người truy cập đồng thời. - Công nghệ ổn định hơn front-end. 	<ul style="list-style-type: none"> - Không tương tác trực tiếp với người dùng. - Phụ thuộc vào front-end để hiển thị dữ liệu và giao diện. - Cần quan tâm đến cơ sở dữ liệu, server, và các vấn đề về bảo mật, mạng.

Bảng 1.1 Bảng đánh giá front-end và back-end

=> Nhìn chung, front-end và back-end đóng vai trò bổ sung cho nhau trong việc xây dựng một ứng dụng web hoàn chỉnh. Cả hai đều rất quan trọng và khó có thể thay thế lẫn nhau.

2.2 Câu 2:

2.2.1: Câu hỏi

Theo anh/chị dùng nodejs để xây dựng front-end hay back-end? Vì sao?

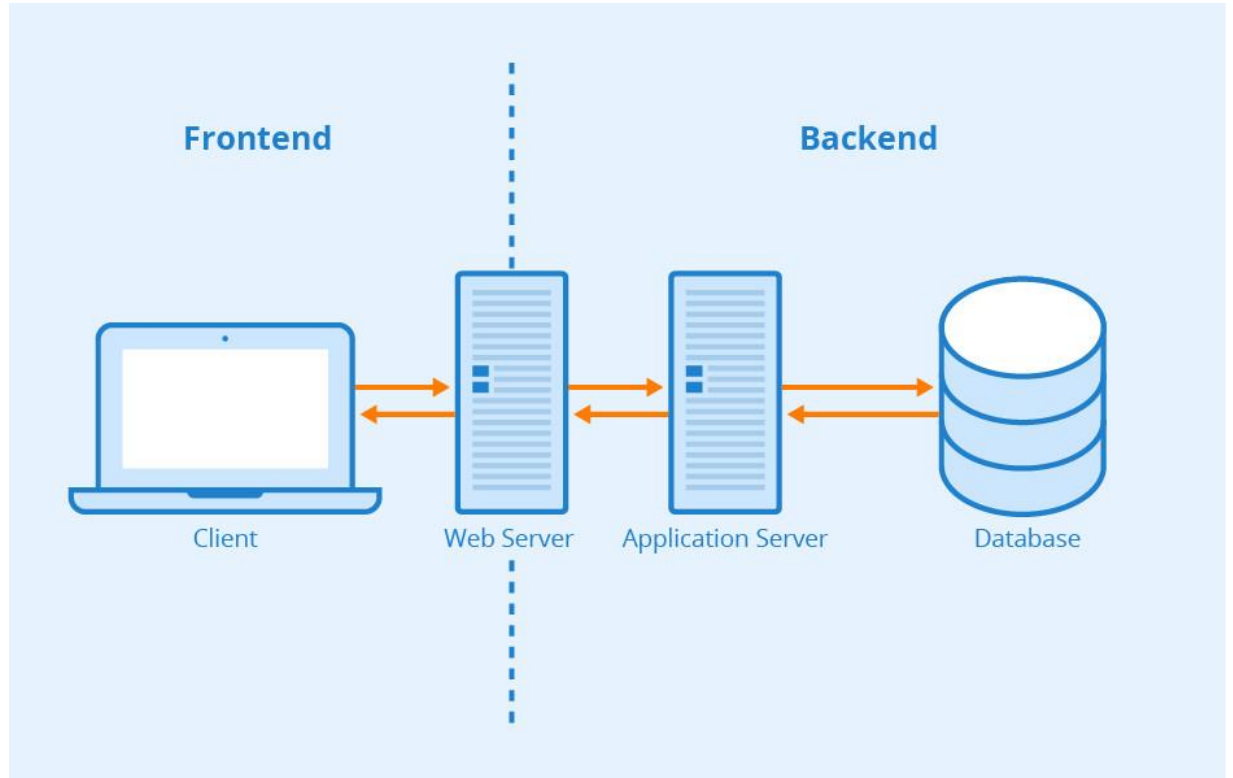
2.2.2: Trả lời

Node.js là một nền tảng phát triển được xây dựng trên JavaScript runtime của Chrome, cho phép chạy mã JavaScript ở phía máy chủ. Node.js đã trở thành một công cụ phổ biến cho việc xây dựng ứng dụng web, nó có thể được dùng để xây dựng cả về front-end lẫn back-end nhờ vào các framework như Express.js hay Meteor.js.

Tuy nhiên, Node.js là một lựa chọn tốt cho việc xây dựng back-end của ứng dụng web. Một trong những lý do chính để sử dụng Node.js là vì nó cho phép phát triển back-end bằng JavaScript, ngôn ngữ phổ biến nhất trong lập trình web. Điều này giúp giảm ngưỡng nhận thức đối với lập trình viên và giảm thời gian học tập và phát triển.

Back-end là một phần của ứng dụng web dùng để xử lý logic và lưu trữ dữ liệu, thường chạy trên máy chủ. Node.js có thể được sử dụng để xây dựng back-end của ứng dụng web. Node.js có khả năng xử lý các kết nối đồng thời (concurrency), giúp tối ưu hóa hiệu suất của ứng dụng. Node.js cũng có thể được sử dụng để xử lý các tác vụ I/O như đọc và ghi file, truy vấn cơ sở dữ liệu và kết nối với các dịch vụ bên ngoài.

Tóm lại, nên sử dụng Node.js để xây dựng back-end của ứng dụng web. Nếu muốn xây dựng front-end của ứng dụng web, Node.js không phải là lựa chọn phổ biến nhất. Thay vào đó, các công cụ như React, Vue.js và Angular thường được sử dụng để xây dựng giao diện người dùng. Các công cụ này cho phép phát triển front-end dễ dàng hơn, bằng cách sử dụng các thành phần tái sử dụng, kiểm thử dễ dàng và các tính năng khác để tăng tốc độ phát triển.



Hình 1.9 Sự tương tác giữ front-end và back-end trong hệ thống Node.JS (nguồn tham khảo)

So sánh Node.js trong việc xây dựng front-end và back-end:

	Ưu điểm	Nhược điểm
Front-end (Node.JS)	<ul style="list-style-type: none"> - Có thể sử dụng trong việc xây dựng cả server-side rendering và client-side rendering: Node.js cho phép sử dụng các công cụ như React để xây dựng ứng dụng web với server-side rendering hoặc client-side rendering. - Thân thiện với các công cụ front-end: Node.js có thể sử dụng chung các công 	<ul style="list-style-type: none"> - Không phù hợp với các ứng dụng có yêu cầu tính toán cao: Node.js không phù hợp với các ứng dụng có yêu cầu tính toán cao, như các ứng dụng xử lý hình ảnh hoặc video.

	cụ front-end phổ biến như NPM hoặc Yarn để quản lý các phụ thuộc và gói. - Công cụ debugging tốt: Node.js cung cấp các công cụ debugging tốt như Node Inspector, giúp lập trình viên dễ dàng theo dõi và sửa lỗi trong mã JavaScript của mình.	- Khả năng mở rộng khó khăn: Node.js không phải là một giải pháp mở rộng tốt cho các ứng dụng lớn hoặc phức tạp.
Back-end (Node.JS)	- Khả năng xử lý đồng thời cao: Node.js sử dụng các khái niệm bất đồng bộ và sự kiện để tối ưu hiệu suất xử lý đồng thời các yêu cầu từ các client khác nhau. - Sử dụng JavaScript: Node.js cho phép lập trình viên viết mã back-end bằng JavaScript, giúp giảm ngưỡng cho việc học và phát triển. - Cộng đồng lớn: Node.js có một cộng đồng lớn và đầy đủ các tài liệu học tập, giúp lập trình viên dễ dàng tìm kiếm thông tin và giúp đỡ khi cần.	- Khó khăn khi xử lý các tác vụ đồng bộ: Node.js sử dụng bất đồng bộ để tối ưu hiệu suất, nhưng điều này đôi khi có thể gây khó khăn cho việc xử lý các tác vụ đồng bộ, ví dụ như truy vấn cơ sở dữ liệu. - Khả năng mở rộng khó khăn: Node.js không phù hợp với các ứng dụng lớn hoặc phức tạp.

Bảng 1.2 Bảng đánh giá front-end và back-end trong Node.JS

=> Tóm lại, Node.js là một công cụ phát triển mạnh mẽ và phổ biến trong việc xây dựng back-end của ứng dụng web, và cũng có thể được sử dụng để xây dựng front-end thông qua các công cụ như React hoặc Angular. Tuy nhiên, Node.js không phù hợp với tất cả các loại ứng dụng và có nhược điểm riêng cho từng trường hợp sử dụng.

2.3 Câu 3:

2.3.1: Câu hỏi

Theo anh/chị dùng expressjs để xây dựng front-end hay back-end? Vì sao?

2.3.2: Trả lời

Express.JS là một framework Node.js chuyên dùng để xây dựng ứng dụng backend. Tuy nhiên, Express.js cũng có thể được sử dụng để xây dựng front-end của ứng dụng web thông qua các công cụ như React hoặc Angular.

Một số lý do chính khiến Express.JS phù hợp để làm back-end hơn front-end là:

- Express.JS chạy trên Node.js, một runtime JavaScript ở phía server. Node.js rất hiệu quả để xử lý các request và response, kết nối database,.. Do đó thích hợp hơn cho việc phát triển back-end.
- Express.JS hỗ trợ việc xây dựng API, routing, các middleware rất tốt. Đây là những yêu cầu cần thiết cho một ứng dụng back-end.
- Express.JS không có DOM (Document Object Model), không hỗ trợ các thư viện front-end như React hay Vue. Nó thiếu nhiều tính năng cần thiết cho front-end như state management, component, virtual DOM,...
- Các template engine hỗ trợ bởi Express.JS (như Pug, EJS) thường đơn giản hơn so với các front-end framework. Chúng thích hợp hơn cho việc render dữ liệu ở phía server.
- Các ứng dụng web thường có kiến trúc theo mô hình MVC (Model-View-Controller), trong đó Express.JS phụ trách xử lý phần Controller và Model ở phía server.

So sánh Express.JS trong việc xây dựng front-end và back-end: (tương tự Node.JS)

	Ưu điểm	Nhược điểm
Front-end (Express.JS)	<ul style="list-style-type: none"> - Có thể sử dụng trong việc xây dựng cả server-side rendering và client-side rendering: Express.js cho phép sử dụng các công cụ như React để xây dựng ứng dụng web với server-side rendering hoặc client-side rendering. - Thân thiện với các công cụ front-end: Express.js có thể sử dụng chung các công cụ front-end phổ biến như NPM hoặc Yarn để quản lý các phụ thuộc và gói. - Công cụ debugging tốt: Express.js cung cấp các công cụ debugging tốt như Node Inspector, giúp lập trình viên dễ dàng theo dõi và sửa lỗi trong mã JavaScript của mình. 	<ul style="list-style-type: none"> - Không phù hợp với các ứng dụng có yêu cầu tính toán cao: Express.js không phù hợp với các ứng dụng có yêu cầu tính toán cao, như các ứng dụng xử lý hình ảnh hoặc video. - Khả năng mở rộng khó khăn: Express.js không phải là một giải pháp mở rộng tốt cho các ứng dụng lớn hoặc phức tạp.
Back-end (Express.JS)	<ul style="list-style-type: none"> - Đơn giản và dễ sử dụng: Express.js được thiết kế để đơn giản và dễ sử dụng, giúp lập trình viên nhanh chóng xây dựng các ứng dụng web đáp ứng các yêu cầu cụ thể. - Cộng đồng lớn: Express.js có một cộng đồng lớn và đầy đủ các tài liệu học tập, giúp lập trình viên dễ dàng tìm kiếm thông tin và giúp đỡ khi cần. 	<ul style="list-style-type: none"> - Khả năng mở rộng khó khăn: Express.js không phù hợp với các ứng dụng lớn hoặc phức tạp, và việc mở rộng ứng dụng sử dụng Express.js có thể gặp khó khăn.

	- Hỗ trợ middleware: Express.js hỗ trợ middleware, cho phép lập trình viên thêm các xử lý trung gian vào quá trình xử lý yêu cầu của ứng dụng.	
--	--	--

Bảng 1.3 Bảng đánh giá front-end và back-end trong Express.js

=> Tóm lại, Express.js là một framework Node.js phổ biến và dễ sử dụng trong việc xây dựng back-end của ứng dụng web. Phần front-end sẽ được xây dựng bằng các công nghệ khác và tương tác với Express.JS thông qua API. Tuy nhiên, Express.js không phù hợp với tất cả các loại ứng dụng và có nhược điểm riêng cho từng trường hợp sử dụng.

2.4 Câu 4:

2.4.1: Câu hỏi

Theo anh/chị expressjs là thư viện mã nguồn mở hay framework hay là ngôn ngữ lập trình web hay công cụ xây dựng web? Vì sao?

2.4.2: Trả lời

Vì ExpressJS cung cấp các tính năng cơ bản của một framework web và được xây dựng trên Node.js nên nó được sử dụng rộng rãi để phát triển các ứng dụng web đơn giản đến phức tạp. Do đó:

- ExpressJS là:
 - Thư viện mã nguồn mở: Vì mã nguồn của Express JS là mã nguồn được công bố và có thể được sửa đổi bởi bất kỳ ai. Đây là một lợi thế cho các nhà phát triển vì họ có thể tùy biến và cải tiến Express JS theo nhu cầu của họ. ExpressJS là một trong những thư viện phổ biến nhất được sử dụng để xây dựng các ứng dụng web và API trên Node.js. Nó cung cấp cho lập trình viên một số tính năng cần thiết chẳng hạn như định tuyến, middleware, xử lý lỗi, đọc dữ liệu POST, và nhiều tính năng khác.

- Framework: vì Express JS vừa là một bộ công cụ và vừa là quy tắc để xây dựng các ứng dụng web. Express JS cung cấp các tính năng như middleware, routing, template engine, error handling, xử lý yêu cầu HTTP, xử lý cookie, xử lý session,... để giúp các nhà phát triển tạo ra các ứng dụng web một cách nhanh chóng và hiệu quả. Vì vậy, ExpressJS là một framework web mạnh mẽ và được sử dụng rộng rãi trong cộng đồng phát triển Node.js để xây dựng các ứng dụng web và API chất lượng cao.
- Express JS không phải là:
 - Ngôn ngữ lập trình web: Vì Express JS không phải là một ngôn ngữ lập trình mà chỉ là một framework sử dụng ngôn ngữ lập trình Javascript. Ngôn ngữ lập trình web phải là một ngôn ngữ được sử dụng để viết mã phía server hoặc phía client của ứng dụng web. Ví dụ, HTML được sử dụng để xác định cấu trúc của một trang web, CSS được sử dụng để định dạng trang web và JavaScript được sử dụng để tương tác với các phần tử HTML và xử lý dữ liệu trên trang web. Các ngôn ngữ lập trình phía server như PHP, Python, Ruby, Java, và C# được sử dụng để xử lý các yêu cầu từ trình duyệt web và tạo ra các phản hồi tương ứng.
 - Công cụ xây dựng website: Vì ExpressJS không phải là một công cụ xây dựng website mà chỉ là một framework để xây dựng các ứng dụng web. Công cụ xây dựng website là một phần mềm hoặc dịch vụ cho phép người dùng tạo ra các trang web mà không cần biết lập trình, ví dụ như WordPress, Wix, Squarespace, ... Do đó, ExpressJS không phải là công cụ xây dựng website đầy đủ, mà chỉ là một công cụ để xây dựng các ứng dụng web và API bằng ngôn ngữ JavaScript trên nền tảng Node.js.

TÀI LIỆU THAM KHẢO

- [1] Cloudinary Upload API Reference - https://cloudinary.com/documentation/image_upload_api_reference
- [2] Socket.IO Documentation - <https://socket.io/docs/v4/>
- [3] ReactJS Documentation - <https://legacy.reactjs.org/docs/getting-started.html>
- [4] Mongoose Documentation - <https://mongoosejs.com/docs/>