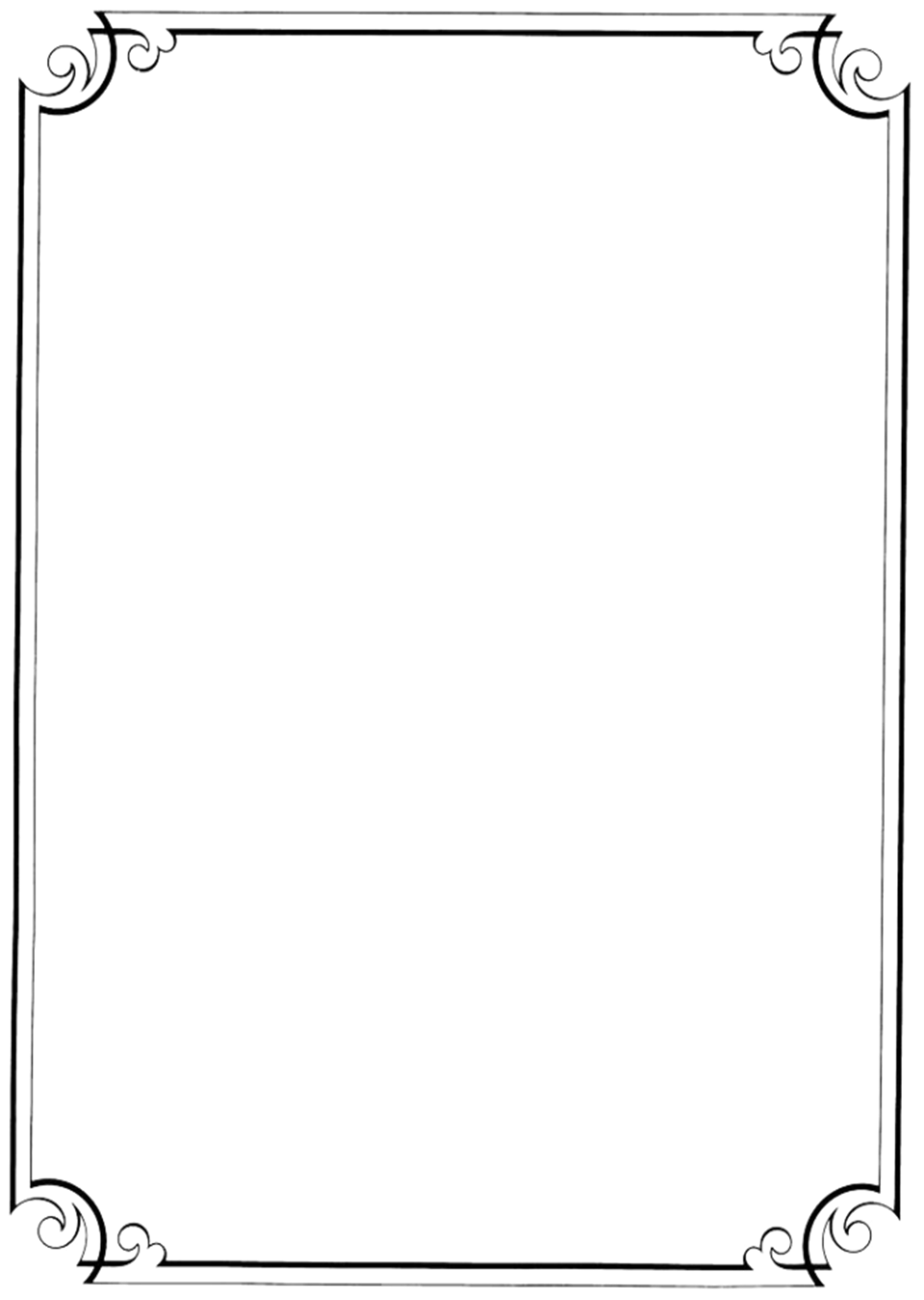
****

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**



**ĐỒ ÁN MÔN HỌC**

**TIỂU LUẬN CHUYÊN NGÀNH**

**ĐỀ TÀI:**

**THIẾT KẾ VÀ XÂY DỰNG WEBCHAT, TÌM HIỂU VÀ TÍCH HỢP CHATBOT API**

GVHD: PGS.TS. Hoàng Văn Dũng

SVTH:

Trần Nhật Hùng 19110213

Nguyễn Hoàng Phúc 19110268

Trần Hữu Thanh 19110007

Tp. HCM, tháng 9 năm 2022

| **Trường Đại học Sư Phạm Kỹ thuật TP. Hồ Chí Minh**  **Khoa Đào tạo Chất lượng cao**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| --- | --- |

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

*Tp. Hồ Chí Minh, ngày …., tháng …., năm 2022*

**Giảng viên hướng dẫn**

PGS.TS. Hoàng Văn Dũng

# 

**MỤC LỤC**

[**LỜI NÓI ĐẦU**](#_heading=h.1fob9te) **5**

[**PHẦN MỞ ĐẦU**](#_heading=h.3znysh7) **6**

[1. Tính cấp thiết đề tài](#_heading=h.2et92p0) 6

[2. Mục đích đề tài](#_heading=h.tyjcwt) 6

[3. Phương pháp nghiên cứu](#_heading=h.3dy6vkm) 6

[4. Khảo sát hiện trạng](#_heading=h.1t3h5sf) 6

[**PHẦN NỘI DUNG**](#_heading=h.4d34og8) **8**

[1. PHÂN TÍCH YÊU CẦU](#_heading=h.2s8eyo1) 8

[1.1. Yêu cầu chức năng](#_heading=h.17dp8vu) 8

[1.2. Yêu cầu phi chức năng](#_heading=h.3rdcrjn) 9

[2. MÔ HÌNH HÓA YÊU CẦU](#_heading=h.26in1rg) 10

[2.1. Use Cases](#_heading=h.lnxbz9) 10

[2.2. Đặc tả Use Cases](#_heading=h.35nkun2) 10

[3. CƠ SỞ LÝ THUYẾT](#_heading=h.1ksv4uv) 20

[3.1. MERN](#_heading=h.44sinio) 20

[3.2. Mã OTP](#_heading=h.2jxsxqh) 21

[3.3. JWT & Cookies](#_heading=h.z337ya) 21

[3.4. Redux & Redux Toolkit](#_heading=h.3j2qqm3) 21

[3.5. Socket.IO & Socket.IO-client](#_heading=h.1y810tw) 22

[3.6. Cloudinary](#_heading=h.4i7ojhp) 22

[3.7. Rasa](#_heading=h.2xcytpi) 23

[4. CÀI ĐẶT](#_heading=h.1ci93xb) 24

[4.1. Thiết kế Models](#_heading=h.3whwml4) 24

[4.2. Setup Project](#_heading=h.2bn6wsx) 25

[4.3. Đăng ký, đăng nhập](#_heading=h.qsh70q) 27

[4.4. Gửi nhận tin nhắn](#_heading=h.3as4poj) 32

[4.5. Gửi nhận file + hình ảnh](#_heading=h.1pxezwc) 34

[4.6. Tích hợp chatbot y tế](#_heading=h.49x2ik5) 35

[4.7. Cấu hình Redux](#_heading=h.2p2csry) 36

[4.8. Cấu hình Realtime](#_heading=h.147n2zr) 40

[5. DEMO](#_heading=h.3o7alnk) 42

[**PHẦN KẾT LUẬN**](#_heading=h.23ckvvd) **46**

[1. Kết quả đạt được](#_heading=h.ihv636) 46

[2. Khó khăn](#_heading=h.32hioqz) 46

[3. Hướng phát triển](#_heading=h.1hmsyys) 46

[**TÀI LIỆU THAM KHẢO**](#_heading=h.41mghml) **47**

[**PHÂN CÔNG CÔNG VIỆC**](#_heading=h.2grqrue) **48**

# 

# LỜI NÓI ĐẦU

Đầu tiên, chúng em xin chân thành cảm ơn thầy Hoàng Văn Dũng đã phụ trách môn học và hướng dẫn chúng em thực hiện dự án này! Môn học này giúp chúng em ôn lại những kiến thức đã học trong các năm qua. Chúng em cũng đã có thể tự học những kiến thức mới và áp dụng vào đồ án môn học, giúp chúng em có những cái nhìn mới về kiến thức ngành CNTT.

Đồ án môn học được thực hiện trong vòng 3 tháng. Tuy nhiên do kiến ​​thức mới nhiều cũng như thời gian chúng em làm qua mỗi tuần chưa tối ưu nên đồ án sẽ có nhiều sai sót là điều không thể tránh khỏi. Rất mong nhận được mọi sự đóng góp ý kiến ​​của thầy để giúp cho những kiến ​​thức còn hạn chế của chúng em được hoàn thiện hơn.

Chân thành cảm ơn thầy!

# PHẦN MỞ ĐẦU

## Tính cấp thiết đề tài

* Việc nhắn tin (SMS) hiện nay đã cũ và có nhiều bất tiện do đó đã có rất nhiều ứng dụng app chat lên ngôi. Nắm bắt nhanh việc kết bạn bốn phương kèm tiện ích mở rộng không tưởng.
* Nhờ công nghệ phát triển, từ đó tương lai con người sẽ còn có những bứt phá hơn nữa về các ứng dụng, phần mềm công nghệ hỗ trợ tương tác giao tiếp giữa người và người. Do đó đề tài của nhóm chúng em được tạo ra với mong muốn có nhiều hơn 1 app chat phổ biến trên thế giới với mục tiêu là tạo ra một ứng dụng chat trong cộng đồng như một trang mạng xã hội thu nhỏ để mọi người có thể dễ dàng sử dụng và kết nối với nhau. Ứng dụng còn được tích hợp thêm một chatbot có khả năng giải đáp các vấn đề về y tế, sức khỏe cho người dùng giúp cho việc tra hỏi thông tin trở nên nhanh gọn, tiện lợi hơn ( có thể tích hợp thêm các chức năng khác như thể thao, giải trí,... trong tương lai ).

## Mục đích đề tài

* Tạo ra 1 trang web giúp người dùng dễ dàng nhắn tin với người thân bạn bè, giao diện thân thiện, dễ sử dụng, tiện lợi và có những tính năng đặc biệt, khác với những ứng dụng khác để người dùng có trải nghiệm thú vị hơn khi sử dụng.
* Kết hợp được AI chatbot có thể giao tiếp một cách tự nhiên về chủ đề y tế.

## Phương pháp nghiên cứu

* Nhóm em sử dụng công nghệ MERN để phát triển đề tài.
* Kết hợp với các Framework hỗ trợ như Next.js, Tailwind cùng với các Library như Firebase OTP, Socket.io, Cloudinary, Styled Component.

## Khảo sát hiện trạng

* **Web 1: Messenger:**
  + Điểm mạnh:
    - Giao diện bắt mắt, dễ dùng
    - Đầy đủ tính năng cần thiết của 1 app chat
  + Điểm yếu:
    - Bị vỡ hình
    - Không được gửi file
    - Quét ra virus ảo
    - Thường xuyên crash app
* **Web 2: Zalo:**
  + Điểm mạnh:
    - Đầy đủ tính năng cần thiết của 1 app chat
    - Tính bảo mật yếu
  + Điểm yếu:
    - Hình bị xóa sau 1 khoảng thời gian
    - Giao diện khó sử dụng

# PHẦN NỘI DUNG

## PHÂN TÍCH YÊU CẦU

### Yêu cầu chức năng

| **Yêu cầu nghiệp vụ** | **Loại** | **Mô tả/Ràng buộc/Công Thức** | **Biểu mẫu (nếu có)** |
| --- | --- | --- | --- |
| Đăng nhập | Tra cứu | Người dùng đăng nhập theo số điện thoại đã đăng ký |  |
| Đăng ký | Lưu Trữ | Số điện thoại đăng ký của người dùng phải được xác thực qua OTP |  |
| Gửi lời mời kết bạn | Tính toán | User có thể gửi lời mời kết bạn tới user khác |  |
| Hủy kết bạn | Tính toán | User có thể hủy kết bạn với những user khác đã kết bạn |  |
| Đồng ý lời mời | Tính toán | User có thể đồng ý lời mời kết bạn từ user khác |  |
| Từ chối lời mời | Tính toán | User có thể từ chối lời mời kết bạn từ user khác |  |
| Tìm kiếm người dùng | Tra cứu | User có thể tra cứu user khác theo số điện thoại hoặc tên |  |
| Xem profile người dùng/bạn bè | Tra cứu | User có thể xem profile user khác hoặc của bản thân |  |
| Chỉnh sửa thông tin cá nhân | Lưu trữ | User có thể chỉnh sửa thông tin cá nhân |  |
| Tạo nhóm chat | Kết xuất | User có thể tạo nhóm chat với nhiều user khác. |  |
| Rời nhóm chat | Kết xuất | User có thể rời nhóm chat mà user đang tham gia. |  |
| Chat (bạn bè/nhóm) | Lưu trữ | User có thể chat với bạn bè hoặc với nhóm chat đang tham gia. |  |
| Chat với Bot qua widget | Tra cứu | User có thể chat với Bot APIs thông qua widget |  |

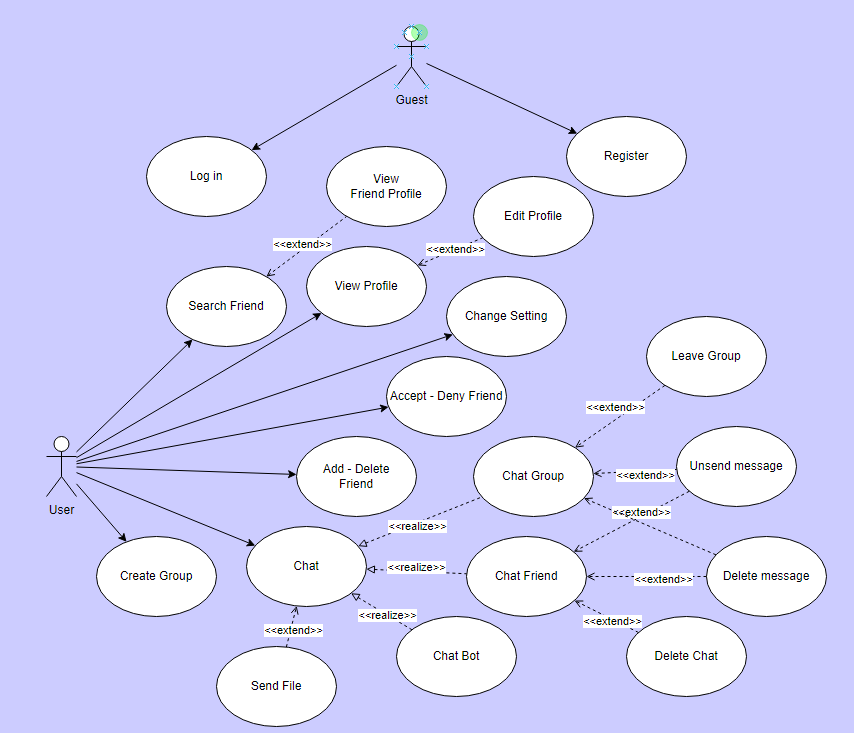
| **Yêu cầu hệ thống** | **Mô tả** | **Ghi chú** |
| --- | --- | --- |
| Hệ thống vận hành ổn định | Hệ thống phải hạn chế bị crash nhất có thể. |  |
| Hình ảnh lưu trữ lâu | Hình phải nên được lưu trữ lâu, sau thời gian dài vẫn có thể truy cập được. |  |
| Hình ảnh được lưu với chất lượng gốc | Hình ảnh nên được kết xuất với chất lượng gốc khi upload. |  |

### Yêu cầu phi chức năng

| **Yêu cầu** | **Loại** | **Ghi chú** |
| --- | --- | --- |
| Giao diện bắt mắt | Tiện dụng |  |
| Giao diện dễ thao tác | Tiện dụng |  |
| Không lưu trữ thông tin nhạy cảm lên browser | Bảo mật |  |

## MÔ HÌNH HÓA YÊU CẦU

### Use Cases



### Đặc tả Use Cases

| **Use Case:** | Login |
| --- | --- |
| **Short Description:** | Người dùng đăng nhập để sử dụng hệ thống**.** |
| **Actors:** | Guest. |
| **Pre-Conditions:** | Người dùng đã có tài khoản của website và đã truy cập vào url website. |
| **Post Conditions:** | Người dùng đăng nhập thành công. |
| **Main Flow:** | 1. Người dùng nhập phone và password và submit. 2. Hệ thống xử lý thông tin và trả kết quả. 3. Giao diện thông báo kết quả thành công [E1]. 4. Giao diện chuyển sang Home. |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Giao diện thông báo kết quả không thành công, yêu cầu guest nhập đúng. |

| **Use Case:** | Register |
| --- | --- |
| **Short Description:** | Người dùng đăng ký để có thể đăng nhập |
| **Actors:** | Guest. |
| **Pre-Conditions:** | Người dùng truy cập vào trang đăng ký |
| **Post Conditions:** | Người dùng chuyển sang trang đăng nhập nếu đăng ký thành công |
| **Main Flow:** | 1. Người dùng nhập các thông tin cần thiết và submit 2. Hệ thống xử lý và trả kết quả 3. Giao diện chuyển qua trang xác nhận mã OTP gửi về số điện thoại của người dùng (A1) 4. Người dùng nhập mã được nhận và submit 5. Hệ thống kiểm tra và trả kết quả 6. Giao diện thông báo thành công và chuyển sang trang đăng nhập (A2) |
| **Alternate Flow:** | (A1) Số điện thoại của người dùng bị trùng, yêu cầu nhập lại  (A2) Mã OTP không đúng, yêu cầu nhập lại hoặc kiểm tra thông tin để gửi lại mã |
| **Exception Flow:** |  |

| **Use Case:** | Create Group |
| --- | --- |
| **Short Description:** | Người dùng tạo nhóm chat |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công |
| **Post Conditions:** | Nhóm chat được tạo thành công |
| **Main Flow:** | 1. Người dùng chọn chức năng tạo nhóm chat 2. Người dùng tìm kiếm và lựa chọn bạn bè vào nhóm chat (E1) 3. Người dùng nhấn tạo nhóm (E1) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng tắt giao diện tạo nhóm |

| **Use Case:** | Search Friend |
| --- | --- |
| **Short Description:** | Người dùng tìm kiếm bạn bè |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công |
| **Post Conditions:** | Danh sách người dùng hiện ra theo số điện thoại hoặc tên đã nhập |
| **Main Flow:** | 1. Người dùng nhập số điện thoại hoặc tên trên thanh tìm kiếm (E1) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1)Người dùng bấm ra ngoài để bỏ tìm kiếm |

| **Use Case:** | Friend Profile |
| --- | --- |
| **Short Description:** | Xem thông tin của bạn bè |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công và tìm kiếm thành công |
| **Post Conditions:** | Thông tin của bạn bè đã chọn sẽ hiện ra sau khi tìm kiếm |
| **Main Flow:** | 1. Người dùng chọn vào bạn bè trong thanh tìm kiếm (E1) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1)Người dùng bấm ra ngoài để hủy thao tác vào xem thông tin bạn bè |

| **Use Case:** | My Profile |
| --- | --- |
| **Short Description:** | Người dùng xem thông tin bản thân |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công |
| **Post Conditions:** | Giao diện thông tin người dùng hiện thành công |
| **Main Flow:** | 1. Người dùng ấn vào tên hoặc avatar người dùng góc trên trái |
| **Alternate Flow:** |  |
| **Exception Flow:** |  |

| **Use Case:** | Edit Profile |
| --- | --- |
| **Short Description:** | Người dùng cập nhật thông tin của bản thân |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công |
| **Post Conditions:** | Cập nhật thông tin cá nhân của người dùng thành công |
| **Main Flow:** | 1. Người dùng ấn vào tên hoặc avatar người dùng góc trên trái 2. Giao diện hiện ra kèm nút cập nhật thông tin 3. Ấn chọn vào nút cập nhật, giao diện cập nhật thông tin sẽ hiện ra (E1) 4. Chỉnh sửa thông tin sau đó đồng ý để cập nhật lại thông tin (E1) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng có thể chọn vào tắt hoặc bấm ra ngoài để hủy thao tác |

| **Use Case:** | Change Setting |
| --- | --- |
| **Short Description:** | Người dùng cài đặt ứng dụng |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công |
| **Post Conditions:** | Cài đặt ứng dụng hiển thị thành công |
| **Main Flow:** | 1. Người dùng bấm vào biểu tượng răng cưa trên ứng dụng để cài đặt 2. Giao diện hiện ra kèm các chức năng cài đặt cần thiết (E1) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng có thể ấn ra bên ngoài hoặc chọn tắt để đóng cài đặt |

| **Use Case:** | Accept-Denied Friend |
| --- | --- |
| **Short Description:** | Người dùng có thể chấp nhận hoặc từ chối lời mời kết bạn |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công |
| **Post Conditions:** | Đồng ý hoặc từ chối lời mời kết bạn thành công |
| **Main Flow:** | 1. Người dùng nhấn vào nút thông báo 2. Giao diện thông báo hiện ra 3. Người dùng có thể đồng ý hoặc từ chối [E1] |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng tắt giao diện thông báo |

| **Use Case:** | Add-Delete Friend |
| --- | --- |
| **Short Description:** | Người dùng có thể thêm hoặc hủy kết bạn |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã đăng nhập thành công |
| **Post Conditions:** | Thêm bạn hoặc hủy kết bạn thành công |
| **Main Flow:** | 1. Người dùng tìm kiếm bạn bè/người dùng khác [E1] 2. Người dùng nhấn kết bạn/hủy với người dùng khác/kết bạn với bạn bè [A1] |
| **Alternate Flow:** | (A1) Người dùng có thể ấn xem thông tin của bạn bè/người dùng khác và sau đó có thể nhấn hủy kết bạn/kết bạn |
| **Exception Flow:** | (E1) Người dùng tắt tìm kiếm bạn bè/người dùng khác |

| **Use Case:** | Chat |
| --- | --- |
| **Short Description:** | Người dùng gửi tin nhắn cho bạn bè, nhóm chat hoặc chatbot |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã chọn một chatroom |
| **Post Conditions:** | Hiện tin nhắn đã được gửi |
| **Main Flow:** | 1. Người dùng nhập tin nhắn text vào Input (E1) 2. Người dùng click send để gửi tin nhắn 3. Tin nhắn được lưu vào database và hiện lên khung chat |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng click chuyển sang chat room khác |

| **Use Case:** | Send File |
| --- | --- |
| **Short Description:** | Người dùng gửi file cho bạn bè, group |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng đã chọn một chatroom |
| **Post Conditions:** | Hiện hiện đã được gửi |
| **Main Flow:** | 1. Người dùng upload file vào Input (E1) 2. Người dùng click send để gửi file 3. File được lưu vào database và hiện lên khung chat |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng click chuyển sang chat room khác |

| **Use Case:** | Leave Group |
| --- | --- |
| **Short Description:** | Người dùng rời nhóm và xóa tin nhắn nhóm |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng chọn nhóm chat muốn rời |
| **Post Conditions:** | Hệ thống thông báo rời nhóm thành công và xóa nhóm chat |
| **Main Flow:** | 1. Người dùng chọn click mở tùy chọn của room chat (E1) 2. Người dùng chọn tùy chọn rời nhóm 3. Hệ thống hiển thị popup xác nhận muốn rời nhóm 4. Người dùng chọn Đồng ý để rời nhóm (E2) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng click chuyển sang chat room khác  (E2) Người dùng chọn Từ chối rời nhóm |

| **Use Case:** | Unsend message |
| --- | --- |
| **Short Description:** | Người dùng thu hồi tin nhắn cá nhân hiển thị trong room chat khi cần |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng chọn một room chat |
| **Post Conditions:** | Hệ thống thông báo thu hồi tin nhắn thành công, nội dung tin nhắn được thay thế bằng thông báo đã thu hồi |
| **Main Flow:** | 1. Người dùng click vào options của tin nhắn muốn thu hồi (E1) 2. Người dùng chọn option thu hồi tin nhắn 3. Hệ thống hiển thị popup xác nhận người dùng có muốn thu hồi tin nhắn với mọi người 4. Người dùng chọn đồng ý để thu hồi tin nhắn (E2) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng click chuyển sang chat room khác  (E2) Người dùng chọn Từ chối thu hồi tin nhắn |

| **Use Case:** | Delete message |
| --- | --- |
| **Short Description:** | Người dùng xóa tin nhắn hiển thị trên tài khoản của người dùng nhưng vẫn hiển thị với tài khoản khác trong room |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng chọn một room chat |
| **Post Conditions:** | Hệ thống thông báo xóa tin nhắn thành công, tin nhắn bị ẩn với tài khoản người dùng |
| **Main Flow:** | 1. Người dùng click vào options của tin nhắn muốn xóa (E1) 2. Người dùng chọn option xóa tin nhắn 3. Hệ thống hiển thị popup xác nhận người dùng có muốn xóa tin nhắn với tài khoản của người dùng 4. Người dùng chọn đồng ý để xóa tin nhắn (E2) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng click chuyển sang chat room khác  (E2) Người dùng chọn Từ chối xóa tin nhắn |

| **Use Case:** | Delete Chat |
| --- | --- |
| **Short Description:** | Người dùng xóa và rời room chat |
| **Actors:** | User |
| **Pre-Conditions:** | Người dùng chọn room chat muốn xóa |
| **Post Conditions:** | Hệ thống thông báo xóa room chat thành công, room chat bị xóa khỏi sidebar |
| **Main Flow:** | 1. Người dùng chọn click mở tùy chọn của room chat (E1) 2. Người dùng chọn tùy chọn xóa room chat 3. Hệ thống hiển thị popup xác nhận muốn xóa room 4. Người dùng chọn đồng ý để xóa room (E2) |
| **Alternate Flow:** |  |
| **Exception Flow:** | (E1) Người dùng click chuyển sang chat room khác  (E2) Người dùng chọn Từ chối xóa room |

## CƠ SỞ LÝ THUYẾT

### MERN

* MERN Stack là một bộ tập hợp các công nghệ open source liên quan đến Javascript:
  + MongoDB: là một noSQL database, thường được sử dụng kèm với Mongoose - một library đơn giản hóa việc giao tiếp với mongoDB.
  + Express: là web framework được xây dựng bằng Javascript chạy trên nền tảng Node.js. Nó hỗ trợ thêm nhiều tính năng cần có của một web framework như: routing, middlewares, template engines,...
  + React/React Native: Là một thư viện Frontend được viết bằng Javascript và hoạt động dựa trên cơ chế Component.
  + Nodejs: là một nền tảng dùng để chạy Javascript runtime. Giúp chúng ta có thể chạy được ứng dụng độc lập ngoài trình duyệt.

### Mã OTP

* Mã OTP (One Time Password) là loại mật khẩu chỉ sử dụng một lần và được xem là lớp bảo vệ thứ hai cho dịch vụ ngân hàng điện tử, thanh toán trực tuyến hay email, mạng xã hội.
* Mã xác thực OTP giúp ngăn chặn, giảm thiểu những rủi ro bị tấn công khi mật khẩu bị lộ hoặc hacker xâm nhập. Nó gồm một dãy gồm các ký tự hoặc chữ số được tự động tạo ra gửi đến số điện thoại của người dùng. Mã OTP chỉ được dùng xác nhận giao dịch một lần duy nhất. Thậm chí khi người dùng chưa sử dụng thì sau khoảng 30 giây đến 2 phút, mã xác nhận này cũng không còn hiệu lực và cũng không thể sử dụng nó cho bất kì giao dịch nào khác.

### JWT & Cookies

* **JWT (JSON Web Token):** là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server , các thông tin trong chuỗi JWT được định dạng bằng JSON . Trong đó chuỗi Token phải có 3 phần là header , phần payload và phần signature được ngăn bằng dấu “.”
* **Cookies:** là một đoạn văn bản mà một Web server có thể lưu trên ổ cứng của người dùng với các tệp được trang web người dùng truy cập tạo ra. Cookie cho phép website lưu thông tin của người dùng trên máy tính và sau đó lấy lại nó. Các mẫu thông tin sẽ được lưu dưới dạng cặp tên – giá trị (name-value).

### Redux & Redux Toolkit

* Redux là công cụ quản lý state cho các ứng dụng Javascript, nó giúp chúng ta viết ứng dụng một cách nhất quán và dễ dàng test trong quá trình dev.
* Các thư viện như React, Angular,... thường hoạt động theo các Component và các Component đó có thể quản lý state của chính nó và có thể chuyển state của bản thân cho các Component con để có thể sử dụng ở nhiều nơi khác. Nhưng nếu như ứng dụng ngày càng mở rộng, số lượng Component càng nhiều thì việc chia sẻ state sẽ càng thêm phức tạp và lằng nhằng.
* Lúc này Redux sẽ đóng vai trò như một kho để giữ tất cả state mà các Component cần sử dụng, Component chỉ việc gọi state trong redux ra để sử dụng hoặc có thể update state ở bất cứ Component nào, từ đó đơn giản hóa việc quản lý state và bỏ các luồng chia sẻ state phức tạp.
* Redux hoạt động dựa trên 3 khái niệm:
  + Action: Là các event mà chúng ta sẽ dùng tùy theo nhu cầu để cập nhật trạng thái của state bằng các input API/người dùng/…
  + Reducer: Là nơi tập trung các action về 1 state mà cho chúng ta setup, nó sẽ lấy dữ liệu từ action được gọi, cập nhật state và trả về trạng thái mới của state.
  + Store: Là nơi tập hợp tất cả Reducer hay là nơi tập hợp tất cả state mà được dùng trong ứng dụng.
* Redux Toolkit là một thư viện giúp cho việc setup Redux dễ dàng hơn cho các coder.

### Socket.IO & Socket.IO-client

* Socket IO là một thư viện của NodeJS với mục đích là tạo môi trường kết nối duy trì giữa hai đầu kết nối (thường là server và client). Socket IO cung cấp sự giao tiếp, trao đổi thông tin hai chiều theo thời gian thực dành cho các ứng dụng cần sự thay đổi data realtime (Chat app, Live app,...)
* Socket.io-client là một thư viện để phía client có thể sử dụng và tạo kết nối với server. Socket IO được sử dụng phần lớn dựa trên phương thức **emit** và **on**, **emit** để gửi sự kiện, **on** để bắt sự kiện đó và thực hiện logic của yêu cầu.

### Cloudinary

* Cloudinary là một cloud-based service, nó cung cấp một giải pháp quản lý file, hình ảnh bao gồm upload, lưu trữ, thao tác, tối ưu hóa và delivery. Cloudinary giúp cho việc upload ảnh lên cloud, tự động thực thi các thao tác với ảnh một cách thông minh mà không cần phải cài đặt bất kỳ một phần mềm phức tạp nào khác. Cloudinary cung cấp các APIs toàn diện và màn hình quản lý giúp chúng ta dễ dàng tích hợp vào các trang web và ứng dụng di động.
* Có ba cách để có thể upload file hoặc hình ảnh lên cloudinary như sau:
  + Gửi file trực tiếp từ frontend:
    - Ưu điểm: Thời gian upload nhanh.
    - Nhược điểm: Không đáp ứng các yêu cầu về bảo mật các API\_SECRET do các key quan trọng được config phía client
  + Gửi file từ frontend xuống backend sau đó backend upload lên cloudinary.
    - Ưu điểm: Đảm bảo tính bảo mật do các key quan trọng được config phía server
    - Nhược điểm: Tăng gấp đôi thời gian upload vì phải gửi file 2 lần.
  + Gửi file từ phía client với signed key từ backend.
    - Ưu điểm: Các key quan trọng được config dưới server và server trả về signed key để cho client upload, từ đó khắc phục được nhược điểm của hai cách trên.
    - Nhược điểm: Khó để setup đúng.

### Rasa

* Rasa là một framework giúp người dùng đặc biệt là dev có thể tạo ra một model chatbot dễ dàng và nhanh chóng. Rasa framework dựa trên cơ chế rule-base. Rasa framework được cấu thành từ các thành phần gồm:
  + **Rasa NLU:** Đây là một thư viện được tạo ra với mục đích là xử lý ngôn ngữ tự nhiên từ người dùng, nó phân tích những thông tin có trong tin nhắn mà con người gửi đến cho chatbot. Các thông tin bao gồm ý định của người dùng (intent) và các đối tượng, thực thế được nhắc đến cần trích xuất (Entities).
  + **Rasa Core:** Sau khi đã phân tích được các thông tin cần thiết có trong tin nhắn của người dùng gửi tới chatbot, việc tiếp theo là dự đoán hành động tiếp theo mà chatbot cần làm để phản hồi lại người dùng. Cái này có thể là phản hồi lại tin nhắn hoặc truy vấn database hay bất cứ hành động nào ta có thể định nghĩa. Và Rasa Core được sinh ra để làm nhiệm vụ dự đoán này.
  + **Rasa X:** Hiểu đơn giản cái này chính là giao diện của toàn Rasa. Ta có thể tạo dữ liệu training trên này, train model trên này, test trên này với một giao diện chatbot cực kỳ thân thiện, gửi URL cho người khác test, thu thập dữ liệu test để training lại,... Và phần này là phần bổ sung, ta có thể dùng hoặc không.
  + **Intent:** Rasa framework cần hiểu ý định từ input người dùng là gì, vì vậy tất cả input phổ biến từ người dùng sẽ được phân nhóm và gán nhãn bằng một hành động cụ thể nào đó (ví dụ “xin chào”, “chào buổi sáng”, “chào bạn”,... sẽ được gom lại vào intent “chào hỏi”)
  + **Entity:** Thực thể là để trích xuất thông tin từ đầu vào của người dùng
  + **Stories:** Là cách để xác định sự tương tác giữa người dùng và chatbot theo intent và action được thực hiện bởi bot. Nó sẽ xác định luồng các hành động tiếp theo mà chatbot nên thực hiện.
  + **Action:** Về cơ bản là các hoạt động được thực hiện bởi bot hoặc yêu cầu thêm một số chi tiết để có được tất cả các thực thể hoặc tích hợp với một số API hoặc truy vấn cơ sở dữ liệu để nhận / lưu một số thông tin.

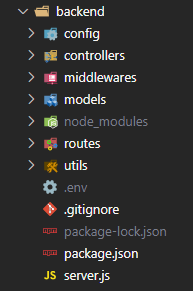
## CÀI ĐẶT

### Thiết kế Models

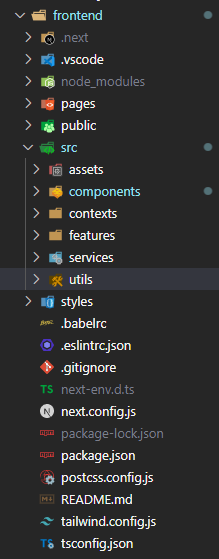
* **Notification Model:**
  + receiveId (String): Id user nhận thông báo (type Object id mongo)
  + requestId (String): Id user gửi thông báo (type Object id mongo)
  + status (String): Trạng thái thông báo dùng kiểm tra xong thông báo kết bạn đang chờ, được chấp nhận, hay bị từ chối (type Pending | Accepted | Denied)
* **Room Model:**
  + groupName (String): Tên phòng chat dùng để hiển thị FE
  + isGroup (Boolean): Là phòng chat nhóm, hay user với user
  + users (Array<Object>):
    - uid (String): Id user trong nhóm
    - role (Boolean): Vai trò user
    - nickname (String): Tên gợi nhớ của user
    - avatar (String): url avatar của user
  + lastMsg (String): Tin nhắn cuối cùng dùng hiển thị FE
* **Friend Model:**
  + uid1 (String): Id user thứ nhất trong quan hệ
  + uid2 (String): Id user thứ hai trong quan hệ
  + status (Object):
    - type (String): Loại quan hệ, có thể là đang có sẵn, chặn 1 phía - cập nhật cùng blockedId, hay chặn 2 phía (type available | oneWayBlock | twoWayBlock)
    - blockedId (String): Id user bị chặn
* **User Model:**
  + avatar (String): Đường dẫn hình ảnh dùng làm avatar
  + banner (String): Đường dẫn hình ảnh dùng làm banner
  + name (String): Tên user đặt hiển thị FE
  + password (String): Mật khẩu đăng nhập của user
  + phone (String): Số điện thoại dùng đăng ký và đăng nhập
  + gender (String): Giới tính user
  + dob (Date): Ngày sinh user
* **Message Model:**
  + roomId (String): Id của room mà tin nhắn thuộc vào
  + senderId (String): Id người gửi
  + msg (String): Tin nhắn
  + files (Array<Object>): Mảng các file gửi kèm
    - url (String): Đường dẫn tới file
    - name: (String): tên của file
    - type (String): loại file (file hoặc image)
  + unSend (Boolean): Kiểm tra tin nhắn có được thu hồi không
  + deleted (Boolean): Kiểm tra tin nhắn có được xóa không

### Setup Project

* + 1. Phía Server
* Folder: config chứa các cấu hình db, controllers chứa các file controller xử lý logic của request từ client, middlewares chứa các logic xử lý bên thứ 3, models chứa các file cấu hình models, routes chứa các route tới các controllers, utils chứa các phương thức global và các cấu hình bên thứ 3

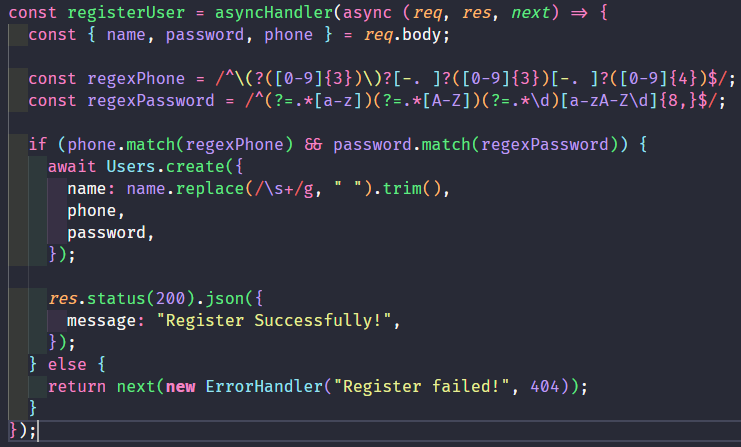


* + 1. Phía Client
* Folder: pages chứa các file tượng trưng cho mỗi page của website, public các icon của website, assets chứa các logo, hình ảnh cố định trong website, components chứa các Component con sử dụng cho các page, contexts chứa cấu hình các biến toàn cục, features chứa các công cụ hỗ trợ cho app, services chứa cấu hình axios và các url apis, utils chứa các type và dữ liệu test trong quá trình code.

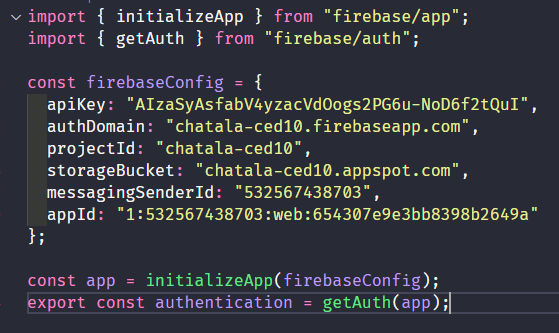


### Đăng ký, đăng nhập

* + 1. Đăng ký
* Tạo API đăng ký

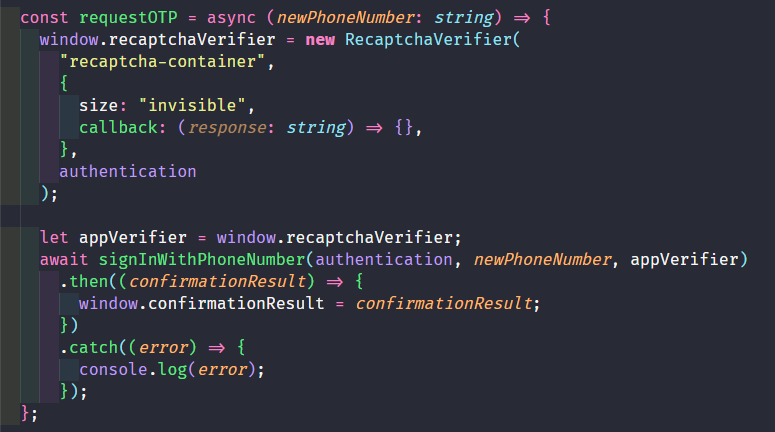


* Cấu hình firebase OTP

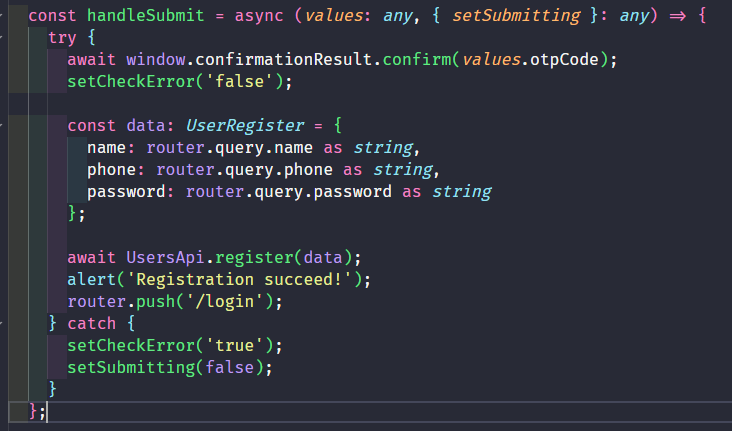


* Tạo recaptcha và gọi phương thức signInWithPhoneNumber từ thư viện firebase sau đó chuyển sang trang OTP

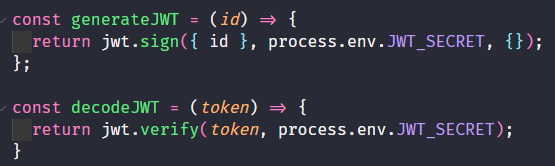




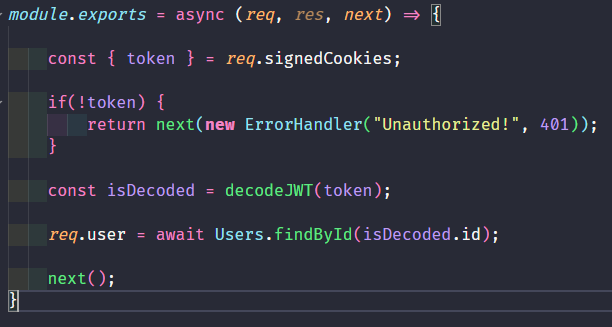
* Trang OTP check code từ input user để kiểm tra và trả về kết quả. Nếu đúng thì sẽ gọi api tạo tài khoản và lưu vào db

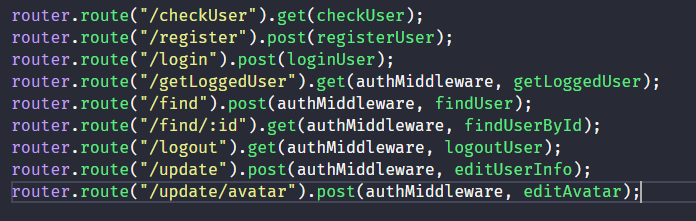


* + 1. Đăng nhập
* Tạo phương thức khởi tạo và giải mã JWT

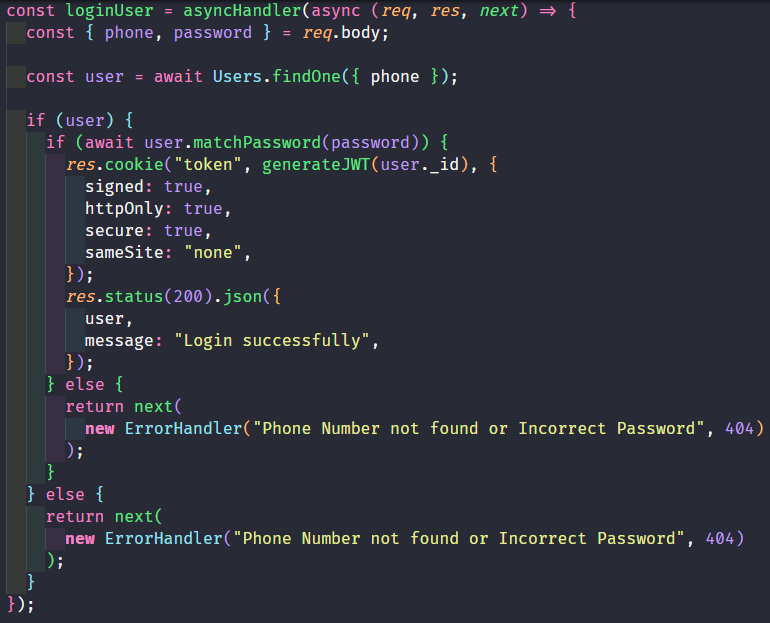


* Tạo middleware nếu như APIs nào cần phải login trước thì sẽ kiểm tra và trả lỗi nếu như chưa login. Và sử dụng middle bằng cách insert trước controller trong route

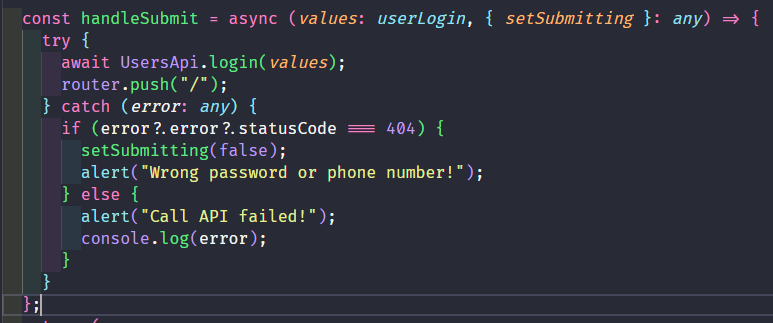




* Tạo API đăng nhập, nếu như input hợp lệ thì sẽ trả về kèm với cookie là JWT được tạo từ id user vừa đăng nhập

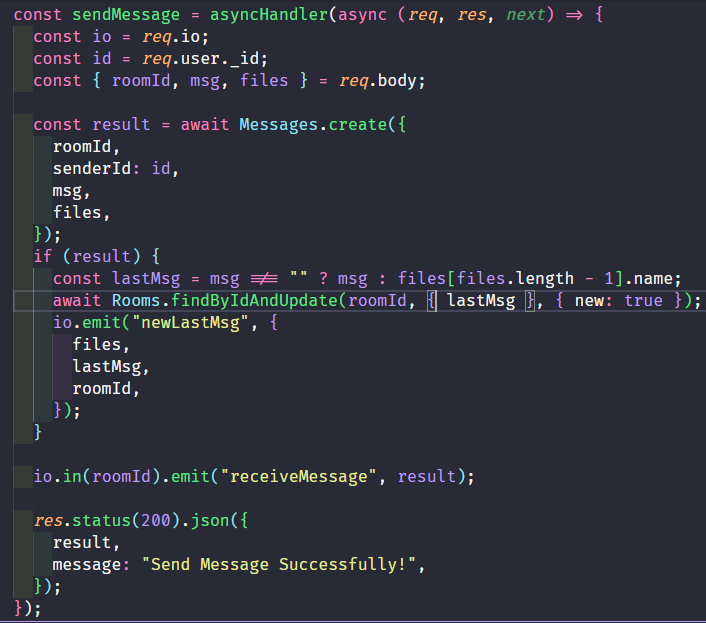


* Phía client khi người dùng điền đầy đủ thông tin và đăng nhập thì sẽ gọi api và chuyển sang trang Home nếu như thông tin hợp lệ

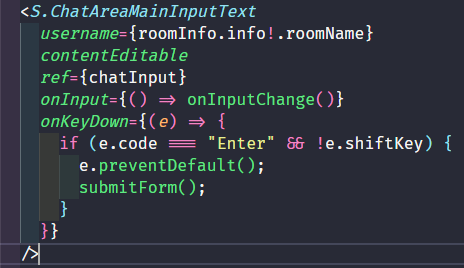


### Gửi nhận tin nhắn

* API: Khi có user gửi tin nhắn, api sendMessage sẽ được gọi để nhận input từ user và lưu vào db sau đó cập nhật lastMsg ở room user đang chat.



* UI: Nhóm em sử dụng thẻ span kèm với thuộc tính contentEditable để thay thế cho thẻ input để có được hiệu ứng scroll khi user nhập tin nhắn dài. Giá trị input của user sẽ được lưu trong biến ref chatInput

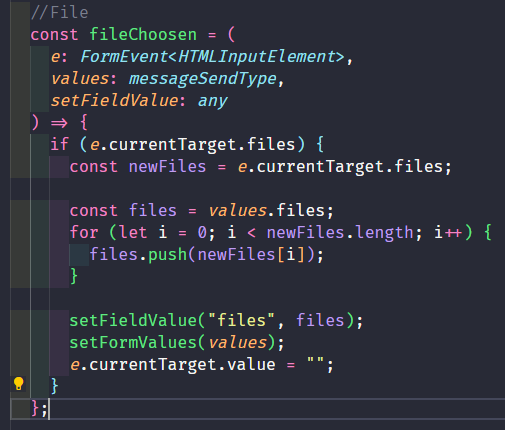


### Gửi nhận file + hình ảnh

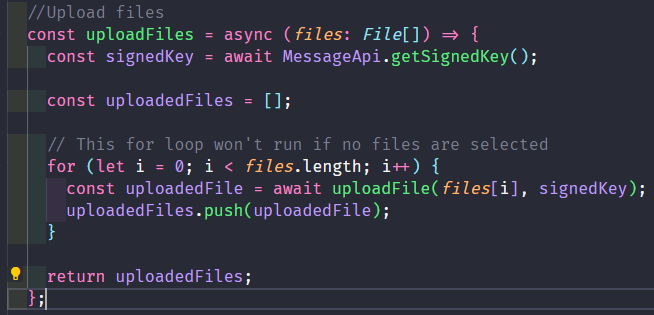
* Setup API tạo signed key

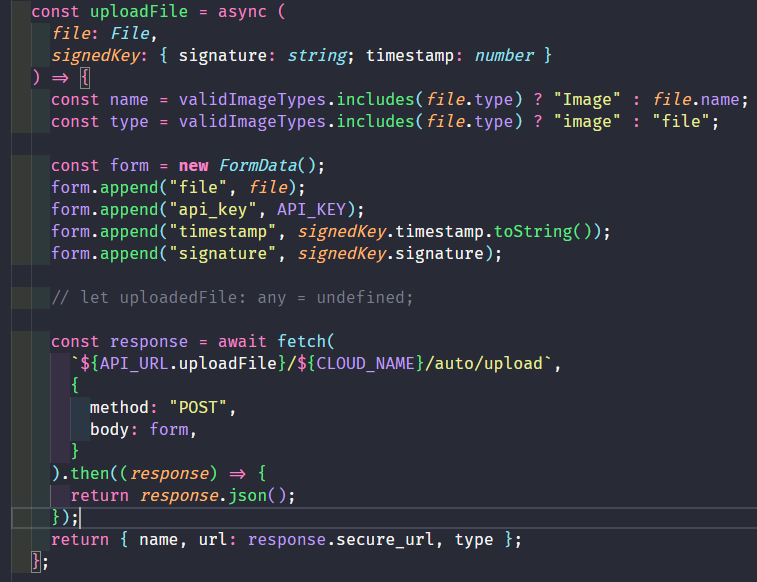


* Cấu hình value formik để chứa các giá trị file upload từ user



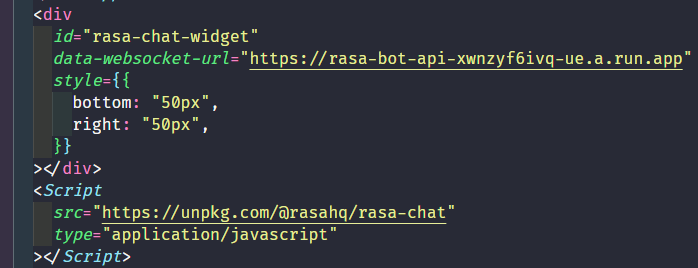
* Sử dụng signed key từ server để upload từng file khi user gửi tin





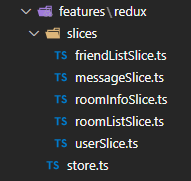
### Tích hợp chatbot y tế

* Project tham khảo: [github.com/Medic-Bot-India/rasaModel](https://github.com/Medic-Bot-India/rasaModel)
* Nhóm chúng em tìm hiểu về rasa framework và tham khảo project tìm được ở trên để train và tích hợp vào webapp.
* Link deploy tham khảo: [github.com/souvikg10/rasa-gcloud-deploy](https://github.com/souvikg10/rasa-gcloud-deploy)
* Tích hợp chat widget vào webapp với config url endpoint từ domain đã được deploy.



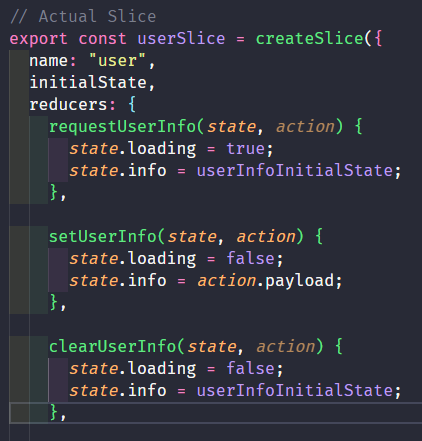
### Cấu hình Redux

* Cấu hình các slices



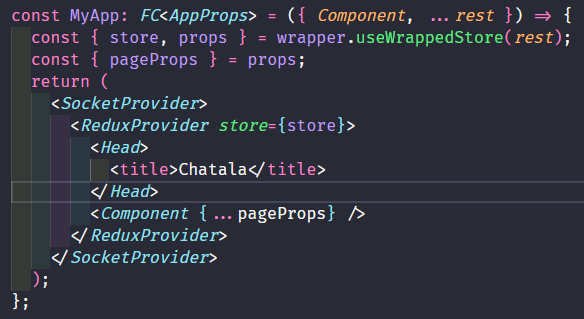
* Trong mỗi slice sẽ cấu hình thông tin của reducer, giá trị khởi tạo và các action cần thiết theo yêu cầu logic





* Store sẽ được cấu hình để lưu trữ tất cả reducer và tích hợp vào app





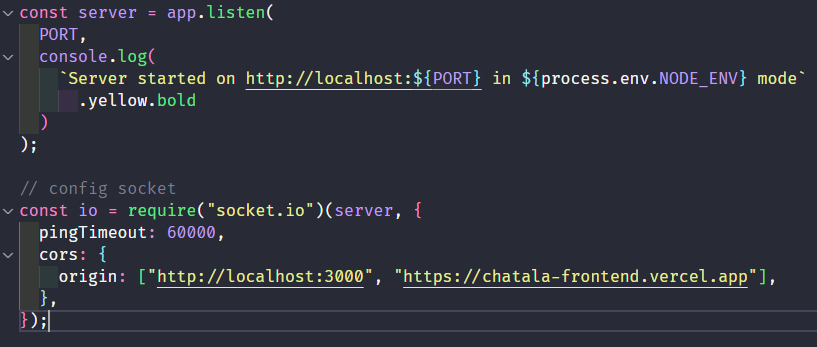
* Khi cần cập nhật hoặc sử dụng state từ redux thì chúng ta sẽ dùng useDispatch và useSelector





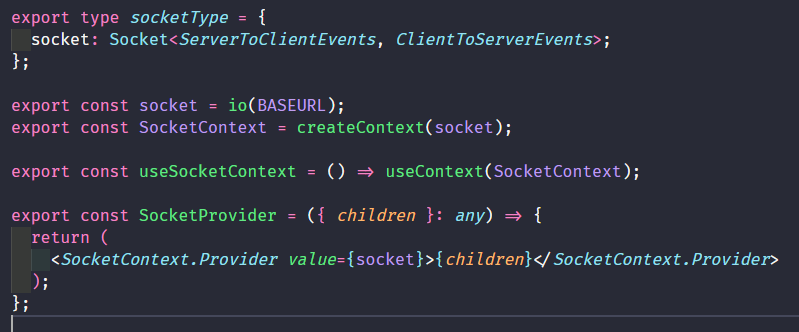
### Cấu hình Realtime

* Phía server: Socket được cấu hình trên instance của server và sử dụng các phương thức **on** hoặc **emit** để lắng nghe các sự kiện khi phía client **emit**





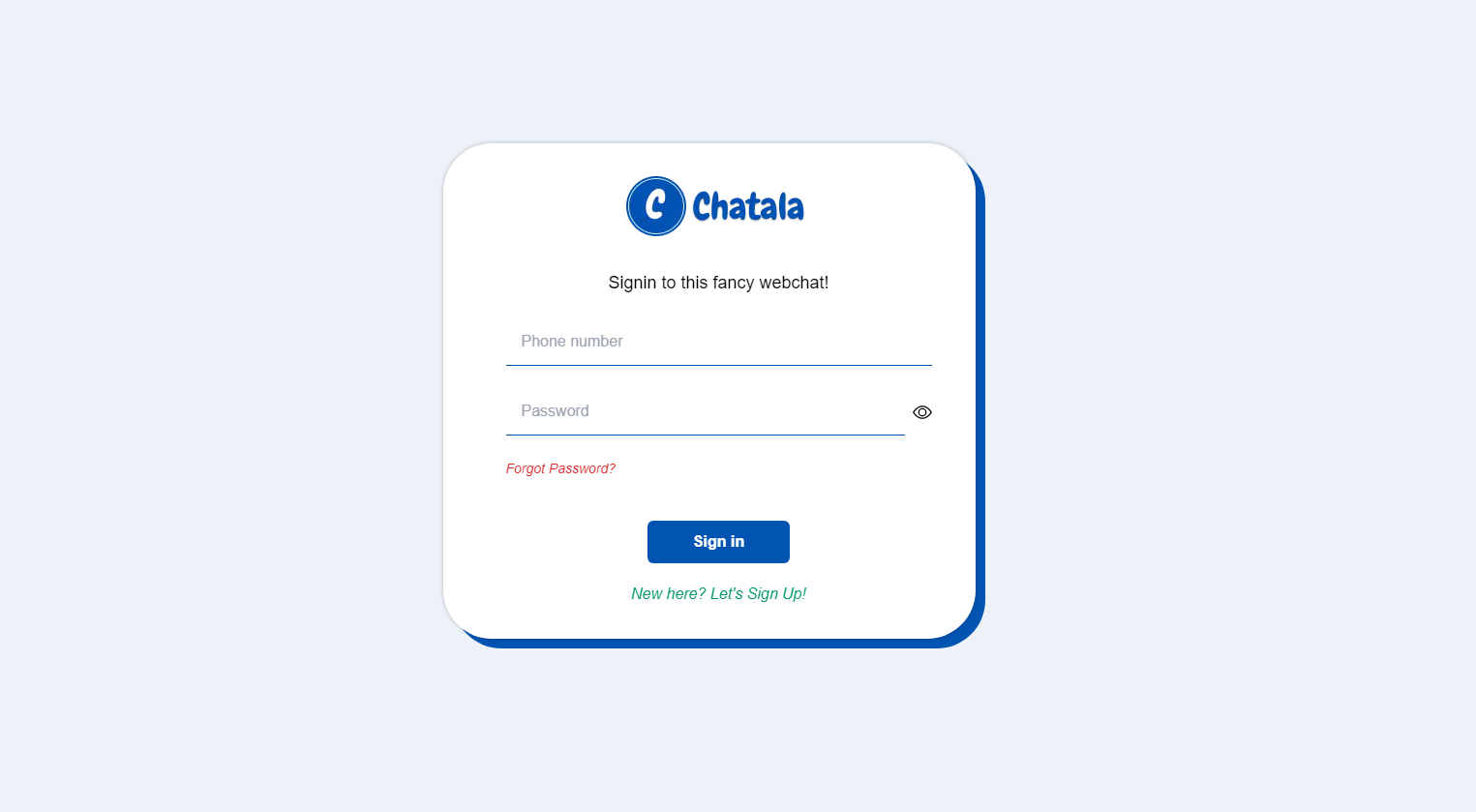
* Phía client: Sử dụng useContext để cấu hình một biến toàn cục là socket, sau đó kết nối tới url của server và cũng cấu hình các trình lắng nghe sự kiện và gửi sự kiện bằng các phương thức **on** và **emit** như phía server.



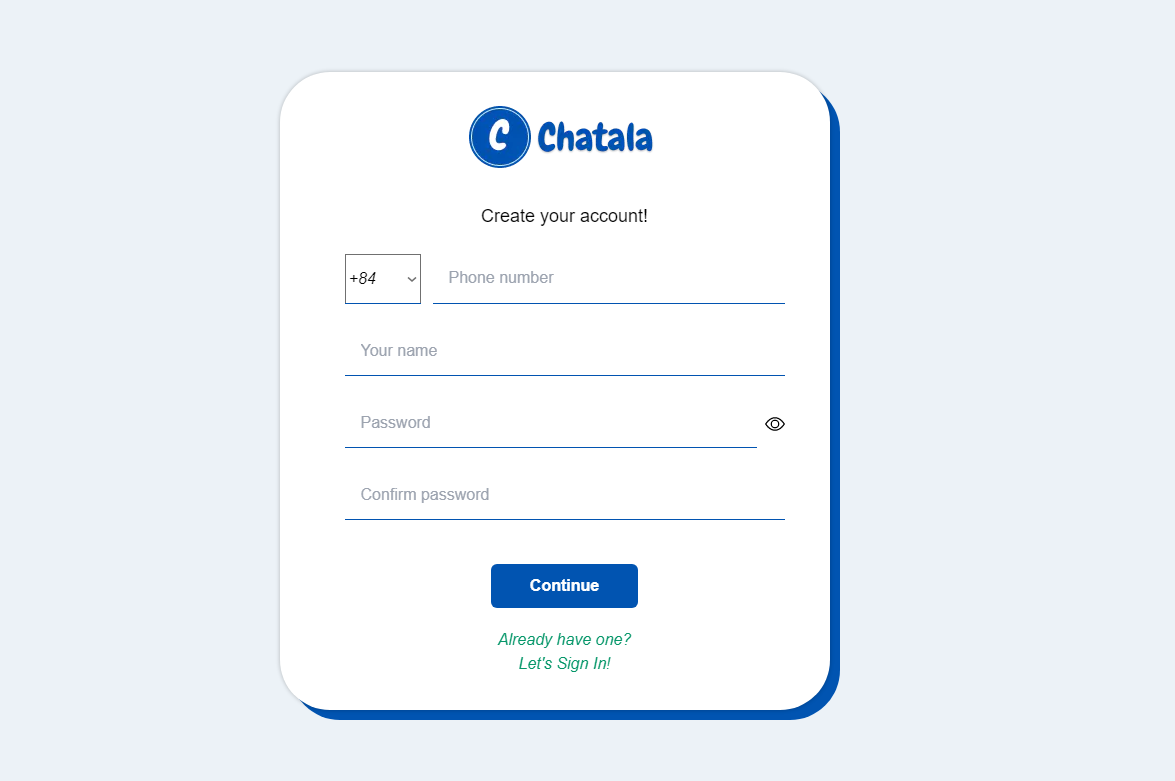


## DEMO

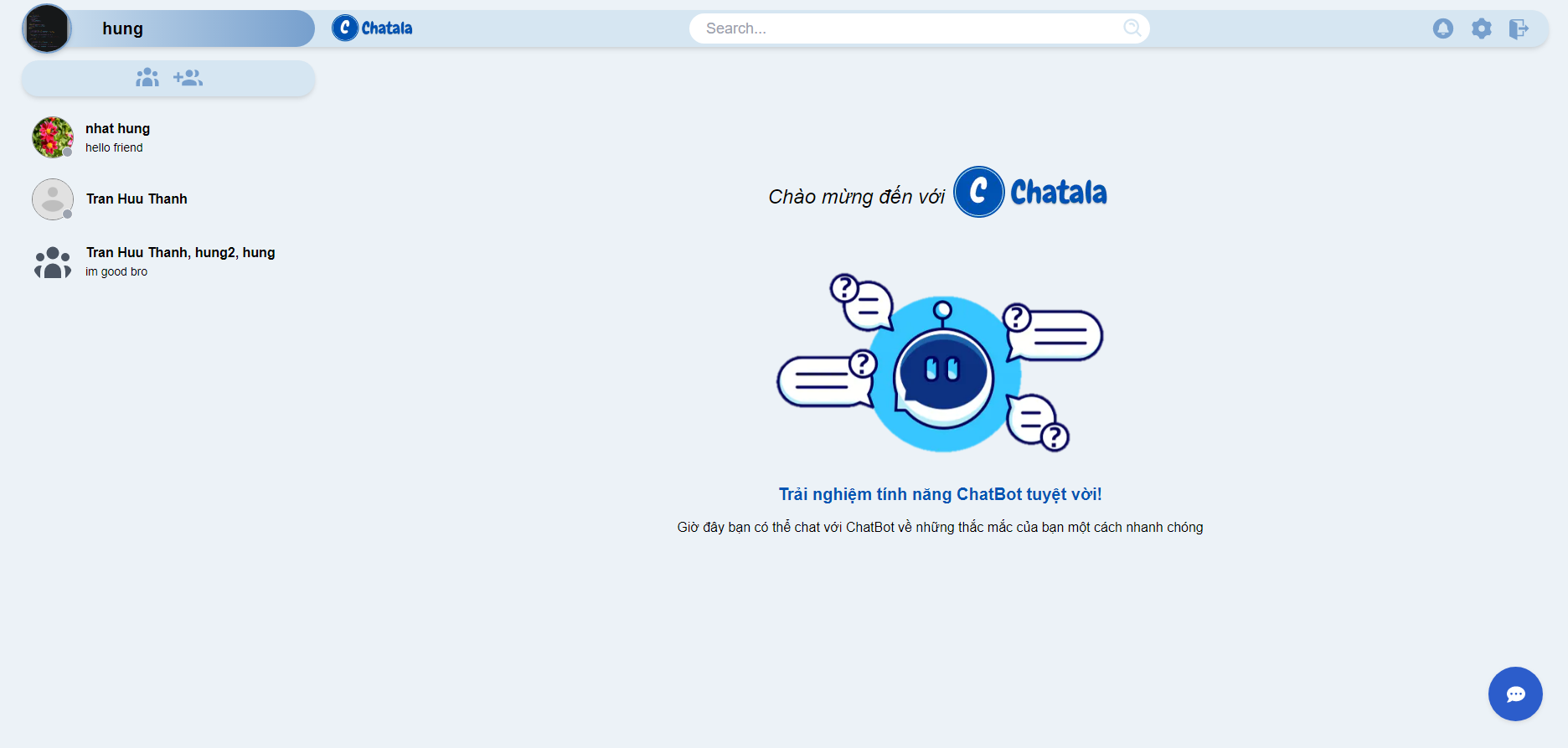
* Link Deploy: [chatala-frontend.vercel.app](https://chatala-frontend.vercel.app)
* Link Github: [github.com/ngphuc2808/Chatala](https://github.com/ngphuc2808/Chatala)
* Giao diện Login



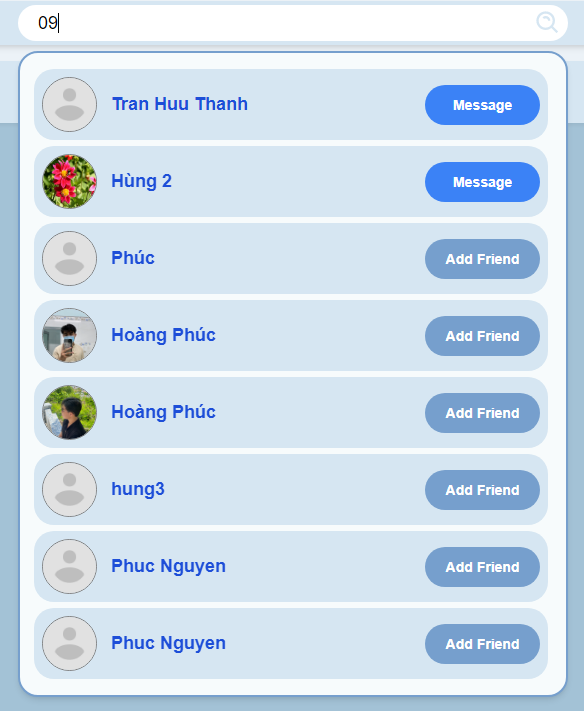
* Giao diện Register



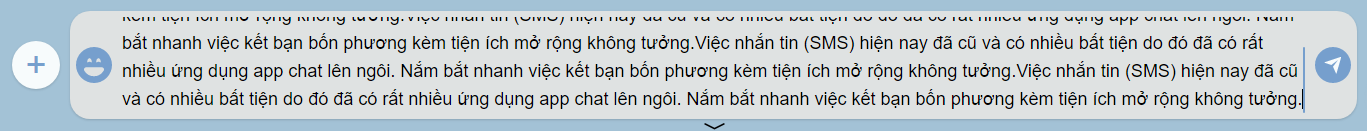
* Giao diện Home



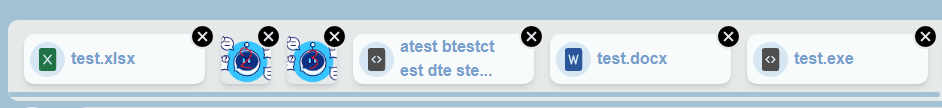
* Giao diện Search



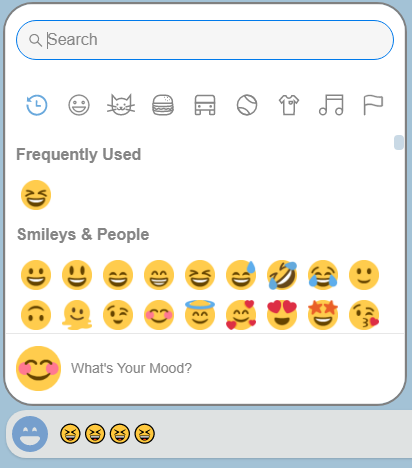
* Giao diện Input, khi text dài thì input sẽ mở rộng theo chiều dọc tới một mức nhất định thì sẽ chuyển qua scroll



* Giao diện khi upload file chuẩn bị gửi, icon sẽ thay đổi theo tùy loại file đc upload.



* Giao diện lựa chọn emoji



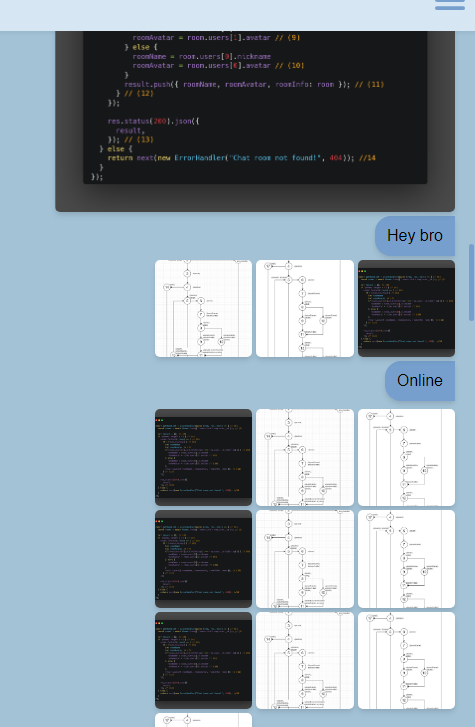
* Biểu tượng typing sẽ hiện lên khi đối phương đang typing



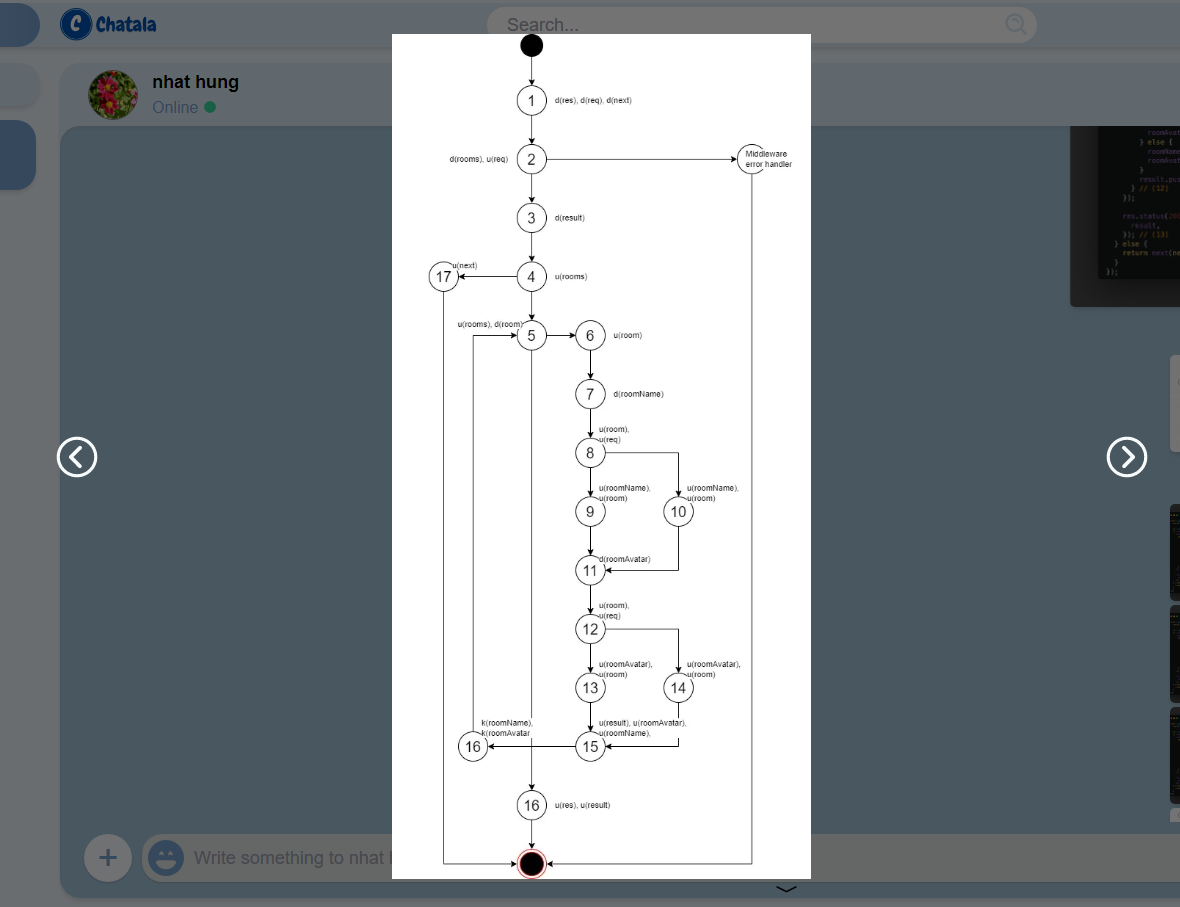
* Giao diện các file đã được gửi



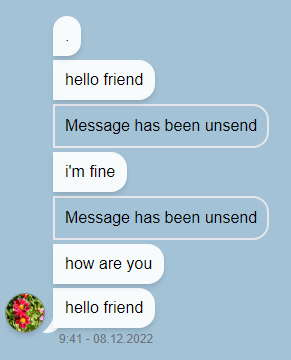
* Giao diện của một hoặc nhiều hình ảnh đã được gửi



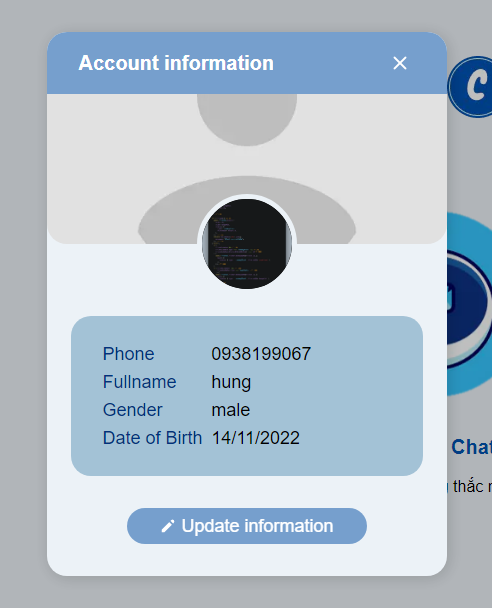
* Người dùng có thể nhấp vào 1 hình ảnh để xem chi tiết và có thể di chuyển qua các hình ảnh khác.



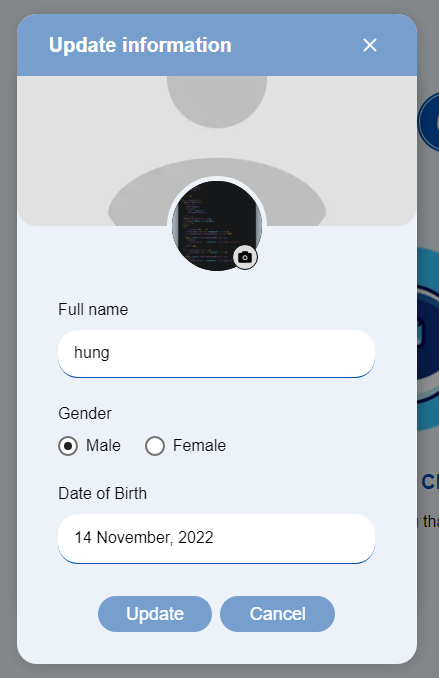
* Giao diện tin nhắn đã được unsend



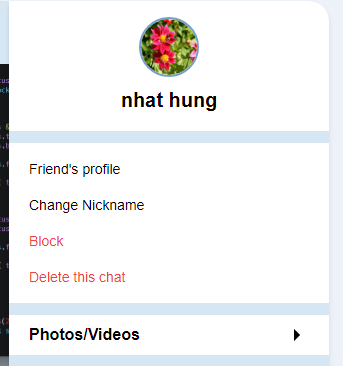
* Giao diện thông tin người dùng



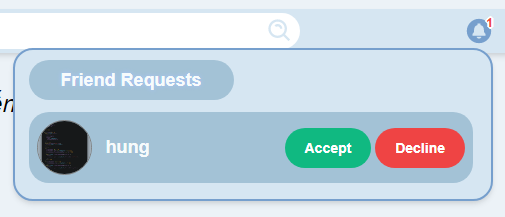
* Giao diện chỉnh sửa thông tin người dùng



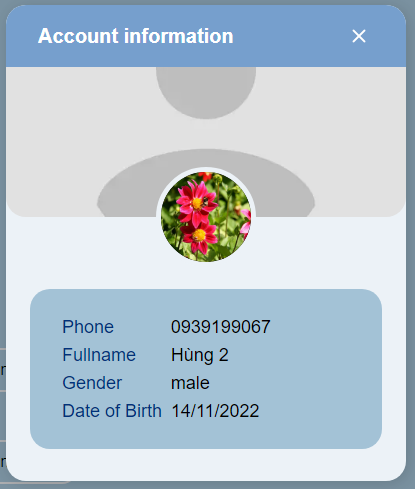
* Giao diện option của khung chat



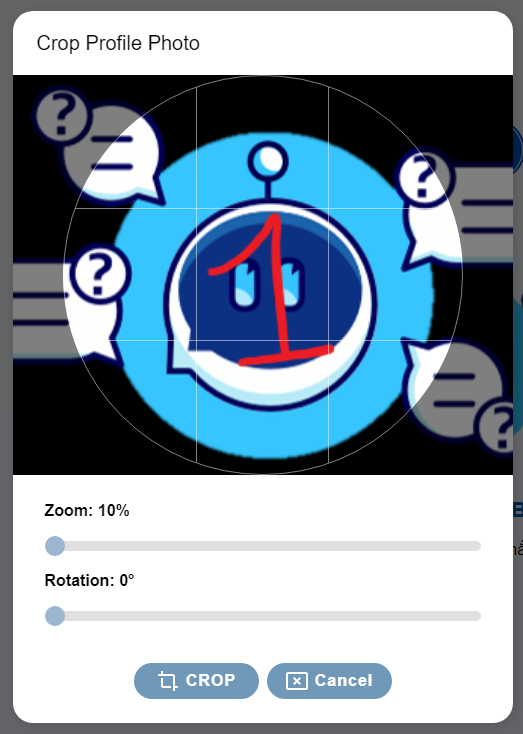
* Giao diện thông báo



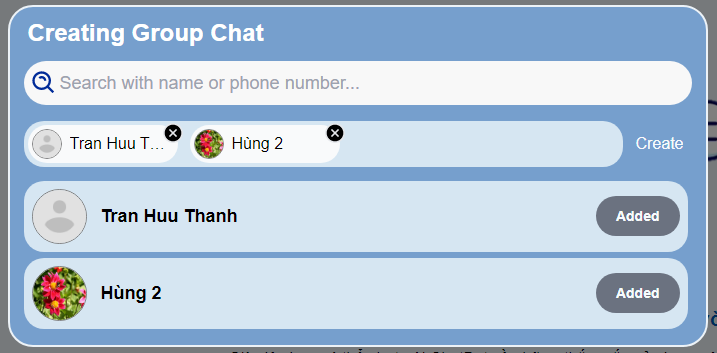
* Giao diện thông tin bạn bè



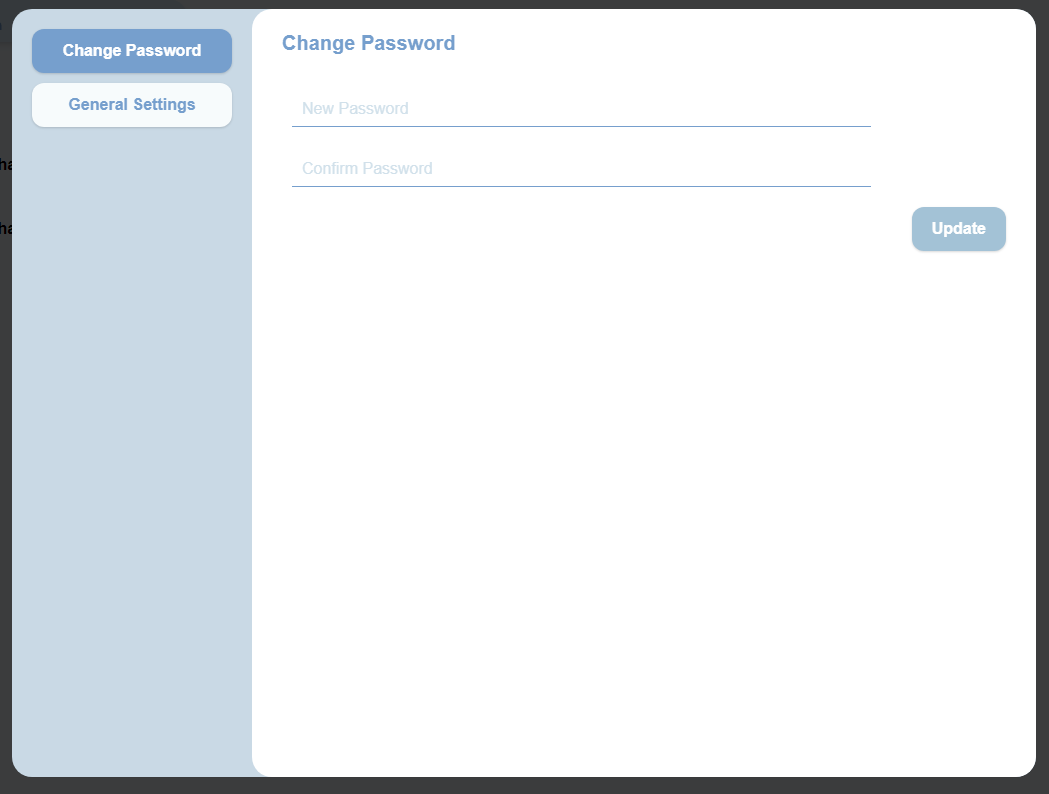
* Giao diện cắt ảnh khi thay đổi avatar



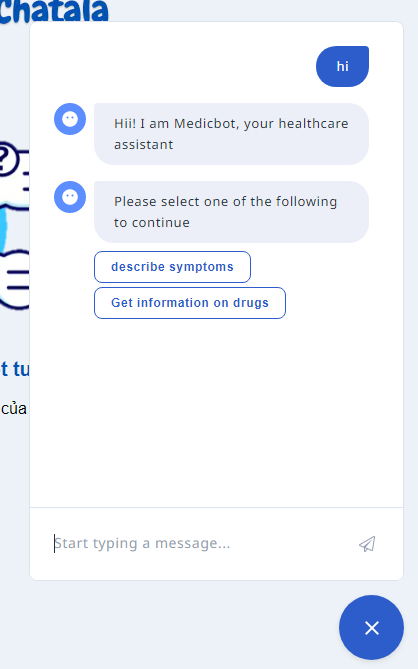
* Giao diện tạo nhóm chat



* Giao diện cài đặt



* Giao diện của chatbot



# PHẦN KẾT LUẬN

## Kết quả đạt được

* Trải qua 2 tháng hơn để tìm hiểu về công nghệ mới, tìm hiểu về lĩnh vực ứng dụng mới. Nhóm em đã thành công trong việc tìm hiểu và phát triển ứng dụng realtime, cụ thể là webchat với các tính năng cốt lõi của một webchat thông thường. Từ đó tụi em cũng được tham khảo thêm nhiều về cách cấu hình cấu trúc của một dự án MERN và workflow giữa client và server.
* Không chỉ thế tụi em còn được tìm hiểu thêm về các công cụ hỗ trợ cho việc code dễ dàng và nhất quán hơn như Redux, Tailwind, Styled Component,...
* Cuối cùng là kiến thức cơ bản về framework Rasa giúp cho việc tạo ra một custom chatbot theo yêu cầu của người thiết kế dễ dàng và nhanh chóng. Cách tích hợp Rasa Widget vào webapp và cách deploy Rasa project.

## Khó khăn

* Do việc lần đầu tiếp xúc các kiến thức mới, thư viện mới, lĩnh vực webapp mới nên nhóm em phải tốn nhiều thời gian hơn để tìm hiểu và ứng dụng, vì vậy cũng gây ra nhiều lỗi trong quá trình phát triển nên từ đó mức độ hoàn thành dự án không được hoàn tất 100%.
* Và cũng do chuyên môn nhóm em không về Machine Learning nên phần chatbot sẽ có thể có sai sót khó tránh khỏi về mặt lý thuyết và cách ứng dụng trong báo cáo.

## Hướng phát triển

* Với việc dự án hiện tại chỉ có những tính năng cốt lõi của một webchat thì nhóm em dự định sẽ bổ sung thêm các tính năng nên có khác của một webchat như là: gửi sticker/gif, tính năng bạn chung, tính năng block, video call,...

# TÀI LIỆU THAM KHẢO

[1] ReactJS: [reactjs.org](https://reactjs.org/)

NextJS: [nextjs.org](https://nextjs.org/)

TypeScript: [www.typescriptlang.org](http://www.typescriptlang.org)

Video EJS, Express cơ bản: [youtube.com/watch?v=A01KtJTv1oc&t=962s](https://www.youtube.com/watch?v=A01KtJTv1oc&t=962s)

[2] Tài liệu về NodeJS: [w3schools.com/nodejs/default.asp](https://www.w3schools.com/nodejs/default.asp)

Tài liệu về Express: [expressjs.com](https://expressjs.com)

[3] MERN: [www.youtube.com/watch?v=98BzS5Oz5E4](http://www.youtube.com/watch?v=98BzS5Oz5E4)

[4] MongoDB: [www.mongodb.com](https://www.mongodb.com)

[5] JWT & Cookies: [topdev.vn/blog/jwt-la-gi/](https://topdev.vn/blog/jwt-la-gi/), [itnavi.com.vn/blog/cookie-la-gi](https://itnavi.com.vn/blog/cookie-la-gi)

[6] Rasa: [rasa.com](https://rasa.com)

[7] Phone Auth Firebase: [firebase.google.com/docs/auth/android/phone-auth](https://firebase.google.com/docs/auth/android/phone-auth)

[8] SocketIO: [socket.io](https://socket.io)

# PHÂN CÔNG CÔNG VIỆC

| **MSSV** | **Họ Tên** | **Công việc** |
| --- | --- | --- |
| 19110213 | Trần Nhật Hùng | * Thiết kế Figma, thiết kế sườn báo cáo * Thiết kế Model, Thiết kế API * Code Frontend, Backend * Tìm hiểu tích hợp Chatbot * Viết báo cáo |
| 19110007 | Trần Hữu Thanh | * Code Frontend, Backend * Tìm hiểu tích hợp Chatbot * Upload file lên Cloudinary * Viết báo cáo |
| 19110268 | Nguyễn Hoàng Phúc | * Thiết kế Figma * Code Frontend, Backend * Tìm hiểu tích hợp OTP * Viết báo cáo |