

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



---

# ASSIGNMENT 1

## NETWORK APPLICATION

---

Lớp: **L09**

Nhóm: **BrSE**

Học kỳ: **HK221**

Người hướng dẫn: **Lê Bảo Khánh**

Sinh viên thực hiện	<b>Huỳnh Văn Nhân</b>	<b>2011734</b>
	<b>Lâu Lâm Tường</b>	<b>2012392</b>
	<b>Bùi Khánh Vĩnh</b>	<b>2010091</b>
	<b>Trần Nguyên Vũ</b>	<b>2012445</b>

Thành phố Hồ Chí Minh, Tháng 11 năm 2022

## Mục lục

<b>I. NHIỆM VỤ</b>	<b>2</b>
<b>II. PHASE 1</b>	<b>3</b>
1 Yêu cầu chức năng . . . . .	3
2 Yêu cầu phi chức năng . . . . .	3
3 Giao thức giao tiếp client - server . . . . .	4
4 Giao thức trò chuyện . . . . .	5
5 Giao thức trao đổi file . . . . .	6
<b>III. PHASE 2</b>	<b>8</b>
1 Mô tả tính năng . . . . .	8
2 Mô tả thiết kế ứng dụng . . . . .	9
3 Kiểm tra và chạy thử ứng dụng . . . . .	12
4 Đánh giá chung . . . . .	15
<b>IV. KẾT LUẬN</b>	<b>16</b>
<b>V. TÀI LIỆU THAM KHẢO</b>	<b>17</b>

## I. NHIỆM VỤ

STT	MSSV	Họ và tên	Nhiệm vụ
1	2011734	Huỳnh Văn Nhân	<ul style="list-style-type: none"><li>• Đề xuất và lên ý tưởng</li><li>• Kiểm tra và chạy thử ứng dụng.</li><li>• Soạn thảo báo cáo.</li></ul>
2	2012392	Lâu Lâm Tường	<ul style="list-style-type: none"><li>• Đề xuất và lên ý tưởng.</li><li>• Phát triển ứng dụng (chính).</li><li>• Soạn thảo slide thuyết trình.</li></ul>
3	2010091	Bùi Khánh Vĩnh	<ul style="list-style-type: none"><li>• Đề xuất và lên ý tưởng.</li><li>• Kiểm tra và chạy thử ứng dụng.</li><li>• Soạn thảo báo cáo.</li></ul>
4	2012445	Trần Nguyên Vũ	<ul style="list-style-type: none"><li>• Đề xuất và lên ý tưởng</li><li>• Phát triển ứng dụng (phụ).</li><li>• Soạn thảo báo cáo</li></ul>

## II. PHASE 1

Trong đề tài này, nhóm sẽ xây dựng một ứng dụng mạng dùng để trò chuyện giữa các người dùng với nhau.

### 1 Yêu cầu chức năng

- **Tạo tài khoản:** Người dùng sẽ thực hiện việc tạo một tài khoản với những thông tin cơ bản (tên tài khoản, mật khẩu, họ và tên, ngày sinh, giới tính). Các thông tin này sẽ được lưu trữ ở phía máy chủ và có thể truy xuất được từ phía máy chủ khi đăng nhập.
- **Quản lý tài khoản:** Người dùng sẽ đăng nhập và truy cập vào máy chủ. Từ đó người dùng có thể theo dõi các thông tin cá nhân cơ bản cũng như danh sách bạn bè trò chuyện. Người dùng cũng có thể thay đổi các thông tin cá nhân cũng như thêm hoặc xóa bạn trò chuyện.
- **Trò chuyện:** Các người dùng sẽ kết nối và trò chuyện 2 người với nhau, đồng thời cho phép các người dùng có thể gửi nhận các file dữ liệu cho nhau trong khi trò chuyện.

### 2 Yêu cầu phi chức năng

- Ứng dụng được xây dựng dựa trên bộ giao thức Internet TCP/IP. Đây là sự kết hợp giữa giao thức TCP (ở tầng transport) có chức năng xác định các ứng dụng và tạo ra các kênh giao tiếp. TCP cũng có chức năng quản lý các thông tin khi được chia nhỏ để truyền tải qua internet. Giao thức này sẽ tập hợp các thông tin này theo đúng thứ tự, đảm bảo truyền tải thông tin chính xác tới địa chỉ đến. Trong khi đó giao thức IP (ở tầng mạng) sẽ đảm bảo thông tin được truyền đến đúng địa chỉ. IP sẽ gán các địa chỉ và định tuyến từng gói thông tin. Mỗi mạng sẽ có 1 địa chỉ IP để xác định được chính xác nơi chuyển/nhận thông tin, dữ liệu.
- Ứng dụng được xây dựng dựa trên mô hình hybrid, là sự kết hợp giữa hai mô hình chính trong một ứng dụng mạng đó là mô hình client - server và mô hình mạng ngang hàng peer - to - peer.
  - Mô hình client - server yêu cầu phải có một máy chủ làm trung tâm, lưu trữ những dữ liệu từ người dùng và có thể được truy xuất bởi người dùng. Máy chủ sẽ xử lý những yêu cầu đến từ người dùng tại cùng một thời điểm khác nhau.
  - Mô hình mạng ngang hàng peer - to - peer cho phép các người dùng có thể trò

chuyện với nhau mà không cần phải gửi nhận dữ liệu qua server như bên thứ ba. Server lúc này chỉ đóng vai trò giúp người dùng truy xuất thông tin cần thiết để có thể thiết lập một kết nối peer - to - peer khi người dùng muốn trò chuyện

- Tại một thời điểm nhất định, server phải xử lý tác vụ đến từ nhiều người dùng khác nhau. Và tại một thời điểm, một người dùng có thể thiết lập kết nối trò chuyện với nhiều người dùng khác nhau trong mạng peer - to - peer.
- Người dùng có thể gửi nhận các file với những định dạng khác nhau, kích thước nhỏ (10MB).
- Tin nhắn được gửi đi trong vòng 1 giây.

### 3 Giao thức giao tiếp client - server

Giao thức này được sử dụng trong tính năng tạo tài khoản và quản lý tài khoản của người dùng

- **Các loại thông điệp được trao đổi:** REQUEST (yêu cầu kết nối), RESPONSE (trả về phản hồi kết nối)
- **Cú pháp thông điệp (Syntax):** Ứng với mỗi loại thông điệp, ta sẽ định nghĩa ra các trường thông tin và ngữ nghĩa như sau
  - **Thông điệp REQUEST:**
    - \* Method: REQUEST
    - \* From: Địa chỉ bên gửi
    - \* To: Địa chỉ bên nhận
    - \* Time: Thời điểm thực hiện
  - **Thông điệp RESPONSE:**
    - \* Method: RESPONSE <code>
    - \* From: Địa chỉ bên gửi
    - \* To: Địa chỉ bên nhận
    - \* Time: Thời điểm thực hiện
- **Quy tắc (Rules):**
  - \* Mã trạng thái (code) sẽ được trả về trong trường Method của thông điệp RESPONSE với những quy ước sau:
    - 100: Kết nối thành công
    - 101: Kết nối thất bại
  - \* Trường Time trong các thông điệp trên sẽ tuân theo định dạng dd/mm/yyyy - hh:mm:ss

## 4 Giao thức trò chuyện

Giao thức này được sử dụng trong tính năng trò chuyện giữa các người dùng với nhau trong mô hình peer - to - peer

- **Các loại thông điệp được trao đổi:** START (khởi tạo kết nối), RESPONSE (phản hồi kết nối), SEND (gửi tin nhắn), QUIT (thoát kết nối)
- **Cú pháp thông điệp (Syntax):** Ứng với mỗi loại thông điệp, ta sẽ định nghĩa ra các trường thông tin và ngữ nghĩa như sau

- **Thông điệp START:**

- \* Method: START
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận
- \* Time: Thời điểm thực hiện

- **Thông điệp RESPONSE:**

- \* Method: RESPONSE <code>
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận
- \* Time: Thời điểm thực hiện

- **Thông điệp SEND:**

- \* Method: SEND
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận
- \* Time: Thời điểm thực hiện
- \* Length: Kích thước thông điệp
- \* Content: Nội dung tin nhắn

- **Thông điệp QUIT:**

- \* Method: QUIT
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận
- \* Time: Thời điểm thực hiện

- **Quy tắc (Rules):**

- \* Mã trạng thái (code) sẽ được trả về trong trường Method của thông điệp RESPONSE với những quy ước sau:
  - 200: Kết nối được thiết lập thành công

– 201: Kết nối thất bại

\* Trường Time trong các thông điệp trên sẽ tuân theo định dạng dd/mm/yyyy - hh:mm:ss

## 5 Giao thức trao đổi file

Giao thức này tương tự như trên cũng được sử dụng trong tính năng trò chuyện giữa các người dùng với nhau trong mô hình peer - to - peer

- **Các loại thông điệp được trao đổi:** START (khởi tạo kết nối), REPOSE (Phản hồi kết nối), ALERT (Cảnh báo), SEND (gửi tin nhắn)
- **Cú pháp thông điệp (Syntax):** Ứng với mỗi loại thông điệp, ta sẽ định nghĩa ra các trường thông tin và ngữ nghĩa như sau

– **Thông điệp START:**

- \* Method: START
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận
- \* Time: Thời điểm thực hiện

– **Thông điệp REPOSE:**

- \* Method: REPOSE
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận
- \* Time: Thời điểm thực hiện
- \* Accept: Trả về connection có được kết nối

– **Thông điệp SEND:**

- \* Method: SEND
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận
- \* Time: Thời điểm thực hiện
- \* Length: Kích thước thông điệp
- \* Content: Nội dung file
- \* Type file: Loại file được gửi

– **Thông điệp ALERT:**

- \* Method: ALERT
- \* From: Địa chỉ bên gửi
- \* To: Địa chỉ bên nhận

- \* Time: Thời điểm thực hiện
- \* Length: Kích thước thông điệp
- \* Content: Nội Dung Cảnh Báo
- **Quy tắc (Rules):** File chỉ được gửi khi có các loại file được cho phép và với dung lượng không quá lớn. Khi vi phạm các điều trên thì method ALERT sẽ thực hiện cảnh báo cho người dùng.



### III. PHASE 2

#### 1 Mô tả tính năng

Ứng dụng do nhóm xây dựng nhằm mục đích cung cấp một giải pháp trò chuyện giữa các người dùng trong mạng internet với nhau. Các tính năng mà ứng dụng cung cấp bao gồm như sau:

- *Tính năng quản lý tài khoản:* Người dùng muốn sử dụng ứng dụng phải tạo một tài khoản bao gồm tên người dùng và mật khẩu. Tài khoản cho phép người dùng lưu danh sách bạn bè trò chuyện. Người dùng muốn tham gia trò chuyện phải thực hiện đăng nhập. Để hiện thực tính năng này, chúng ta sẽ sử dụng giao thức được gọi là **giao thức giao tiếp client - server**.
- *Tính năng quản lý bạn bè:* Người dùng có thể truy xuất thông tin từ máy về thông tin danh sách bạn bè, trạng thái kết nối của bạn bè. Người dùng có thể chuyển đổi cuộc trò chuyện bằng việc thao tác với danh sách bạn bè. Người dùng có thể bắt đầu cuộc trò chuyện với bạn bè đang trong trạng thái sẵn sàng kết nối. Để hiện thực tính năng này, chúng ta sẽ sử dụng giao thức được gọi là **giao thức giao tiếp client - server**.
- *Tính năng trò chuyện:* Cho phép người dùng có thể trò chuyện với một người dùng khác thông qua mô hình mạng peer - to - peer. Người dùng có thể trò chuyện với nhiều người dùng khác trong ứng dụng, tuy nhiên tại một thời điểm nhất định, người dùng chỉ có thể gửi nhận tin nhắn đến một người dùng nhất định. Để hiện thực tính năng này, chúng ta sẽ sử dụng giao thức được gọi là **giao thức trò chuyện**.
- *Tính năng gửi nhận file:* Trong khi trò chuyện, người dùng có thể gửi nhận file với kích thước tối đa là 10MB, các file có thể tuân theo những định dạng khác nhau. Để hiện thực tính năng này, chúng ta sẽ sử dụng giao thức được gọi là **giao thức trao đổi file**.
- *Tính năng biểu tượng cảm xúc:* Trong khi trò chuyện, người dùng có thể gửi nhận các biểu tượng cảm xúc tương tự như các ứng dụng trò chuyện phổ biến hiện nay. Ứng dụng sẽ cung cấp một phương thức để người dùng tự do chọn lựa các biểu tượng cảm xúc với các thể loại khác nhau.

Ứng dụng sẽ hoạt động dựa trên mô hình hybrid, là sự kết hợp giữa mô hình client - server và mô hình peer - to - peer để thích ứng với các tính năng mà ứng dụng cung cấp:

- *Mô hình client - server:* Tạo ra một máy chủ để lưu trữ thông tin của người dùng

cũng như cho phép người dùng lưu trữ danh sách bạn bè. Khi người dùng muốn bắt đầu một cuộc trò chuyện, người dùng có thể lấy thông tin kết nối từ danh sách bạn bè ở trên máy chủ.

- *Mô hình peer - to - peer*: Cho phép người dùng có thể trực tiếp trò chuyện cũng như gửi nhận file với bạn bè mà không cần thông qua máy chủ. Tuy nhiên, người dùng sẽ cần lấy thông tin kết nối của bạn bè từ máy chủ.

Các yêu cầu khác mà ứng dụng cần đáp ứng:

- Hỗ trợ được nhiều trình duyệt phổ biến như Chrome, Edge, Firefox, Safari....
- Ứng dụng có giao diện đẹp, tương tác thân thiện với người dùng.
- Có thể triển khai và chạy được ở trên internet.
- Gửi nhận tin nhắn, file với độ trễ thấp.
- Đáp ứng các tương tác của người dùng nhanh và mượt.

## 2 Mô tả thiết kế ứng dụng

### \* Giới thiệu MERN Stack



MERN Stack là một tập hợp các công cụ open source để giúp các nhà phát triển có thể xây dựng một ứng dụng Web hoàn chỉnh bao gồm:

- *MongoDB*: Là hệ quản trị cơ sở dữ liệu phi quan hệ phổ biến hiện nay. Thích hợp trong việc lưu trữ dữ liệu của ứng dụng web với quy mô không lớn cho phép thực hiện truy vấn với tốc độ nhanh. MongoDB được sử dụng kết hợp với thư viện Mongoose, giúp cho việc giao tiếp giữa ứng dụng và dữ liệu trở nên dễ dàng hơn.
- *ExpressJS*: Là một framework được xây dựng trên nền tảng của Nodejs. Nó cung cấp

các tính năng mạnh mẽ để phát triển web hoặc mobile. Expressjs hỗ trợ các method HTTP và middleware tạo ra API mạnh mẽ và dễ sử dụng.

- *ReactJS*: Là một thư viện có thể giúp tạo giao diện người dùng động một cách dễ dàng. Là một thư viện các công cụ và chức năng có thể giúp thiết kế, phát triển và hiển thị các thành phần giao diện người dùng. ReactJS là cách tốt nhất để kiểm soát trạng thái của các sự kiện khi có một lượng lớn dữ liệu động cần được cập nhật.
- *NodeJS*: Là một JavaScript runtime được build dựa trên engine JavaScript V8 của Chrome. Node.js sử dụng kiến trúc hướng sự kiện event-driven, mô hình non-blocking I/O làm cho nó nhẹ và hiệu quả hơn. Hệ thống nén của Node.js, npm, là hệ thống thư viện nguồn mở lớn nhất thế giới.

### \* Giới thiệu Socket IO

Socket IO là 1 module trong Node.js được nhà sáng chế tạo ra và phát triển từ năm 2010. Mục đích lớn nhất của Socket io là để tạo môi trường giao tiếp thuận lợi trên Internet giúp trả về các giá trị thực ngay tại thời điểm giao tiếp giữa các bên với nhau (thường là giữa server và client). Việc giao tiếp 2 chiều giữa máy khách và máy chủ được thực hiện bởi Socket IO khi và chỉ khi máy khách có module này trong trình duyệt và máy chủ cũng đã tích hợp sẵn gói socket io. Các ứng dụng sử dụng Socket IO thường đòi hỏi tốc độ phản hồi ngay lập tức. Một số ví dụ điển hình như xổ số, trực tiếp bóng đá, chat... Socket IO không phải là 1 ngôn ngữ, vì vậy nó phải được sử dụng kết hợp với những ngôn ngữ khác như NodeJS, PHP, ASP.NET... Các đặc điểm của Socket IO:

- *Bảo mật cao*: Socket IO được xây dựng dựa trên Engine.IO. Nó sẽ khởi chạy phương thức long-polling trước nhất để kết nối. Sau đó nó mới sử dụng các phương thức giao tiếp tốt hơn như là Websocket. Vì được thiết lập chặt chẽ như vậy nên khi Socket IO xuất hiện nó sẽ tự động tạo những kết nối bảo mật như là: proxy và cân bằng tải hoặc là tường lửa cá nhân và phần mềm chống virus...
- *Kết nối tự động tới server*: Giả sử trong quá trình khởi chạy bị mất kết nối giữa client và server thì Socket IO sẽ tự động gắn kết nối mãi mãi cho đến khi nào server phản hồi lại. Và đây là tính năng có thể tùy chỉnh được nên bạn có quyền chọn không kết nối tự động đến bất kỳ server nào mà mình muốn.
- *Mã hóa nhị phân*: Socket IO có thể hỗ trợ mã hóa nhị phân như ArrayBuffer và Blob trên trình duyệt hoặc là ArrayBuffer và Buffer trong Node.js.
- *Cho phép tạo kênh và phòng*: Đây là 1 tính năng khá nổi bật khi mà socket io có thể tạo ra mối quan hệ giữa các phần hoặc các module riêng lẻ bằng cách tạo ra những

kênh riêng biệt khác nhau. Ngoài việc tạo kênh, nó còn hỗ trợ tạo phòng cho các clients tham gia với mục đích gửi thông báo đến 1 nhóm người dùng được kết nối với 1 số thiết bị nào đó.

### \* Kiến trúc ứng dụng

Ứng dụng được xây dựng và tổ chức trên hai project chính đó là chat-client và chat-server, tuân thủ theo kiến trúc MVC (*Model - View - Controller*). Trong đó View được hiện thực trong project chat-client, còn Model và Controller được hiện thực trong project chat-server.

- *Model*: Cung cấp những phương thức để truy xuất trực tiếp với hệ cơ sở dữ liệu đối với các chức năng tạo tài khoản, đăng nhập, truy xuất danh sách các kết nối người dùng hiện tại. Các phương thức được gọi đến theo yêu cầu đến từ controller.
- *View*: Render các giao diện khác nhau dựa trên sự điều khiển từ controller bao gồm:
  - Hiện thị giao diện đăng nhập và đăng ký cho người dùng, hiển thị các kết quả xử lý liên quan đến lỗi đăng nhập và đăng ký.
  - Hiện thị giao diện trò chuyện khi người dùng đăng nhập thành công, hiển thị tin nhắn từ bên gửi và bên nhận theo thời gian thực. Chuyển đổi cuộc trò chuyện qua lại giữa các người dùng đang kết nối vào hệ thống.
- *Controller*: Hiện thực xử lý cho toàn bộ chương trình, điều khiển View và tương tác với Model một cách thích hợp.
  - Xử lý các yêu cầu đăng ký tài khoản và đăng nhập từ người dùng. Từ đó thực hiện gọi đến Model để trả về kết quả phù hợp và hiển thị lại cho người dùng thông qua View.
  - Xử lý các trạng thái của người dùng, khi một người dùng kết nối đến ứng dụng hoặc thoát khỏi ứng dụng, gọi chung là trạng thái kết nối, để đưa ra phương thức gửi nhận các tin nhắn cũng như file giữa các người dùng một cách thích hợp. Các trạng thái của người dùng sẽ được Controller gọi đến Model để lưu trữ vào cơ sở dữ liệu.
  - Xử lý các hành động gửi nhận tin nhắn cũng như gửi nhận file dữ liệu giữa các người dùng với nhau.
  - Tương tác với API để lưu trữ các file trên nền tảng cloud mà các người dùng trao đổi với nhau. Điều này cho phép dữ liệu file mà các người dùng gửi nhận có thể được truy xuất và tải về thông qua giao diện trò chuyện.

### 3 Kiểm tra và chạy thử ứng dụng

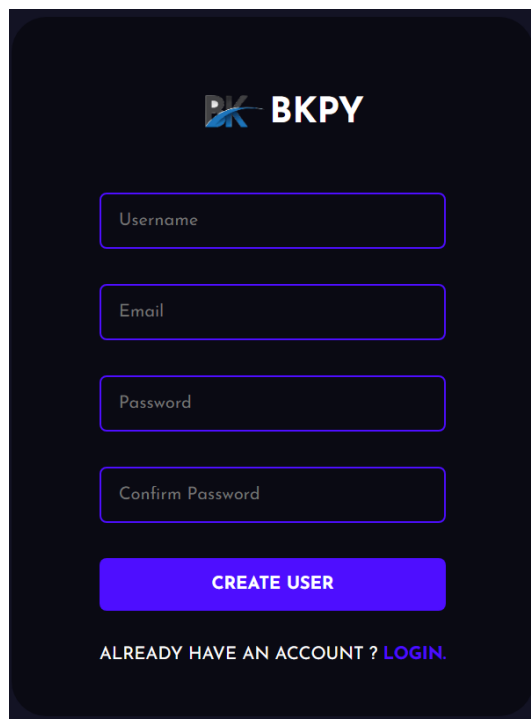
Ứng dụng được triển khai thông qua nền tảng Vercel, qua đó cho phép các người dùng trong public network có thể trò chuyện được với nhau, ứng dụng hoạt động ở đường dẫn <https://chat-client-nine.vercel.app/>. Dưới đây là các tính năng quan trọng mà ứng dụng cung cấp:

#### \* Tính năng đăng ký và đăng nhập:

Khi người dùng truy cập vào đường dẫn như ở trên, nếu người dùng chưa đăng nhập vào ứng dụng lần nào, thì ứng dụng sẽ hiển thị trang giao diện để người dùng đăng nhập vào ứng dụng. Để thực hiện đăng nhập vào ứng dụng, người dùng cần phải có một tài khoản, do đó nếu chưa có tài khoản thì người dùng cần phải thực hiện đăng ký tài khoản.

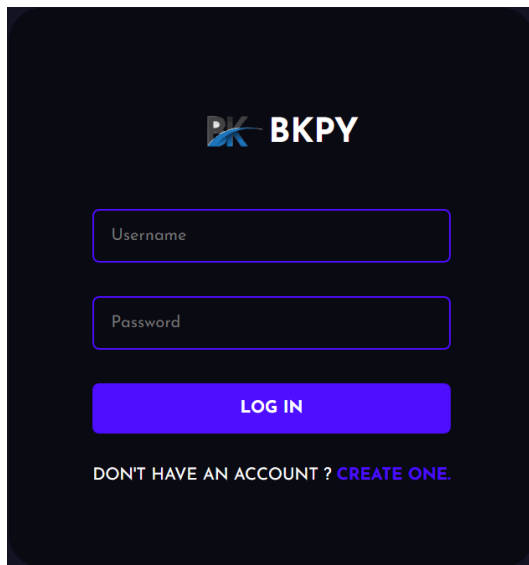
Khi thực hiện đăng ký tài khoản, ứng dụng sẽ xác thực các thông tin mà người dùng cung cấp. Ứng dụng sẽ báo lỗi nếu gặp các trường hợp sau: username hoặc email đã được sử dụng, password ngắn hơn 8 ký tự, password và confirm password không khớp. Nếu người dùng đăng ký thành công, ứng dụng sẽ hiển thị trang đăng nhập để người dùng thực hiện đăng nhập và sử dụng tính năng của ứng dụng.

Khi người dùng đăng nhập thành công, ứng dụng sẽ hiển thị giao diện chat để người dùng có thể bắt đầu sử dụng

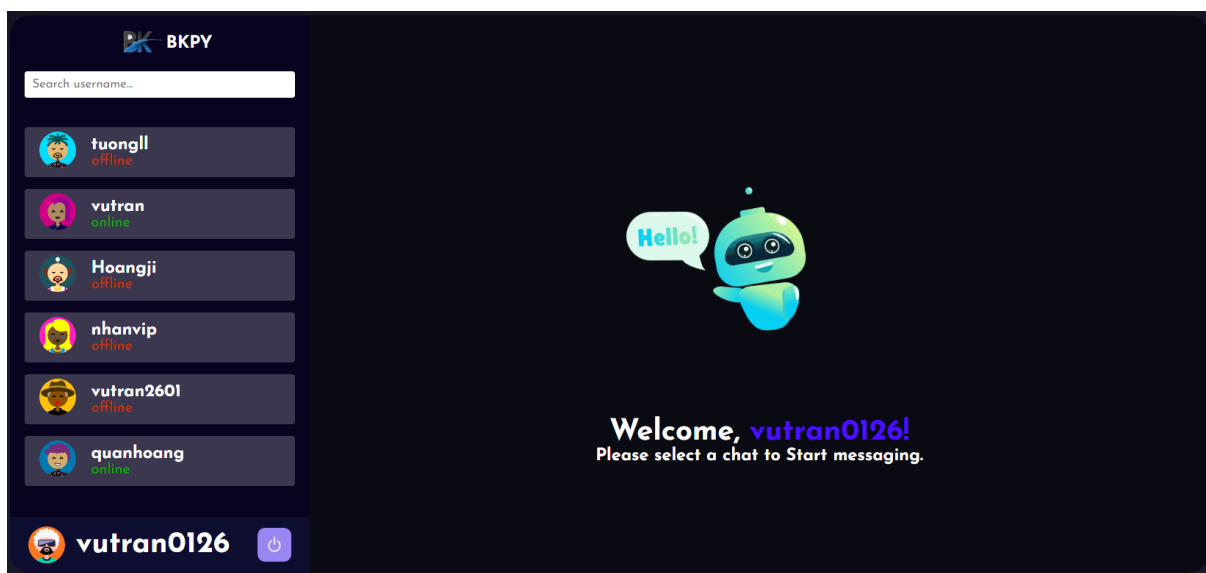


The image shows a registration form for BKPYPY. At the top, there is a logo with the letters 'BK' and the word 'PY' next to it. Below the logo, there are four input fields: 'Username', 'Email', 'Password', and 'Confirm Password'. Each field has a placeholder text. Below the input fields, there is a blue button labeled 'CREATE USER'. At the bottom, there is a link that says 'ALREADY HAVE AN ACCOUNT ? LOGIN.'.

Hình 1: Giao diện đăng ký tài khoản



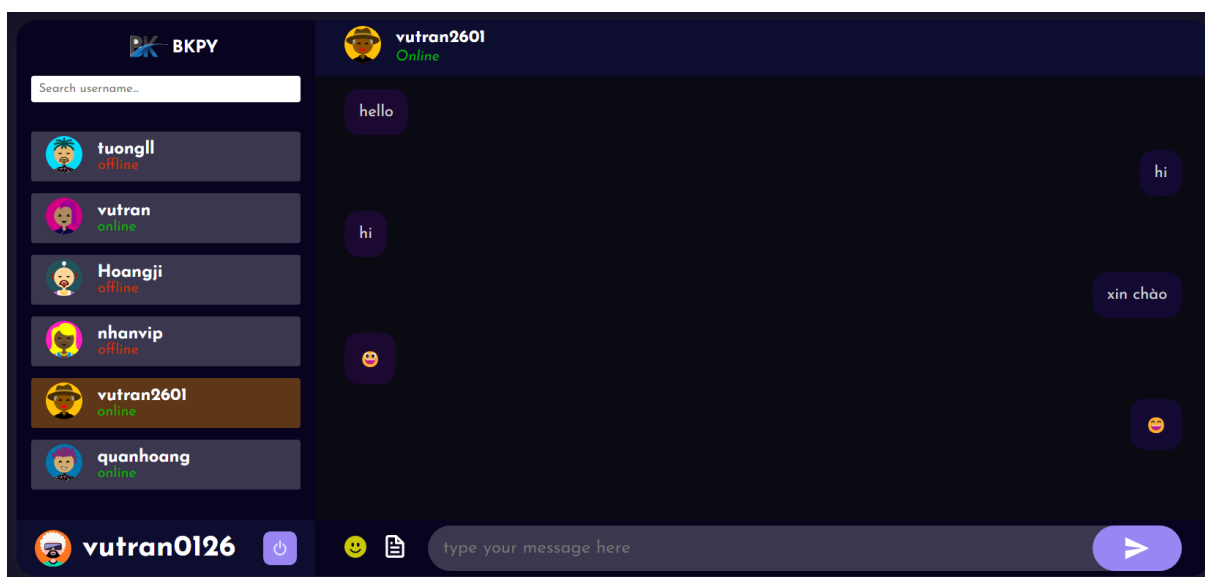
Hình 2: Giao diện đăng ký tài khoản



Hình 3: Giao diện đăng ký tài khoản

### \* Tính năng trò chuyện P2P:

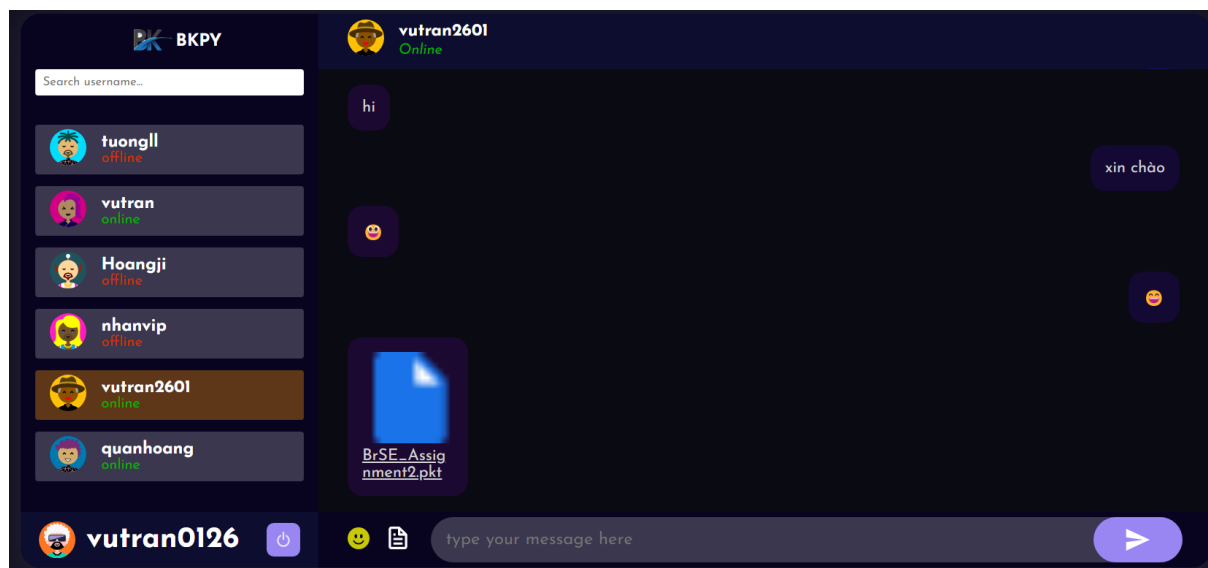
Sau khi đăng nhập, ứng dụng sẽ hiển thị danh sách những người dùng sử dụng kèm theo trạng thái online của người dùng đó. Người dùng có thể tìm kiếm tên của một người dùng nhất định khác trên thanh tìm kiếm. Người dùng có thể bắt đầu một cuộc trò chuyện với bất kỳ người dùng khác ở trong danh sách. Ứng dụng cũng hỗ trợ tính năng gửi các biểu tượng cảm xúc (emoji). Ứng dụng cho một trải nghiệm tốt, tuy nhiên cũng có độ trễ nhất định đối với việc gửi nhận tin nhắn, nhưng không ảnh hưởng quá nhiều trong quá trình trò chuyện.



Hình 4: Trò chuyện với người dùng khác

### \* Tính năng gửi nhận file:

Ứng dụng cũng cho phép người dùng thực hiện gửi nhận file thông qua giao diện trò chuyện. Người dùng có thể gửi nhận file với các định dạng khác nhau và kích thước không quá 10MB. Để hỗ trợ tính năng này, máy chủ ứng dụng sẽ cần một không gian lưu trữ đám mây (cloud service), trong trường hợp này, đơn giản chúng ta sẽ sử dụng Google Drive để lưu trữ tất cả các file mà tất cả các người dùng gửi nhận với nhau. Điều này có nghĩa là, khi một người dùng gửi một file cho người dùng khác, thực chất là người dùng đó đã tải file lên cloud (do vậy, việc gửi nhận file sẽ cần một độ trễ nhất định để ứng dụng có thể xử lý file và tải file lên cloud) và trả về một liên kết đến với file đó để cho phép người gửi và người nhận có thể tải về từ giao diện trò chuyện ở bất cứ thời điểm nào.



Hình 5: Gửi nhận file

#### 4 Đánh giá chung

Sau khi hiện thực ứng dụng, triển khai và chạy thử, nhóm tự đánh giá những mặt tốt và mặt hạn chế một cách tổng quát như sau:

##### \* Mặt tốt:

- Đáp ứng được các chức năng đã đề ra: đăng ký, đăng nhập, trò chuyện, gửi nhận file, biểu tượng cảm xúc.
- Thiết kế giao diện đẹp, thân thiện với người dùng.
- Triển khai được trên internet.
- Hỗ trợ được nhiều loại trình duyệt phổ biến hiện nay.

##### \* Mặt hạn chế:

- Thiết kế giao thức dành riêng cho ứng dụng.
- Các thao tác trên ứng dụng nhìn chung còn độ trễ nhất định, ảnh hưởng một phần đến trải nghiệm người dùng.
- Ảnh được gửi nhận không thể hiển thị trực tiếp ở trên khung trò chuyện.



## IV. KẾT LUẬN

Qua việc thực hiện dự án này, nhóm được ôn lại những kiến thức tổng quát về mạng máy tính, về tổ chức mạng theo mô hình client - server và mô hình peer - to - peer, hiểu rõ hơn về giao thức ở tầng ứng dụng, tầng giao vận và tầng mạng. Cùng với đó nhóm xây dựng được kỹ năng trong việc phát triển một ứng dụng web hoàn chỉnh, đáp ứng được nhu cầu trong thực tế. Dự án mà nhóm thực hiện về cơ bản đáp ứng được những tính năng cần thiết, tuy nhiên vẫn gặp phải một số những mặt hạn chế nhất định, những điều này sẽ là kinh nghiệm để nhóm có thể áp dụng trong việc thực hiện những dự án khác ở trong tương lai. Dự án này cũng giúp nhóm có thể học hỏi kinh nghiệm lẫn nhau, cũng như học hỏi thêm kinh nghiệm làm việc nhóm trong một dự án. Cùng với đó, nhóm cũng trang bị được thêm những kỹ năng cần thiết liên quan đến việc lên kế hoạch và thiết kế một ứng dụng hoàn chỉnh theo hướng cấu trúc và sử dụng framework, hệ thống quản lý phiên bản để từ đó giúp rút ngắn quá trình phát triển cũng như giảm thiểu những lỗi mắc phải trong quá trình thiết kế.

## V. TÀI LIỆU THAM KHẢO

- [1] Difference between Client-Server and Peer-to-Peer Network: <https://www.geeksforgeeks.org/difference-between-client-server-and-peer-to-peer-network>
- [2] Protocols in Application Layer: <https://www.geeksforgeeks.org/protocols-application-layer>
- [3] MERN Stack Explained: <https://www.mongodb.com/mern-stack>
- [4] Introduction Socket.IO <https://socket.io/docs/v4/>