TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN,
ĐẠI HỌC QUỐC GIA HÀ NỘI

Quản lý người dùng trang Hẹn hò trực tuyến

Ngày 12 tháng 1 năm 2024

Giảng viên hướng dẫn
Thầy Vũ Tiến Dũng
Thầy Phạm Duy Phương

Môn học
Cơ sở dữ liệu Web & Hệ thống thông tin

Nhóm 14

Họ và tên	Mã sinh viên
Nguyễn Thị Hương Huế	21002148
Lê Trọng Minh	20002072
Phan Sĩ Nguyên	21000207
Lê Thị Phương	21002167

Mục lục

Thuật ngữ viết tắt

Thuật ngữ viết tắt	Giải thích	Mô tả
API	Application Programming Interface	Giao diện lập trình ứng dụng
NoSQL	Not Only SQL	Hệ quản trị cơ sở dữ liệu phi quan hệ
RDBMS	Relational Database Management System	Hệ quản trị cơ sở dữ liệu quan hệ
UI	User Interface	Giao diện người dùng

Danh mục bảng

Danh mục hình ảnh

Giới thiệu

Swife là một ứng dụng hẹn hò trực tuyến dựa trên vị trí, cho phép người dùng kết nối với những người khác dựa trên sở thích và vị trí địa lý. Ứng dụng đang được phát triển như một dự án cho môn học của nhóm chúng tôi.

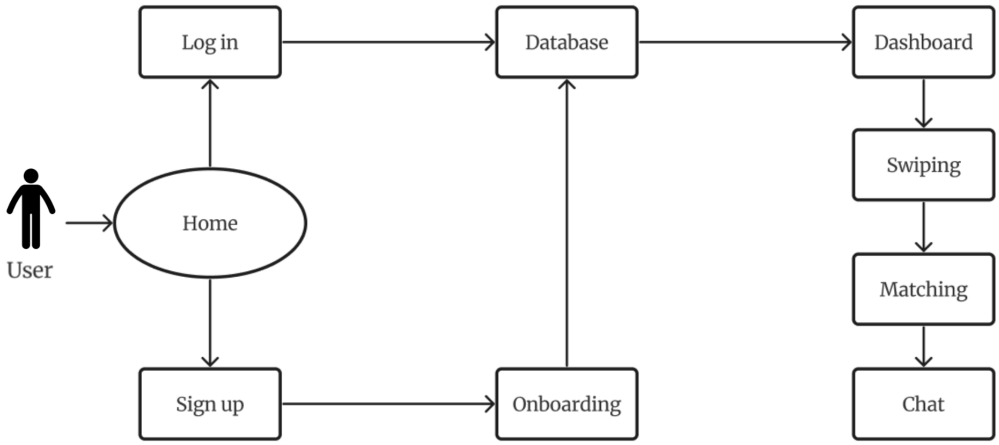
Swife hoạt động bằng cách hiển thị cho người dùng các hồ sơ của những người khác ở gần họ. Người dùng có thể vuốt sang trái để "không thích"hoặc vuốt sang phải để "thích"một hồ sơ. Nếu cả hai người dùng đều thích hồ sơ của nhau, họ sẽ được kết nối và có thể bắt đầu trò chuyện.

Swife cung cấp một số tính năng để giúp người dùng tìm kiếm các kết nối tiềm năng. Người dùng có thể đặt các bộ lọc để giới hạn kết quả tìm kiếm của họ theo giới tính, độ tuổi, khoảng cách và sở thích. Đây là tính năng hiện vẫn đang được phát triển và chưa đưa vào sử dụng.

Dưới đây là một số tính năng chính của Swife:

- **Vuốt để thích hoặc không thích:** Người dùng có thể vuốt sang trái để "không thích"hoặc vuốt sang phải để "thích"một hồ sơ.
- **Kết nối khi cả hai bên đều thích nhau:** Nếu cả hai người dùng đều thích hồ sơ của nhau, họ sẽ được kết nối và có thể bắt đầu trò chuyện.
- **Bộ lọc:** Người dùng có thể đặt các bộ lọc để giới hạn kết quả tìm kiếm của họ theo giới tính, độ tuổi, khoảng cách và sở thích.

1 Use Case



Hình 1: Sơ đồ UseCase

Interaction giữa web và người dùng

Người dùng khi mới sử dụng sẽ được chuyển đến đến trang chủ (Home), tại đây người dùng có thể chọn đăng nhập (Log In) hoặc tạo tài khoản mới (Sign Up) nếu chưa có tài khoản.

Trong trường hợp Log In, người dùng sẽ được đưa tới trang để quẹt (Dashboard).

Nếu người dùng chọn Sign Up, họ sẽ được chuyển sang trang đăng ký thông tin (Onboarding) để điền thông tin người dùng vào. Email và password của người dùng đăng ký mới sẽ được lưu trữ trong cookie để tiếp tục await gắn tiếp thông tin vào đó. Khi hoàn thành trang Onboarding, thông tin (bao gồm cả email và password từ trang Home và các thông tin từ trang Onboarding) sẽ được gửi lên database dưới dạng 1 file JSON.

Trang Dashboard dùng để quẹt sẽ có 2 phần chính:

- Phần quẹt: Mỗi user sẽ hiển thị dưới dạng 1 Tindercard (thư viện có sẵn trên GitHub) và hiển thị hình ảnh cũng như thông tin của người dùng.
- Phần nhắn tin (chat): Khi hai user đã match với nhau (cùng quẹt nhau), sẽ hiển thị ô chat để người dùng có thể trò chuyện với nhau. Các message này sẽ được lưu trữ trong 1 collection trong database.

2 Database

2.1 Cơ sở lý thuyết

MongoDB

1. Giới thiệu MongoDB
   MongoDB là hệ cơ sở dữ liệu mã nguồn mở, là hệ cơ sở dữ liệu phi quan hệ hay còn gọi là NoSQL (Non-Relationship SQL). NoSQL được phát triển trên JavaScript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value. NoSQL ra đời là sự bổ sung cho những khuyết điểm thiếu sót cũng như hạn chế của mô hình quan hệ RDBMS (Relational Database Management System - Hệ quản trị cơ sở dữ liệu quan hệ) về tốc độ, tính năng và khả năng mở rộng. Với NoSQL có thể mở rộng mà không lo khóa chính, khóa ngoại, kiểm tra ràng buộc. NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng. MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như cơ sở dữ liệu quan hệ nên truy vấn sẽ rất nhanh.

   Với cơ sở dữ liệu quan hệ (như MySQL, SQL Server,...) sử dụng các bảng để lưu trữ dữ liệu thì với MongoDB dùng khái niệm "collection"thay vì bảng. So với RDBMS thì trong MongoDB collection ứng với table, còn document sẽ ứng với row, MongoDB sẽ dùng các document thay cho row trong RDBMS.

   Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định. Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB.

2. Ưu điểm của MongoDB
   - Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng document JSON nên mỗi collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong lưu trữ dữ liệu, do đó việc chèn thêm dữ liệu không bị hạn chế.
   - Dữ liệu trong MongoDB không bị ràng buộc như cơ sở dữ liệu quan hệ và không có khái niệm kết nối. Do đó, khi thêm mới, xóa hoặc cập

nhật dữ liệu, không cần thời gian để kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.

- MongoDB rất dễ mở rộng. Trong MongoDB, có khái niệm cụm dữ liệu (cluster) là một nhóm các node chứa dữ liệu giao tiếp với nhau. Khi muốn mở rộng hệ thống, ta chỉ cần thêm một node vào cụm dữ liệu.
- Trường dữ liệu "_id" luôn được tự động đánh chỉ mục để tăng tốc độ truy vấn thông tin và đạt hiệu suất cao nhất.
- Khi có một truy vấn dữ liệu, các bản ghi sẽ được ghi tạm thời vào bộ nhớ RAM để phục vụ các truy vấn tiếp theo. Điều này giúp cải thiện hiệu suất xử lý dữ liệu mà không cần phải đọc lại từ ổ cứng.
- MongoDB có hiệu năng cao. Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn rất nhiều so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Khi thử nghiệm với một lượng dữ liệu đủ lớn, tốc độ chèn dữ liệu của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.

3. Nhược điểm của MongoDB
   Hệ quản trị cơ sở dữ liệu MongoDB có nhiều ưu điểm, tuy nhiên cũng có một số nhược điểm.

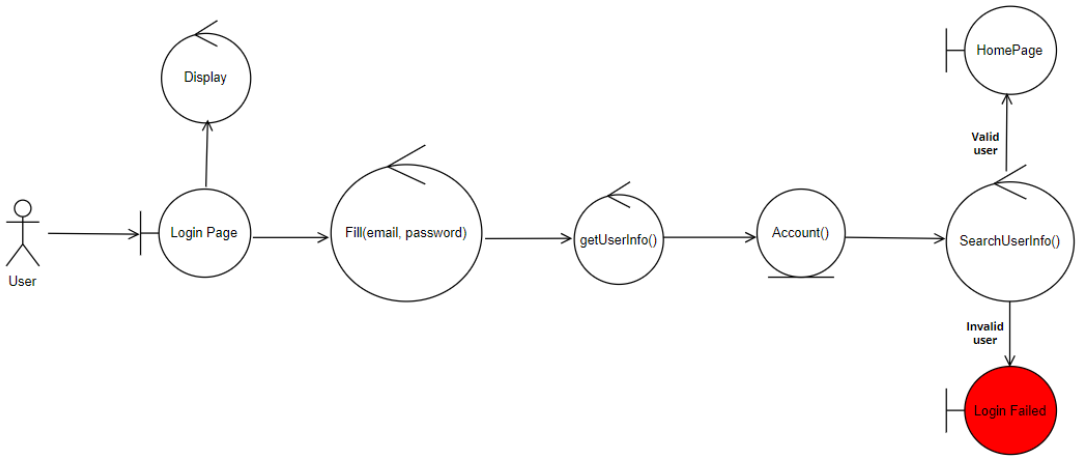
   - Thiếu tính chất ràng buộc: MongoDB không có các tính chất ràng buộc như trong cơ sở dữ liệu quan hệ. Điều này đồng nghĩa với việc chuyên gia phải tự xử lý các mối quan hệ giữa các dữ liệu trong MongoDB. Việc quản lý mối quan hệ và đảm bảo tính nhất quán của dữ liệu trở nên phức tạp hơn.
   - Dư thừa dữ liệu: Vì MongoDB không hỗ trợ liên kết, dữ liệu có thể bị dư thừa. Thay vì sử dụng liên kết, MongoDB khuyến khích nhúng dữ liệu trong các document. Điều này có thể dẫn đến việc lưu trữ các bản sao dữ liệu trong nhiều document, gây tăng kích thước bộ nhớ lưu trữ.
   - Ghi trễ (write delay): Khi thực hiện các thao tác insert, update, hoặc remove, MongoDB không cập nhật ngay lập tức vào ổ cứng. Thay vào đó, MongoDB sẽ ghi tạm thời các thay đổi vào bộ nhớ RAM và sau một khoảng thời gian (thường là 60 giây), MongoDB mới thực hiện ghi toàn bộ dữ liệu từ RAM xuống thiết bị lưu trữ. Điều này gây nguy cơ mất dữ liệu khi xảy ra các tình huống như mất điện trong khoảng thời gian ghi trễ.

2.2 Thiết kế hệ thống

2.2.1 Robustness Diagram

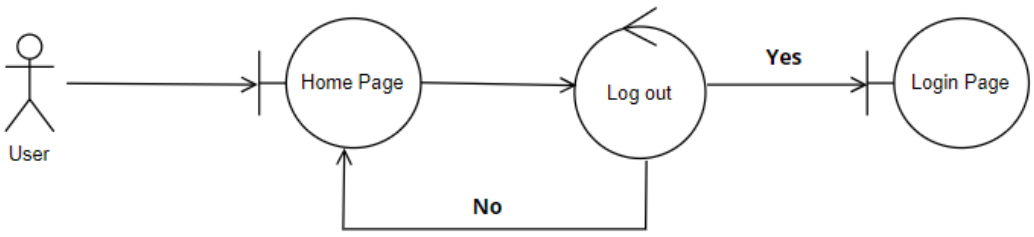
1. Các chức năng chung của người dùng hệ thống

a) Đăng nhập



Hình 2: Robustness Diagram Đăng nhập

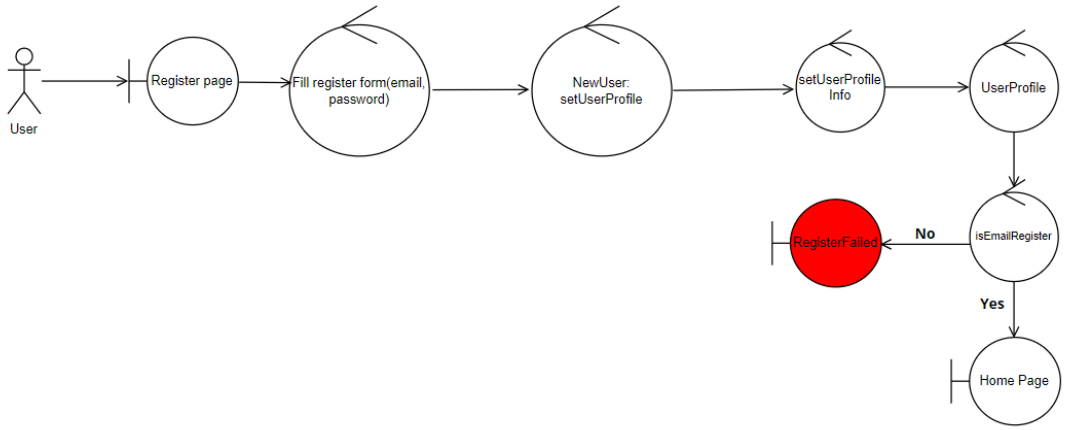
b) Đăng xuất



Hình 3: Robustness Diagram Đăng xuất

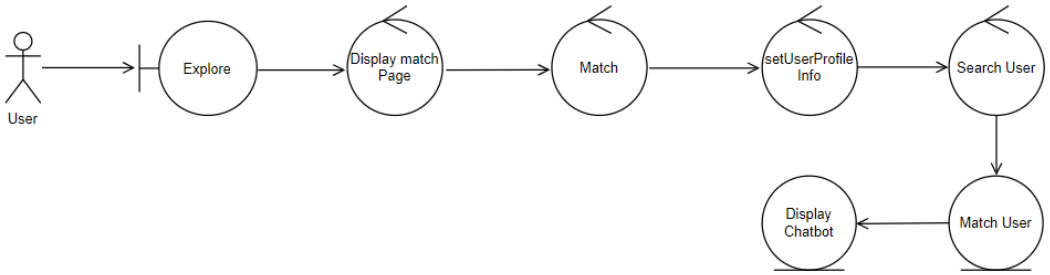
2. Các chức năng của người dùng

a) Đăng kí



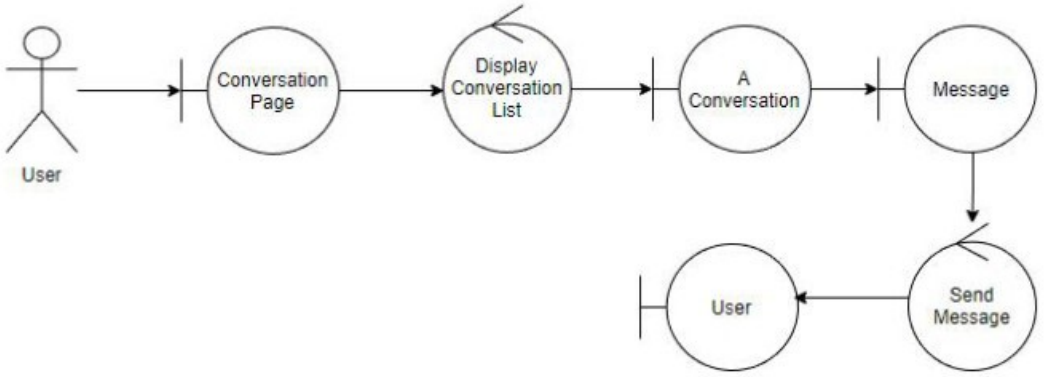
Hình 4: Robustness Diagram Đăng kí

b) Match



Hình 5: Robustness Diagram Match

c) Trò chuyện

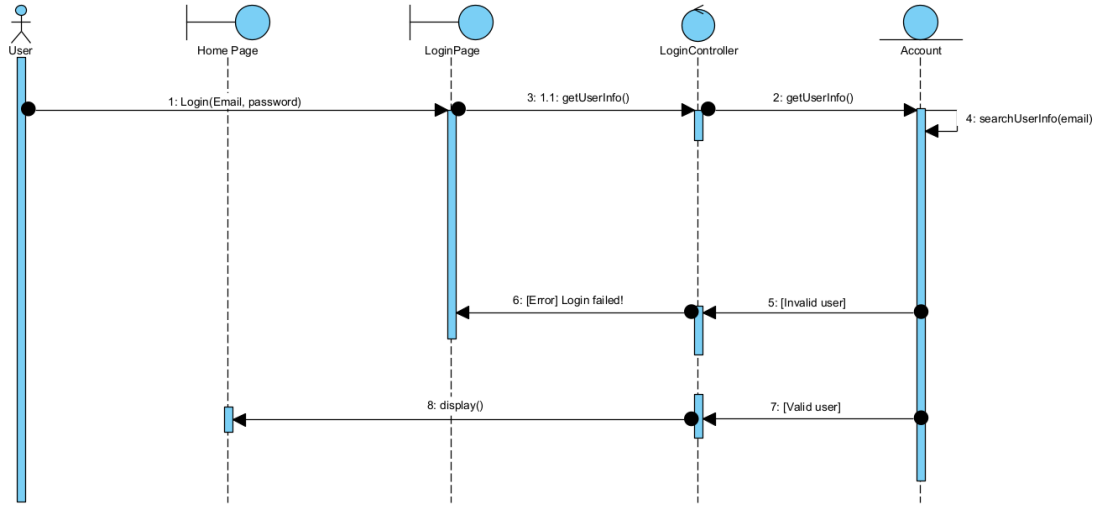


Hình 6: Robustness Diagram Nhắn tin

2.2.2 Sequence Diagram

1. Các chức năng chung của người dùng hệ thống

a) Đăng nhập

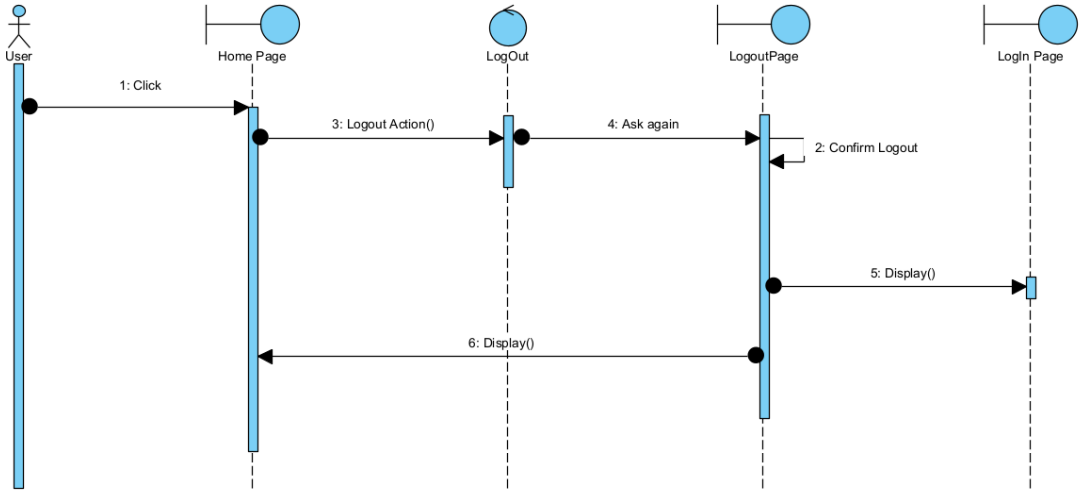


Hình 7: Biểu đồ tuần tự chức năng đăng nhập

Mô tả tóm tắt:

- Người dùng nhập email và mật khẩu để đăng nhập vào hệ thống.
- Hệ thống sẽ kiểm tra và đối chiếu với thông tin người dùng đã đăng kí trong cơ sở dữ liệu.
- Nếu thông tin đúng hệ thống sẽ thông báo đăng nhập thành công.
- Sau khi đăng nhập thành công người dùng có thể thao tác trên giao diện trang chủ.

b) Đăng xuất



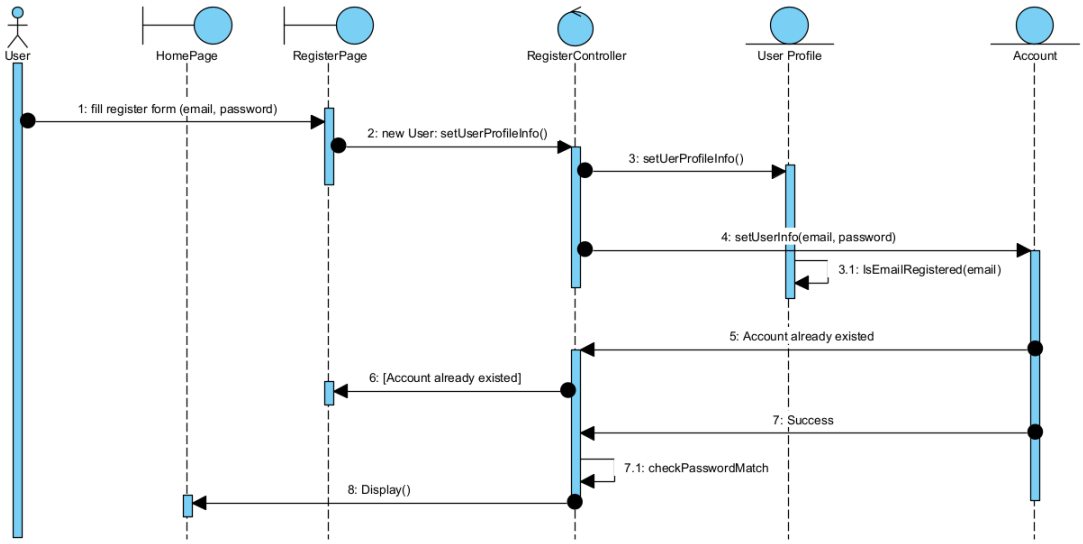
Hình 8: Biểu đồ tuần tự chức năng đăng xuất

Mô tả tóm tắt:

- Người dùng kích chọn đăng xuất, hệ thống sẽ hỏi và xác nhận đăng xuất.
- Sau khi đăng xuất thành công sẽ trở về giao diện trang đăng nhập.

2. Các chức năng của người dùng

a) Đăng kí

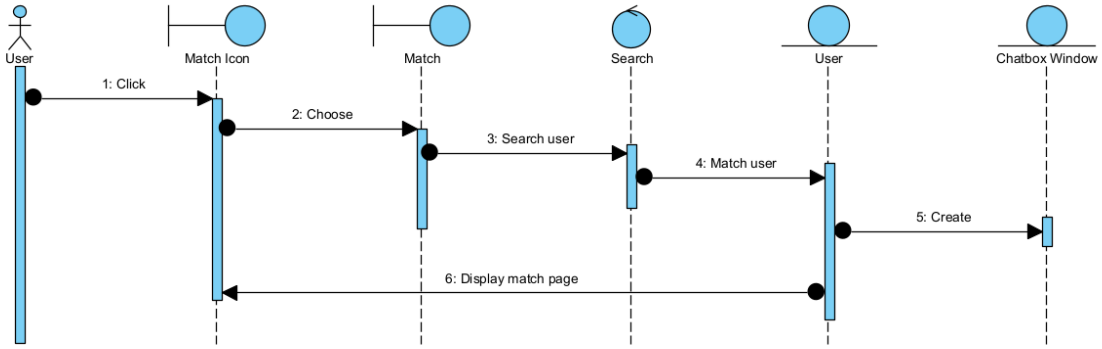


Hình 9: Biểu đồ tuần tự chức năng đăng kí

Mô tả tóm tắt:

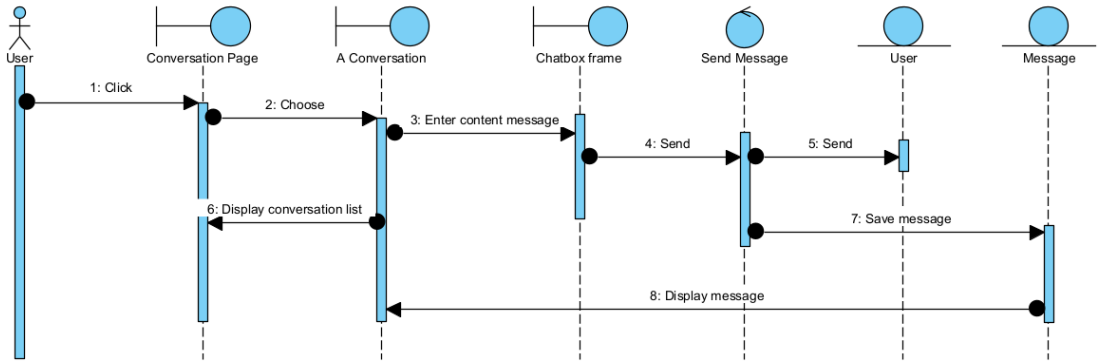
- Người dùng nhập email và mật khẩu để đăng kí vào hệ thống.
- Hệ thống sẽ kiểm tra email đã tồn tại chưa, nếu chưa sẽ xác nhận lại mật khẩu. Nếu mật khẩu trùng khớp sẽ tiến hành vào trang đăng kí tiếp theo (Họ, tên, ngày tháng năm sinh, giới tính, giới tính quan tâm, sở thích, mô tả bản thân, link ảnh).
- Sau khi đăng kí thành công người dùng có thể thao tác trên giao diện trang chủ.

b) Match



Hình 10: Biểu đồ tuần tự chức năng match

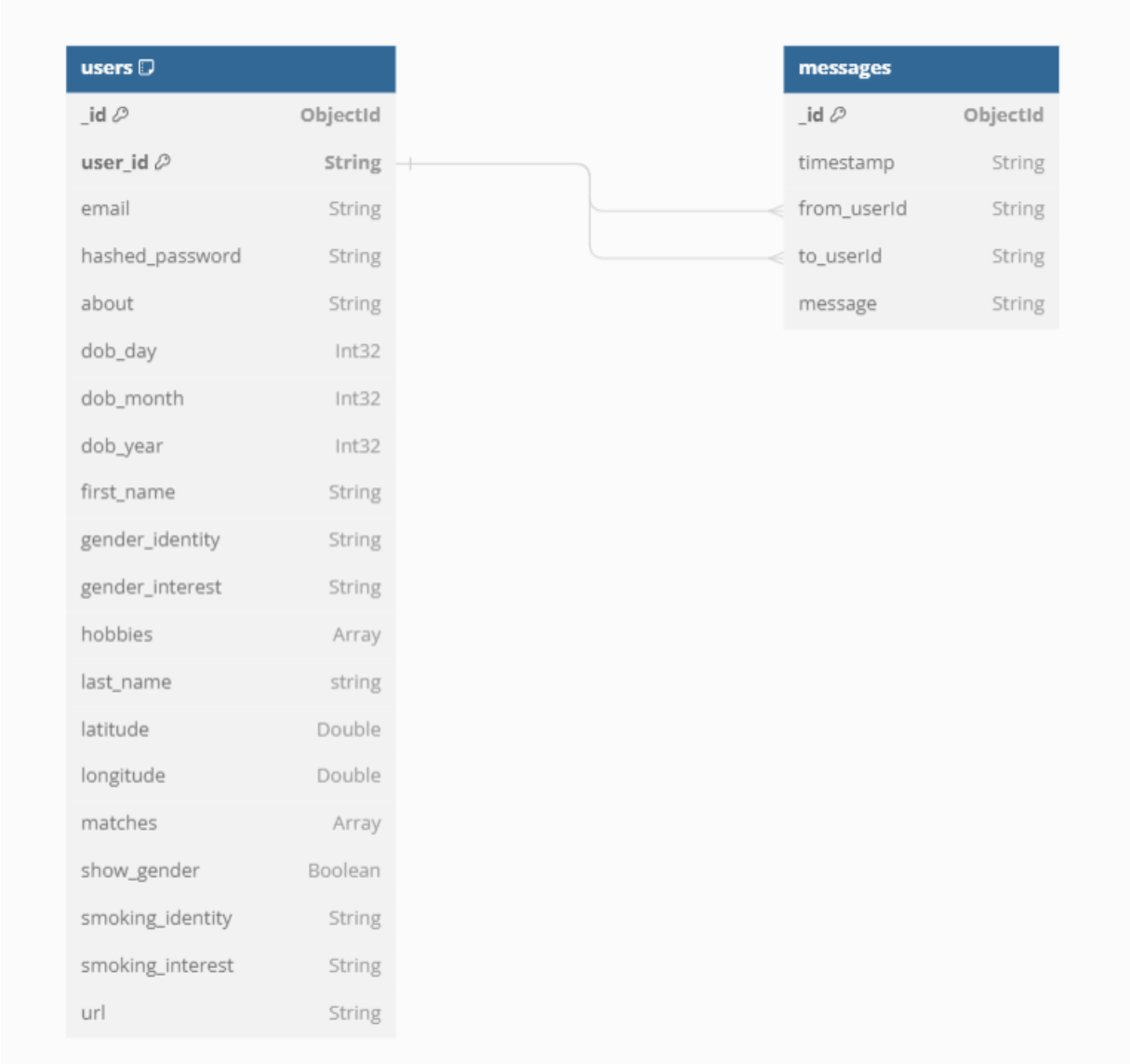
c) Trò chuyện



Hình 11: Biểu đồ tuần tự chức năng nhắn tin

2.2.3 Thiết kế cơ sở dữ liệu

1. Mô hình quan hệ



Hình 12: Mô hình quan hệ

2. Danh sách các bảng trong mô hình quan hệ

Bảng 1: Collection users

Collection *users* lưu trữ thông tin về tất cả người dùng.

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	**_id**	ObjectId	Đại diện cho tính duy nhất của một document trong một collection. Khóa chính của bảng.
2	**user_id**	String	Mã duy nhất đại diện cho mỗi người dùng. Đây là khóa chính.
3	Email	String	Email của người dùng.
4	hashed_password	String	Mật khẩu đã được mã hóa.
5	About	String	Mô tả bản thân người dùng.
6	dob_day	Int	Ngày sinh của người dùng.
7	dob_month	Int	Tháng sinh của người dùng.
8	dob_year	Int	Năm sinh của người dùng.
9	first_name	String	Tên người dùng.
10	gender_identity	String	Giới tính người dùng.
11	gender_interest	String	Giới tính mà người dùng quan tâm.
12	hobbies	Array	Mảng chứa các sở thích của người dùng.
13	last_name	String	Họ và tên đệm người dùng.
14	latitude	Double	Vĩ độ vị trí của người dùng.
15	longitude	Double	Kinh độ vị trí của người dùng.
16	matches	Array	Mảng chứa các user_id của người đã ghép đôi.
17	show_gender	Boolean	Trạng thái hiển thị giới tính của người dùng.
18	smoking_identity	String	Thói quen hút thuốc của người dùng.
19	smoking_interest	String	Thói quen hút thuốc mà người dùng quan tâm.
20	url	String	Link file ảnh người dùng.

Bảng 1: Collection users

Bảng 2: Collection messages

Collection *users* lưu trữ thông tin về tin nhắn của người dùng.

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	**_id**	ObjectId	Đại diện cho tính duy nhất của một document trong một collection. Khóa chính của bảng.
2	from_userId	String	Mã của người gửi tin nhắn. Khóa ngoại, liên kết là **user_id** của users.
3	to_userId	String	Mã của người nhận tin nhắn. Khóa ngoại, liên kết là **user_id** của users.
4	message	String	Nội dung tin nhắn.

Bảng 2: Collection messages

3. Tổng quát

Hiện tại, chúng tôi sử dụng cơ sở dữ liệu MongoDB để lưu trữ thông tin khoảng 200 người dùng. Mỗi người dùng được biểu diễn bởi một bản ghi trong cơ sở dữ liệu, chứa thông tin cá nhân, sở thích và các thuộc tính khác. Mặc dù số lượng người dùng còn hạn chế, nhưng thông tin về từng

người dùng là đa dạng. Mỗi người dùng có những sở thích như chơi game, đọc sách, lập trình, du lịch, nấu ăn,... Điều này tạo ra sự đa dạng trong cơ sở dữ liệu của chúng tôi.

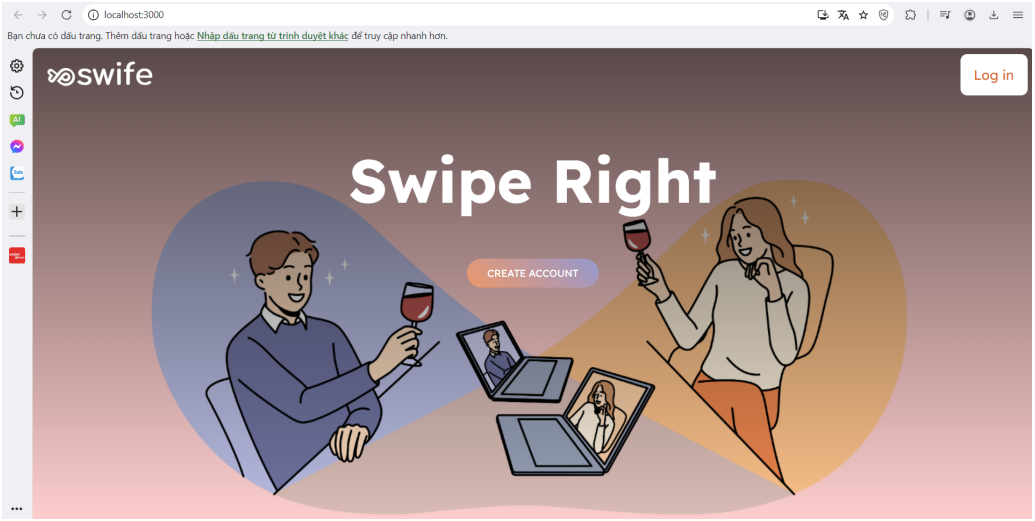
3 Front-end

Tổng quan

- Homepage UI & AuthModal UI
- OnBoarding UI
- Dashboard UI
- ChatContainer UI

3.1 Homepage UI & AuthModal UI

3.1.1 Homepage Interface



Hình 13: Giao diện trang Homepage

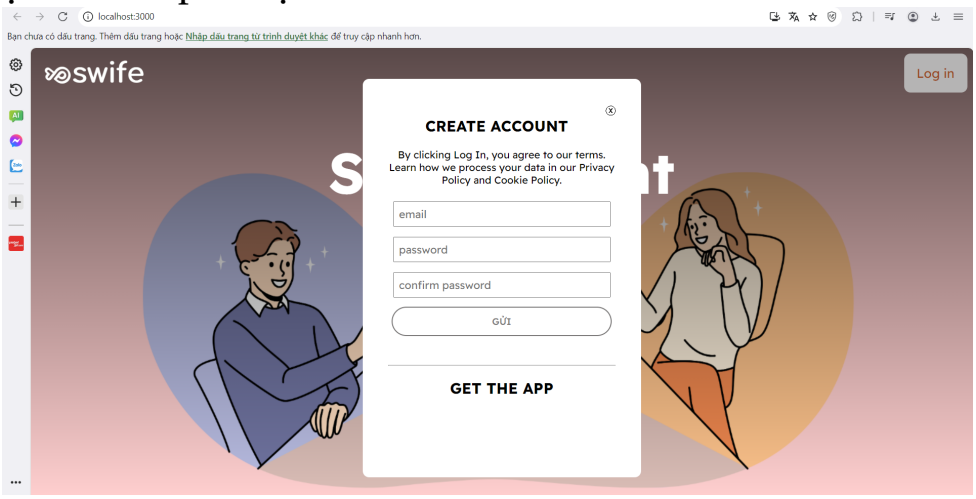
Đây sẽ là giao diện đầu tiên khi người dùng chạy/truy cập dự án. Người dùng có thể lựa chọn đăng ký tài khoản mới bằng cách bấm vào **CREATE ACCOUNT** hoặc đăng nhập vào tài khoản sẵn có bằng cách bấm **LOG IN** vào góc trên cùng bên phải giao diện trang chủ.

3.1.2 AuthModal Interface

Hàm *handleSubmit* kiểm tra mật khẩu đã trùng với mật khẩu đã đăng ký ở trên (xác nhận mật khẩu) và phải đạt yêu cầu bảo mật.
Hàm *response* nhận thao tác của người dùng xem họ đăng ký tài khoản mới

hay đăng nhập vào tài khoản đã có.
Biến *success* sẽ được trả lại khi chuyển tiếp thành công sang các trang khác (onboarding và dashboard)

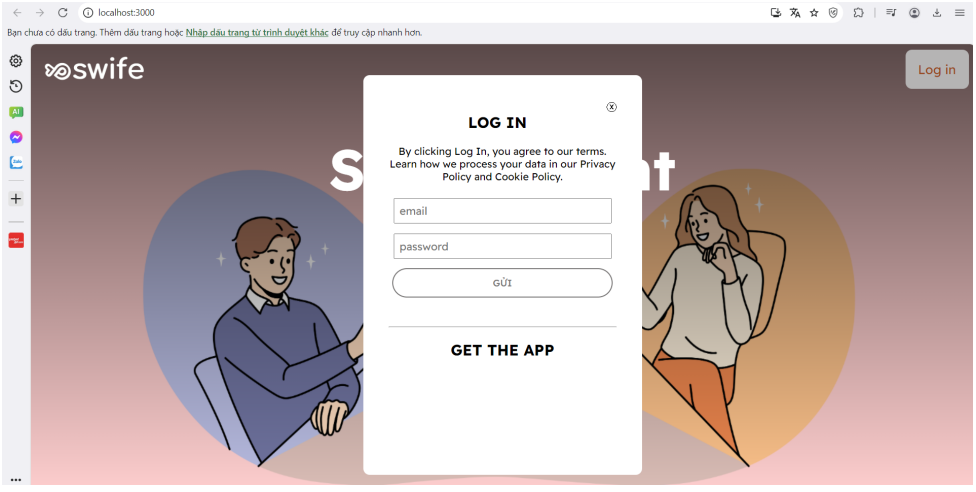
- Giao diện khi nhấp chuột vào **CREATE ACCOUNT**:



Hình 14: Giao diện Create Account

Khi nhấp chuột vào **CREATE ACCOUNT**, giao diện AuthModal để tạo tài khoản sẽ được hiển thị, giao diện sẽ yêu cầu người dùng điền email, mật khẩu, và xác nhận lại mật khẩu để hoàn thiện quá trình tạo tài khoản. Sau khi điền những thông tin trên, người dùng sẽ được chuyển tiếp sang giao diện OnBoarding để hoàn thiện thông tin về tài khoản Swife.

- Giao diện khi nhấp chuột vào **LOG IN**:



Hình 15: Giao diện Log In

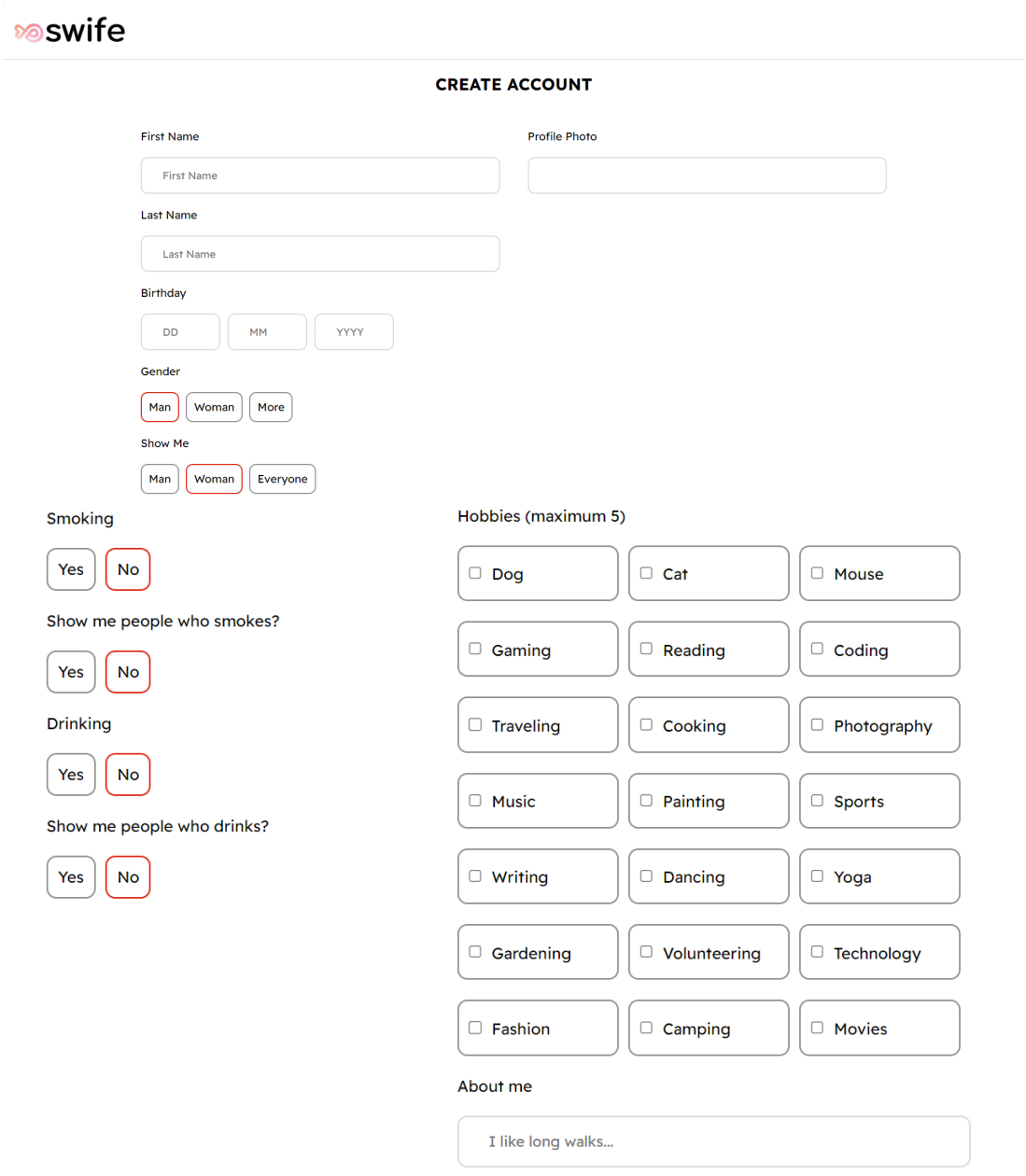
Đây là giao diện AuthModal để đăng nhập vào tài khoản đã có sẵn trong database. Người dùng sẽ được yêu cầu điền email và mật khẩu để đăng nhập vào tài khoản. Sau khi đăng nhập thành công, người dùng sẽ được

chuyển tiếp sang giao diện DashBoard của web app.

3.2 OnBoarding UI

Hàm *handle change* dùng để nhập thông tin người dùng, hàm *isInRange* để kiểm tra giá trị trong khoảng cho trước.
Hàm *updateDateField* sẽ kiểm tra input được nhập vào gồm ngày sinh, tháng sinh, năm sinh.
Hàm *updatedHobbies* lọc các yêu cầu từ input.

OnBoarding Interface



Hình 16: Giao diện trang Onboarding

Đây là giao diện nơi người dùng có thể hoàn thiện thông tin về tài khoản của người dùng. Người dùng sẽ cần điền các thông tin sau:

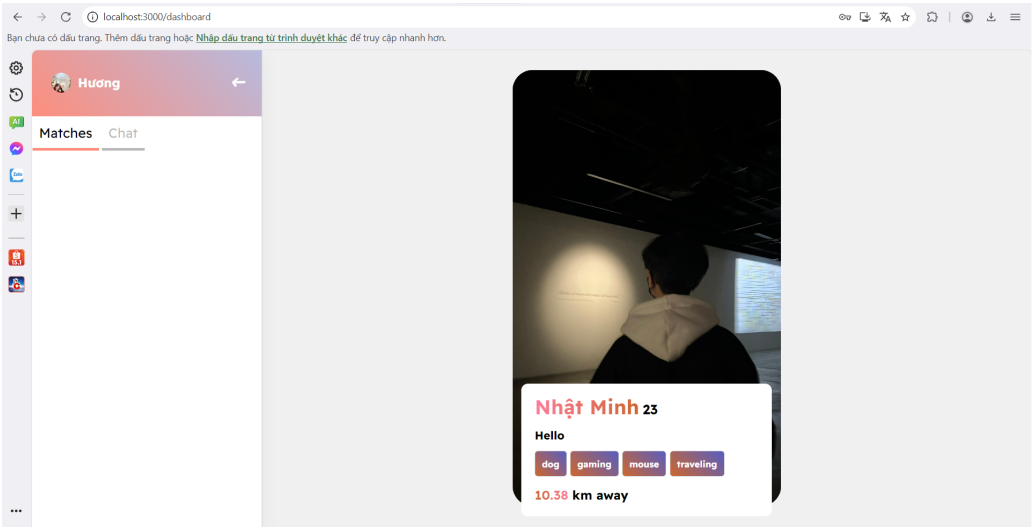
- First name - Tên
- Last name - Họ
- Birthday - Ngày/tháng/năm sinh
- Gender - Giới tính
- Show me - Lựa chọn hiển thị cho người dùng các hồ sơ có giới tính nào
- About me - Người dùng viết về bản thân
- Smoking/Drinking - Thói quen của người dùng
- Show me people who smokes/drinks - Lựa chọn hiển thị của người dùng
- Hobbies - Sở thích của người dùng (Tối đa 5)
- Profile Photo - Ảnh đại diện của người dùng, người dùng cần điền đường dẫn ảnh mong muốn làm ảnh đại diện cho hồ sơ

Sau khi người dùng xác nhận thông tin mà họ đã hoàn thiện, những thông tin như email, password và những thông tin vừa nêu trên sẽ được lưu vào database người dùng của web app.

3.3 Dashboard UI

Deg2rad là hàm trả về góc radian từ góc degree (dùng cho tính toán vị trí người dùng).
MatchedUserIds dùng để trả về các users matched với nhau.
filteredGenderedUsers trả về các users đã được phân theo giới tính và điều kiện.

3.3.1 Dashboard Interface:



Hình 17: Giao diện trang Dashboard

Swife sử dụng một giao diện đơn giản và trực quan để tạo trải nghiệm tương tác dễ dàng cho người dùng.
Trang chủ của Swife hiển thị các hồ sơ người dùng một cách tuần tự và ngẫu nhiên trong danh sách.
Mỗi hồ sơ được hiển thị bằng một thẻ (Tindercard) chứa hình ảnh và thông tin ngắn về người dùng.

3.3.2 Dashboard operations:

Người dùng có thể quẹt sang trái hoặc phải trên thẻ hồ sơ để thể hiện sự quan tâm của mình.
Thao tác quẹt sang trái được sử dụng để từ chối (dislike) hồ sơ người dùng hiện tại. Thao tác quẹt sang phải được sử dụng để thể hiện sự quan tâm (like) đến hồ sơ người dùng hiện tại.

3.3.3 Visual Effects:

Khi người dùng quẹt sang trái hoặc phải, thẻ hồ sơ sẽ di chuyển theo hướng quẹt và biến mất khỏi màn hình.
Hiệu ứng hình ảnh thường được sử dụng để tạo ra cảm giác mượt mà và thú vị cho người dùng.

3.3.4 Swipe result:

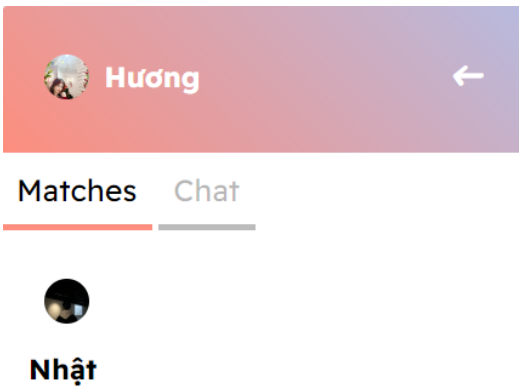
Sau khi người dùng quẹt phải, Swife sẽ hiển thị một thông báo ngắn "You Swiped Right"để xác nhận hành động của người dùng. Và tương tự, khi người

dùng quẹt trái, Swife sẽ hiển thị một thông báo ngắn "You Swiped Left"để xác nhận hành động của người dùng.
Nếu hai người dùng thể hiện sự quan tâm (like) lẫn nhau, sẽ xảy ra một kết hợp (match) và cả hai có thể bắt đầu trò chuyện (chat) với nhau.

3.4 ChatContainer UI

Phần *Chat* hiển thị các tin nhắn người dùng bao gồm ảnh và tên người dùng đó theo trình tự thời gian (tin nhắn mới sẽ ở dưới).
Phần *ChatContainer* sẽ bao gồm tất cả phần chat header, phần hiển thị các người match và cả phần ChatDisplay.
Phần *ChatDisplay* sẽ hiển thị các tin nhắn theo trình tự thời gian, phân theo *clickedUser* là người dùng được match và User là người dùng.
Phần *ChatHeader* gồm ảnh nhỏ của người dùng và tên người dùng và một nút Log Out (có hình mũi tên ←) để người dùng đăng xuất và trở về trang Home.
Phần *ChatInput* dùng để nhập tin nhắn từ người dùng và các thêm các message vào Database MongoDB collection message.
Phần *MatchesDisplay* sẽ lấy ra các matched và hiển thị ảnh cũng như tên những người dùng đã matched lên giao diện.

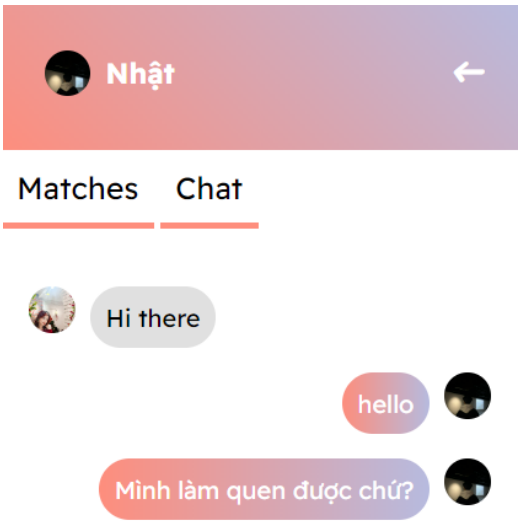
ChatContainer Interface:



Hình 18: Giao diện ChatContainer

Sau khi đã match được 1 hồ sơ ở phần Dashboard, người dùng mà bạn match sẽ được hiện ở phía giao diện bên trái màn hình.

Bấm vào hồ sơ đó, người dùng sẽ được chuyển tiếp sang giao diện chat để có thể nhắn tin cho đối tượng đã match. Tin nhắn sẽ được lưu trữ trong database của web app.



Hình 19: Giao diện Chat

4 API

4.1 API

Tên API	Phương thức
app.post('/login')	POST
app.post('/signup')	POST
app.get('/user')	GET
app.put('/addmatch')	PUT
app.get('/users')	GET
app.get('/gendered-users')	GET
app.put('/user')	PUT
app.get('/messages')	GET
app.post('/message')	POST

Bảng 3: API

Tham số

Tên	Kiểu dữ liệu	Bắt buộc	Mô tả
email	string	Có	Email của người dùng
password	string	Có	Mật khẩu của người dùng
userId	string	Có	ID của người dùng
name	string	Có	Tên của người dùng
gender_identity	string	Có	Giới tính của người dùng
location	object	Có	Vị trí của người dùng
correspondingUserId	string	Có	ID của người dùng đối diện
content	string	Có	Nội dung của tin nhắn

Bảng 4: Bảng tham số

Đầu ra

Tên	Kiểu dữ liệu	Mô tả
token	string	Mã thông báo JWT
user	object	Thông tin của người dùng
matches	array	Danh sách người dùng đã được ghép đôi
users	array	Danh sách người dùng
gendered_users	array	Danh sách người dùng có giới tính tương ứng
user	object	Thông tin của người dùng đã được cập nhật
messages	array	Danh sách tin nhắn
message	object	Thông tin của tin nhắn đã được thêm

Bảng 5: Bảng đầu ra

4.2 Các endpoint/API

4.2.1 app.post('/login')

Endpoint POST "/login" để xử lí yêu cầu đăng nhập của người dùng. Nó tìm kiếm người dùng trong collection users dựa trên email được cung cấp, nếu tồn tại thì kiểm tra mật khẩu bằng cách so sánh mật khẩu đã nhập và mật khẩu đã được mã hóa. Nếu thông tin đăng nhập hợp lệ, mã sẽ tạo một mã thông báo JWT để xác thực người dùng đã đăng nhập thành công. Mã thông báo này sẽ hết hạn sau 1 ngày.

4.2.2 app.post('/signup')

Endpoint POST "/signup" để xử lý yêu cầu đăng ký người dùng mới khi được cung cấp email và password. Khi email chưa được đăng kí sẽ tạo ra 1 'user_id' ngẫu nhiên và mật khẩu được mã hóa bằng 'bcrypt'. Sau đó insert vào collection 'users' trong cơ sở dữ liệu. Một mã thông báo JWT được tạo để xác thực người dùng đã đăng ký thành công.

4.2.3 app.get('/user')

Endpoint GET "/user" dùng để lấy thông tin của một người dùng dựa trên 'user_id' từ collection users. Sau khi thiết lập kết nối và truy vấn cơ sở dữ liệu, nó tìm kiếm người dùng theo 'user_id' và trả về thông tin người dùng đó trong phản hồi.

4.2.4 app.put('/addmatch')

Một endpoint PUT "/addmatch" để cập nhật thông tin của người dùng sau khi người dùng đó được ghép đôi với một người dùng khác. API này sẽ thêm một bản ghi vào mảng matches của người dùng. Bản ghi này sẽ bao gồm thông tin về người dùng được ghép đôi.

4.2.5 app.get('/users')

Một endpoint GET "/users" lấy danh sách 'user_id' người dùng từ tham số userIds trong yêu cầu và trả về thông tin của tất cả người dùng có 'user_id' trong danh sách đó.

4.2.6 app.get('/gendered-users')

Một endpoint GET "/gendered-users" để trả về danh sách tất cả người dùng có giới tính tương ứng với tham số gender. Đầu tiên, endpoint lấy tham số gender từ yêu cầu. Sau đó, nó sử dụng tham số này để xây dựng một truy vấn tìm kiếm. Truy vấn này sẽ tìm tất cả các tài liệu trong collection users có trường gender_identity trùng khớp với giá trị gender. Kết quả truy vấn được trả về dưới dạng một mảng. Mảng này được chuyển đổi thành một đối tượng JSON và trả về trong phản hồi của yêu cầu.

4.2.7 app.put('/user')

Endpoint PUT "/user" được thực hiện để cập nhật thông tin người dùng. Truy vấn để xác định người dùng cần cập nhật dựa trên 'user_id' lấy từ formData. Hàm getLocationByIP() được gọi để lấy thông tin vị trí dựa trên địa chỉ IP của người dùng. Kết quả trả về chứa thông tin vị trí gồm kinh độ và vĩ độ. Một đối tượng updateDocument được tạo chứa thông tin cần cập nhật của người dùng. Sau đó, phương thức updateOne() được sử dụng để cập nhật thông tin người dùng tương ứng với truy vấn và đối tượng updateDocument. Kết quả trả về chứa thông tin về người dùng đã được cập nhật. Cuối cùng, thông tin người dùng đã được cập nhật được trả về dưới dạng một đối tượng JSON qua hàm res.json().

4.2.8 app.get('/messages')

Endpoint GET "/message" nhận yêu cầu GET để lấy danh sách tin nhắn. Đầu tiên, endpoint lấy hai tham số 'userId' và 'correspondingUserId' từ yêu cầu.

Sau đó, nó sử dụng hai tham số này để xây dựng một truy vấn tìm kiếm. Truy vấn này tìm kiếm các tin nhắn trong collection messages có ‘from_userId‘ là ‘userId‘ và ‘to_userId‘ là ‘correspondingUserId‘. Sau đó gửi danh sách tin nhắn đã tìm thấy trong phản hồi của yêu cầu.

4.2.9 app.post(’/message’)

Endpoint POST “/message” để thêm một tin nhắn mới vào database dựa trên nội dung được gửi trong body của yêu cầu. Đầu tiên, endpoint lấy nội dung tin nhắn từ body của yêu cầu. Sau đó, nó sử dụng nội dung này để tạo một đối tượng tin nhắn. Đối tượng tin nhắn này được thêm vào collection messages trong database bằng phương thức insertOne(). Phương thức này trả về một đối tượng chứa thông tin về tin nhắn đã được thêm (bao gồm mã ID của tin nhắn) và thông tin này được gửi trong phản hồi của yêu cầu.

5 Phương hướng phát triển

Dự án hiện tại đã khá hoàn thiện, tuy nhiên còn cần được phát triển để chất lượng tốt hơn, phù hợp và thu hút người dùng.

Sau đây là một số phương hướng phát triển trong tương lai cho dự án:

- Thêm lựa chọn hình ảnh được tải lên từ máy tính người dùng và có nhiều ảnh hơn.
- Tối ưu hoá giao diện người dùng: dễ dùng và hợp thị hiếu người dùng hơn, có tính thẩm mỹ hơn.
- Nâng cao bộ lọc tìm kiếm: Tối ưu hoá thuật toán tìm kiếm để cung cấp kết quả chính xác và nhanh chóng dựa trên sở thích, khoảng cách,...
- Xác minh tài khoản: Tăng cường quy trình xác minh để đảm bảo mỗi người dùng đều là thật, giảm nguy cơ tài khoản giả mạo, đủ tuổi sử dụng (18 tuổi).
- Hệ thống báo cáo và đánh giá người dùng: báo cáo người dùng không đúng tuổi, tài khoản giả mạo, có hành vi không lành mạnh,...; đánh giá người dùng khác và khoá các tài khoản có đánh giá thấp.
- Tích hợp Video và giọng nói: Phát triển tính năng gọi video và gọi thoại để tăng cường trải nghiệm người dùng.

- Tăng cường bảo mật và quyền riêng tư: Liên tục cập nhật và tăng cường các biện pháp bảo mật để ngăn chặn mọi rủi ro liên quan đến an toàn thông tin, cung cấp tuỳ chọn quyền riêng tư linh hoạt cho người dùng kiểm soát thông tin cá nhân của mình.
- Áp dụng Trí tuệ nhân tạo và học máy: cung cấp gợi ý người dùng dựa trên các hành vi trước đó.
- Thêm ngôn ngữ.
- ...

Kết luận

Swife là một trang web hẹn hò được phát triển với mục đích tạo ra trải nghiệm kết nối độc đáo và thoải mái cho người dùng. Trang web gồm nhiều chức năng quan trọng và được lưu trữ dữ liệu trong cơ sở dữ liệu MongoDB.
Trong quá trình phát triển ***Swife***, chúng tôi đã tạo ra một giao diện thân thiện và thuận tiện cho người người dùng.
Cuối cùng, chúng tôi hy vọng rằng ***Swife*** sẽ đem lại trải nghiệm tích cực và đáp ứng nhu cầu kết nối của cộng đồng người sử dụng, đồng thời tiếp tục phát triển và cải thiện để trang web ngày càng tốt hơn.

Lời cảm ơn

Trong thời gian học tập và hoàn thiện bài báo cáo này, nhóm chúng tôi xin chân thành cảm ơn sự giúp đỡ nhiệt tình của thầy Vũ Tiến Dũng và thầy Phạm Duy Phương cùng các thầy cô trợ giảng vì đã giúp đỡ, cung cấp nhiều thông tin quý báu cũng như tạo điều kiện để nhóm có cơ hội được thực hành và học tập các khía cạnh liên quan đến Cơ sở dữ liệu Web và Hệ thống thông tin để có thể hoàn thiện được bài báo cáo này trong suốt quá trình học tập.

Mặc dù nhóm đã cố gắng hoàn thiện bài báo cáo này một cách tốt nhất. Tuy nhiên do kiến thức, kinh nghiệm còn hạn chế nên bài báo cáo của nhóm vẫn chưa được trọn vẹn. Nhóm chúng tôi rất mong nhận được những ý kiến đóng góp của giảng viên và các bạn để bài báo cáo trở nên tốt hơn.

Chúng tôi xin chân thành cảm ơn!

Tài liệu

[1] Veronika Abramova, Jorge Bernardino, (2013). *NoSQL databases: MongoDB*, 14-22.

[2] Emil Drkušić, (2018). *A Dating App Data Model*.

[3] Ania Kubow, (2022). *A Tinder Clone for educational purposes*.