

## This image shows a full page of a handwriting practice worksheet. It consists of multiple horizontal rows, each defined by two parallel dashed lines. The rows are evenly spaced across the entire page, providing a guide for letter height and placement. There is no text or other markings on the page.

(Ký và ghi rõ họ tên)

## LỜI CẢM ƠN

Đầu tiên, nhóm em xin gửi lời cảm ơn chân thành đến giảng viên bộ môn người đã hết mực chỉ bảo, hướng dẫn và sẵn sàng giải đáp thắc mắc, cung cấp những kiến thức hữu ích giúp nhóm em hoàn thành công việc trong suốt quá trình thực hiện báo cáo này.

Mặc dù đã cố gắng để thực hiện đề tài một cách nghiêm túc và tập trung trong suốt quá trình thu thập, tìm hiểu, phân tích và tổng hợp nhiều nguồn tài liệu khác nhau, song do sự hiểu biết còn chưa cao, trình độ chuyên môn còn hạn chế ở nhiều mặt và chưa có nhiều kỹ năng trong việc thực hiện các công việc trong chuyên ngành nên trong quá trình biên soạn đồ án không thể tránh khỏi các thiếu sót. Do đó nếu như có những điểm trong báo cáo còn chưa hợp lý hoặc sai sót nhóm em rất mong được đóng góp thêm các ý kiến để có thể học hỏi và tiếp thu thêm các kiến thức còn thiếu sót.

Chúng em xin chân thành cảm ơn!

*Sinh viên ký và ghi rõ họ và tên*

Ngô Huỳnh Quốc Khang

Phạm Trung Hiếu

Nguyễn Hoàng Lãm

## MỤC LỤC

CHƯƠNG 1. Giới thiệu.....	1
1.1. Tên dự án và chủ đề.....	1
1.2. Mục tiêu của ứng dụng .....	1
1.3. Lý do chọn đề tài .....	2
CHƯƠNG 2. Phân tích yêu cầu .....	3
2.1. Các chức năng chính của hệ thống (Functional Requirements) .....	3
2.2. Các yêu cầu phi chức năng (Non-functional Requirements).....	4
CHƯƠNG 3. Thiết kế hệ thống.....	5
3.1. Kiến trúc tổng thể .....	5
3.1.1. Frontend – Giao diện người dùng .....	5
3.1.2. Backend – Xử lý nghiệp vụ.....	5
3.1.3. Database – Lưu trữ dữ liệu.....	5
3.1.4. Triển khai và môi trường.....	5
3.1.5. Sơ đồ kiến trúc hệ thống .....	6
3.2. Thiết kế cơ sở dữ liệu.....	7
3.2.1. Bảng users – Quản lý người dùng.....	7
3.2.2. Bảng profile – Hồ sơ cá nhân.....	7
3.2.3. Bảng posts – Bài viết .....	7
3.2.4. Bảng comments – Bình luận .....	7
3.2.5. Bảng conversations – Cuộc trò chuyện.....	7
3.2.6. Bảng conversation_members – Thành viên trong cuộc trò chuyện.....	8
3.2.7. Bảng messages – Tin nhắn .....	8

3.2.8. Bảng message_attachments – Đính kèm tin nhắn.....	8
3.2.9. Bảng message_reads – Trạng thái đã đọc .....	8
3.3. Sơ đồ cấu trúc cơ sở dữ liệu.....	9
3.4. Thiết kế API .....	10
3.4.1. API xác thực .....	10
3.4.2. API bài viết.....	10
3.4.3. API bình luận và tương tác.....	11
3.4.4. API bạn bè .....	11
3.4.5. API tin nhắn.....	12
3.5. Thiết kế giao diện (UI/UX).....	13
3.5.1. Đăng nhập, Đăng ký.....	13
3.5.2. Newsfeed.....	14
3.5.3. Hồ sơ cá nhân .....	15
3.5.4. Trang đăng bài viết.....	16
3.5.5. Trang nhắn tin .....	17
CHƯƠNG 4. Triển khai và công nghệ sử dụng .....	18
4.1. Frontend .....	18
4.2. Backend .....	19
4.3. Database.....	19
4.4. Triển khai .....	19
4.5. CI/CD.....	19
CHƯƠNG 5. Quản lý dự án .....	20
5.1. Jira.....	20
5.1.1. Quản lý Sprint .....	20
5.1.2. Tạo User Story và Task .....	20

5.1.3. Quản lý backlog .....	20
5.2. Phân công công việc .....	21
CHƯƠNG 6. Kiểm thử.....	22
6.1. Chiến lược kiểm thử .....	22
6.1.1. Kiểm thử API bằng Postman.....	22
6.1.2. Kiểm thử luồng người dùng chính .....	22
6.1.3. Kiểm tra chất lượng mã nguồn qua CI.....	22
6.2. Kết quả kiểm thử.....	22
CHƯƠNG 7. Đánh giá và kết luận.....	23
7.1. Kết quả .....	23
7.2. Hạn chế và khó khăn.....	23
7.3. Hướng phát triển .....	23

## **DANH SÁCH HÌNH ẢNH**

Hình 3.1 Sơ đồ kiến trúc hệ thống.....	6
Hình 3.2 Sơ đồ cấu trúc cơ sở dữ liệu .....	9
Hình 3.3 Giao diện đăng ký.....	13
Hình 3.4 Giao diện đăng nhập.....	13
Hình 3.5 Giao diện trang Newsfeed .....	14
Hình 3.6 Giao diện trang hồ sơ cá nhân .....	15
Hình 3.7 Giao diện trang đăng bài viết.....	16
Hình 3.8 Giao diện trang nhấn tin .....	17

## **DANH SÁCH BẢNG BIỂU**

Bảng 5.1 Bảng phân công công việc .....	21
---	----

## **CHƯƠNG 1. GIỚI THIỆU**

### **1.1. Tên dự án và chủ đề**

Tên dự án: Xây dựng mạng xã hội dành cho sinh viên

Chủ đề: Nghiên cứu xây dựng ứng dụng và viết báo cáo kết thúc môn học Công Nghệ Phần Mềm

### **1.2. Mục tiêu của ứng dụng**

Mục tiêu chính của dự án là xây dựng một website mạng xã hội cơ bản, phục vụ đối tượng sinh viên, với các chức năng thiết yếu như:

- Đăng ký và đăng nhập tài khoản
- Tạo và chia sẻ bài viết
- Kết bạn và quản lý danh sách bạn bè
- Nhắn tin, trao đổi thông tin giữa người dùng

Ngoài ra, dự án còn hướng đến việc triển khai sản phẩm lên môi trường thực tế để kiểm thử tính ổn định, khả năng sử dụng và các vấn đề liên quan đến hiệu năng, bảo mật.

Dự án hoàn toàn không mang mục tiêu thương mại. Đây là một sản phẩm học thuật, nhằm phục vụ việc học, thực hành và rèn luyện kỹ năng trong quy trình phát triển phần mềm hiện đại.

### 1.3. Lý do chọn đề tài

“Xây dựng mạng xã hội cho sinh viên” được nhóm lựa chọn với mong muốn xây dựng một nền tảng kết nối học thuật hiện đại, phù hợp với nhu cầu thực tế của cộng đồng sinh viên trong thời đại số. Trong bối cảnh mạng xã hội đang có mức độ ảnh hưởng sâu rộng và ngày càng phát triển, việc thực hiện đề tài này không chỉ mang tính thời sự mà còn là cơ hội để nhóm:

Tiếp cận và thực hành các giai đoạn trong quy trình phát triển phần mềm như phân tích yêu cầu, thiết kế hệ thống, triển khai, kiểm thử và bảo trì.

Vận dụng kiến thức đã học vào một dự án thực tiễn nhằm nâng cao kỹ năng lập trình web, quản lý dự án, làm việc nhóm và sử dụng các công cụ công nghệ như React, Node.js, Docker, CI/CD, kiểm thử API...

Tạo ra một môi trường học tập tương tác, nơi sinh viên có thể chia sẻ tài liệu, kết nối với bạn bè, cập nhật xu hướng học tập và công nghệ.

Phát triển tư duy giải quyết vấn đề, rèn luyện kỹ năng sáng tạo và khả năng thích ứng với công nghệ mới.



## CHƯƠNG 2. PHÂN TÍCH YÊU CẦU

### 2.1. Các chức năng chính của hệ thống (Functional Requirements)

Đăng ký, đăng nhập, đăng xuất tài khoản: Người dùng có thể tạo tài khoản mới, đăng nhập vào hệ thống để sử dụng các chức năng, và đăng xuất khi kết thúc phiên làm việc.

Đăng bài viết, chỉnh sửa và xóa bài: Người dùng có thể tạo nội dung mới dưới dạng bài viết, chỉnh sửa nội dung đã đăng, hoặc xóa bài viết khi không còn cần thiết.

Bình luận và tương tác với bài viết: Cho phép người dùng bình luận dưới các bài viết và thể hiện cảm xúc bằng tính năng thả like.

Newsfeed: Hiển thị các bài viết từ bạn bè hoặc chính người dùng theo thứ tự thời gian hoặc mức độ tương tác, giúp người dùng dễ dàng theo dõi các hoạt động mới.

Quản lý hồ sơ cá nhân: Mỗi người dùng có một trang hồ sơ riêng, hiển thị thông tin cá nhân và các bài viết đã đăng.

Kết bạn và quản lý danh sách bạn bè: Người dùng có thể gửi lời mời kết bạn, chấp nhận hoặc từ chối yêu cầu, cũng như quản lý danh sách bạn bè của mình.

Nhắn tin trực tiếp: Hệ thống hỗ trợ chức năng nhắn tin, giúp người dùng trao đổi thông tin một cách riêng tư.

## 2.2. Các yêu cầu phi chức năng (Non-functional Requirements)

Giao diện người dùng đơn giản, dễ sử dụng: Thiết kế UI/UX trực quan, giúp người dùng mới dễ làm quen và sử dụng hệ thống hiệu quả.

Bảo mật thông tin người dùng: Dữ liệu nhạy cảm như mật khẩu được mã hóa bằng thuật toán băm (hash), cơ chế xác thực sử dụng JSON Web Token (JWT) để bảo vệ phiên đăng nhập.

Khả năng mở rộng và bảo trì: Mã nguồn được tổ chức rõ ràng. Các thành phần trong hệ thống được tách biệt, hỗ trợ phát triển mở rộng về sau.

Tích hợp CI/CD: Sử dụng GitHub Actions để tự động hóa quá trình kiểm thử và triển khai, rút ngắn thời gian đưa sản phẩm lên môi trường thực tế.

Triển khai bằng Docker: Hệ thống có thể được đóng gói bằng Docker, thuận tiện cho việc triển khai và quản lý môi trường.

Quản lý dự án hiệu quả: Mã nguồn được quản lý tập trung qua GitHub. Nhóm sử dụng Jira để theo dõi tiến độ, phân công công việc và quản lý công việc một cách rõ ràng và trực quan.

## CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

### 3.1. Kiến trúc tổng thể

Hệ thống mạng xã hội dành cho sinh viên được xây dựng theo mô hình Client – Server – Database, một kiến trúc phổ biến, phù hợp với các ứng dụng web hiện đại. Mô hình này chia tách rõ ràng giữa phần giao diện người dùng (client), phần xử lý nghiệp vụ (server) và hệ quản trị cơ sở dữ liệu (database), giúp dễ bảo trì, mở rộng và phân chia công việc trong nhóm phát triển.

#### 3.1.1. Frontend – Giao diện người dùng

Sử dụng React 18 kết hợp với Vite 5, react là một thư viện JavaScript mạnh mẽ, hỗ trợ phát triển giao diện theo hướng component, giúp tăng khả năng tái sử dụng mã nguồn và quản lý trạng thái hiệu quả. Vite giúp tăng tốc quá trình build và phát triển.

Chức năng chính: Giao diện cung cấp các trang như đăng nhập, đăng ký, newsfeed, trang hồ sơ cá nhân, form đăng bài viết, trang nhấn tin

#### 3.1.2. Backend – Xử lý nghiệp vụ

Sử dụng Node.js kết hợp với Express.js cho phép viết backend bằng JavaScript, cùng một ngôn ngữ với frontend, giúp đồng nhất mã nguồn trong toàn bộ dự án. Express là một framework nhẹ, dễ sử dụng và hỗ trợ tốt cho việc xây dựng RESTful API.

Chức năng chính: Xử lý yêu cầu HTTP từ client, tương tác với cơ sở dữ liệu, xác thực người dùng, gửi và nhận tin nhắn, lưu trữ bài viết, xử lý logic kết bạn

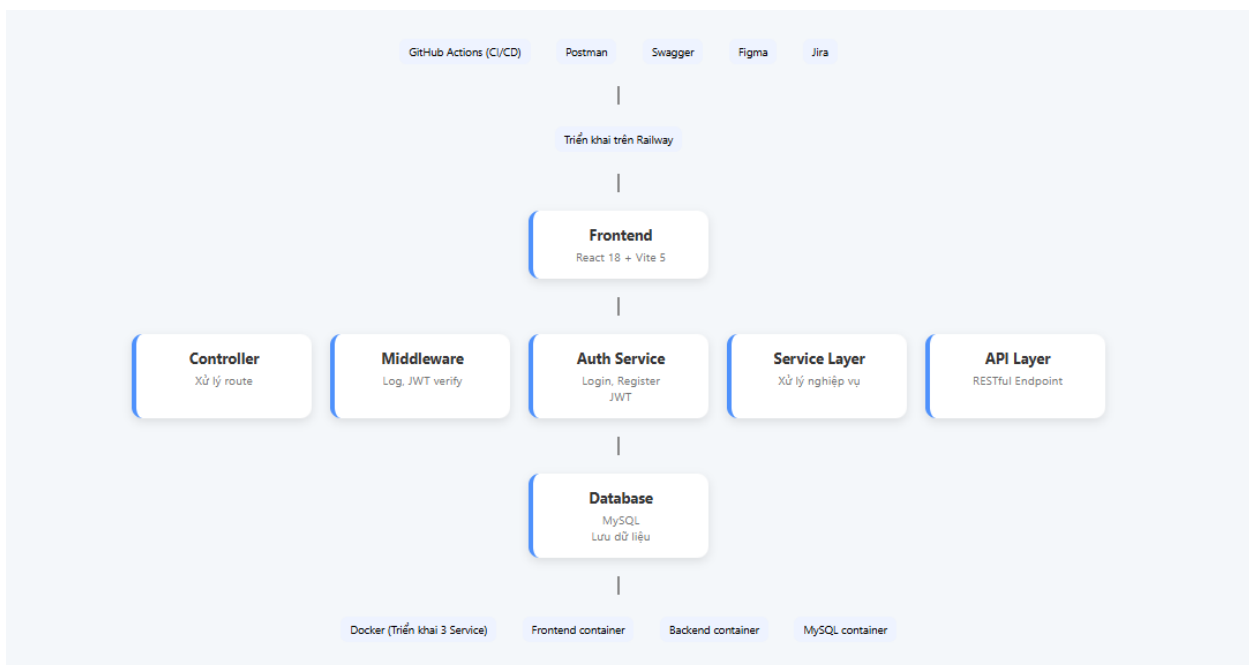
#### 3.1.3. Database – Lưu trữ dữ liệu

Sử dụng MySQL là hệ quản trị cơ sở dữ liệu quan hệ phổ biến, có hiệu năng ổn định và hỗ trợ tốt cho các quan hệ giữa bảng.

#### 3.1.4. Triển khai và môi trường

Hệ thống được triển khai thông qua Docker với 3 service chính: frontend, backend và database. Điều này giúp đảm bảo tính nhất quán của môi trường giữa các thành viên trong nhóm cũng như khi triển khai lên server thực tế.

### 3.1.5. Sơ đồ kiến trúc hệ thống



Hình 3.1 Sơ đồ kiến trúc hệ thống

## 3.2. Thiết kế cơ sở dữ liệu

### 3.2.1. Bảng users – Quản lý người dùng

Lưu trữ thông tin tài khoản cơ bản: email, mật khẩu (dưới dạng hash), trạng thái online, thời gian hoạt động gần nhất. Trường id là khóa chính, tăng tự động.

Ràng buộc:

- email là duy nhất.
- last\_active tự động cập nhật mỗi khi bản ghi thay đổi.

### 3.2.2. Bảng profile – Hồ sơ cá nhân

Lưu thông tin chi tiết của người dùng: họ tên, biệt danh, ngày sinh, địa chỉ, mô tả bản thân, ảnh đại diện và ảnh bìa. user\_id là khóa ngoại trỏ đến bảng users, quan hệ 1-1.

Ràng buộc: Khi người dùng bị xóa, hồ sơ cũng bị xóa (ON DELETE CASCADE).

### 3.2.3. Bảng posts – Bài viết

Chứa các bài viết của người dùng. Mỗi bài viết gồm tiêu đề, nội dung và thời gian tạo. User\_id là khóa ngoại liên kết với bảng users. Khi người dùng bị xóa, bài viết cũng bị xóa theo.

### 3.2.4. Bảng comments – Bình luận

Lưu các bình luận dưới bài viết. Mỗi bình luận liên kết với 1 bài viết và 1 người dùng.

Ràng buộc toàn vẹn: Khi người dùng hoặc bài viết bị xóa, bình luận tương ứng cũng bị xóa.

### 3.2.5. Bảng conversations – Cuộc trò chuyện

Đại diện cho các cuộc hội thoại (có thể là nhóm hoặc cá nhân). Có thể có tên nhóm và người quản trị (admin\_id). Nếu admin bị xóa, trường admin\_id được set NULL (ON DELETE SET NULL).

### **3.2.6. Bảng conversation\_members – Thành viên trong cuộc trò chuyện**

Quan hệ nhiều-nhiều giữa users và conversations. Ghi nhận thời gian người dùng tham gia cuộc trò chuyện. Khóa chính là tổ hợp (conversation\_id, user\_id).

### **3.2.7. Bảng messages – Tin nhắn**

Lưu trữ nội dung tin nhắn trong mỗi cuộc trò chuyện. Tin nhắn có thể là văn bản, hình ảnh, hoặc tệp tin. Trường attachment\_url chứa đường dẫn đến tệp tin nếu có.

### **3.2.8. Bảng message\_attachments – Đính kèm tin nhắn**

Lưu chi tiết các tệp tin đính kèm tin nhắn như ảnh, video, âm thanh. Ràng buộc chặt với bảng messages.

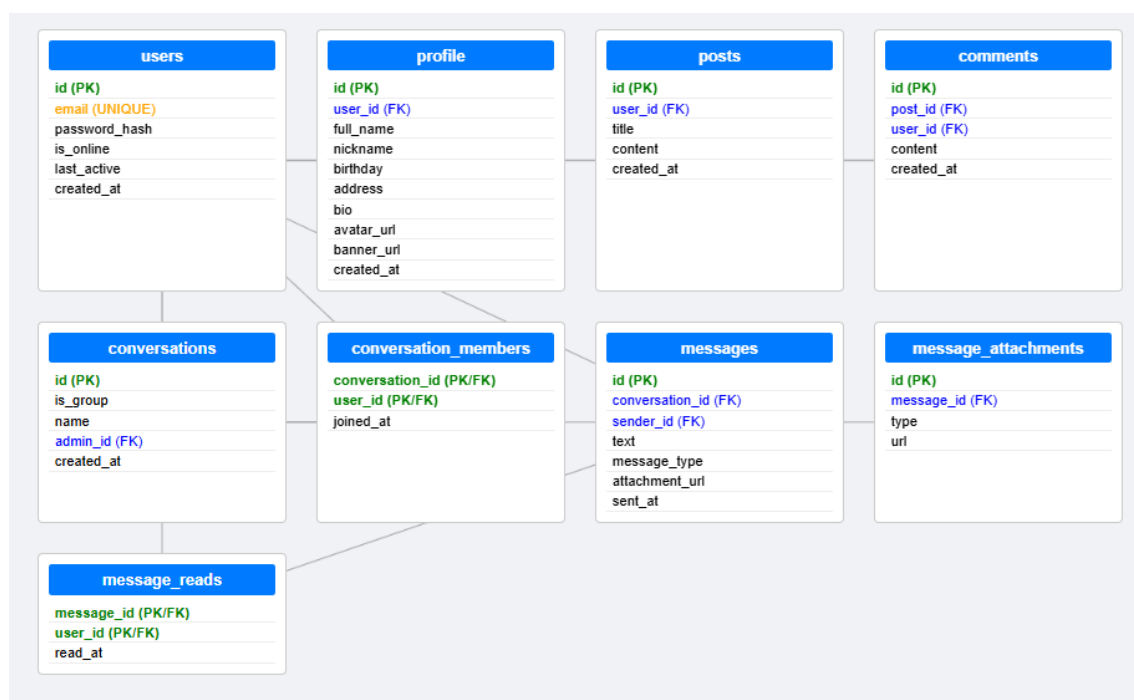
### **3.2.9. Bảng message\_reads – Trạng thái đã đọc**

Xác định người dùng nào đã đọc tin nhắn nào.

Lưu thời gian đọc (read\_at).

Khóa chính là tổ hợp (message\_id, user\_id)

### 3.3. Sơ đồ cấu trúc cơ sở dữ liệu



Hình 3.2 Sơ đồ cấu trúc cơ sở dữ liệu

## 3.4. Thiết kế API

### 3.4.1. API xác thực

Việc xác thực người dùng là bước đầu tiên và bắt buộc để đảm bảo tính bảo mật của hệ thống. Các API xác thực giúp tạo và quản lý phiên làm việc của người dùng, thường kết hợp với cơ chế mã hóa mật khẩu và sử dụng JSON Web Token (JWT) để xác minh danh tính.

Khi người dùng đăng ký, hệ thống sẽ kiểm tra tính hợp lệ của email, mã hóa mật khẩu và tạo một bản ghi mới trong bảng users. Trường password\_hash dùng để lưu chuỗi mã hóa thay vì mật khẩu gốc, nhằm đảm bảo an toàn. Khi đăng nhập, hệ thống xác minh thông tin và nếu hợp lệ, sẽ tạo một JWT token trả về phía client. Token này sẽ được dùng cho tất cả các API cần xác thực sau đó.

Ngoài ra, hệ thống còn lưu trạng thái online của người dùng thông qua trường is\_online và tự động cập nhật thời gian hoạt động cuối cùng (last\_active). Điều này rất quan trọng để xác định trạng thái thời gian thực của người dùng, đặc biệt trong các ứng dụng có tính tương tác cao.

### 3.4.2. API bài viết

Bài viết là nội dung trung tâm trong hệ thống mạng xã hội. Mỗi bài viết được liên kết với một người dùng thông qua khóa ngoại user\_id. Hệ thống API cho phép người dùng tạo bài viết mới, chỉnh sửa nội dung cũ và xóa bài viết khi cần thiết. Khi một bài viết được tạo, các trường như title, content, và created\_at sẽ được ghi nhận. Việc thiết kế đơn giản hóa để tạo điều kiện mở rộng thêm các tính năng như đính kèm hình ảnh, hashtag hoặc gắn thẻ bạn bè trong tương lai.

Khi người dùng hoặc khách truy cập xem bài viết, API sẽ truy xuất dữ liệu từ bảng posts, kết hợp với thông tin người đăng (từ bảng users hoặc profile) cũng như các tương tác liên quan như bình luận và lượt thích. Đây là một quy trình quan trọng nhằm hiển thị thông tin đầy đủ, tạo trải nghiệm người dùng phong phú và liền mạch.



### 3.4.3. API bình luận và tương tác

Tính năng tương tác cho phép người dùng tham gia vào các cuộc thảo luận và thể hiện cảm xúc trên bài viết. Phần này được xây dựng dựa trên hai bảng: comments và likes.

Bình luận là hành động tạo nội dung mới liên quan đến một bài viết cụ thể. Khi người dùng gửi bình luận, hệ thống sẽ lưu thông tin vào bảng comments, bao gồm nội dung, người viết và thời gian tạo. Khi cần hiển thị, API sẽ lấy toàn bộ danh sách bình luận của một bài viết, thường được sắp xếp theo thời gian hoặc mức độ phổ biến.

Tương tự, mỗi lượt thích được lưu thành một bản ghi riêng biệt trong bảng likes. Để tránh việc một người dùng thích nhiều lần trên cùng một bài viết, hệ thống sử dụng khóa UNIQUE (post\_id, user\_id) để đảm bảo tính duy nhất. Người dùng cũng có thể bỏ thích bằng cách gọi API xoá lượt thích. Những dữ liệu này sau đó có thể được dùng để thống kê mức độ phổ biến của bài viết, cũng như cá nhân hóa nội dung hiển thị cho từng người.

### 3.4.4. API bạn bè

Quan hệ xã hội là phần không thể thiếu trong bất kỳ nền tảng mạng xã hội nào. Trong hệ thống này, mối quan hệ bạn bè không được lưu trực tiếp như một bảng liên kết đơn giản, mà thay vào đó sử dụng mô hình “yêu cầu kết bạn” – tức là trước khi trở thành bạn bè, phải có một người gửi lời mời và người kia chấp nhận.

Bảng friend\_requests lưu các yêu cầu đang chờ xử lý, đã chấp nhận hoặc bị từ chối. Mỗi bản ghi gồm sender\_id, receiver\_id, trạng thái status và các mốc thời gian. Điều này không chỉ giúp kiểm soát logic hệ thống (ai được kết bạn với ai), mà còn hỗ trợ việc thống kê hoặc gợi ý bạn bè dựa trên các tương tác trước đó.

API trong nhóm này cho phép người dùng gửi lời mời, xem danh sách yêu cầu đến, chấp nhận hoặc từ chối kết bạn. Sau khi được chấp nhận, hệ thống sẽ xác nhận quan hệ hai chiều (có thể tạo bảng trung gian nếu cần tối ưu hoá tìm kiếm bạn bè sau này). Ngoài ra, người dùng cũng có quyền huỷ kết bạn, điều này dẫn tới xoá quan hệ đã xác lập.

### 3.4.5. API tin nhắn

Tin nhắn riêng tư là thành phần nâng cao trong hệ thống, cho phép người dùng giao tiếp trực tiếp hoặc theo nhóm. Cơ sở dữ liệu hỗ trợ điều này thông qua các bảng: `conversations`, `conversation_members`, `messages`, `message_attachments` và `message_reads`.

Một cuộc trò chuyện có thể là một nhóm hoặc giữa hai người, được xác định qua trường `is_group`. Mỗi cuộc trò chuyện lưu thông tin cơ bản như tên, ảnh đại diện và người quản trị nhóm (`admin_id`). Thành viên tham gia được lưu trong bảng `conversation_members`, giúp mở rộng và quản lý linh hoạt danh sách người dùng.

Mỗi tin nhắn được lưu trong bảng `messages`, có thể là văn bản, hình ảnh hoặc file đính kèm (thể hiện qua các trường `message_type` và `attachment_url`). Để tăng tính cá nhân hóa và tương tác, hệ thống còn ghi nhận trạng thái đã đọc thông qua bảng `message_reads`. Điều này hỗ trợ tính năng “seen” quen thuộc với người dùng.

API tin nhắn không chỉ hoạt động theo kiểu truyền thống (gửi/nhận qua các request HTTP), mà còn tích hợp giao tiếp thời gian thực thông qua WebSocket. Nhờ đó, khi một người gửi tin nhắn, người nhận có thể nhận ngay lập tức mà không cần phải tải lại trang.


### 3.5. Thiết kế giao diện (UI/UX)

#### 3.5.1. Đăng nhập, Đăng ký



The registration form is titled "ĐĂNG KÝ" in bold black text. It features three input fields: "Email", "Mật khẩu" (Password), and "Nhập lại mật khẩu" (Confirm Password). A blue button labeled "ĐĂNG KÝ" is positioned below the fields. To the right, a blue vertical bar contains a white circular logo with a megaphone icon and the text "CHÉMNET".

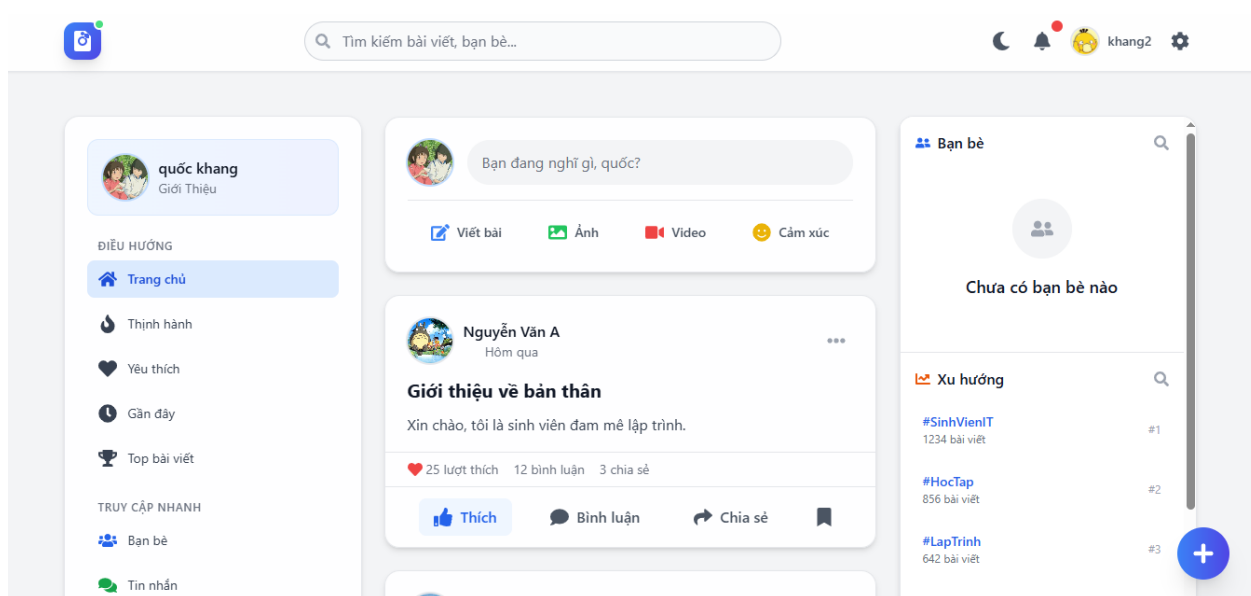
Hình 3.3 Giao diện đăng ký



The login form is titled "ĐĂNG NHẬP" in bold black text. It features two input fields: "Email" and "Nhập mật khẩu" (Enter password). A blue button labeled "ĐĂNG NHẬP" is positioned below the fields. To the left, a blue vertical bar contains a white circular logo with a megaphone icon and the text "CHÉMNET".

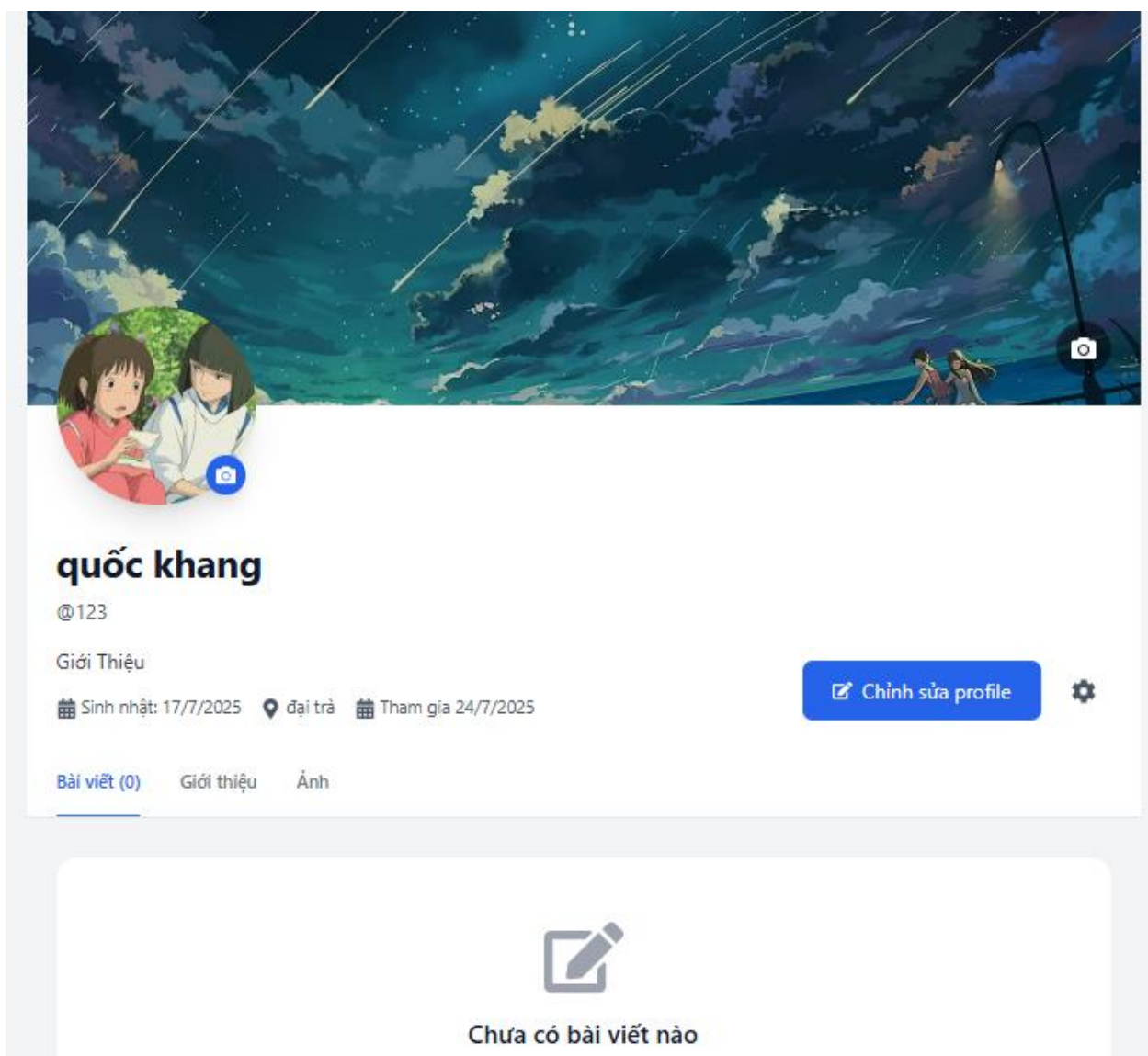
Hình 3.4 Giao diện đăng nhập

### 3.5.2. Newsfeed



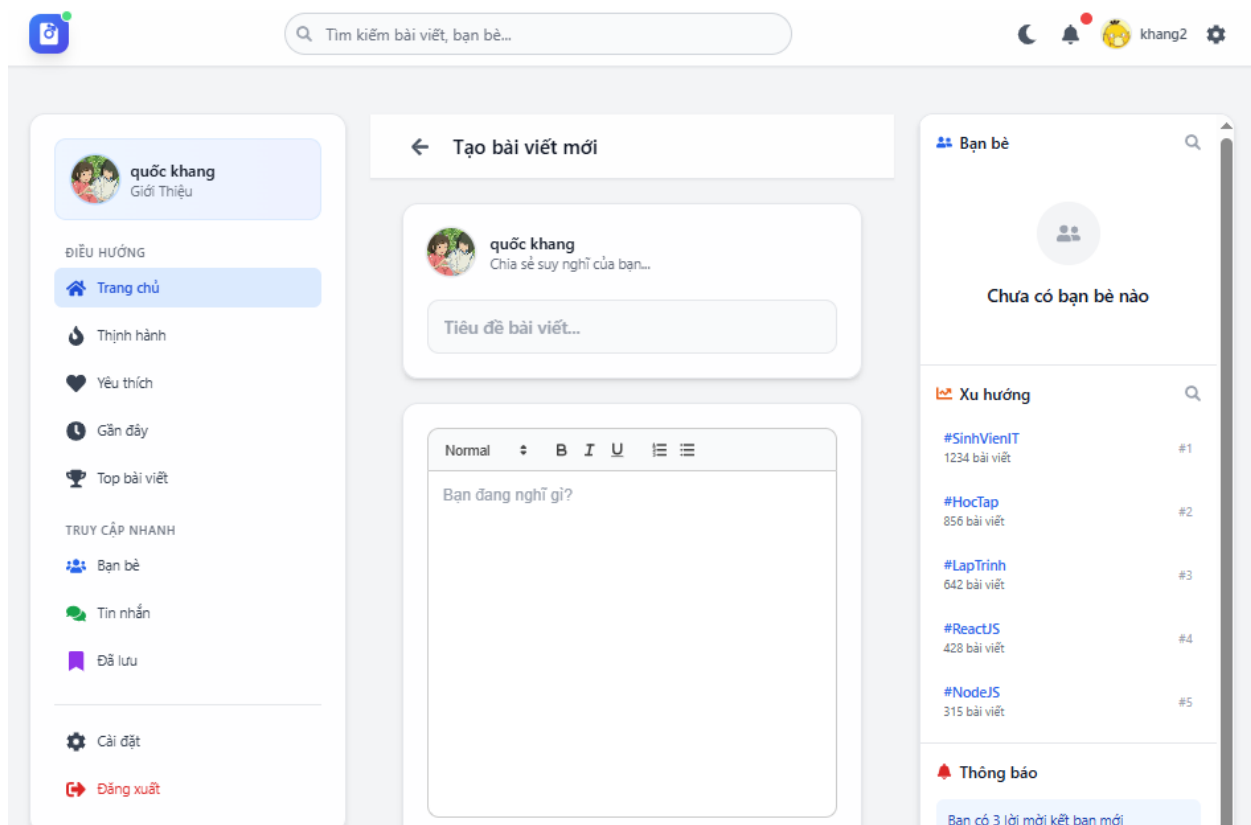
Hình 3.5 Giao diện trang Newsfeed

### 3.5.3. Hồ sơ cá nhân



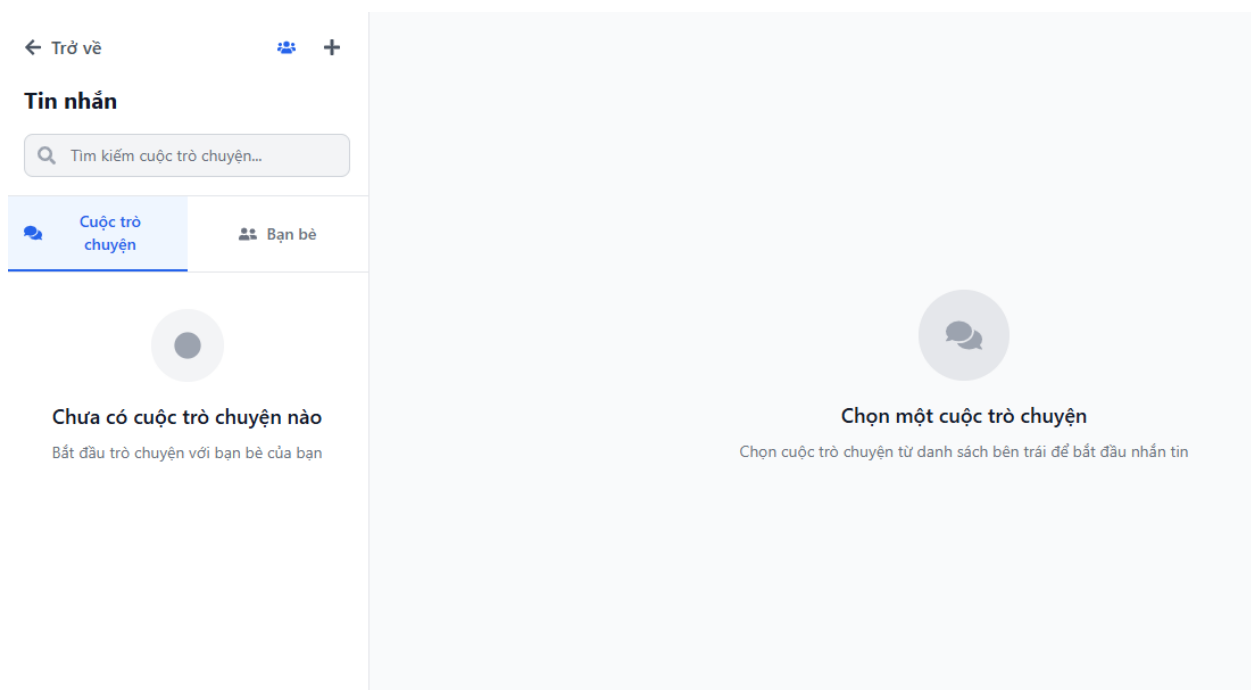
Hình 3.6 Giao diện trang hồ sơ cá nhân

### 3.5.4. Trang đăng bài viết



Hình 3.7 Giao diện trang đăng bài viết

### 3.5.5. Trang nhắn tin



Hình 3.8 Giao diện trang nhắn tin

## CHƯƠNG 4. TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG

### 4.1. Frontend

ReactJS 18, Vite 5 giúp phát triển giao diện nhanh, tối ưu hiệu năng.

Tailwind CSS hỗ trợ thiết kế giao diện hiện đại, linh hoạt.

react-router-dom: định tuyến trang.

axios: gọi API.

zustand: quản lý trạng thái ứng dụng.

react-toastify: hiển thị thông báo.

emoji-picker-react: hỗ trợ chèn biểu tượng cảm xúc.

react-quill: trình soạn thảo văn bản.

jwt-decode: giải mã JWT token.

lucide-react, react-icons: icon cho UI.

dompurify: lọc nội dung HTML để tăng tính bảo mật.



## 4.2. Backend

Node.js và Express 5.1.0 để xây dựng hệ thống API.

bcryptjs: mã hóa mật khẩu.

cors: xử lý vấn đề chia sẻ tài nguyên giữa các domain.

cookie-parser: đọc và ghi cookie.

dotenv: quản lý biến môi trường.

jsonwebtoken: xác thực người dùng bằng JWT.

multer: xử lý upload ảnh.

mysql2: tương tác cơ sở dữ liệu MySQL.

socket.io: tích hợp tính năng chat thời gian thực.

swagger-jsdoc, swagger-ui-express: tài liệu hóa API.

## 4.3. Database

MySQL: lưu trữ dữ liệu theo mô hình quan hệ.

## 4.4. Triển khai

Docker: triển khai từng phần frontend, backend, database trong các container riêng biệt và độc lập.

Railway: nền tảng cloud hỗ trợ triển khai hệ thống nhanh chóng mà không cần cấu hình máy chủ thủ công.

## 4.5. CI/CD

Cài đặt dependencies: Tự động cài các thư viện cần thiết cho backend và frontend.

Chạy kiểm tra lint: Đảm bảo mã nguồn tuân thủ các quy chuẩn mã hóa.

Chạy kiểm thử: Các đoạn mã kiểm thử sẽ được thực hiện tự động nhằm kiểm tra logic và độ ổn định của hệ thống.

## CHƯƠNG 5. QUẢN LÝ DỰ ÁN

### 5.1. Jira

#### 5.1.1. Quản lý Sprint

Nhóm chia quá trình phát triển thành nhiều Sprint, mỗi Sprint kéo dài 2 tuần.

Mỗi Sprint bắt đầu bằng phiên họp lập kế hoạch, nơi các thành viên thảo luận, ước lượng thời gian và xác định các công việc sẽ thực hiện trong Sprint đó.

#### 5.1.2. Tạo User Story và Task

Dự án được chia nhỏ thành các User Story, mỗi User Story mô tả một tính năng cụ thể từ góc nhìn người dùng.

Các User Story sau đó được chia thành các task nhỏ hơn là sub-tasks, phù hợp với khả năng thực hiện của từng thành viên.

#### 5.1.3. Quản lý backlog

Tất cả các đầu việc đều được ghi lại trong Product Backlog, đảm bảo không bỏ sót yêu cầu nào.

Bảng được sử dụng để theo dõi tiến độ theo thời gian thực với các cột "To Do", "In Progress", "Review", "Done".

## 5.2. Phân công công việc

*Bảng 5.1 Bảng phân công công việc*

Họ tên thành viên	Chức năng phụ trách	Nhiệm vụ cụ thể
Nguyễn Hoàng Lãm	Đăng ký, đăng nhập, xác thực	Xây dựng API đăng ký/đăng nhập, xử lý JWT, giao diện và kiểm thử tính năng
Ngô Huỳnh Quốc Khang	Trang newsfeed hiển thị nội dung bài viết	Hiển thị newsfeed các nội dung bài viết của người dùng đã tạo trước đó
Nguyễn Hoàng Lãm	Kết bạn và hệ thống bạn bè	Xây dựng API gửi/duyet lời mời kết bạn, quản lý danh sách bạn bè, giao diện tương tác
Nguyễn Hoàng Lãm	Hệ thống nhắn tin	Thiết kế cấu trúc lưu trữ tin nhắn, xây dựng API chat, tạo giao diện nhắn tin thời gian thực
Phạm Trung Hiếu	Hồ sơ cá nhân	Thiết kế trang profile, chỉnh sửa thông tin người dùng
Ngô Huỳnh Quốc Khang	Quản lý bài viết	Tạo API bài viết, thiết kế giao diện bài viết

## CHƯƠNG 6. KIỂM THỬ

### 6.1. Chiến lược kiểm thử

#### 6.1.1. Kiểm thử API bằng Postman

Postman được sử dụng để kiểm thử hầu hết các API trong hệ thống.

Các request được cấu hình chi tiết kèm theo thông tin đầu vào (headers, body), kiểm tra phản hồi trả về (status code, dữ liệu JSON).

Các luồng chính được kiểm tra gồm: tạo tài khoản, đăng nhập, đăng bài viết, gửi bình luận, gửi lời mời kết bạn, nhắn tin.

#### 6.1.2. Kiểm thử luồng người dùng chính

Các luồng quan trọng như: đăng ký, đăng nhập, đăng bài, nhắn tin đều được kiểm thử xuyên suốt để đảm bảo tính ổn định và mạch lạc giữa các chức năng.

Các luồng đăng nhập không hợp lệ, xử lý lỗi mạng, thông báo lỗi giao diện cũng được đưa vào kiểm thử.

#### 6.1.3. Kiểm tra chất lượng mã nguồn qua CI

Các workflow CI thiết lập bằng GitHub Actions có bước kiểm tra lint để đảm bảo mã sạch, tuân thủ quy tắc viết code.

Hệ thống cũng tự động build và ghi log kết quả tại từng bước, giúp dễ dàng truy vết lỗi nếu có.

### 6.2. Kết quả kiểm thử

API backend: Các endpoint đều hoạt động đúng, trả về kết quả đúng định dạng, xử lý logic chính xác.

Luồng người dùng: Các bước đăng ký, đăng nhập, đăng bài, nhắn tin được kiểm thử thành công liên tục trong nhiều phiên bản triển khai.

CI/CD: code nhánh chính chạy qua workflow CI gặp phải phát sinh lỗi cấu trúc trong lúc test, sau vài lần khắc phục đã không còn xảy ra lỗi cấu trúc.

## CHƯƠNG 7. ĐÁNH GIÁ VÀ KẾT LUẬN

### 7.1. Kết quả

Hoàn thiện đầy đủ các chức năng cốt lõi như: đăng ký, đăng nhập, tạo bài viết, kết bạn, nhắn tin.

Giao diện người dùng đơn giản, dễ sử dụng, đáp ứng tốt trải nghiệm cơ bản.

Hệ thống backend hoạt động ổn định, quản lý dữ liệu hiệu quả thông qua các API.

Tích hợp thành công CI/CD và Docker để đảm bảo quá trình phát triển hiện đại, dễ mở rộng.

Áp dụng được các công cụ thực tế như Jira, Postman vào quy trình làm việc.

### 7.2. Hạn chế và khó khăn

Các thành viên mất nhiều thời gian để cấu hình và chạy Docker, đặc biệt trong việc kết nối mạng giữa các container và gỡ lỗi khi build image thất bại.

Trong quá trình giao tiếp giữa frontend và backend, một số lỗi liên quan đến CORS phát sinh do cấu hình chưa đồng bộ, gây gián đoạn kết nối tạm thời.

### 7.3. Hướng phát triển

Trong tương lai, nhóm mong muốn tiếp tục mở rộng và hoàn thiện hệ thống theo các hướng tăng tính bảo mật như áp dụng thêm các cơ chế bảo vệ như xác thực hai bước, giới hạn truy cập API theo vai trò, Nâng cấp UI/UX để phù hợp với nhiều thiết bị, hỗ trợ responsive tốt hơn trên điện thoại.