HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CNTT 1

BỘ MÔN THỰC TẬP CƠ SỞ

HỆ THỐNG CHAT REALTIME TÍCH HỢP AI PHÂN LOẠI NỘI DUNG ĐỘC HẠI

Giảng viên : **Kim Ngọc Bách**

Họ và tên sinh viên : **Phạm Nhật Minh**

Mã sinh viên : **B23DCCE066**

Lớp : **D23CQCE06-B**

Hà Nội – 2026

MỤC LỤC

1. Giới thiệu dự án..... 3
1.1. Lý do chọn đề tài.....3
1.2. Ý nghĩa và tính ứng dụng của hệ thống..... 4
1.3. Giá trị học thuật và đóng góp nghiên cứu.....4
2. Cơ sở lý thuyết và công nghệ sử dụng.....5
2.1. Kiến trúc luồng thời gian thực (MERN & WebSocket).....5
2.2. Hệ thống đệm và đồng bộ dữ liệu với Redis.....6
2.3. Phương pháp lai (Hybrid) và Mạng Bayes.....6
2.4. Mô hình tích hợp kiểm duyệt AI vào hệ thống Web..... 7
3. Phân tích yêu cầu của dự án..... 7
3.1. Yêu cầu chức năng (Functional Requirements)..... 7
3.2. Yêu cầu phi chức năng (Non-Functional Requirements).....8
4. Kế hoạch thực hiện dự án..... 8
TÀI LIỆU THAM KHẢO.....10

1. Giới thiệu dự án

Ngày nay, sự phát triển bùng nổ của kỷ nguyên công nghệ số đã làm thay đổi hoàn toàn cách thức con người chúng ta giao tiếp và làm việc. Thay vì phải gặp mặt trực tiếp để giải quyết vấn đề, chúng ta hoàn toàn có thể kết nối, thảo luận và xử lý công việc ngay lập tức thông qua các nền tảng trực tuyến. Từ những công cụ gửi nhận tin nhắn cơ bản, các ứng dụng hiện đại ngày nay đã tiến hóa thành những không gian làm việc chung, đòi hỏi khả năng trao đổi dữ liệu liên tục với độ trễ thấp và tính ổn định cao.

Tuy nhiên, cùng với sự phát triển đó, việc quản lý nội dung và tạo ra một môi trường giao tiếp văn minh, an toàn cũng đang trở thành một thách thức lớn đối với các nhà phát triển.

Xuất phát từ những nhu cầu thực tế trên, báo cáo này sẽ tập trung nghiên cứu và xây dựng một *Hệ thống nhắn tin thời gian thực (Real-time Chat System)*. Web được phát triển dựa trên nền tảng công nghệ của *MERN Stack* bao gồm *MongoDB, Express.js, React.js, Node.js* cùng với đó là kết hợp với *Redis* để tối ưu hóa tốc độ và khả năng đồng bộ dữ liệu. Điểm nổi bật của hệ thống là việc sẽ tích hợp mô hình *Học máy (Machine Learning)* và *Xử lý ngôn ngữ tự nhiên (NLP)* nhằm tự động phân tích, nhận diện và ngăn chặn các tin nhắn chứa nội dung độc hại, từ đó nâng cao trải nghiệm và bảo vệ người dùng trên không gian mạng.

1.1. Lý do chọn đề tài

Hiện nay, các ứng dụng nhắn tin thời gian thực (real-time chat) đã trở thành công cụ nền tảng không thể thiếu trong cả giao tiếp cá nhân lẫn môi trường công sở. Điều này sẽ đặt ra yêu cầu cho các hệ thống này ngày càng khắt khe: không chỉ phải đảm bảo tốc độ gửi nhận tin nhắn tức thì, mà còn đòi hỏi tính ổn định và khả năng chịu tải (scalability) khi có nhiều kết nối đồng thời. Để giải quyết bài toán hiệu năng này, chúng ta sẽ áp dụng các ngăn xếp công nghệ hiện đại như *MERN (MongoDB, Express.js, React.js, Node.js)* kết hợp với hệ thống bộ nhớ đệm *Redis* đang là tiêu chuẩn thiết kế kiến trúc phần mềm tối ưu.

Tuy nhiên thì, khi một nền tảng giao tiếp được vận hành, trải nghiệm của người dùng lại dễ dàng bị suy giảm bởi sự xuất hiện của các tin nhắn rác, ngôn từ thù ghét hay quấy rối. Để hệ thống chat thực sự hoàn thiện và an toàn, bên cạnh việc xây dựng luồng dữ liệu mượt mà, cần có một cơ chế kiểm duyệt tự động hoạt động ngầm mà không làm ảnh hưởng đến độ trễ của luồng chat.

Xuất phát từ những thực tiễn đó, đề tài *"Xây dựng ứng dụng Chat Realtime tích hợp AI phân loại lọc nội dung độc hại"* được thực hiện. Trọng tâm của đề tài sẽ là thiết kế, phát triển một hệ thống web giao tiếp thời gian thực hoàn chỉnh và tối ưu hóa hiệu năng bằng Redis. Đồng thời, dự án tiến hành tích hợp thêm module AI (Học máy) đóng vai trò như một màng lọc tự động, giúp nhận diện và ngăn chặn tức thời các nội dung tiêu cực, hướng tới một môi trường giao tiếp mạng văn minh và an toàn.

1.2. Ý nghĩa và tính ứng dụng của hệ thống

Việc thiết kế và triển khai thành công hệ thống này mang lại những giá trị ứng dụng thực tiễn to lớn trong việc làm sạch không gian mạng như sau:

- **Bảo vệ người dùng và sức khỏe tinh thần:** Bằng cách chủ động ngăn chặn các luồng tin nhắn quấy rối và bắt nạt, web này sẽ tạo ra một môi trường số an toàn, lành mạnh, khuyến khích sự tự do biểu đạt tích cực.
- **Tối ưu hóa chi phí vận hành:** Việc ứng dụng các thuật toán Học máy giúp tự động hóa khâu kiểm duyệt nội dung, giảm thiểu đáng kể chi phí nhân sự và loại bỏ các sai sót do yếu tố con người. Hệ thống có thể ngay lập tức phát hiện các hành vi vi phạm và thậm chí gửi email cảnh báo tự động cho người quản trị hoặc người dùng vi phạm.
- **Ứng dụng đa lĩnh vực:** Nền tảng này có thể được áp dụng rộng rãi không chỉ trong các ứng dụng mạng xã hội mà còn tích hợp vào các nền tảng giáo dục trực tuyến, hệ thống bình luận của báo điện tử, hay các ứng dụng phát sóng trực tiếp (livestream), nơi luồng tin nhắn xuất hiện ồ ạt và liên tục.

1.3. Giá trị học thuật và đóng góp nghiên cứu

Đề tài là minh chứng cho việc vận dụng hiệu quả các công nghệ lập trình Web hiện đại để giải quyết bài toán xử lý dữ liệu thời gian thực và quản trị nội dung. Những đóng góp cốt lõi bao gồm:

- **Tối ưu hóa hiệu năng và khả năng mở rộng:** Giải quyết rào cản về việc đồng bộ hóa dữ liệu của giao thức WebSocket khi hệ thống mở rộng nhiều máy chủ thông qua cơ chế Redis Pub/Sub.
- **Thiết kế hệ thống linh hoạt:** Triển khai tách biệt luồng xử lý logic (Node.js/Express) và luồng xử lý trí tuệ nhân tạo (Python/FastAPI) để tối ưu hóa tài nguyên và dễ dàng bảo trì.
- **Xây dựng luồng kiểm duyệt đa lớp (Cascade Moderation):** Thiết lập quy trình lọc nội dung thông minh theo thứ tự: *Luật cứng (Blacklist) -> Bộ nhớ đệm (Redis Cache) -> AI Phân tích ngữ nghĩa.* Quy trình này giúp đảm bảo độ trễ

thấp nhất cho tin nhắn trong khi vẫn duy trì khả năng kiểm soát nội dung độc hại một cách tự động.

2. Cơ sở lý thuyết và công nghệ sử dụng

Hệ thống được thiết kế xoay quanh ngăn xếp công nghệ MERN kết hợp cùng nền tảng lưu trữ cấu trúc dữ liệu trên RAM Redis và giao thức truyền tải WebSocket. Phần này sẽ phân tích về cơ sở lý thuyết và công nghệ được ứng dụng trong hệ thống.

2.1. Kiến trúc luồng thời gian thực (MERN & WebSocket)

MongoDB: Là một hệ quản trị cơ sở dữ liệu mã nguồn mở thuộc học NoSQL. Nó được thiết kế theo kiểu hướng đối tượng, các bảng trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ trên bảng không cần tuân theo một cấu trúc nhất định nào cả. Thêm vào đó, MongoDB lưu trữ dữ liệu theo hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.[1]

Express.js và Node.js: Xử lý theo Restful API và điều phối luồng WebSocket. Node.js hoạt động dựa trên cơ chế I/O không chặn, cho phép nó vừa duy trì hàng ngàn kết nối WebSocket đồng thời, vừa giao tiếp không đồng bộ (asynchronous) với hệ thống Học máy để kiểm duyệt tin nhắn trước khi chạy. [2]

React.js: Giao diện người dùng sử dụng Virtual DOM để tối ưu hóa hiển thị. Giao diện này sẽ phản hồi theo thời gian thực với các nhãn dán cảnh báo hoặc che mờ các tin nhắn bị hệ thống AI đánh giá là độc hại.[2]

WebSocket & Socket.io: Giúp đảm bảo giao tiếp thời gian thực đáng tin cậy trên nhiều môi trường khác nhau. Socket.io đóng vai trò quan trọng trong việc triển khai các tính năng như nhắn tin thời gian thực, cập nhật trạng thái hiện diện (online/offline) và thông báo trong ứng dụng trò chuyện. Một trong những ưu điểm của Socket.io là khả năng duy trì giao tiếp với độ trễ thấp, điều này rất cần thiết để mang lại trải nghiệm người dùng liền mạch trong các ứng dụng thời gian thực.[2]

2.2. Hệ thống đệm và đồng bộ dữ liệu với Redis

Chat theo thời gian thực sẽ cần đòi hỏi nền tảng giao tiếp trực tuyến phải xử lý hội thoại ngay lập tức mà gần như không có độ trễ. Để giải quyết bài toán này, chúng ta sẽ sử dụng Redis nhờ vào ba ưu điểm cốt lõi sau:

- **Tốc độ xử lý siêu tốc:** Ta sẽ dùng Redis Pub/Sub, nó giúp đảm nhận vai trò trung gian giữa các chat clients và server. Khi một người dùng gửi tin nhắn, server sẽ gửi tin nhắn đó vào Redis. Redis sau đó sẽ đẩy tin nhắn ra cho các người dùng khác ngay lập tức thông qua kênh pub/sub. [3]
- **Cấu trúc dữ liệu phong phú:** Không chỉ lưu trữ dữ liệu đơn thuần, Redis hỗ trợ mạnh mẽ nhiều kiểu dữ liệu phức tạp như *Lists* (Danh sách), *Sets* (Tập hợp), *Sorted Sets* (Tập hợp có sắp xếp) và *Hashes* (Bảng băm). Sự linh hoạt này giúp chúng ta dễ dàng xây dựng các tính năng như lưu trữ lịch sử tin nhắn, quản lý danh sách người dùng trực tuyến hay phân loại phòng chat.[4]
- **Khả năng mở rộng mạnh mẽ với Pub/Sub:** Redis tích hợp sẵn mô hình truyền thông điệp Publish/Subscribe (Pub/Sub). Cơ chế này cho phép chúng ra dễ dàng mở rộng quy mô (scale) hệ thống backend bằng cách tạo ra nhiều máy chủ (server instances) hoạt động song song, giúp ứng dụng chịu tải tốt ngay cả khi lượng người dùng nhắn tin tăng vọt.[4]

2.3. Phương pháp lai (Hybrid) và Mạng Bayes.

Hệ thống kiểm duyệt tin nhắn thời gian thực áp dụng phương pháp lai (Hybrid) để đảm bảo độ trễ thấp nhất:

- **1. Bộ lọc luật cứng (Rule-based Blacklist):** Tin nhắn được đối chiếu tức thì với danh sách đen. Nếu phát hiện từ hoặc cụm từ vi phạm, hệ thống chặn ngay lập tức với độ tin cậy 100%, bỏ qua bước AI để tiết kiệm tối đa tài nguyên server.
- **2. Tiền xử lý NLP:** Sử dụng thư viện underthesea để tách từ tiếng Việt chuẩn xác. Sau đó, thuật toán TfidfVectorizer (áp dụng n-gram từ 1-3) sẽ chuyển đổi văn bản thành các vector toán học, giúp hệ thống nắm bắt được ngữ cảnh thay vì chỉ đọc từng từ rời rạc.
- **3. Phân loại bằng AI (Complement Naive Bayes - CNB):** Dữ liệu vector được đưa vào mô hình CNB. Thuật toán này được chọn vì tốc độ tính toán xác suất cực kỳ nhanh, rất lý tưởng cho chat realtime. Nó có thể dự đoán nhãn (Tích cực/Tiêu cực/Trung tính) và cấu hình chỉ đánh dấu là độc hại nếu độ chắc chắn (Confidence Score) vượt mức 60%.[5]

2.4. Mô hình tích hợp kiểm duyệt AI vào hệ thống Web

Để tối ưu hiệu suất và giữ độ trễ thấp nhất cho ứng dụng chat thời gian thực, hệ thống áp dụng kiến trúc tách biệt kết hợp cùng cơ chế kiểm duyệt đa lớp:

- **Kiến trúc Microservices:** Mô hình AI chạy hoàn toàn độc lập trên Python (FastAPI). Điều này giúp tận dụng tối đa sức mạnh tính toán của Python mà không làm nghẽn luồng xử lý sự kiện (Event-loop) của máy chủ Node.js.
- **Kiểm duyệt 3 lớp tối ưu:** Tin nhắn nhận từ Socket.io sẽ đi qua quy trình:
    1. Lọc nhanh qua danh sách từ khóa cấm (Blacklist).
    2. Tra cứu bộ nhớ đệm (Redis Cache) xem nội dung này đã từng được AI phân tích trước đó chưa.
    3. Chỉ gọi API sang AI Server (FastAPI) để xử lý ngữ nghĩa nếu đây là nội dung mới.
- **Xử lý thời gian thực:** Dựa trên kết quả trả về, Node.js sẽ quyết định từ chối tin nhắn hoặc tiến hành lưu trữ vào MongoDB và phát sóng ngay lập tức đến các người dùng khác trong phòng.

3. Phân tích yêu cầu của dự án

3.1. Yêu cầu chức năng (Functional Requirements)

Các chức năng của web sẽ có thể bao gồm như sau:

- **Giao tiếp và Quản lý phòng Chat:** Đăng ký, đăng nhập bằng JWT. Tạo các kênh chat 1-1 và chat nhóm. Hiển thị trạng thái trực tuyến (online/offline).
- **Kết bạn, hủy kết bạn,...**: Thực hiện các chức năng như gửi lời mời, chấp nhận, hủy kết bạn,...
- **Tính năng tạo cuộc hội thoại:** Người dùng có thể tạo cuộc hội thoại, lấy tất cả tin, đánh dấu đã xem, chưa đọc,...
- **Tính năng tin nhắn:** Nhắn tin với 1-1 hoặc nhóm,....
- **Tùy biến giao diện hồ sơ:** Có thể tùy biến giao diện hồ sơ như tải ảnh lên,...
- **Kiểm duyệt nội dung tự động (Automated Content Moderation):** Mọi tin nhắn văn bản khi gửi đi đều tự động chạy qua mô hình Học máy để trích xuất đặc trưng và dự đoán cấp độ độc hại. Hệ thống phải xử lý được các nhãn phân loại như: Tích cực, Trung tính, Tiêu cực.
- **Gắn cờ và Xử lý vi phạm:** Nếu tin nhắn được mô hình đánh giá là độc hại, hệ thống không hiển thị nội dung đó ra công khai mà thay thế bằng cảnh báo dán nhãn (ví dụ: "Tin nhắn đã bị ẩn do vi phạm tiêu chuẩn cộng đồng").
- **Hệ thống phân tích quản trị (Admin Dashboard):** Cung cấp giao diện cho ban quản trị theo dõi số lượng tin nhắn độc hại bị chặn, hiệu suất của bộ lọc AI.

3.2. Yêu cầu phi chức năng (Non-Functional Requirements)

- **Hiệu năng và Độ trễ:** Thời gian dự đoán của mô hình học máy cộng với thời gian trễ mạng không được phép làm gián đoạn trải nghiệm thời gian thực. Tổng độ trễ từ lúc người dùng nhấn "Gửi" đến khi tin nhắn qua bộ lọc AI và hiển thị trên màn hình người nhận phải duy trì dưới mốc 100ms.
- **Khả năng mở rộng (Scalability):** Hệ thống phân tách rõ ràng giữa máy chủ Node.js và máy chủ Python AI. Khi lượng tin nhắn tăng vọt, quản trị viên có thể dễ dàng nhân bản các node xử lý AI độc lập mà không ảnh hưởng đến cụm web backend.
- **Tính bảo mật:** Đảm bảo toàn vẹn dữ liệu cá nhân. Các tin nhắn đẩy qua bộ lọc AI phải được mã hóa truyền tải và hệ thống chỉ sử dụng dữ liệu trong RAM để quét, không lưu trữ vĩnh viễn các đoạn chat cá nhân cho mục đích huấn luyện trái phép.

4. Kế hoạch thực hiện dự án (khoảng 13 tuần)

a, Tìm hiểu kiến thức và thiết kế đăng nhập, đăng ký (1 tuần)

- **Công việc:** Tìm hiểu về công nghệ MERN stack, thiết kế cấu trúc cơ sở dữ liệu (vẽ ERD chi tiết cho Người dùng, Tin nhắn, Bạn bè, ..), tổ chức thư mục, kết nối cơ sở dữ liệu.

b. Xây dựng kiến trúc backend, frontend xử lý đăng ký, đăng nhập, đăng xuất (1 tuần)

- **Công việc:** Xây dựng API Backend và UI Frontend cho tính năng Đăng ký / Đăng nhập.

c. Xây dựng tính năng chat realtime phần backend (3 tuần)

- **Công việc:** Xây dựng API Backend cho các tính năng: Kết bạn, tạo phòng chat, và nhắn tin realtime, ….

d. Xây dựng tính năng chat realtime phần frontend (3 tuần)

- **Công việc:** Xây dựng giao diện Frontend cho các tính năng: Kết bạn, tạo phòng chat, và nhắn tin realtime, ….

e. Tối ưu hiệu năng và Trải nghiệm (2 tuần)

- **Công việc:** Tích hợp Redis để lưu trạng thái Online/Offline của user và cache các dữ liệu thường xuyên truy xuất. Bổ sung các tính năng phụ như thông báo (notifications), "đang gõ phím...", và tinh chỉnh UI/UX.

f. Tìm hiểu về AI, xây dựng kiến trúc phân loại độc hại, tích hợp (2 tuần).

- **Công việc:** Thu thập dữ liệu text, huấn luyện mô hình phân loại ngôn từ độc hại và viết API tích hợp mô hình đó vào luồng chat Node.js.

g. Deploy, kiểm thử, đóng gói (1 tuần)

- **Công việc:** Sửa bug nếu còn. Đưa ứng dụng lên môi trường Internet (Deploy Backend lên Render/AWS, Database lên MongoDB Atlas, Frontend lên Vercel). Viết báo cáo cuối kỳ.

TÀI LIỆU THAM KHẢO

[1] Khoa Nguyễn/Viblo. (2017). Những điều cần biết về MongoDB [Online]. Từ: https://viblo.asia/p/nhung-dieu-can-biet-ve-mongodb-ByEZkwnEZQ0

[2] IJISRT. (2024). Title of the Article [Online]. Từ: https://www.ijisrt.com/assets/upload/files/IJISRT24DEC729.pdf

[3] VNTalking. (2023). Giảm độ trễ trong hệ thống chat real-time bằng Redis Pub/Sub: Bí quyết tối ưu hóa [Online]. Từ: https://vntalking.com/giam-do-tre-trong-he-thong-chat-real-time-bang-redis-pub-sub-bi-quyet-toi-uu-hoa.html

[4] TopDev. (2023). Redis là gì? Ưu điểm và các trường hợp sử dụng Redis [Online]. Từ: https://topdev.vn/blog/redis-la-gi/

[5] Phạm Hoàng Anh/Viblo. (2019). Lý thuyết về mạng Bayes và ứng dụng vào bài toán lọc thư rác [Online]. Từ: https://viblo.asia/p/ly-thuyet-ve-mang-bayes-va-ung-dung-vao-bai-toan-loc-thu-rac-07LKXzkelV4

[6] PHẠM HIẾU/Viblo. (2020). Lập trình ứng dụng chat real-time với ReactJS, NodeJS và Socket.io [Online]. Từ: https://viblo.asia/p/lap-trinh-ung-dung-chat-real-time-voi-reactjs-nodejs-va-socketio-ORNZqVeMl0n

[7] Socket.io. (2024). Logging and Debugging [Online]. Từ: https://socket.io/docs/v4/logging-and-debugging/