



## **ĐỒ ÁN MÔN HỌC**

**Đề tài:**

# **XÂY DỰNG ỨNG DỤNG CHAT MESSAGING: NHẮN TIN NHÓM VÀ FILE**

Chuyên ngành: **KHOA HỌC DỮ LIỆU**

Giảng viên hướng dẫn : ThS. Nguyễn Hữu Trung

Sinh viên thực hiện :

2286400031 – Phạm Văn Thân

2286400037 – Phạm Ninh Thuận

2286400030 – Nguyễn Anh Thảo

Lớp: 22DKHA1

TP. Hồ Chí Minh, 2025



## **ĐỒ ÁN MÔN HỌC**

**Đề tài:**

# **XÂY DỰNG ỨNG DỤNG CHAT MESSAGING: NHẮN TIN NHÓM VÀ FILE**

Chuyên ngành: **KHOA HỌC DỮ LIỆU**

Giảng viên hướng dẫn : ThS. Nguyễn Hữu Trung

Sinh viên thực hiện :

2286400031 – Phạm Văn Thân

2286400037 – Phạm Ninh Thuận

2286400030 – Nguyễn Anh Thảo

Lớp: 22DKHA1

TP. Hồ Chí Minh, 2025

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

TP.HCM, Ngày....tháng....năm 2025

## **Giảng viên hướng dẫn**

(Ký tên, đóng dấu)

# LỜI CAM ĐOAN

Nhóm chúng tôi gồm Nguyễn Anh Thao, Phạm Văn Thành, Phạm Ninh Thuận xin cam đoan rằng toàn bộ nội dung được trình bày trong đồ án môn học này là kết quả của quá trình học tập, nghiên cứu và thực hiện nghiêm túc của bản thân dưới sự chỉ dạy của giảng viên môn học. Trong suốt quá trình thực hiện, chúng tôi đã cố gắng tiếp cận vấn đề một cách khoa học, trung thực và khách quan, đồng thời vận dụng các kiến thức đã học để phân tích và giải quyết bài toán một cách hiệu quả.

Báo cáo không sao chép bất kỳ nội dung nào từ các tài liệu, nguồn thông tin hoặc công trình nghiên cứu khác mà không trích dẫn rõ ràng. Tất cả dữ liệu, hình ảnh, bảng biểu và kết quả phân tích được sử dụng trong báo cáo đều có nguồn gốc rõ ràng và được thu thập, xử lý, phân tích bởi chính tôi hoặc thông qua các nguồn tài liệu hợp pháp, có dẫn chứng cụ thể.

Nhóm chúng tôi ý thức rõ tầm quan trọng của tính trung thực trong học thuật cũng như đạo đức nghề nghiệp. Do đó, chúng tôi xin hoàn toàn chịu trách nhiệm trước giảng viên đánh giá về mọi nội dung trình bày trong báo cáo này. Nếu phát hiện có bất kỳ hành vi gian lận, sao chép, đạo văn hoặc vi phạm quy định học thuật nào, chúng tôi xin chấp nhận mọi hình thức xử lý theo quy định mà không có bất kỳ khiếu nại nào.

Chúng tôi kính mong nhận được sự xem xét, góp ý từ giảng viên môn học để chúng tôi có thể hoàn thiện hơn về kiến thức chuyên môn và phương pháp nghiên cứu trong tương lai.

TP.HCM, Ngày 07 tháng 01 năm 2026

**Sinh viên thực hiện**  
(Ký và ghi rõ họ tên)

Phạm Văn Thành

Phạm Ninh Thuận

Nguyễn Anh Thao

# DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT VÀ TỪ KHÓA

---

## Ký hiệu/Viết tắt   Giải thích / Định nghĩa

---

<b>API</b>	Application Programming Interface / Giao diện lập trình ứng dụng
<b>BaaS</b>	Backend-as-a-Service / Mô hình dịch vụ phía máy chủ cung cấp sẵn
<b>CDN</b>	Content Delivery Network / Mạng phân phối nội dung giúp tải tệp nhanh
<b>FCM</b>	Firebase Cloud Messaging / Dịch vụ thông báo đẩy dự phòng
<b>JSON</b>	JavaScript Object Notation / Định dạng trao đổi dữ liệu văn bản nhẹ
<b>NoSQL</b>	Non-Relational / SQL Cơ sở dữ liệu phi quan hệ của Firestore
<b>UID</b>	User Identifier / Mã định danh duy nhất của người dùng trên Firebase
<b>UI/UX</b>	User Interface / User Experience Giao diện và Trải nghiệm người dùng
<b>URL</b>	Uniform Resource Locator Đường dẫn tài nguyên trên Internet
<b>Dart</b>	Ngôn ngữ lập trình mã nguồn mở do Google phát triển
<b>Flutter</b>	Framework mã nguồn mở hỗ trợ xây dựng ứng dụng đa nền tảng
<b>Firestore</b>	Cơ sở dữ liệu đám mây thời gian thực của Firebase
<b>Cloudinary</b>	Dịch vụ quản lý và lưu trữ tệp tin đa phương tiện trực tuyến

---

# MỤC LỤC

<b>LỜI CAM ĐOAN</b> . . . . .	iii
<b>DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT VÀ TỪ KHÓA</b> . . . . .	iv
<b>DANH SÁCH HÌNH VẼ</b> . . . . .	vii
<b>1 TỔNG QUAN VỀ ĐỀ TÀI</b> . . . . .	1
<b>1.1 Lý do chọn đề tài</b> . . . . .	1
<b>1.2 Mục tiêu của đề tài</b> . . . . .	1
1.2.1 Mục tiêu chính . . . . .	1
1.2.2 Mục tiêu phụ . . . . .	1
<b>1.3 Phạm vi nghiên cứu</b> . . . . .	2
1.3.1 Kiến trúc hệ thống . . . . .	2
1.3.2 Công nghệ phát triển . . . . .	2
1.3.3 Lưu trữ và xử lý dữ liệu . . . . .	2
1.3.4 Giới hạn phạm vi . . . . .	2
<b>1.4 Đối tượng nghiên cứu</b> . . . . .	3
1.4.1 Đối tượng thực thể . . . . .	3
1.4.2 Đối tượng kỹ thuật . . . . .	3
<b>2 CƠ SỞ LÝ THUYẾT</b> . . . . .	4
<b>2.1 Ngôn ngữ lập trình Dart</b> . . . . .	4
<b>2.2 Giới thiệu về Flutter</b> . . . . .	4
<b>2.3 Quản lý trạng thái trong Flutter với Provider</b> . . . . .	5
<b>2.4 Kiến trúc tổng thể ứng dụng</b> . . . . .	5
<b>2.5 Giao tiếp thời gian thực (Real-time Communication)</b> . . . . .	5
<b>2.6 Firebase và Cloud Firestore cho ứng dụng thời gian thực</b> . . . . .	6
<b>2.7 Firebase Authentication</b> . . . . .	6
<b>2.8 Mô hình dữ liệu NoSQL cho ứng dụng chat</b> . . . . .	6
<b>2.9 Bảo mật dữ liệu với Firestore Security Rules</b> . . . . .	6
<b>2.10 Quản lý tệp tin với Cloudinary</b> . . . . .	7
<b>2.11 Xử lý file và media trong Flutter</b> . . . . .	7
<b>2.12 Bảo mật tổng quát trong ứng dụng di động</b> . . . . .	7

<b>3</b>	<b>THIẾT KẾ VÀ TRIỂN KHAI CÁC TÍNH NĂNG CHÍNH . . . . .</b>	<b>8</b>
3.1	Tổng quan về các tính năng . . . . .	8
3.2	Xác thực và đăng ký người dùng . . . . .	8
3.3	Màn hình hồ sơ cá nhân (Profile) . . . . .	9
3.4	Danh sách bạn bè và nhắn tin cá nhân . . . . .	10
3.5	Quản lý nhóm chat . . . . .	11
3.6	Tương tác tin nhắn thời gian thực . . . . .	13
3.7	Chia sẻ tệp tin và hình ảnh . . . . .	14
<b>4</b>	<b>GIAO DIỆN NGƯỜI DÙNG VÀ TRẢI NGHIỆM . . . . .</b>	<b>16</b>
4.1	Tổng quan về thiết kế giao diện . . . . .	16
4.2	Hệ thống chủ đề và chế độ sáng/tối . . . . .	16
4.3	Hiệu ứng và animation . . . . .	16
4.4	Typography và bố cục . . . . .	17
4.5	Điều hướng và luồng người dùng . . . . .	17
4.6	Trải nghiệm tổng thể . . . . .	18
<b>5</b>	<b>KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN . . . . .</b>	<b>19</b>
5.1	Kết quả đạt được . . . . .	19
5.2	Hạn chế của ứng dụng . . . . .	19
5.3	Hướng phát triển trong tương lai . . . . .	20
5.4	Đóng góp của đề tài . . . . .	20
	<b>TÀI LIỆU THAM KHẢO . . . . .</b>	<b>22</b>

# DANH SÁCH HÌNH VẼ

3.1	Màn hình Đăng ký tài khoản . . . . .	8
3.2	Màn hình Đăng nhập . . . . .	9
3.3	Màn hình Cá nhân chế độ sáng . . . . .	10
3.4	Màn hình Cá nhân chế độ tối . . . . .	10
3.5	Danh sách bạn bè . . . . .	11
3.6	Menu quản lý trong danh sách bạn bè . . . . .	11
3.7	Màn hình tạo nhóm chat mới . . . . .	12
3.8	Danh sách nhóm chat . . . . .	12
3.9	Menu quản lý trong nhóm chat . . . . .	12
3.10	Màn hình chat với menu ghim/xóa tin nhắn . . . . .	13
3.11	Tin nhắn sau khi đã ghim . . . . .	14
3.12	Tin nhắn sau khi bị xóa . . . . .	14
3.13	Gửi ảnh và tệp tin bằng nút đính kèm trong chat . . . . .	15
3.14	Màn hình xem ảnh toàn màn hình với hiệu ứng zoom . . . . .	15

# CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

## 1.1 Lý do chọn đề tài

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ hiện nay, nhu cầu giao tiếp tức thời và chia sẻ thông tin nhanh chóng đã trở thành một phần thiết yếu trong đời sống cá nhân cũng như trong môi trường làm việc nhóm và học tập từ xa. Các ứng dụng nhắn tin phổ biến như Zalo, Facebook Messenger, WhatsApp hay Telegram đã đáp ứng tốt phần lớn nhu cầu này, tuy nhiên chúng thường là các dịch vụ tập trung, phụ thuộc vào nhà cung cấp lớn, dẫn đến một số hạn chế như vấn đề bảo mật quyền riêng tư dữ liệu, chi phí sử dụng nâng cao hoặc thiếu khả năng tùy chỉnh theo nhu cầu riêng của tổ chức và doanh nghiệp [1].

Việc tự nghiên cứu và xây dựng một ứng dụng nhắn tin nhóm với tính năng chia sẻ tệp tin mang lại giá trị lớn cho người thực hiện khi giúp củng cố và nắm vững các công nghệ hiện đại như phát triển ứng dụng đa nền tảng bằng Flutter [1], xây dựng hệ thống giao tiếp thời gian thực sử dụng Firebase Firestore [2], quản lý tệp tin đa phương tiện trên đám mây thông qua Cloudinary [3], cùng các kỹ thuật quản lý trạng thái và thiết kế giao diện người dùng hiện đại. Đề tài mang tính thực tiễn cao, có thể áp dụng trực tiếp vào các hệ thống nhắn tin nội bộ của doanh nghiệp nhỏ, nhóm dự án, lớp học hoặc cộng đồng cần kiểm soát tốt dữ liệu cá nhân và không muốn phụ thuộc hoàn toàn vào các nền tảng thương mại. Chính vì những lý do trên, đề tài “Xây dựng ứng dụng chat messaging: Nhắn tin nhóm và chia sẻ tệp tin” được lựa chọn để thực hiện trong môn học.

## 1.2 Mục tiêu của đề tài

### 1.2.1 Mục tiêu chính

Mục tiêu chính của đề tài là xây dựng một ứng dụng nhắn tin hoàn chỉnh trên nền tảng di động với các tính năng cốt lõi bao gồm hỗ trợ nhắn tin thời gian thực cho cả chat cá nhân và nhóm, cho phép chia sẻ đa dạng định dạng tệp tin như hình ảnh, tài liệu PDF, ZIP hay video ngắn, đồng thời đảm bảo xác thực người dùng an toàn và mang lại trải nghiệm người dùng tốt cùng giao diện thân thiện.

### 1.2.2 Mục tiêu phụ

Ngoài mục tiêu chính, đề tài còn hướng tới việc áp dụng Flutter để phát triển ứng dụng chạy được trên cả Android và iOS từ một codebase duy nhất, tích hợp Firebase Authentication và

Cloud Firestore nhằm xử lý xác thực cũng như đồng bộ dữ liệu thời gian thực [4], sử dụng Cloudinary làm dịch vụ lưu trữ tệp tin để tối ưu hiệu suất và chi phí , triển khai giao diện hiện đại hỗ trợ chế độ sáng/tối (Dark Mode) cùng các hiệu ứng chuyển cảnh mượt mà nhằm dễ sử dụng cho người dùng mới, và cuối cùng là đánh giá hiệu suất ứng dụng qua các trường hợp thử nghiệm thực tế như đo lường độ trễ tin nhắn hay khả năng chịu tải đồng thời.

## 1.3 Phạm vi nghiên cứu

### 1.3.1 Kiến trúc hệ thống

Ứng dụng được thiết kế theo mô hình Client-Server trong đó phần client là ứng dụng Flutter và backend tận dụng các dịch vụ Backend-as-a-Service (BaaS) của Firebase bao gồm Authentication và Firestore, kết hợp với Cloudinary để lưu trữ tệp tin [5].

### 1.3.2 Công nghệ phát triển

Công nghệ phát triển được lựa chọn bao gồm ngôn ngữ lập trình Dart, framework frontend Flutter, quản lý trạng thái bằng Provider [6], backend dựa trên Firebase Authentication và Cloud Firestore, lưu trữ tệp tin qua Cloudinary Public API, cùng các package hỗ trợ như image\_picker, file\_picker, uuid và intl.

### 1.3.3 Lưu trữ và xử lý dữ liệu

Dữ liệu người dùng, nhóm chat và lịch sử tin nhắn được lưu trữ trên Cloud Firestore dưới dạng các collection như users, groups, chats và messages. Tệp tin được upload lên Cloudinary và chỉ lưu URL tham chiếu trong Firestore nhằm tối ưu băng thông cũng như tốc độ truy xuất.

### 1.3.4 Giới hạn phạm vi

Phạm vi nghiên cứu tập trung vào triển khai trên nền tảng di động Android và iOS, chưa hỗ trợ Flutter Web hoặc Desktop. Các tính năng nâng cao như gọi thoại/video call, mã hóa tin nhắn end-to-end phức tạp hay thông báo đẩy thời gian thực đầy đủ chưa được thực hiện. Push Notification thông qua Firebase Cloud Messaging chỉ đóng vai trò dự phòng, kích hoạt để gửi thông báo lên thanh trạng thái khi người dùng đã thoát hoặc chạy ngầm ứng dụng nhằm tránh bỏ lỡ tin nhắn, trong khi cơ chế chính vẫn là cập nhật dữ liệu trực tiếp qua Stream thời gian thực để hiển thị tin nhắn ngay lập tức khi người dùng đang mở ứng dụng, giúp tối ưu tốc độ và tránh thông báo trùng lặp. Quy mô thử nghiệm giới hạn ở nhóm nhỏ, và đề tài không đi sâu vào tối ưu hóa offline-first hay xử lý xung đột dữ liệu phức tạp.

## **1.4 Đối tượng nghiên cứu**

### *1.4.1 Đối tượng thực thể<sup>2</sup>*

Đối tượng thực thể bao gồm người dùng cá nhân có nhu cầu nhắn tin nhanh chóng và chia sẻ tệp tin hàng ngày, cùng với các nhóm làm việc nhỏ, nhóm dự án, lớp học trực tuyến hoặc cộng đồng cần một công cụ nhắn tin nội bộ bảo mật mà không phụ thuộc vào các nền tảng thương mại lớn.

### *1.4.2 Đối tượng kỹ thuật*

Đối tượng kỹ thuật bao gồm cơ chế đồng bộ dữ liệu thời gian thực sử dụng Firestore Streams và StreamBuilder [?], quy trình upload và quản lý tệp tin đa phương tiện trên Cloudinary [?], các kỹ thuật xác thực người dùng và bảo vệ dữ liệu cơ bản bằng Firebase Authentication, cùng thiết kế mô hình dữ liệu NoSQL phù hợp với bài toán chat nhóm.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1 Ngôn ngữ lập trình Dart

Dart là ngôn ngữ lập trình mã nguồn mở do Google phát triển, được thiết kế đặc biệt để xây dựng các ứng dụng client-side đa nền tảng một cách hiệu quả [7]. Với đặc tính hướng đối tượng kết hợp hỗ trợ lập trình chức năng, Dart sở hữu cú pháp đơn giản, dễ đọc và gần gũi với các ngôn ngữ phổ biến như Java, JavaScript hay C.

Những ưu điểm nổi bật của Dart khi kết hợp với Flutter bao gồm tính năng sound null safety giúp phát hiện lỗi null ngay tại thời điểm biên dịch, giảm đáng kể lỗi runtime; hỗ trợ đồng thời JIT cho hot reload nhanh trong quá trình phát triển và AOT để biên dịch thành mã máy đạt hiệu suất gần native khi triển khai; cơ chế hot reload cho phép cập nhật giao diện tức thì mà không mất trạng thái ứng dụng yếu tố quan trọng nâng cao năng suất lập trình viên; cùng với hệ thống kiểu dữ liệu mạnh mẽ, async/await native và generics hiện đại giúp mã nguồn dễ bảo trì và ít lỗi hơn. Trong ứng dụng chat, async/await được sử dụng rộng rãi để xử lý các tác vụ bắt đồng bộ như upload tệp tin hoặc lấy dữ liệu mà không làm gián đoạn giao diện người dùng. Dart được chọn làm ngôn ngữ chính cho Flutter nhờ khả năng cân bằng hoàn hảo giữa tốc độ phát triển và hiệu suất cuối cùng trên nhiều nền tảng [1].

## 2.2 Giới thiệu về Flutter

Flutter là framework mã nguồn mở do Google phát triển, cho phép lập trình viên xây dựng ứng dụng đa nền tảng từ một codebase duy nhất, hỗ trợ Android, iOS, Web, Windows, macOS và Linux [1]. Flutter sử dụng Dart làm ngôn ngữ lập trình và sở hữu cơ chế render riêng thông qua engine Skia (và Impeller trên một số nền tảng mới), nhờ đó giao diện người dùng được vẽ trực tiếp mà không phụ thuộc vào các thành phần native của hệ điều hành, mang lại trải nghiệm mượt mà và nhất quán trên mọi thiết bị.

Flutter nổi bật với hiệu suất cao khi render trực tiếp giao diện mà không cần bridge trung gian, tính năng hot reload giúp cập nhật giao diện gần như tức thì trong quá trình phát triển, thư viện widget phong phú hỗ trợ cả Material Design của Android và Cupertino của iOS, đồng thời giúp tiết kiệm đáng kể chi phí và thời gian so với việc phát triển ứng dụng native riêng lẻ cho từng nền tảng [8]. Trong đề tài này, Flutter được lựa chọn làm framework chính để xây dựng giao diện người dùng và xử lý logic phía client cho ứng dụng nhắn tin thời gian thực.

## 2.3 Quản lý trạng thái trong Flutter với Provider

Provider là một trong những package quản lý trạng thái được đội ngũ Flutter khuyên nghị chính thức, hoạt động dựa trên cơ chế InheritedWidget để truyền dữ liệu xuống cây widget một cách hiệu quả mà không cần truyền tham số thủ công qua nhiều lớp [6]. Provider thường được kết hợp với ChangeNotifier để theo dõi và thông báo sự thay đổi trạng thái, chẳng hạn như chuyển đổi theme sáng/tối hoặc cập nhật thông tin người dùng.

Package này mang lại sự đơn giản trong sử dụng, hiệu suất cao nhờ chỉ rebuild những widget thực sự cần thiết, và đặc biệt phù hợp với các ứng dụng có quy mô từ nhỏ đến trung bình như ứng dụng chat hiện tại [9]. Trong đề tài, Provider được áp dụng để quản lý theme toàn cục và các trạng thái chung khác của ứng dụng.

## 2.4 Kiến trúc tổng thể ứng dụng

Ứng dụng được thiết kế theo mô hình kiến trúc phân lớp (layered architecture) nhằm đảm bảo tính tách biệt và dễ bảo trì. Lớp trình bày (Presentation Layer) bao gồm các widget và màn hình Flutter chịu trách nhiệm hiển thị giao diện và tiếp nhận tương tác từ người dùng. Lớp logic nghiệp vụ (Business Logic Layer) sử dụng Provider kết hợp với các service class để quản lý trạng thái và xử lý các quy tắc nghiệp vụ như tạo chatId hay xử lý tin nhắn. Lớp dữ liệu (Data Layer) bao gồm Firebase để xử lý dữ liệu thời gian thực và Cloudinary để lưu trữ tệp tin đa phương tiện. Kiến trúc này giúp mã nguồn rõ ràng, dễ kiểm thử và thuận tiện cho việc mở rộng trong tương lai.

## 2.5 Giao tiếp thời gian thực (Real-time Communication)

Giao tiếp thời gian thực là yếu tố cốt lõi của mọi ứng dụng chat, cho phép tin nhắn được đồng bộ gần như tức thì giữa các thiết bị mà không cần người dùng phải làm mới thủ công. Trong Firebase Firestore, cơ chế này được thực hiện thông qua Streams – một luồng dữ liệu liên tục lắng nghe mọi thay đổi trên database [10].

Về mặt lý thuyết, Firestore sử dụng WebSockets ở tầng dưới để duy trì kết nối hai chiều giữa client và server, kết hợp với StreamBuilder trong Flutter để tự động rebuild giao diện ngay khi có dữ liệu mới, chẳng hạn khi một tin nhắn vừa được gửi đến. Ưu điểm lớn nhất là độ trễ rất thấp (thường dưới 1 giây), hỗ trợ hoạt động offline và tự động đồng bộ khi kết nối trở lại. So với các giải pháp như Socket.io, Firestore Streams linh hoạt hơn nhờ tích hợp sẵn khả năng query dữ liệu phức tạp, dù Socket.io có thể phù hợp hơn với các ứng dụng cần backend tùy biến hoàn toàn [11]. Trong đề tài, Streams được sử dụng để đồng bộ tin nhắn và trạng thái online của người dùng.

## 2.6 Firebase và Cloud Firestore cho ứng dụng thời gian thực

Firebase là nền tảng Backend-as-a-Service (BaaS) do Google cung cấp, bao gồm nhiều dịch vụ hữu ích như Authentication, Cloud Firestore và Cloud Messaging [2]. Cloud Firestore là cơ sở dữ liệu NoSQL dạng document-oriented, nổi bật với khả năng đồng bộ thời gian thực qua streams, hỗ trợ lưu trữ offline và thực hiện các truy vấn phức tạp [4].

So với Realtime Database trước đây, Firestore phù hợp hơn hẳn cho ứng dụng chat nhờ hỗ trợ query mạnh mẽ (lọc, sắp xếp, phân trang), khả năng mở rộng cao hơn và cơ chế offline persistence ổn định [12]. Trong đề tài, Firestore được sử dụng để lưu trữ thông tin người dùng, nhóm chat, tin nhắn và metadata của tệp tin.

## 2.7 Firebase Authentication

Firebase Authentication cung cấp dịch vụ xác thực người dùng an toàn và dễ tích hợp, hỗ trợ nhiều phương thức như email/password và OAuth từ các nhà cung cấp bên thứ ba [2]. Trong ứng dụng chat, dịch vụ này được sử dụng để quản lý quá trình đăng ký và đăng nhập, tạo UID duy nhất làm cơ sở phân quyền, đồng thời kết hợp chặt chẽ với Firestore Security Rules để kiểm soát truy cập dữ liệu. Nhờ đó, hệ thống đảm bảo được tính bảo mật cơ bản và nhất quán trong việc xác định danh tính người gửi tin nhắn.

## 2.8 Mô hình dữ liệu NoSQL cho ứng dụng chat

Firestore tổ chức dữ liệu theo mô hình collection và document, rất phù hợp với cấu trúc phi quan hệ của ứng dụng chat. Thông tin người dùng được lưu trong collection `users` với các trường `UID`, `email` và đường dẫn ảnh đại diện. Thông tin nhóm chat nằm trong collection `groups` bao gồm danh sách thành viên và ID quản trị viên. Tin nhắn được lưu dưới dạng subcollection `messages` của từng phòng chat hoặc nhóm, kèm theo timestamp để sắp xếp theo thời gian và các trường như `text`, `fileUrl`, `senderId` [13]. StreamBuilder trong Flutter được sử dụng để lắng nghe mọi thay đổi và cập nhật giao diện thời gian thực.

## 2.9 Bảo mật dữ liệu với Firestore Security Rules

Firestore Security Rules cho phép định nghĩa quy tắc truy cập dữ liệu ngay tại phía server, đảm bảo chỉ những người dùng đã được xác thực mới có quyền đọc hoặc ghi dữ liệu phù hợp [4]. Đối với ứng dụng chat, các best practices bao gồm kiểm tra `UID` của người dùng có nằm trong mảng `members` của phòng chat trước khi cho phép truy cập, ngăn chặn việc đọc toàn bộ database và chỉ cho phép truy vấn subcollection cụ thể của từng phòng, đồng thời sử dụng custom claims để phân quyền admin nếu cần [14]. Cách tiếp cận này giúp bảo vệ quyền riêng

tư và ngăn chặn truy cập trái phép.

## 2.10 Quản lý tệp tin với Cloudinary

Cloudinary là dịch vụ quản lý media chuyên dụng, cung cấp khả năng upload, tự động tối ưu hóa, transformation động và phân phối nội dung qua CDN toàn cầu [3]. So với Firebase Storage, Cloudinary nổi bật nhờ khả năng tự động chuyển đổi định dạng (ví dụ sang WebP), resize, crop hoặc thêm watermark, cùng với tier miễn phí hào phóng hơn dành cho các ứng dụng media [15]. Trong đề tài, Cloudinary được sử dụng để upload ảnh đại diện và các tệp tin chia sẻ từ người dùng, sau đó chỉ lưu URL bảo mật vào Firestore nhằm tối ưu hiệu suất và chi phí [5].

## 2.11 Xử lý file và media trong Flutter

Trong môi trường Flutter, việc xử lý file và media được thực hiện thông qua các package chuyên dụng như `image_picker` để chọn ảnh từ thư viện hoặc camera và `file_picker` để chọn các định dạng tệp tin đa dạng [16]. Các tác vụ upload được thực hiện bắt đồng bộ để tránh làm gián đoạn giao diện người dùng, thường sử dụng `MultipartRequest` hỗ trợ chunking cho tệp lớn. Best practices bao gồm nén tệp trước khi upload để tiết kiệm băng thông, xử lý lỗi với cơ chế retry tự động và kiểm tra quyền truy cập bộ nhớ trước khi thực hiện thao tác. Trong đề tài, các package này được kết hợp chặt chẽ với Cloudinary để đảm bảo quá trình chia sẻ tệp tin diễn ra an toàn và nhanh chóng.

## 2.12 Bảo mật tổng quát trong ứng dụng di động

Bảo mật ứng dụng di động cần tuân thủ các nguyên tắc trong OWASP Mobile Top 10, bao gồm xác thực mạnh mẽ thông qua Firebase Authentication, không lưu trữ thông tin nhạy cảm như mật khẩu trên thiết bị, phòng chống các dạng tấn công phổ biến như injection trong cơ sở dữ liệu NoSQL hay XSS qua nội dung tin nhắn, đồng thời luôn sử dụng HTTPS cho mọi kết nối API [17]. Trong đề tài, các biện pháp này được áp dụng toàn diện để bảo vệ dữ liệu tin nhắn và tệp tin của người dùng.

# CHƯƠNG 3: THIẾT KẾ VÀ TRIỂN KHAI CÁC TÍNH NĂNG CHÍNH

## 3.1 Tổng quan về các tính năng

Ứng dụng được thiết kế với các tính năng cốt lõi của một hệ thống nhắn tin hiện đại: xác thực người dùng, quản lý hồ sơ cá nhân, nhắn tin cá nhân và nhóm, chia sẻ tệp tin đa định dạng, cùng các tính năng hỗ trợ như ghim tin nhắn và xóa tin nhắn. Tất cả các tính năng đều hoạt động thời gian thực nhờ tích hợp Cloud Firestore và lưu trữ tệp trên Cloudinary.

## 3.2 Xác thực và đăng ký người dùng

Ứng dụng cung cấp hai màn hình chính cho việc xác thực.

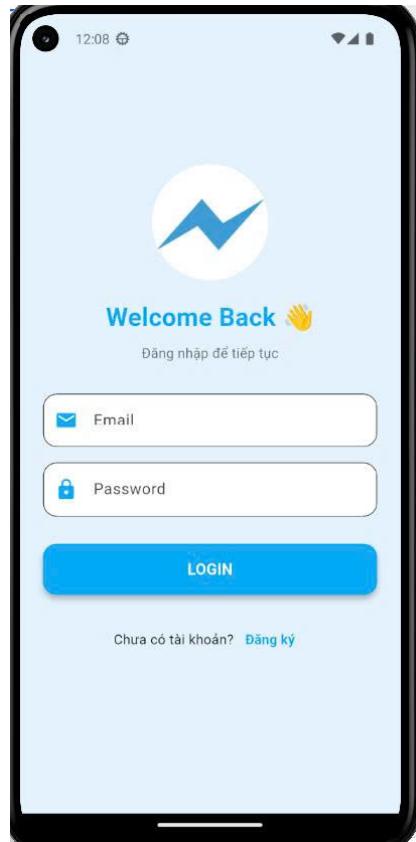
Màn hình Đăng ký cho phép tạo tài khoản mới với đầy đủ thông tin: họ tên, email, mật khẩu (xác nhận lại) và đặc biệt là chọn ảnh đại diện ngay từ bước đăng ký. Ảnh đại diện được tải lên Cloudinary ngay sau khi tạo tài khoản thành công và lưu URL vào Firestore.



Hình 3.1: Màn hình Đăng ký tài khoản

Quá trình xác thực sử dụng Firebase Authentication với phương thức email/password, đảm bảo bảo mật cơ bản và tốc độ xử lý nhanh.

Màn hình Đăng nhập có giao diện đơn giản, hiện đại với logo đặc trưng và thông điệp chào mừng. Người dùng chỉ cần nhập email và mật khẩu mà mình đã đăng ký để truy cập.

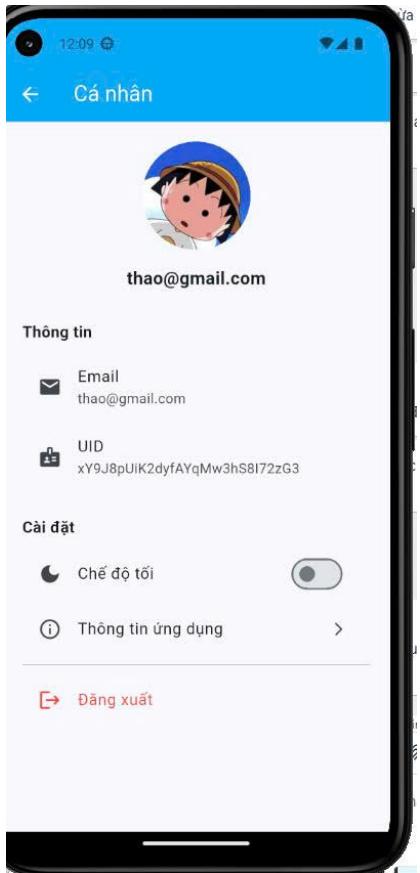


Hình 3.2: Màn hình Đăng nhập

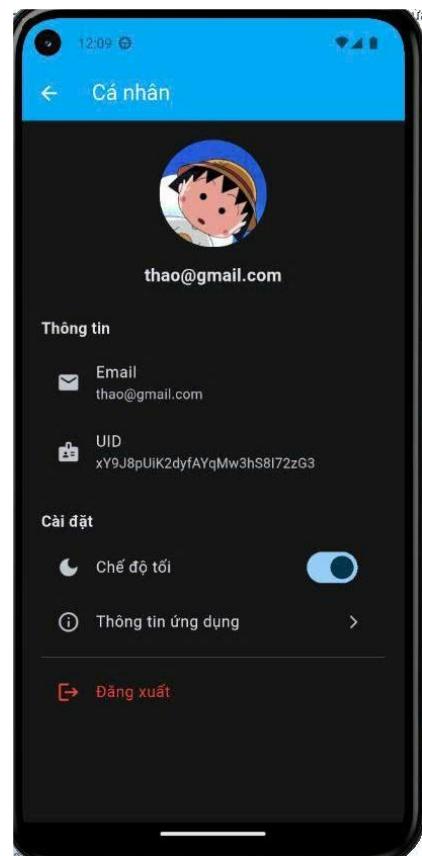
### 3.3 Màn hình hồ sơ cá nhân (Profile)

Sau khi đăng nhập, người dùng có thể truy cập màn hình cá nhân để xem và quản lý thông tin của mình.

Màn hình này hiển thị ảnh đại diện lớn, tên hiển thị (email làm tên tạm thời), thông tin chi tiết gồm email và UID từ Firebase Auth, công tắc chuyển đổi chế độ tối (Dark Mode), nút xem thông tin ứng dụng và đăng xuất.



Hình 3.3: Màn hình Cá nhân chế độ sáng



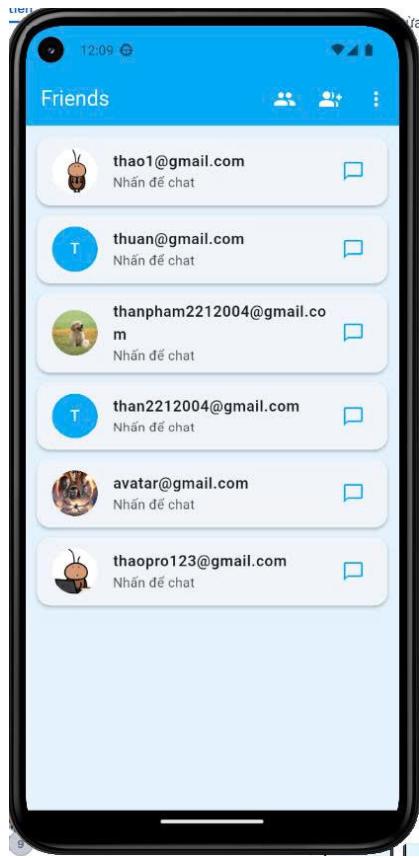
Hình 3.4: Màn hình Cá nhân chế độ tối

Ứng dụng hỗ trợ cả hai chế độ sáng và tối, với giao diện tự động điều chỉnh màu nền, chữ và icon.

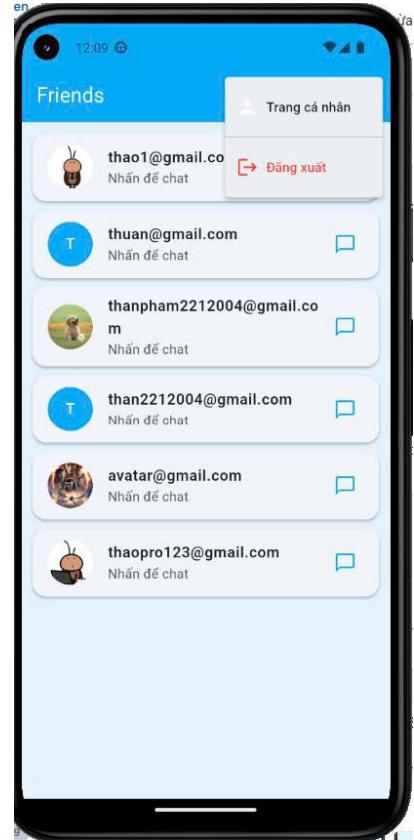
### 3.4 Danh sách bạn bè và nhắn tin cá nhân

Màn hình Friends hiển thị danh sách tất cả người dùng đã đăng ký trong hệ thống, hoạt động như một danh bạ.

Mỗi mục bạn bè bao gồm ảnh đại diện, email (làm tên hiển thị), trạng thái online/offline và nút chat nhanh. Khi nhấn vào một người dùng, hệ thống tự động tạo hoặc mở hội thoại cá nhân với chatId được sinh duy nhất từ việc sắp xếp và nối hai UID.



Hình 3.5: Danh sách bạn bè



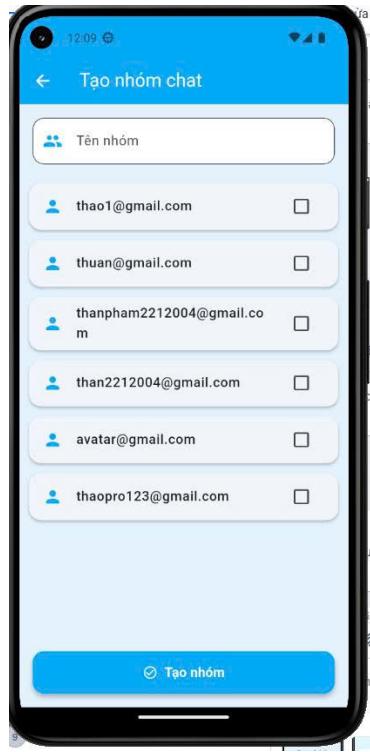
Hình 3.6: Menu quản lý trong danh sách bạn bè

Người dùng cũng có thể thực hiện các thao tác quản lý như xem trang cá nhân hoặc đăng xuất từ menu góc trên.

### 3.5 Quản lý nhóm chat

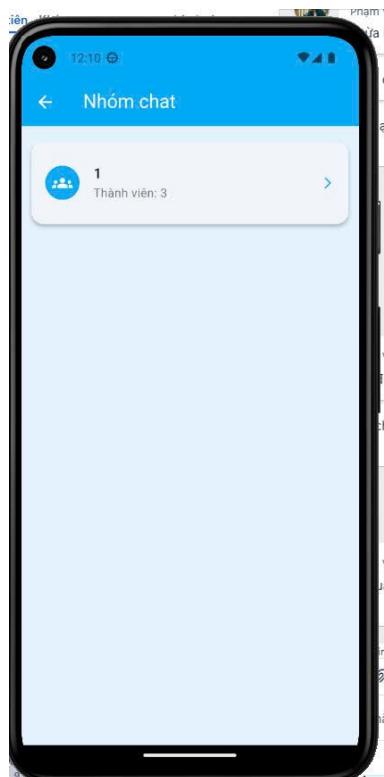
Ứng dụng hỗ trợ đầy đủ tính năng nhóm chat.

**Tạo nhóm mới:** Người dùng nhập tên nhóm và chọn thành viên từ danh sách bạn bè thông qua checkbox. Sau khi nhấn "Tạo nhóm", người tạo tự động trở thành quản trị viên (admin).

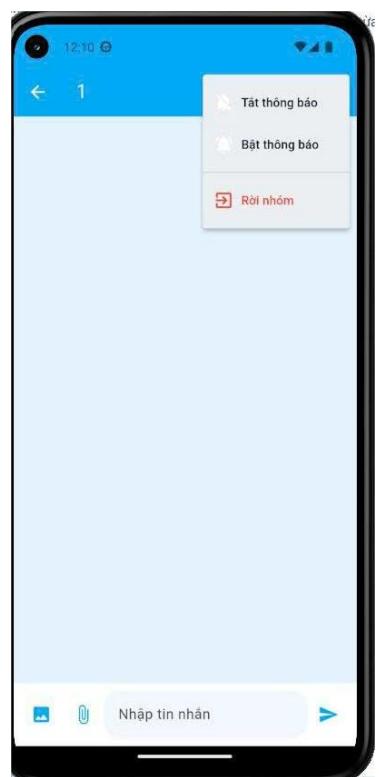


Hình 3.7: Màn hình tạo nhóm chat mới

**Danh sách nhóm :** Hiển thị các nhóm đã tham gia với số lượng thành viên. Trong nhóm, người dùng có thể tắt/bật thông báo, rời nhóm hoặc thực hiện các thao tác quản lý khác.



Hình 3.8: Danh sách nhóm chat

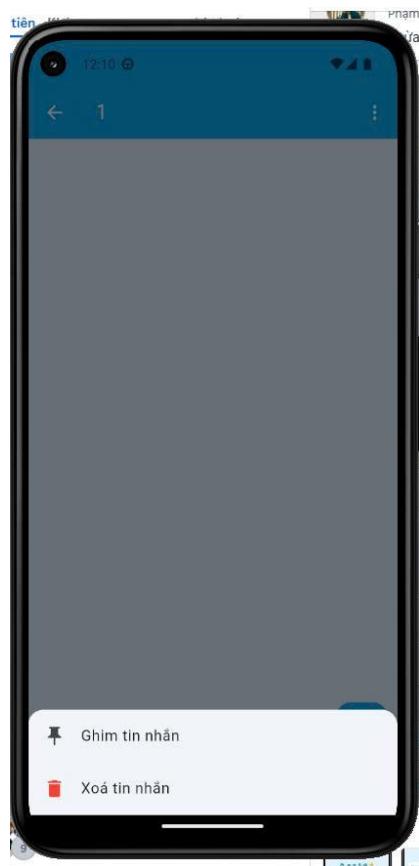


Hình 3.9: Menu quản lý trong nhóm chat

### 3.6 Tương tác tin nhắn thời gian thực

Màn hình chat cả cá nhân và nhóm đều hoạt động hoàn toàn thời gian thực nhờ Firestore Streams.

Các tính năng chính bao gồm hiển thị tin nhắn theo thứ tự thời gian, hỗ trợ ghim tin nhắn quan trọng lên đầu hội thoại, xóa tin nhắn với thông báo "Tin nhắn đã bị xoá", và menu ngữ cảnh khi nhấp giữ tin nhắn.



Hình 3.10: Màn hình chat với menu ghim/xóa tin nhắn



Hình 3.11: Tin nhắn sau khi đã ghim



Hình 3.12: Tin nhắn sau khi bị xóa

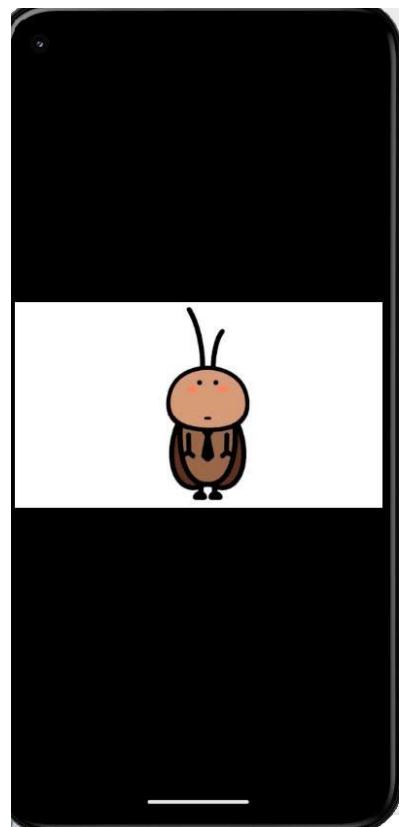
### 3.7 Chia sẻ tệp tin và hình ảnh

Ứng dụng hỗ trợ gửi đa dạng định dạng tệp tin thông qua nút đính kèm, bao gồm hình ảnh (hình nhỏ đại diện cho nội dung chính và xem full-screen với hiệu ứng zoom), tài liệu PDF, file nén ZIP và các định dạng khác.

Quy trình xử lý: tệp được chọn từ thiết bị, upload lên Cloudinary, lưu URL vào tin nhắn trong Firestore và người nhận có thể tải về hoặc xem trực tiếp.



Hình 3.13: Gửi ảnh và tệp tin bằng nút đính kèm trong chat



Hình 3.14: Màn hình xem ảnh toàn màn hình với hiệu ứng zoom

# CHƯƠNG 4: GIAO DIỆN NGƯỜI DÙNG VÀ TRẢI NGHIỆM

## 4.1 Tổng quan về thiết kế giao diện

Giao diện ứng dụng được xây dựng dựa trên nguyên tắc Material Design 3, sử dụng tông màu chủ đạo là xanh lightBlue kết hợp với các màu trung tính trắng/xám. Thiết kế tập trung vào sự đơn giản, nhất quán và dễ sử dụng, đảm bảo người dùng có thể nhanh chóng làm quen và thao tác mà không gặp khó khăn. Các thành phần giao diện như button, card, text field và icon đều tuân thủ chuẩn Material 3, mang lại vẻ ngoài hiện đại và chuyên nghiệp.

## 4.2 Hệ thống chủ đề và chế độ sáng/tối

Ứng dụng hỗ trợ đầy đủ hai chế độ hiển thị: sáng và tối. Người dùng có thể chuyển đổi chế độ này thông qua công tắc trong màn hình Cá nhân.

Khi ở chế độ tối:

- Nền chính chuyển sang màu đen đậm
- Các surface (card, dialog) sử dụng màu xám tối
- Màu chữ chuyển sang trắng hoặc xám nhạt để đảm bảo độ tương phản cao
- Màu accent xanh chủ đạo được giữ nguyên nhưng tăng nhẹ độ sáng để nổi bật trên nền tối

Việc chuyển đổi theme được thực hiện tức thì trên toàn ứng dụng mà không cần tải lại, mang lại trải nghiệm mượt mà và thoải mái cho người dùng trong các điều kiện ánh sáng khác nhau.

## 4.3 Hiệu ứng và animation

Để tăng tính tương tác và cảm giác sống động, ứng dụng tích hợp nhiều hiệu ứng tinh tế:

- **Hero Animation:** Ảnh đại diện hoặc hình ảnh trong tin nhắn chuyển cảnh mượt khi mở xem chi tiết full-screen
- **Pinch-to-Zoom và Pan:** Trong màn hình xem ảnh, người dùng có thể phóng to, thu nhỏ và di chuyển ảnh bằng cử chỉ tự nhiên

- **Ripple Effect:** Hiệu ứng lan tỏa khi nhấn vào button hoặc card
- **Fade và Slide Transition:** Chuyển cảnh giữa các màn hình với hiệu ứng mờ dần và trượt nhẹ
- **CircularProgressIndicator:** Hiển thị trạng thái loading khi thực hiện các tác vụ nền như đăng nhập, upload tệp

Các hiệu ứng này không chỉ nâng cao tính thẩm mỹ mà còn giúp người dùng nhận biết rõ ràng trạng thái và hành động đang diễn ra.

## 4.4 Typography và bố cục

Hệ thống typography tuân thủ chuẩn Material 3 với font Roboto mặc định:

- Tiêu đề màn hình: Display Medium hoặc Large
- Tên người dùng, tên nhóm: Title Medium
- Nội dung tin nhắn: Body Large
- Thời gian gửi tin nhắn và thông tin phụ: Body Small hoặc Caption

Bố cục được thiết kế theo lối 8dp, với padding và margin hợp lý, đảm bảo khoảng cách thoáng và dễ đọc. Các card được sử dụng rộng rãi để nhóm thông tin, tạo cảm giác phân tầng rõ ràng. Nội dung chính luôn được căn chỉnh phù hợp (left-aligned cho tin nhắn, center cho logo và thông điệp chào mừng).

## 4.5 Điều hướng và luồng người dùng

Hệ thống điều hướng được thiết kế đơn giản, trực quan với các luồng chính:

- Splash Screen kiểm tra trạng thái đăng nhập đến chuyển tự động đến màn hình phù hợp
- Từ Login/Register đến Home (danh sách hội thoại)
- Truy cập nhanh các phần chính: Friends, Groups, Profile qua menu hoặc icon
- Nút back hệ thống hoạt động nhất quán trên mọi màn hình
- Sử dụng BottomSheet và PopupMenu cho các hành động phụ (quản lý nhóm, menu tin nhắn)

## 4.6 Trải nghiệm tổng thể

Ứng dụng mang lại trải nghiệm người dùng tốt nhờ:

- Kích thước button và khu vực chạm đủ lớn, phù hợp thao tác bằng ngón tay
- Phản hồi nhanh chóng với mọi thao tác nhờ Flutter engine
- Thông báo lỗi rõ ràng qua Snackbar khi có sự cố mạng hoặc upload thất bại
- Giao diện nhất quán trên mọi màn hình, giúp người dùng mới dễ dàng làm quen

Tổng thể, thiết kế tập trung vào sự tiện lợi, thẩm mỹ và hiệu suất, đáp ứng tốt nhu cầu sử dụng hàng ngày của một ứng dụng nhắn tin hiện đại.

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1 Kết quả đạt được

Đề tài đã hoàn thành việc xây dựng một ứng dụng nhắn tin nhóm và chia sẻ tệp tin hoàn chỉnh trên nền tảng Flutter với các tính năng chính sau:

- **Xác thực người dùng:** Đăng nhập và đăng ký bằng email/password thông qua Firebase Authentication, kèm theo khả năng tải lên ảnh đại diện ngay khi tạo tài khoản.
- **Quản lý hồ sơ cá nhân:** Hiển thị thông tin chi tiết (email, UID), hỗ trợ chuyển đổi chế độ sáng/tối và đăng xuất an toàn.
- **Nhắn tin thời gian thực:** Hỗ trợ chat cá nhân và nhóm với đồng bộ tức thì nhờ Cloud Firestore Streams.
- **Quản lý nhóm:** Tạo nhóm mới, chọn thành viên, hiển thị danh sách nhóm, quyền quản trị cơ bản (rời nhóm, tắt thông báo).
- **Tương tác tin nhắn:** Ghim tin nhắn quan trọng, xóa tin nhắn (soft delete) với thông báo rõ ràng.
- **Chia sẻ tệp tin:** Gửi đa định dạng (hình ảnh, PDF, ZIP, v.v.) qua Cloudinary, hiển thị thumbnail và xem ảnh full-screen với hiệu ứng zoom.
- **Giao diện người dùng:** Thiết kế hiện đại theo Material Design 3, hỗ trợ đầy đủ Light/Dark mode, các hiệu ứng animation mượt mà và bố cục responsive.

Ứng dụng hoạt động ổn định trên thiết bị di động, độ trễ tin nhắn thấp, quá trình upload/download tệp nhanh chóng và giao diện thân thiện, dễ sử dụng.

## 5.2 Hạn chế của ứng dụng

Mặc dù đã đạt được các mục tiêu đề ra, ứng dụng vẫn tồn tại một số hạn chế:

- Chưa triển khai thông báo đẩy (Push Notification) bằng Firebase Cloud Messaging khi có tin nhắn mới.

- Không hỗ trợ gọi thoại hoặc video call.
- Chưa có cơ chế mã hóa end-to-end nâng cao cho tin nhắn.
- Chức năng tìm kiếm tin nhắn hoặc người dùng trong danh sách còn hạn chế.
- Quy mô thử nghiệm chỉ ở mức nhỏ nhóm dưới 50 người dùng đồng thời, chưa kiểm tra khả năng mở rộng với hàng nghìn người dùng.
- Chưa hỗ trợ nền tảng web hoặc desktop (Flutter Web/Desktop).

### **5.3 Hướng phát triển trong tương lai**

Dựa trên các hạn chế hiện tại, ứng dụng có thể được mở rộng và cải thiện theo các hướng sau:

- Tích hợp Firebase Cloud Messaging để gửi thông báo đẩy khi nhận tin nhắn mới hoặc có thành viên mới trong nhóm.
- Thêm tính năng gọi thoại và video call sử dụng các SDK như Agora hoặc WebRTC.
- Triển khai mã hóa end-to-end cho tin nhắn bằng thư viện encrypt hoặc signal protocol.
- Phát triển phiên bản Flutter Web để sử dụng trên trình duyệt và Flutter Desktop cho Windows/macOS/Linux.
- Thêm chức năng tìm kiếm tin nhắn, lọc danh sách bạn bè, và online status chính xác hơn.
- Xây dựng trang quản trị (admin panel) để quản lý người dùng và nhóm từ phía server.
- Tối ưu hiệu suất cho quy mô lớn bằng cách sử dụng sharding Firestore hoặc chuyển sang backend tự xây (Node.js + Socket.io).

### **5.4 Đóng góp của đề tài**

Đề tài đã mang lại những giá trị sau:

- Cung cấp một sản phẩm thực tế ứng dụng nhắn tin nhóm hoàn chỉnh có thể sử dụng ngay trong các nhóm làm việc nội bộ hoặc học tập.
- Giúp người thực hiện nắm vững quy trình phát triển ứng dụng đa nền tảng bằng Flutter kết hợp Firebase và Cloudinary.
- Áp dụng thành công các kỹ thuật hiện đại: quản lý trạng thái với Provider, đồng bộ dữ liệu thời gian thực, lưu trữ media đám mây, thiết kế theo Material 3.

- Làm tài liệu tham khảo hữu ích cho các đề tài tương tự về phát triển ứng dụng chat trên Flutter.

Tổng thể, đề tài đã đạt được mục tiêu đề ra, tạo nền tảng vững chắc để tiếp tục phát triển thành một ứng dụng nhắn tin chuyên nghiệp hơn trong tương lai.

# TÀI LIỆU THAM KHẢO

## Tài liệu tham khảo

- [1] Google. Flutter documentation, 2025.
- [2] Google. Firebase documentation, 2025.
- [3] Cloudinary. Flutter sdk documentation, 2025.
- [4] Google. Get started with cloud firestore, 2025.
- [5] Various Authors. Cloudinary vs firebase storage comparison, 2025.
- [6] Remi Rousselet. Provider package for flutter, 2025.
- [7] Google. Dart overview, 2025.
- [8] Flutter Team. Why flutter?, 2025.
- [9] Flutter Team. Simple app state management, 2025.
- [10] Various Authors. Building real-time chat with flutter and firebase, 2024.
- [11] Various Authors. Socket.io vs firestore for realtime apps, 2025.
- [12] Google. Choose a database: Cloud firestore or realtime database, 2025.
- [13] Henry Ifebunandu. Cloud firestore data modeling for chat application, 2024.
- [14] Google. Firestore security rules best practices, 2025.
- [15] Cloudinary. Flutter quick start, 2025.
- [16] Miguel Ruivo. File picker package for flutter, 2025.
- [17] OWASP Foundation. Owasp mobile top 10, 2025.