

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN CUỐI KÌ

Học phần: Dữ liệu NoSQL

**Đề tài: Tìm hiểu công cụ và thiết kế CSDL tài liệu bằng
Firebase Firestore cho “Quản lý ứng dụng chat nhóm
sinh viên”. Xây dựng ứng dụng minh họa.**

Sinh viên thực hiện	Giảng viên bộ môn
Nhóm 3 1. 2001222641 – Trần Công Minh (NT) 2. 2001225676 – Lê Đức Trung 3. 2001225916 – Tạ Nguyên Vũ	Nguyễn Thị Định

TP. HỒ CHÍ MINH, THÁNG 11 NĂM 2025

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN CUỐI KÌ

Học phần: Dữ liệu NoSQL

**Đề tài: Tìm hiểu công cụ và thiết kế CSDL tài liệu bằng
Firebase Firestore cho “Quản lý ứng dụng chat nhóm
sinh viên”. Xây dựng ứng dụng minh họa.**

Sinh viên thực hiện	Giảng viên bộ môn
Nhóm 3 1. 2001222641 – Trần Công Minh (NT) 2. 2001225676 – Lê Đức Trung 3. 2001225916 – Tạ Nguyên Vũ	Nguyễn Thị Định

TP. HỒ CHÍ MINH, THÁNG 11 NĂM 2025

BẢNG PHÂN CÔNG CÔNG VIỆC

Tên thành viên	Nội dung được phân công	Đánh giá mức độ hoàn thành
Trần Công Minh	<ul style="list-style-type: none"> Nghiên cứu và thiết kế chi tiết cấu trúc các collection trên Firebase Firestore (users, rooms, messages...). Viết, kiểm thử và triển khai các quy tắc bảo mật trong tệp <code>firestore.rules</code>. Phân tích các yêu cầu truy vấn và định nghĩa các chỉ mục hiệu suất cao trong <code>firestore.indexes.json</code>. Xây dựng toàn bộ tầng dịch vụ (service layer) trong thư mục <code>src/firebase/services/</code> để trừu tượng hóa các thao tác với Firestore. Thiết lập và cấu hình Firebase Authentication (cho cả Google và Email/Password) và Supabase Storage. Word + PowerPoint. 	100%
Lê Đức Trung	<ul style="list-style-type: none"> Xây dựng cấu trúc layout và các component chính của ứng dụng. Phát triển các Context Provider (<i>AuthProvider</i>, <i>AppProvider</i>) để quản lý trạng thái toàn cục của ứng dụng (thông tin người dùng, phòng chat đang chọn...). Xây dựng các custom hooks phức tạp để xử lý việc lắng nghe dữ liệu thời gian thực và tải dữ liệu theo trang. Tích hợp logic từ tầng dịch vụ vào các component và hooks. 	90%
Tạ Nguyên Vũ	<ul style="list-style-type: none"> Thiết kế giao diện người dùng (UI) và luồng trải nghiệm người dùng (UX). Viết CSS, thực hiện styling cho toàn bộ ứng dụng, đảm bảo tính nhất quán và hỗ trợ Dark Mode. Xây dựng các component chi tiết và chuyên biệt như: các loại Modals (tạo phòng, thêm bạn...), các component hiển thị tin nhắn (văn bản, hình ảnh, file, vote...), component tải file. Triển khai các tính năng tương tác nhỏ nhưng quan trọng như bộ chọn emoji, reactions cho tin nhắn, chỉ báo "Đang nhập...". 	90%

MỤC LỤC

LỜI MỞ ĐẦU	1
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN CÔNG CỤ	2
1.1. Giới thiệu về Cơ sở dữ liệu NoSQL.....	2
1.1.1. Khái niệm và lịch sử phát triển	2
1.1.2. Các loại CSDL NoSQL phổ biến.....	2
1.1.3. Ưu và nhược điểm so với CSDL quan hệ (SQL).....	4
1.2. Tổng quan về nền tảng Google Firebase.....	5
1.2.1. Firebase là gì? Các tính năng chính	5
1.2.2. Giới thiệu Firebase Cloud Firestore - CSDL tài liệu	6
1.3. Cài đặt và thiết lập môi trường	7
1.3.1. Hướng dẫn tạo dự án trên Firebase Console.....	7
1.3.2. Tích hợp Firebase SDK vào ứng dụng.....	9
CHƯƠNG 2. PHÂN TÍCH VÀ SO SÁNH FIREBASE FIRESTORE 12	
2.1. Phân tích chi tiết các chức năng của Firebase Firestore	12
2.1.1. Mô hình dữ liệu.....	12
2.1.2. Các thao tác dữ liệu cơ bản (CRUD)	13
2.1.3. Truy vấn dữ liệu	14
2.1.4. Lắng nghe dữ liệu thời gian thực (Real-time Listeners).....	14
2.1.5. Bảo mật với Firebase Security Rules	15
2.2. So sánh Firebase Firestore với MongoDB Atlas	16
2.2.1. Giới thiệu về MongoDB và MongoDB Atlas	16

2.2.2. So sánh điểm mạnh, điểm yếu	16
2.3. Đánh giá khả năng ứng dụng của Firebase Firestore	17
2.3.1. Các loại ứng dụng phù hợp nhất với Firestore	17
2.3.2. Lý do lựa chọn Firestore cho ứng dụng	18
CHƯƠNG 3. XÂY DỰNG CSDL CHO ỨNG DỤNG	19
3.1. Phân tích yêu cầu và thiết kế CSDL	19
3.1.1. Mô tả các chức năng chính	19
3.1.2. Thiết kế cấu trúc các Collections	19
3.2. Xây dựng và khởi tạo dữ liệu	23
3.2.1. Tạo các Collections trên Firebase Console	23
3.2.2. Insert dữ liệu	23
3.3. Quản lý dữ liệu	24
3.3.1. Import và Export dữ liệu	24
3.3.2. Backup và Restore dữ liệu	25
CHƯƠNG 4. TRIỂN KHAI TRUY VẤN VÀ KẾT NỐI ỨNG DỤNG	26
4.1. Tổng quan về cách triển khai truy vấn trong	26
4.2. Các truy vấn cơ bản trong ứng dụng	26
4.3. Các truy vấn nâng cao trong ứng dụng	29
4.4. Cập nhật giao diện người dùng tự động với Real-time Listeners	32
4.5. Tối ưu hóa truy vấn với Firestore Indexes	34
4.6. Cơ chế bảo mật và xác thực trong truy vấn	35
4.7. Kết nối và cấu hình Firebase trong ứng dụng	35
CHƯƠNG 5. DEMO ỨNG DỤNG VÀ KẾT QUẢ	37

5.1. Giao diện và các chức năng chính của ứng dụng.....	37
5.2. Chức năng quản trị người dùng.....	37
5.3. Chức năng thao tác trên CSDL	39
5.4. Chức năng quản lý chuyên biệt.....	41
CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	43
6.1. Tóm tắt kết quả đạt được.....	43
6.2. Đánh giá ưu, nhược điểm của giải pháp	43
6.3. Hướng phát triển trong tương lai	43
TÀI LIỆU THAM KHẢO	45

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
BaaS	Backend-as-a-Service	Nền tảng dịch vụ backend tích hợp
DBaaS	Database-as-a-Service	Nền tảng cơ sở dữ liệu
GUI	Graphical User Interface	Giao diện đồ họa
CLI	Command Line Interface	Giao diện dòng lệnh
SDK	Software Development Kit	Bộ công cụ phát triển phần mềm
CRUD	Create, Read, Update, Delete	Tạo, Đọc, Cập nhật, Xóa
RDBMS	Relational Database Management System	Nền tảng cơ sở dữ liệu quan hệ
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc

DANH MỤC HÌNH

Hình 1. Các loại CSDL NoSQL phổ biến.....	3
Hình 2. Hệ sinh thái của Firebase	5
Hình 3. Sơ đồ minh họa cấu trúc dữ liệu của Firestore	13
Hình 4. Giao diện chính của ứng dụng	37

DANH MỤC BẢNG

Bảng 1. Ưu và nhược điểm so với CSDL quan hệ (SQL)	4
Bảng 2. So sánh điểm mạnh, điểm yếu của Firebase Firestore với MongoDB Atlas.....	17

LỜI MỞ ĐẦU

1. Mục tiêu nghiên cứu

Đề án tập trung vào các mục tiêu chính sau:

- Nghiên cứu sâu về cơ sở dữ liệu NoSQL, cụ thể là CSDL tài liệu Firebase Firestore.
- Phân tích, so sánh ưu và nhược điểm của Firestore so với các công cụ khác.
- Thiết kế một cấu trúc cơ sở dữ liệu tối ưu, bảo mật và hiệu quả cho một ứng dụng chat nhóm.
- Xây dựng một ứng dụng minh họa trên nền tảng ReactJS, kết nối và sử dụng Firebase Firestore để triển khai các chức năng cốt lõi.

2. Đối tượng và phạm vi nghiên cứu

- **Đối tượng nghiên cứu:** Nền tảng Firebase, đặc biệt là dịch vụ Cloud Firestore.
- **Phạm vi nghiên cứu:**
 - Tập trung vào việc thiết kế, triển khai và quản lý cơ sở dữ liệu Firestore.
 - Xây dựng các chức năng chính của một ứng dụng chat: quản lý người dùng, chat nhóm, chat 1-1, gửi tệp đa phương tiện, và các tính năng hỗ trợ học tập cơ bản.
 - Đề án không đi sâu vào các vấn đề tối ưu hóa hạ tầng ở quy mô cực lớn hay các thuật toán xử lý ngôn ngữ tự nhiên phức tạp.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN CÔNG CỤ

1.1. Giới thiệu về Cơ sở dữ liệu NoSQL

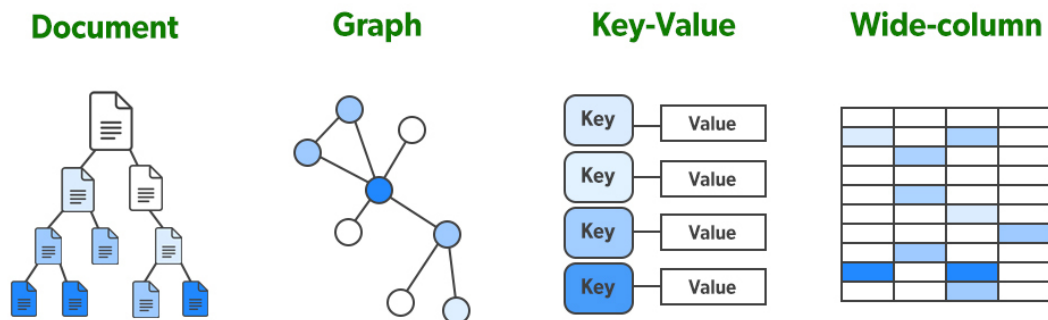
1.1.1. Khái niệm và lịch sử phát triển

NoSQL, thường được hiểu là "Not Only SQL" (Không chỉ SQL), là một thuật ngữ dùng để chỉ một lớp các hệ quản trị cơ sở dữ liệu không tuân theo mô hình quan hệ truyền thống (RDBMS). Sự ra đời của NoSQL gắn liền với sự bùng nổ của Internet và dữ liệu lớn (Big Data) vào đầu những năm 2000. Các ứng dụng web quy mô lớn như Google, Amazon, Facebook phải đối mặt với những thách thức về khả năng mở rộng (scalability), hiệu suất cao và tính linh hoạt trong cấu trúc dữ liệu mà các hệ CSDL quan hệ truyền thống khó đáp ứng.

CSDL NoSQL được thiết kế để giải quyết các bài toán này bằng cách ưu tiên tốc độ, khả năng mở rộng theo chiều ngang (horizontal scaling) và cho phép lưu trữ dữ liệu phi cấu trúc hoặc bán cấu trúc một cách linh hoạt.

1.1.2. Các loại CSDL NoSQL phổ biến

CSDL NoSQL được phân thành bốn loại chính, mỗi loại có một mô hình lưu trữ và các trường hợp sử dụng tối ưu riêng:



Hình 1. Các loại CSDL NoSQL phổ biến

1. **Document Databases (CSDL tài liệu):** Lưu trữ dữ liệu dưới dạng các tài liệu (documents), thường là ở định dạng JSON, BSON hoặc XML. Mỗi tài liệu chứa các cặp *key-value* và có cấu trúc riêng, không cần tuân theo một lược đồ (schema) cứng nhắc.
 - *Đặc điểm:* Linh hoạt, dễ dàng cho lập trình viên làm việc vì cấu trúc dữ liệu gần giống với các đối tượng trong mã nguồn.
 - *Ví dụ:* Firebase Firestore, MongoDB, CouchDB.
2. **Key-Value Stores (CSDL dạng Key-Value):** Là loại CSDL đơn giản nhất, lưu trữ dữ liệu dưới dạng các cặp *key-value*. Mỗi *key* là một định danh duy nhất và *value* có thể là bất kỳ loại dữ liệu nào, từ chuỗi đơn giản đến đối tượng phức tạp.
 - *Đặc điểm:* Tốc độ truy xuất cực nhanh, lý tưởng cho việc lưu trữ *cache*, *session* người dùng.
 - *Ví dụ:* Redis, Amazon DynamoDB.
3. **Column-family Stores (CSDL họ cột):** Dữ liệu được lưu trữ trong các bảng, nhưng thay vì tổ chức theo hàng, nó được tổ chức theo các họ cột. Điều này giúp tối ưu hóa việc đọc/ghi trên các tập dữ liệu khổng lồ.

- *Đặc điểm: Khả năng mở rộng và hiệu suất cao cho các truy vấn tổng hợp.*
- *Ví dụ: Apache Cassandra, HBase.*

4. **Graph Databases (CSDL đồ thị):** Được thiết kế để lưu trữ và truy vấn các mối quan hệ phức tạp. Dữ liệu được biểu diễn dưới dạng các nút (nodes), cạnh (edges) và thuộc tính (properties).

- *Đặc điểm: Rất hiệu quả cho các bài toán liên quan đến mạng xã hội, hệ thống gợi ý (recommendation engines), phát hiện gian lận.*
- *Ví dụ: Neo4j, Amazon Neptune.*

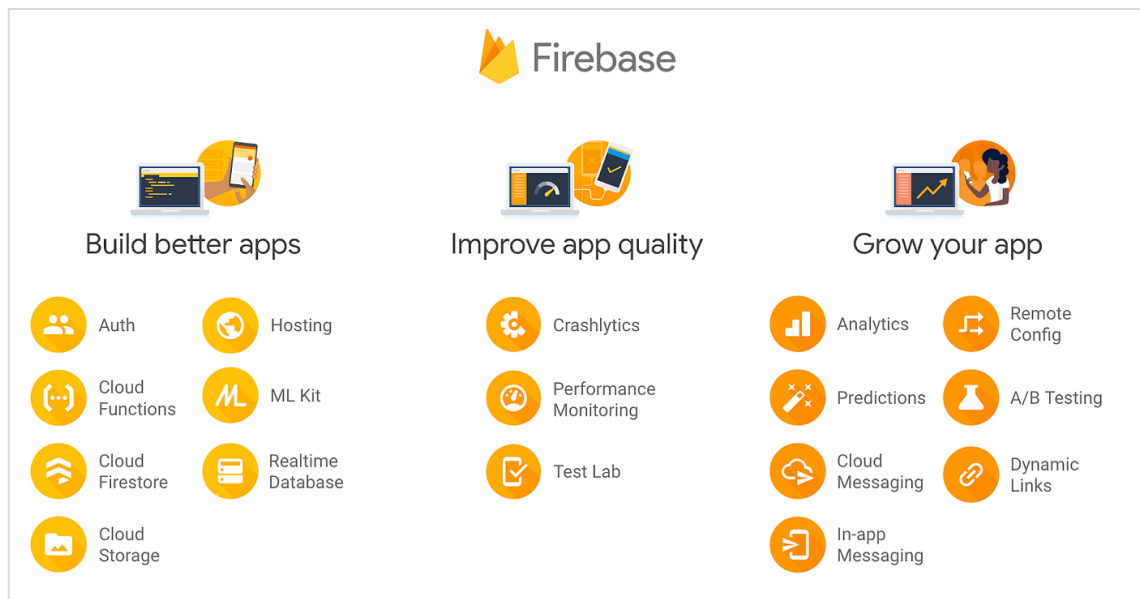
1.1.3. Ưu và nhược điểm so với CSDL quan hệ (SQL)

Tiêu chí	CSDL Quan hệ (SQL)	CSDL NoSQL
Mô hình dữ liệu	Dữ liệu có cấu trúc, được lưu trong các bảng với các hàng và cột.	Dữ liệu phi cấu trúc, bán cấu trúc hoặc có cấu trúc. Đa dạng mô hình (tài liệu, key-value, đồ thị...).
Lược đồ (Schema)	Cứng nhắc (Schema-on-write): Lược đồ phải được định nghĩa trước khi nhập dữ liệu.	Linh hoạt (Schema-on-read): Lược đồ có thể thay đổi động, không cần định nghĩa trước.
Khả năng mở rộng	Mở rộng theo chiều dọc (Vertical Scaling): Tăng cường tài nguyên (CPU, RAM) cho một máy chủ duy nhất.	Mở rộng theo chiều ngang (Horizontal Scaling): Thêm nhiều máy chủ vào hệ thống để phân tán tải.
Tính nhất quán	Tuân thủ nguyên tắc ACID (Atomicity, Consistency, Isolation, Durability), đảm bảo tính nhất quán cao.	Thường tuân thủ định lý CAP và mô hình BASE (Basically Available, Soft state, Eventual consistency), ưu tiên tính sẵn sàng.
Ngôn ngữ truy vấn	Sử dụng SQL (Structured Query Language), một ngôn ngữ mạnh mẽ và được chuẩn hóa.	Không có ngôn ngữ truy vấn chung. Mỗi hệ CSDL có API hoặc ngôn ngữ truy vấn riêng.
Trường hợp sử dụng	Phù hợp cho các ứng dụng yêu cầu giao dịch phức tạp, tính toàn vẹn dữ liệu cao như hệ thống ngân hàng, kế toán.	Phù hợp cho các ứng dụng dữ liệu lớn, yêu cầu tính linh hoạt, khả năng mở rộng nhanh và xử lý thời gian thực.

Bảng 1. Ưu và nhược điểm so với CSDL quan hệ (SQL)

1.2. Tổng quan về nền tảng Google Firebase

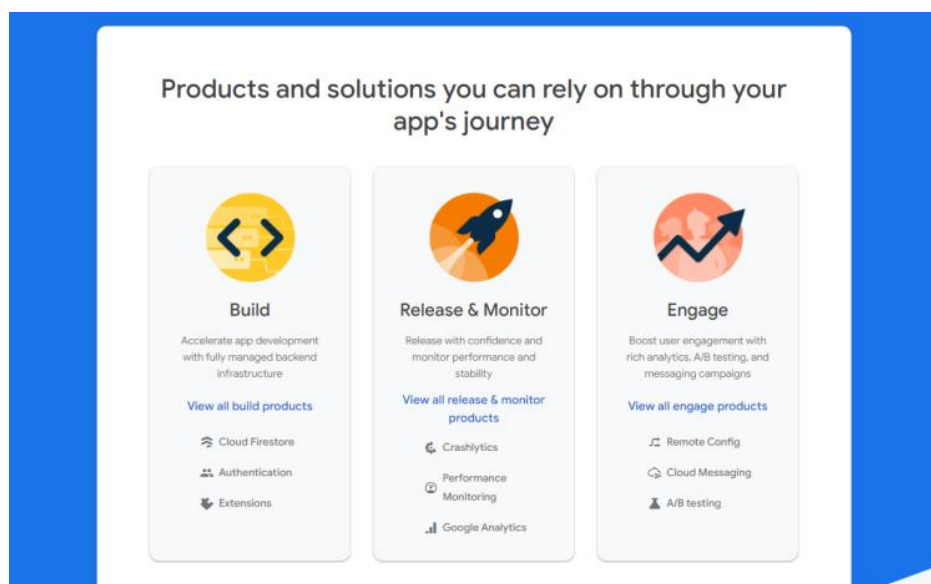
1.2.1. Firebase là gì? Các tính năng chính



Hình 2. Hệ sinh thái của Firebase

Firebase là một nền tảng phát triển ứng dụng di động và web do Google cung cấp. Nó hoạt động theo mô hình Backend-as-a-Service (BaaS), cung cấp cho các nhà phát triển một bộ công cụ và dịch vụ mạnh mẽ để xây dựng ứng dụng nhanh hơn, chất lượng cao hơn mà không cần phải tự xây dựng và quản lý hạ tầng backend phức tạp.

Các tính năng chính của Firebase bao gồm:



- **Build (Xây dựng):**

- Authentication: Xác thực người dùng qua email/mật khẩu, các nhà cung cấp bên thứ ba (Google, Facebook,...) và số điện thoại.
- Cloud Firestore & Realtime Database: Hai giải pháp CSDL NoSQL mạnh mẽ, đồng bộ dữ liệu thời gian thực.
- Storage: Lưu trữ và quản lý các tệp do người dùng tải lên như hình ảnh, video.
- Hosting: Dịch vụ hosting nhanh, an toàn cho các ứng dụng web.
- Cloud Functions: Chạy mã backend mà không cần quản lý máy chủ (serverless).

- **Release & Monitor (Phát hành & Giám sát):**

- Crashlytics: Báo cáo lỗi và sự cố ứng dụng theo thời gian thực.
- Performance Monitoring: Theo dõi hiệu suất của ứng dụng.

- **Engage (Tương tác):**

- Cloud Messaging (FCM): Gửi thông báo đẩy (push notifications) đến người dùng.
- Google Analytics: Phân tích hành vi người dùng trong ứng dụng.

1.2.2. Giới thiệu Firebase Cloud Firestore - CSDL tài liệu

Cloud Firestore (hay Firestore) là giải pháp cơ sở dữ liệu NoSQL dạng tài liệu của Firebase. Đây là sản phẩm kế thừa và cải tiến từ Firebase Realtime Database, được thiết kế để giải quyết các bài toán phức tạp hơn về truy vấn, khả năng mở rộng và cấu trúc dữ liệu.

Firestore



Các đặc điểm nổi bật của Firestore:

- **Mô hình dữ liệu linh hoạt:** Dữ liệu được lưu trữ trong các tài liệu (documents), được sắp xếp vào các bộ sưu tập (collections). Mô hình này cho phép tạo các cấu trúc dữ liệu phân cấp và truy vấn phức tạp.
- **Truy vấn mạnh mẽ:** Hỗ trợ các truy vấn phức hợp như lọc, sắp xếp trên nhiều trường dữ liệu cùng lúc.
- **Đồng bộ hóa thời gian thực:** Dữ liệu được tự động đồng bộ trên tất cả các thiết bị client đang kết nối, giúp xây dựng các ứng dụng cộng tác và real-time một cách dễ dàng.
- **Hỗ trợ ngoại tuyến (Offline Support):** Tự động lưu trữ dữ liệu vào bộ đệm trên thiết bị, cho phép ứng dụng hoạt động ngay cả khi không có kết nối mạng. Dữ liệu sẽ được đồng bộ lên máy chủ ngay khi có kết nối trở lại.
- **Khả năng mở rộng toàn cầu:** Được xây dựng trên hạ tầng của Google Cloud, đảm bảo khả năng mở rộng tự động và độ tin cậy cao.

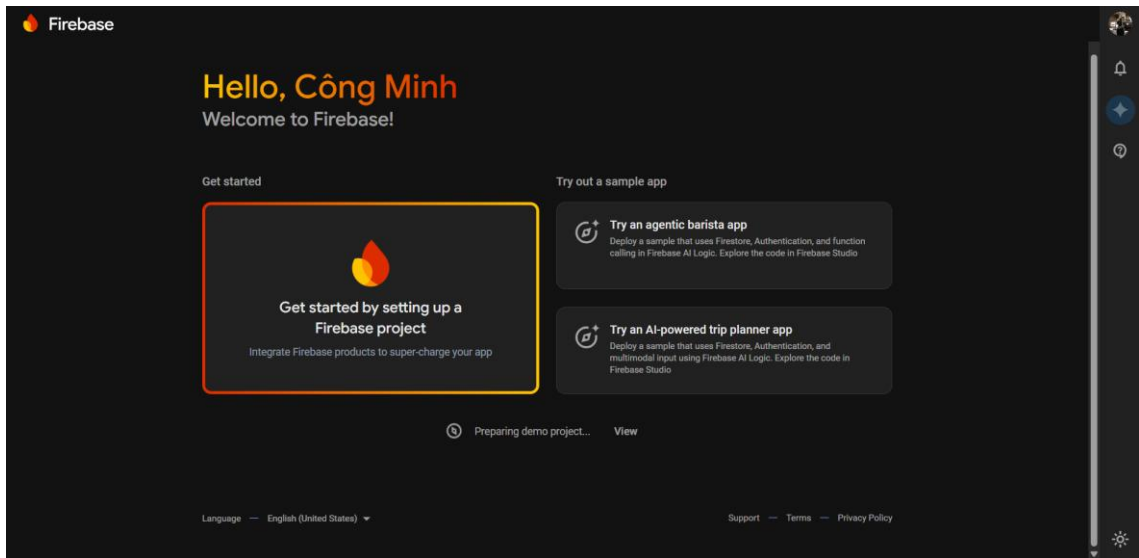
1.3. Cài đặt và thiết lập môi trường

1.3.1. Hướng dẫn tạo dự án trên Firebase Console

Để bắt đầu sử dụng Firestore, trước tiên cần tạo một dự án trên Firebase Console.

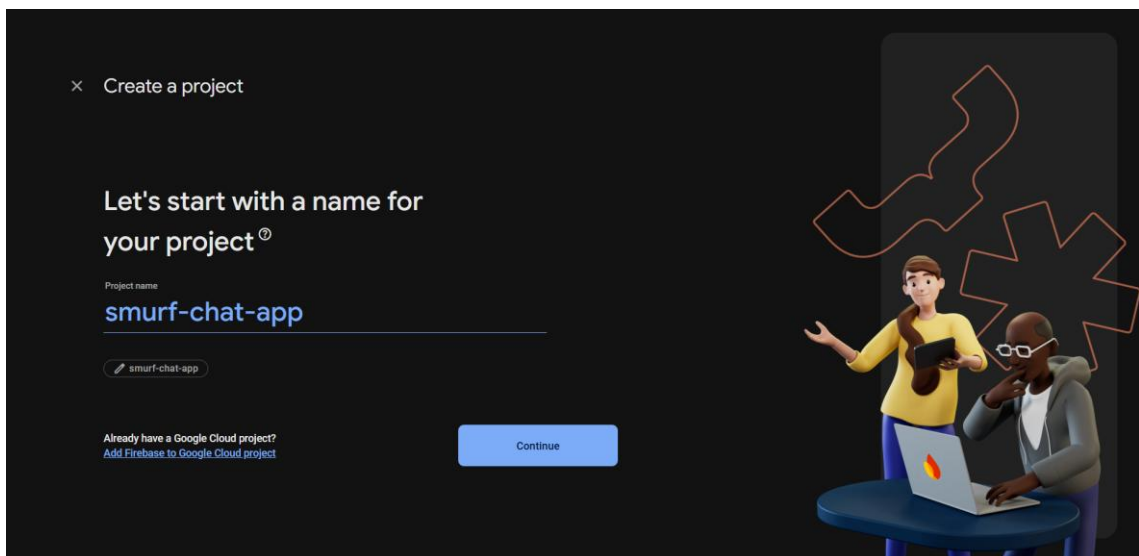
Bước 1: Truy cập trang web Firebase Console tại

<https://console.firebase.google.com/> và đăng nhập bằng tài khoản Google.

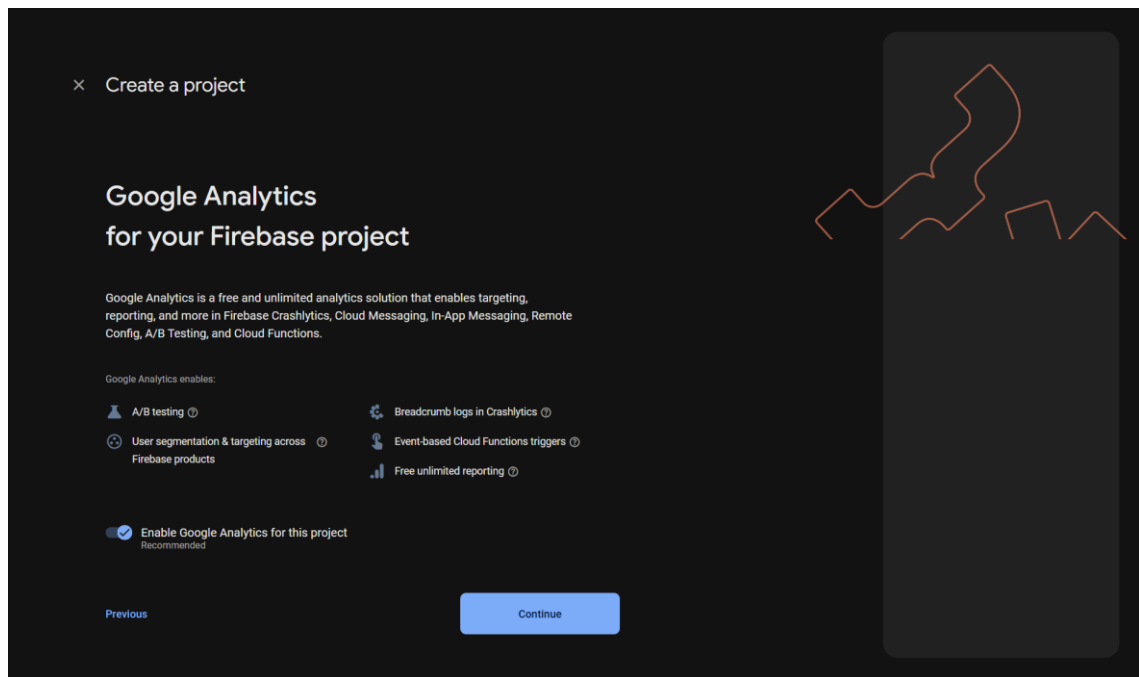


Bước 2: Nhấp vào "Add project" (Thêm dự án).

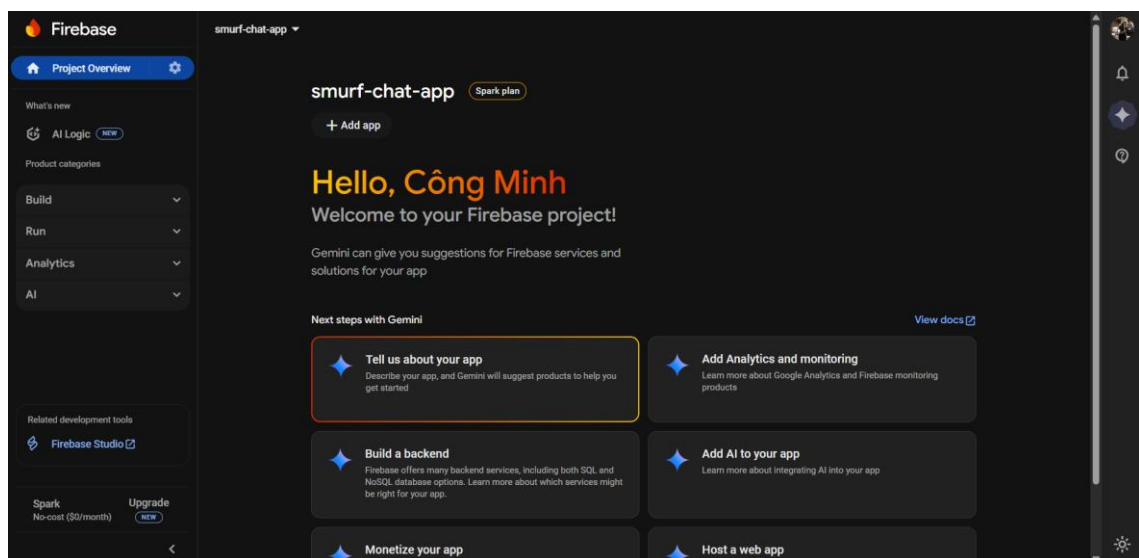
Bước 3: Nhập tên cho dự án (ví dụ: *smurf-chat-app*) và chấp nhận các điều khoản.



Bước 4: (Tùy chọn) Bật Google Analytics cho dự án để có thêm các công cụ phân tích người dùng.



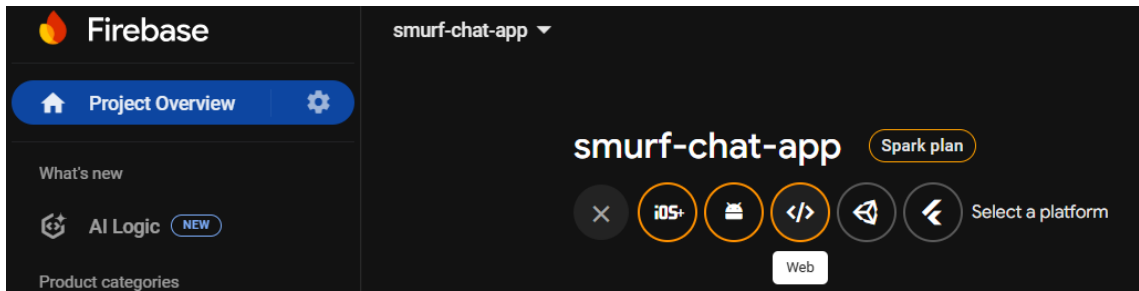
Bước 5: Chờ trong giây lát để Firebase khởi tạo dự án. Sau khi hoàn tất, bạn sẽ được chuyển đến trang tổng quan (Project Overview) của dự án.



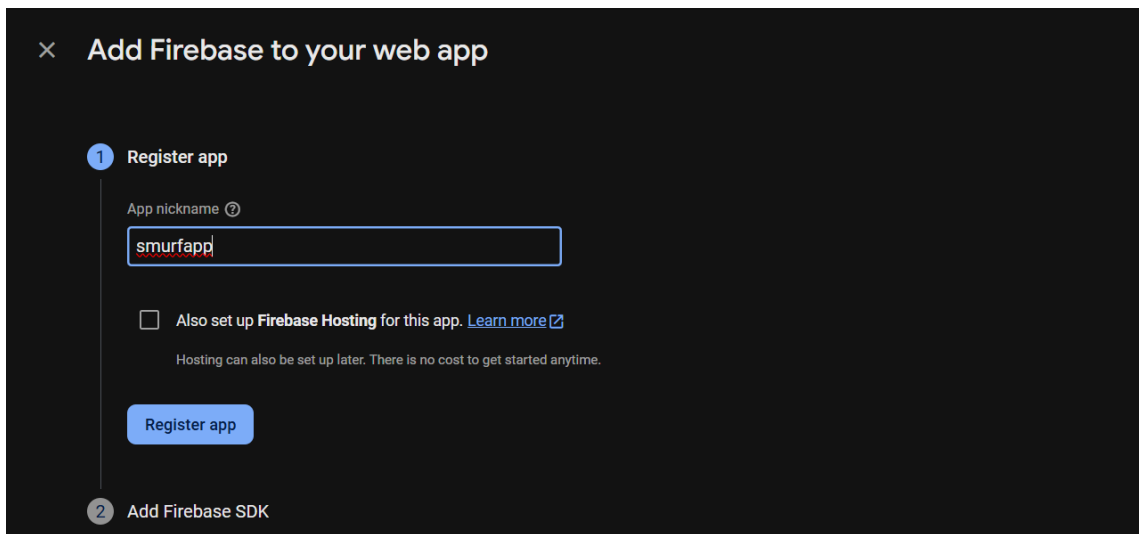
1.3.2. Tích hợp Firebase SDK vào ứng dụng

Sau khi có dự án Firebase, bước tiếp theo là kết nối nó với ứng dụng tùy theo môi trường phát triển ứng dụng.

- **Bước 1:** Tại trang tổng quan dự án, nhấp vào biểu tượng Web </> để thêm một ứng dụng web vào dự án.



- **Bước 2:** Đặt một biệt danh (nickname) cho ứng dụng của bạn và nhấp vào "Register app".



- **Bước 3:** Firebase sẽ cung cấp một đối tượng cấu hình firebaseConfig. Hãy sao chép đối tượng này, nó chứa các khóa API cần thiết để kết nối ứng dụng với dự án Firebase.

×

Add Firebase to your web app

Go to docs

✓

Register app

2

Add Firebase SDK

☒ Use npm☐ Use a <script> tag

If you're already using [npm](#) and a module bundler such as [webpack](#) or [Rollup](#), you can run the following command to install the latest SDK ([learn more](#)):

\$ npm install firebase

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

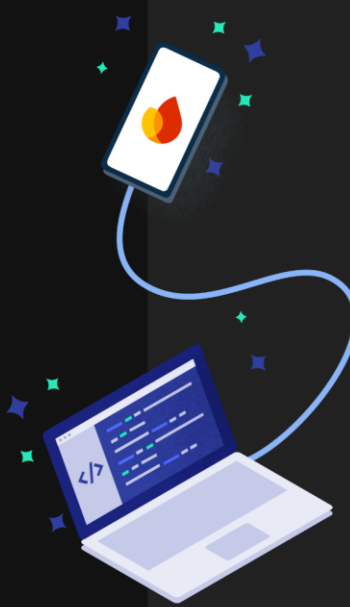
// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyBrl0T3t42waGRkHl1rIn8MuW17pM1qw58",
  authDomain: "smurf-chat-app.firebaseio.com",
  projectId: "smurf-chat-app",
  storageBucket: "smurf-chat-app.firebaseio.com",
  messagingSenderId: "82986884465",
  appId: "1:82986884465:web:5fa3686383bd95731aeadd8"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Note: This option uses the [modular JavaScript SDK](#), which provides reduced SDK size.

Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

Continue to console



CHƯƠNG 2. PHÂN TÍCH VÀ SO SÁNH FIREBASE FIRESTORE

2.1. Phân tích chi tiết các chức năng của Firebase Firestore

2.1.1. Mô hình dữ liệu

Firestore sử dụng một mô hình dữ liệu NoSQL phân cấp, trực quan và linh hoạt. Dữ liệu không được lưu trữ trong các bảng và hàng truyền thống, mà thay vào đó là:

- **Document (Tài liệu):** Là đơn vị lưu trữ cơ bản trong Firestore. Một tài liệu thực chất là một đối tượng JSON, chứa các cặp trường (field) và giá trị (value). Các giá trị có thể là nhiều kiểu dữ liệu khác nhau như chuỗi, số, boolean, mảng, đối tượng lồng nhau (map), hoặc thậm chí là null. Mỗi tài liệu được xác định bởi một ID duy nhất.

Ví dụ: Một tài liệu đại diện cho người dùng có thể trông như sau:

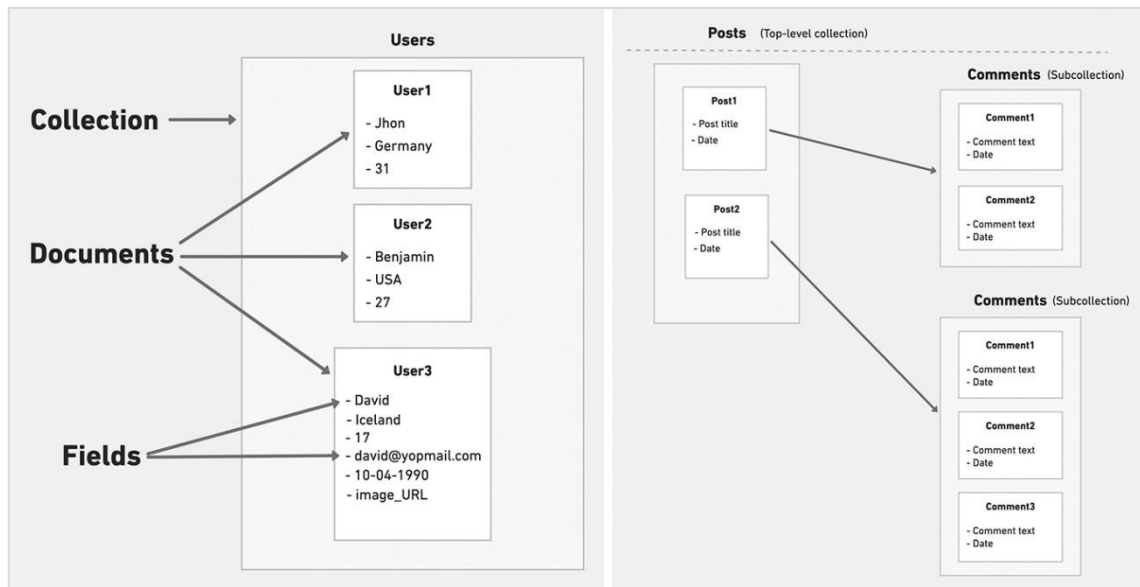
```
{
  "displayName": "Nguyen Van A",
  "email": "a.nguyen@example.com",
  "photoURL": "url-to-avatar.png",
  "providerId": "google.com"
}
```

Collection (Bộ sưu tập): Là một tập hợp chứa các tài liệu. Ví dụ, bạn có thể có một collection users để chứa tất cả các tài liệu người dùng, một collection rooms để chứa các tài liệu về phòng chat. Collection giúp tổ chức dữ liệu một cách logic.

Sub-collection (Bộ sưu tập con): Để tạo cấu trúc dữ liệu phức tạp và phân cấp hơn, Firestore cho phép một tài liệu chứa các collection con. Điều này rất hữu ích để nhóm các dữ liệu liên quan mà không làm phình to tài liệu cha.

Ví dụ: Một tài liệu trong collection rooms có thể chứa một sub-collection tên là messages. Mỗi tài liệu trong sub-collection messages sẽ đại diện cho một

tin nhắn trong phòng chat đó. Cấu trúc này giúp truy vấn tin nhắn của một phòng cụ thể rất hiệu quả.



Hình 3. Sơ đồ minh họa cấu trúc dữ liệu của Firestore

2.1.2. Các thao tác dữ liệu cơ bản (CRUD)

Firestore cung cấp một bộ API đơn giản và mạnh mẽ để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên dữ liệu.

- **Create (Tạo):** Thêm một tài liệu mới vào một collection. Firestore cho phép tự động tạo ID duy nhất cho tài liệu hoặc chỉ định ID tùy chỉnh.
- **Read (Đọc):** Đọc một tài liệu cụ thể bằng ID của nó hoặc đọc tất cả các tài liệu trong một collection.
- **Update (Cập nhật):** Sửa đổi các trường của một tài liệu đã tồn tại. Nếu tài liệu không tồn tại, thao tác cập nhật sẽ thất bại. Firestore cũng hỗ trợ cập nhật lồng nhau trong các đối tượng map.
- **Delete (Xóa):** Xóa một tài liệu khỏi collection. Lưu ý rằng việc xóa một tài liệu sẽ không tự động xóa các sub-collection bên trong nó.

2.1.3. Truy vấn dữ liệu

Đây là một trong những điểm mạnh nhất của Firestore so với người tiền nhiệm Realtime Database.

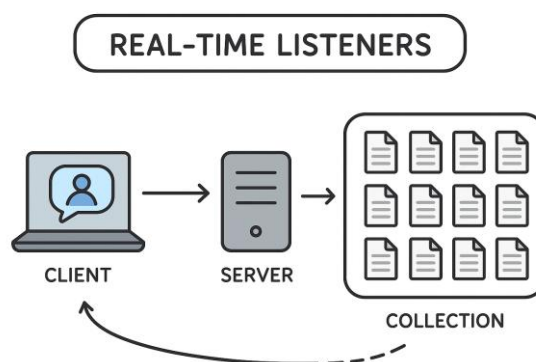
- **Truy vấn đơn giản:** Lấy một tài liệu hoặc toàn bộ tài liệu trong một collection.
- **Lọc (Filtering):** Sử dụng các mệnh đề *where()* để lọc các tài liệu dựa trên giá trị của một hoặc nhiều trường. Firestore hỗ trợ các toán tử so sánh như *==, <, >, <=, >=, !=, array-contains, in, not-in*.

➤ Ví dụ: Lấy tất cả các phòng chat có tên là "Đồ án tốt nghiệp".

- **Sắp xếp (Sorting):** Sử dụng mệnh đề *orderBy()* để sắp xếp kết quả trả về theo một trường cụ thể theo thứ tự tăng dần (*asc*) hoặc giảm dần (*desc*).
- **Truy vấn phức hợp (Compound Queries):** Kết hợp nhiều mệnh đề *where()* để tạo ra các truy vấn phức tạp hơn.

➤ Ví dụ: Tìm tất cả tin nhắn trong một phòng chat được gửi bởi một người dùng cụ thể và sắp xếp chúng theo thời gian gửi.

2.1.4. Lắng nghe dữ liệu thời gian thực (Real-time Listeners)



Đây là tính năng cốt lõi giúp xây dựng các ứng dụng có tính tương tác cao như ứng dụng chat. Thay vì phải liên tục "hỏi" server xem có dữ liệu mới hay không (polling), client có thể đăng ký một "người nghe" (listener) vào

một tài liệu hoặc một collection. Bất cứ khi nào dữ liệu tại đó thay đổi (thêm, sửa, xóa), Firestore sẽ tự động đẩy (push) bản cập nhật đó đến tất cả các client đang lắng nghe. Điều này giúp giao diện người dùng luôn được cập nhật tức thì.

2.1.5. Bảo mật với *Firestore Security Rules*

Firestore Security Rules là một hệ thống mạnh mẽ để kiểm soát quyền truy cập vào dữ liệu Firestore. Các quy tắc này được viết bằng một cú pháp riêng và được lưu trữ trên server của Firestore. Chúng cho phép định nghĩa ai có thể đọc, ghi, cập nhật hoặc xóa dữ liệu ở bất kỳ đường dẫn nào trong CSDL của bạn.

- **Xác thực (Authentication):** Quy tắc có thể kiểm tra xem người dùng đã đăng nhập hay chưa.
- **Ủy quyền (Authorization):** Quy tắc có thể xác minh rằng người dùng có quyền thực hiện một hành động cụ thể hay không (ví dụ: chỉ người tạo ra phòng chat mới có quyền xóa nó).
- **Xác thực dữ liệu (Data Validation):** Quy tắc có thể kiểm tra xem dữ liệu được ghi vào có đúng định dạng, kiểu dữ liệu hay tuân thủ các ràng buộc khác hay không.

Một quy tắc cơ bản sẽ có cấu trúc như sau:

```
{
  "rules": {
    ".read": "condition",
    ".write": "condition"
  }
}
```

Condition: Là các điều kiện kiểm tra như xác thực người dùng (auth), giá trị dữ liệu (data), hoặc yêu cầu từ người dùng (request).

Ví dụ: Chỉ cho phép người dùng đã đăng nhập đọc dữ liệu:

```
{
```



```

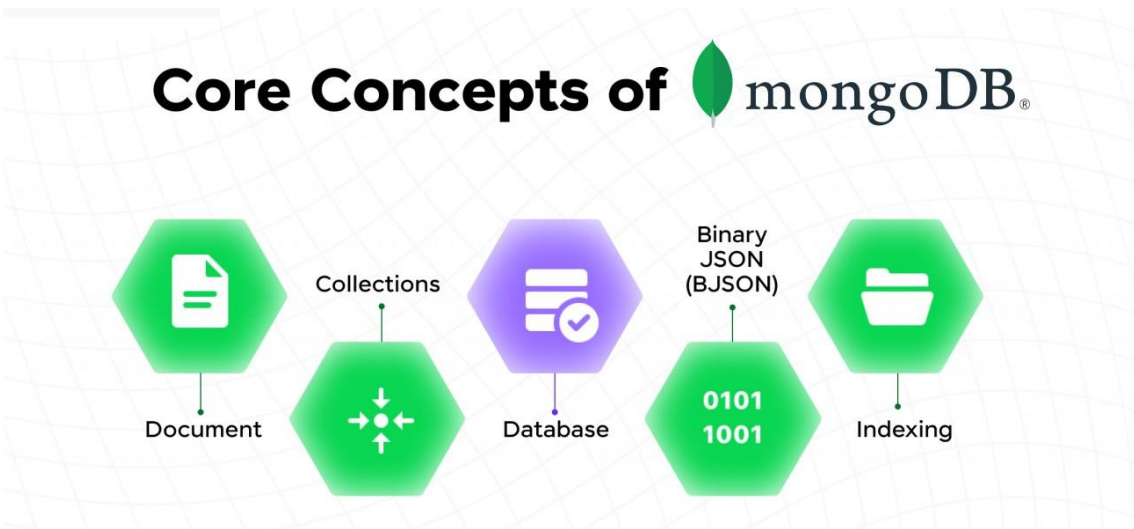
"rules": {
  ".read": "auth != null",
  ".write": "auth != null"
}
}

```

2.2. So sánh Firebase Firestore với MongoDB Atlas

Để có cái nhìn toàn diện hơn, chúng ta sẽ so sánh Firestore với MongoDB Atlas, một dịch vụ CSDL tài liệu trên nền tảng đám mây rất phổ biến.

2.2.1. Giới thiệu về MongoDB và MongoDB Atlas



MongoDB là một trong những CSDL NoSQL dạng tài liệu mã nguồn mở phổ biến nhất. Nó lưu trữ dữ liệu dưới dạng các tài liệu BSON (một dạng nhị phân của JSON). MongoDB Atlas là dịch vụ Database-as-a-Service (DBaaS) chính thức cho MongoDB, được quản lý hoàn toàn trên các nền tảng đám mây lớn như AWS, Google Cloud, và Azure.

2.2.2. So sánh điểm mạnh, điểm yếu

Tiêu chí	Firebase Firestore	MongoDB Atlas
Mô hình dịch vụ	Backend-as-a-Service (BaaS). Tích hợp sâu với các dịch vụ Firebase khác (Auth, Storage, Functions).	Database-as-a-Service (DBaaS). Tập trung chuyên sâu vào cung cấp CSDL.

Đồng bộ Real-time	Điểm mạnh: Tích hợp sẵn và là tính năng cốt lõi. Rất dễ triển khai.	Có thể thực hiện thông qua Change Streams, nhưng yêu cầu nhiều thiết lập và logic phía client/server hơn.
Hỗ trợ Offline	Điểm mạnh: Tích hợp sẵn trên SDK di động và web, hoạt động gần như tự động.	Không có sẵn. Phải tự triển khai logic lưu trữ và đồng bộ cục bộ.
Khả năng truy vấn	Mạnh mẽ cho hầu hết các trường hợp sử dụng, nhưng có một số hạn chế (ví dụ: truy vấn OR phức tạp).	Điểm mạnh: Cực kỳ mạnh mẽ với Aggregation Pipeline, cho phép các truy vấn tổng hợp và biến đổi dữ liệu phức tạp. Hỗ trợ tìm kiếm văn bản (full-text search).
Mô hình giá	Dựa trên số lượng lượt đọc/ghi/xóa tài liệu và dung lượng lưu trữ/mạng. Có bậc miễn phí (free tier) hào phóng.	Dựa trên tài nguyên hạ tầng (RAM, CPU, dung lượng lưu trữ). Có bậc miễn phí nhưng giới hạn hơn.
Bảo mật	Dựa trên Firebase Security Rules, tích hợp chặt chẽ với Firebase Authentication.	Dựa trên vai trò (Role-Based Access Control), danh sách IP, VPC Peering. Mạnh mẽ nhưng phức tạp hơn để cấu hình.
Hệ sinh thái	Điểm mạnh: Là một phần của hệ sinh thái Firebase/Google Cloud, dễ dàng kết hợp với các dịch vụ khác.	Tập trung vào CSDL, nhưng có các công cụ mạnh mẽ như Compass (GUI), Atlas Search, Charts.
Đối tượng phù hợp	Lập trình viên front-end, ứng dụng di động, startup muốn xây dựng nhanh sản phẩm (MVP), các ứng dụng yêu cầu real-time cao.	Các doanh nghiệp có đội ngũ backend, ứng dụng yêu cầu phân tích dữ liệu phức tạp, cần toàn quyền kiểm soát hạ tầng CSDL.

Bảng 2. So sánh điểm mạnh, điểm yếu của Firebase Firestore với MongoDB Atlas

2.3. Đánh giá khả năng ứng dụng của Firebase Firestore

2.3.1. Các loại ứng dụng phù hợp nhất với Firestore

Dựa trên các đặc điểm đã phân tích, Firestore tỏ ra vượt trội cho các loại ứng dụng sau:

- **Ứng dụng cộng tác và thời gian thực:** Bảng trắng online, ứng dụng chỉnh sửa tài liệu chung, và đặc biệt là ứng dụng chat.

- **Ứng dụng di động (Mobile-first):** Với khả năng hỗ trợ offline mạnh mẽ, Firestore đảm bảo trải nghiệm người dùng mượt mà ngay cả khi kết nối mạng không ổn định.
- **Ứng dụng có quy mô thay đổi nhanh:** Các startup và dự án cần phát triển nhanh (MVP) có thể tận dụng mô hình serverless và khả năng mở rộng tự động của Firestore mà không cần lo lắng về quản trị hạ tầng.
- **Ứng dụng mạng xã hội và game:** Cập nhật trạng thái, bảng xếp hạng, tin tức (feed) theo thời gian thực.

2.3.2. Lý do lựa chọn Firestore cho ứng dụng

Việc lựa chọn Firebase Firestore cho đề tài "Quản lý ứng dụng chat nhóm sinh viên" là hoàn toàn hợp lý dựa trên các lý do sau:

- **Yêu cầu cốt lõi là Real-time:** Chức năng chat đòi hỏi tin nhắn phải được gửi và nhận gần như ngay lập tức. Tính năng Real-time Listeners của Firestore đáp ứng hoàn hảo yêu cầu này mà không cần triển khai các giải pháp phức tạp như WebSockets.
- **Tích hợp xác thực dễ dàng:** Ứng dụng cần quản lý người dùng. Việc kết hợp Firestore với Firebase Authentication giúp đơn giản hóa việc đăng ký, đăng nhập và bảo vệ dữ liệu dựa trên danh tính người dùng.
- **Phát triển nhanh:** Firestore (BaaS) cho phép xây dựng một ứng dụng hoàn chỉnh với ít công sức quản trị server nhất.
- **Mô hình dữ liệu phù hợp:** Cấu trúc Collection-Document của Firestore rất tự nhiên để mô hình hóa các thực thể như *users*, *rooms*, và *messages*.

CHƯƠNG 3. XÂY DỰNG CSDL CHO ỨNG DỤNG

3.1. Phân tích yêu cầu và thiết kế CSDL

3.1.1. Mô tả các chức năng chính

Dựa trên mục tiêu của đề tài là xây dựng một ứng dụng chat nhóm cho sinh viên, cần đáp ứng các chức năng sau:

- **Quản lý người dùng:** Đăng ký, đăng nhập, cập nhật thông tin cá nhân.
- **Giao tiếp đa dạng:** Hỗ trợ cả chat nhóm (Rooms) và chat 1-1 (Conversations).
- **Tương tác thời gian thực:** Gửi/nhận tin nhắn, hiển thị trạng thái "đang gõ" (typing status) và "đã xem" (last seen) theo thời gian thực.
- **Quản lý phòng chat:** Tạo phòng, thêm/xóa thành viên, phân quyền quản trị viên, giải tán phòng.
- **Quản lý bạn bè và quyền riêng tư:** Gửi lời mời kết bạn, chấp nhận/từ chối, và chặn người dùng không mong muốn.
- **Tính năng nâng cao:** Tạo cuộc bình chọn (vote) trong phòng chat, ghim các cuộc trò chuyện quan trọng, lưu trữ (archive) cuộc trò chuyện.

3.1.2. Thiết kế cấu trúc các Collections

Để đáp ứng các yêu cầu trên, ứng dụng được thiết kế một cấu trúc CSDL trên Firestore bao gồm các collection chính sau:

1. users

Mục đích: Lưu trữ thông tin định danh và cài đặt của mỗi người dùng.

```

{
  uid: string,           // Firebase Auth UID
  displayName: string,   // Tên hiển thị
  email: string,         // Email
  photoURL: string|null, // Avatar URL
  providerId: string,    // Provider đăng nhập ('password', 'google', etc.)
  keywords: string[],    // Từ khóa tìm kiếm
  isOnline: boolean,     // Trạng thái online
  lastSeen: timestamp,   // Thời gian hoạt động cuối
  createdAt: timestamp,  // Thời gian tạo tài khoản
  updatedAt: timestamp   // Thời gian cập nhật cuối
}

```

2. rooms

Mục đích: Lưu trữ thông tin về các phòng chat nhóm.

```

{
  id: string,
  name: string,           // Tên phòng
  admin: string,         // UID admin
  members: string[],     // Array UID thành viên
  avatar: string|null,   // Avatar phòng
  lastMessage: string,   // Tin nhắn cuối
  lastMessageAt: timestamp, // Thời gian gửi tin nhắn cuối
  lastSeen: object,      // {userId: timestamp}
  typingStatus: object,  // {userId: boolean}
  pinned: boolean,       // Đã pin
  pinnedAt: timestamp|null, // Thời gian pin
  updatedAt: timestamp,  // Thời gian cập nhật
  updatedBy: string      // UID người cập nhật lastMessage
}

```

3. conversations

Mục đích: Lưu trữ thông tin về các cuộc trò chuyện 1-1. Cấu trúc tương tự rooms nhưng đơn giản hơn.

```

{
  id: string,
  participants: string[], // [uid1, uid2] sorted
  lastMessage: string, // Tin nhắn cuối
  lastMessageAt: timestamp, // Thời gian gửi tin nhắn cuối
  lastSeen: object, // {userId: timestamp}
  typingStatus: object, // {userId: boolean}
  pinned: boolean, // Đã pin
  pinnedAt: timestamp|null, // Thời gian pin
  updatedAt: timestamp, // Thời gian cập nhật
  updatedBy: string // UID người cập nhật lastMessage
}

```

4. Messages

Mục đích: Lưu trữ tất cả tin nhắn từ cả rooms và conversations. Đây là collection lớn nhất và hoạt động nhiều nhất.

```

{
  id: string,
  text: string, // Nội dung text
  uid: string, // Người gửi
  displayName: string, // Tên người gửi
  photoURL: string|null, // Avatar người gửi
  chatType: 'room'|'direct', // Loại chat
  chatId: string, // ID phòng hoặc cuộc trò chuyện
  messageType: 'text'|'file'|'voice'|'location', // Loại tin nhắn
  fileData: object|null, // Dữ liệu file
  locationData: object|null, // Dữ liệu vị trí
  readByDetails: object, // {userId: timestamp}
  reactions: object, // {emoji: [userIds]}
  recalled: boolean, // Đã thu hồi
  recalledAt: timestamp|null, // Thời gian thu hồi
  originalText: string|null, // Text gốc
  originalFileData: object|null, // File data gốc
  originalLocationData: object|null, // Location data gốc
  lastReadAt: timestamp|null, // Thời gian đọc cuối
  forwarded: boolean, // Đã forward
  createdAt: timestamp, // Thời gian tạo
  updatedAt: timestamp // Thời gian cập nhật (khi add reaction)
}

```

5. friends

Mục đích: Lưu trữ danh sách bạn bè.

```
{
  id: string,
  participants: string[],    // [uid1, uid2] sorted
  createdAt: timestamp      // Thời gian tạo
}
```

6. friend_requests

Mục đích: Lưu trữ danh sách lời mời kết bạn

```
{
  id: string,
  from: string,              // UID người gửi
  to: string,                // UID người nhận
  participants: string,      // Combined key for queries
  status: 'pending'|'accepted'|'declined'|'cancelled', // Trạng thái
  createdAt: timestamp,      // Thời gian tạo
  updatedAt: timestamp       // Thời gian cập nhật
}
```

7. blocked_users

Mục đích: Lưu trữ danh sách người bị chặn

```
{
  id: string,
  blocker: string,           // UID người chặn
  blocked: string,           // UID bị chặn
  createdAt: timestamp
}
```

8. votes

Mục đích: Lưu trữ danh sách các bình chọn

```

{
  id: string,
  roomId: string,           // ID phòng chat
  creatorId: string,        // UID người tạo vote
  title: string,            // Tiêu đề vote
  options: string[],        // Mảng các lựa chọn
  votes: object,            // {userId: optionIndex[]} - votes của từng user
  voteCounts: object,       // {optionIndex: count} - số lượng vote cho mỗi option
  createdAt: timestamp,
  updatedAt: timestamp
}

```

9. archived_chats

Mục đích: Lưu trữ danh sách tin nhắn lưu trữ

```

{
  id: string,
  userId: string,           // UID người archive
  chatId: string,           // ID room/conversation
  isConversation: boolean,  // true nếu là conversation, false nếu là room
  archivedAt: timestamp     // Thời gian archive
}

```

3.2. Xây dựng và khởi tạo dữ liệu

3.2.1. Tạo các Collections trên Firebase Console

Việc tạo collection trong Firestore được thực hiện một cách linh hoạt. Collection sẽ tự động được tạo ra khi tài liệu đầu tiên được thêm vào.

- Ví dụ, khi một người dùng đăng ký tài khoản lần đầu, một hàm sẽ tạo một document mới trong collection **users** với **uid** tương ứng. Tương tự, khi một phòng chat mới được tạo, một document sẽ được thêm vào collection **rooms**.

3.2.2. Insert dữ liệu

Để phục vụ quá trình phát triển và kiểm thử, việc chèn dữ liệu mẫu là rất cần thiết. Dữ liệu này có thể được thêm vào bằng nhiều cách:

- **Thủ công:** Sử dụng giao diện của Firebase Console để thêm các document mới vào các collection. Cách này phù hợp cho việc tạo một vài dữ liệu ban đầu.
- **Sử dụng Script:** Viết một đoạn script để tự động tạo ra một lượng lớn dữ liệu mẫu. Cách này giúp mô phỏng môi trường sử dụng thực tế một cách hiệu quả.

3.3. Quản lý dữ liệu

3.3.1. Import và Export dữ liệu

Firebase cung cấp các công cụ mạnh mẽ để di chuyển dữ liệu.

Recent imports and exports						
You'll see the most recent ongoing and completed imports and exports here.						
Filter operations						
Started ↓	Type	Collection groups	Bucket	Size	Completed	
Oct 16, 2020, 1:49:48 PM	Import	All Collections		112.67 MB		CANCEL

- **Export:** Có thể xuất toàn bộ CSDL hoặc chỉ một vài collection cụ thể ra Google Cloud Storage. Lệnh này thường được chạy thông qua gcloud CLI. Đây là cách chính để tạo ra các bản sao lưu (backup).

```
1 gcloud firestore export gs://[BUCKET_NAME] --collection-ids='users','rooms'
```

- **Import:** Dữ liệu đã được xuất ra Cloud Storage có thể được nhập lại vào Firestore. Quá trình này sẽ ghi đè lên dữ liệu hiện có, do đó cần được thực hiện một cách cẩn trọng.



```
1 gcloud firestore export gs://[BUCKET_NAME] --collection-ids='users','rooms'
```

Tuy nhiên, một vài lưu ý khi sử dụng tính năng xuất/nhập dữ liệu:

- ⚠ Việc xuất dữ liệu từ Cloud Firestore sẽ phát sinh một thao tác đọc cho mỗi tài liệu được xuất. Tuy nhiên, các lượt đọc này sẽ không xuất hiện trong phần mức sử dụng của bảng điều khiển. Vì vậy cần đảm bảo hiểu rõ quy trình xuất/nhập trước khi thiết lập chế độ xuất định kỳ để tránh bị tính phí ngoài dự kiến.
- ⚠ Để có thể sử dụng dịch vụ xuất và nhập được quản lý, phải bật tính năng thanh toán cho dự án Google Cloud. Chỉ những dự án Google Cloud đã bật tính năng thanh toán mới có thể sử dụng chức năng xuất và nhập.

3.3.2. *Backup và Restore dữ liệu*

- **Backup (Sao lưu):** Quá trình sao lưu chính là thực hiện thao tác Export đã nêu ở trên. Google Cloud cho phép tự động hóa quá trình này bằng cách thiết lập các công việc định kỳ (cron jobs) để chạy lệnh export hàng ngày hoặc hàng tuần, đảm bảo dữ liệu luôn có một bản sao lưu gần nhất.
- **Restore (Phục hồi):** Phục hồi dữ liệu là quá trình Import từ một bản sao lưu đã có trên Cloud Storage. Đây là thao tác quan trọng để khôi phục hệ thống sau khi xảy ra sự cố mất mát dữ liệu.

CHƯƠNG 4. TRIỂN KHAI TRUY VẤN VÀ KẾT NỐI ỨNG DỤNG

4.1. Tổng quan về cách triển khai truy vấn trong

Tất cả các truy vấn được triển khai thông qua mã nguồn JavaScript/React sử dụng Firebase SDK. Cách tiếp cận này cho phép ứng dụng thực hiện các truy vấn phức tạp, được tối ưu hóa và bảo mật một cách tự động.

Các truy vấn được tổ chức theo mô hình phân lớp:

- **Tầng Service (Service Layer):** Chứa các hàm thực hiện truy vấn trực tiếp với Firestore.
- **Tầng Hook (Custom Hooks):** Bao bọc logic truy vấn và cung cấp giao diện dễ sử dụng cho các component React.
- **Tầng Component:** Sử dụng các hook để lấy dữ liệu và hiển thị lên giao diện người dùng.

4.2. Các truy vấn cơ bản trong ứng dụng

Các truy vấn cơ bản bao gồm các thao tác CRUD cơ bản với từng document cụ thể. Dưới đây là ví dụ về cách triển khai thực tế:

Ví dụ 1: Gửi tin nhắn mới (Create)

```

1 import { collection, addDoc, serverTimestamp } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3
4 // Hàm gửi tin nhắn mới
5 const sendMessage = async (messageData) => {
6   // Tạo tham chiếu đến collection messages trong Firestore
7   const docRef = collection(db, 'messages');
8
9   // Thêm document mới với dữ liệu tin nhắn và thời gian tạo tự động
10  const result = await addDoc(docRef, {
11    ...messageData, // Dữ liệu tin nhắn (text, senderId, chatId, etc.)
12    createdAt: serverTimestamp(), // Thời gian tạo được Firestore tự động tạo
13  });
14
15  // Trả về ID của tin nhắn vừa tạo để sử dụng sau này
16  return result.id;
17 };

```

Ví dụ 2: Cập nhật thông tin phòng chat (Update)

```

1 import { doc, updateDoc, serverTimestamp } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3
4 // Hàm cập nhật tên phòng chat
5 const updateRoomName = async (roomId, newName) => {
6   // Tạo tham chiếu đến document phòng chat cụ thể
7   const roomRef = doc(db, 'rooms', roomId);
8
9   // Cập nhật tên mới và thời gian cập nhật
10  await updateDoc(roomRef, {
11    name: newName, // Tên phòng mới
12    updatedAt: serverTimestamp(), // Thời gian cập nhật
13  });
14 };

```

Ví dụ 3: Đọc thông tin người dùng (Read)



```
1 import { doc, getDoc } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3
4 // Hàm lấy thông tin người dùng
5 const getUserProfile = async (userId) => {
6   // Tạo tham chiếu đến document người dùng
7   const userRef = doc(db, 'users', userId);
8
9   // Lấy dữ liệu từ Firestore
10  const userDoc = await getDoc(userRef);
11
12  // Kiểm tra document có tồn tại không
13  if (userDoc.exists()) {
14    // Trả về dữ liệu người dùng
15    return userDoc.data();
16  }
17
18  // Trả về null nếu không tìm thấy người dùng
19  return null;
20 };
```

Ví dụ 4: Xóa tin nhắn (Delete)



```
1 import { doc, deleteDoc } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3
4 // Hàm xóa tin nhắn
5 const deleteMessage = async (messageId) => {
6   // Tạo tham chiếu đến document tin nhắn cần xóa
7   const messageRef = doc(db, 'messages', messageId);
8
9   // Xóa document khỏi Firestore
10  await deleteDoc(messageRef);
11  };
```

4.3. Các truy vấn nâng cao trong ứng dụng

Các truy vấn nâng cao sử dụng hàm *query()* kết hợp với *where()*, *orderBy()*, và *limit()* để lọc, sắp xếp và giới hạn kết quả:

Ví dụ 1: Tìm kiếm tin nhắn trong cuộc trò chuyện

```

1 import { collection, query, where, orderBy, limit, getDocs } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3
4 // Hàm tìm kiếm tin nhắn trong một cuộc trò chuyện
5 const searchMessagesInChat = async (chatId, searchTerm, limitCount = 50) => {
6   // Tạo tham chiếu đến collection messages
7   const messagesRef = collection(db, 'messages');
8
9   // Xây dựng truy vấn với điều kiện lọc và sắp xếp
10  const searchQuery = query(
11    messagesRef, // Collection cần truy vấn
12    where('chatId', '==', chatId), // Lọc theo ID cuộc trò chuyện
13    orderBy('createdAt', 'desc'), // Sắp xếp theo thời gian giảm dần (mới nhất trước)
14    limit(limitCount) // Giới hạn số kết quả trả về
15  );
16
17  // Thực hiện truy vấn và lấy kết quả
18  const querySnapshot = await getDocs(searchQuery);
19
20  // Xử lý kết quả truy vấn
21  const messages = [];
22  querySnapshot.forEach((doc) => {
23    const messageData = doc.data();
24    // Kiểm tra nội dung tin nhắn có chứa từ khóa tìm kiếm không
25    if (messageData.text && messageData.text.toLowerCase().includes(searchTerm.toLowerCase())) {
26      messages.push({ id: doc.id, ...messageData });
27    }
28  });
29
30  return messages;
31 };

```

Ví dụ 2: Lấy danh sách phòng chat của người dùng

```

1 import { collection, query, where, orderBy, getDocs } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3
4 // Hàm lấy danh sách phòng chat mà người dùng là thành viên
5 const getUserRooms = async (userId) => {
6   // Tạo tham chiếu đến collection rooms
7   const roomsRef = collection(db, 'rooms');
8
9   // Xây dựng truy vấn tìm phòng chat có người dùng là thành viên
10  const roomsQuery = query(
11    roomsRef, // Collection cần truy vấn
12    where('members', 'array-contains', userId), // Lọc phòng có userId trong mảng members
13    orderBy('lastMessageAt', 'desc') // Sắp xếp theo thời gian tin nhắn cuối cùng
14  );
15
16  // Thực hiện truy vấn
17  const querySnapshot = await getDocs(roomsQuery);
18
19  // Xử lý kết quả
20  const rooms = [];
21  querySnapshot.forEach((doc) => {
22    rooms.push({ id: doc.id, ...doc.data() });
23  });
24
25  return rooms;
26 };

```

Ví dụ 3: Lấy tin nhắn chưa đọc trong phòng chat


```

1 import { collection, query, where, orderBy, getDocs } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3
4 // Hàm lấy tin nhắn chưa đọc của người dùng trong phòng chat
5 const getUnreadMessages = async (roomId, userId, lastSeenTime) => {
6   // Tạo tham chiếu đến collection messages
7   const messagesRef = collection(db, 'messages');
8
9   // Xây dựng truy vấn lấy tin nhắn chưa đọc
10  const unreadQuery = query(
11    messagesRef, // Collection cần truy vấn
12    where('chatId', '==', roomId), // Lọc theo ID phòng chat
13    where('createdAt', '>', lastSeenTime), // Lọc tin nhắn gửi sau thời điểm xem cuối
14    orderBy('createdAt', 'asc') // Sắp xếp theo thời gian tăng dần
15  );
16
17  // Thực hiện truy vấn
18  const querySnapshot = await getDocs(unreadQuery);
19
20  // Xử lý kết quả
21  const unreadMessages = [];
22  querySnapshot.forEach((doc) => {
23    unreadMessages.push({ id: doc.id, ...doc.data() });
24  });
25
26  return unreadMessages;
27 };

```

4.4. Cập nhật giao diện người dùng tự động với Real-time Listeners

Đây là tính năng cốt lõi giúp ứng dụng có trải nghiệm thời gian thực:

Ví dụ triển khai Real-time Listener

```

1 import { collection, query, where, orderBy, limit, onSnapshot } from 'firebase/firestore';
2 import { db } from '../firebase/config';
3 import { useState, useEffect } from 'react';
4
5 // Custom Hook để lắng nghe tin nhắn trong phòng chat
6 const useChatMessages = (roomId) => {
7   // State để lưu trữ danh sách tin nhắn và trạng thái loading
8   const [messages, setMessages] = useState([]);
9   const [loading, setLoading] = useState(true);
10
11   useEffect(() => {
12     // Không thực hiện nếu không có roomId
13     if (!roomId) return;
14
15     // Tạo tham chiếu đến collection messages
16     const messagesRef = collection(db, 'messages');
17
18     // Xây dựng truy vấn lấy 50 tin nhắn gần nhất của phòng
19     const messagesQuery = query(
20       messagesRef,
21       where('chatId', '==', roomId),
22       orderBy('createdAt', 'desc'),
23       limit(50)
24     );
25
26     // Đăng ký listener thời gian thực với onSnapshot
27     const unsubscribe = onSnapshot(messagesQuery, (querySnapshot) => {
28       const newMessages = [];
29       querySnapshot.forEach((doc) => {
30         newMessages.push({ id: doc.id, ...doc.data() });
31       });
32
33       // Đảo ngược mảng để hiển thị từ tin nhắn cũ đến mới
34       setMessages(newMessages.reverse());
35       setLoading(false);
36     });
37
38     // Hàm cleanup để hủy đăng ký listener khi component unmount
39     return () => unsubscribe();
40   }, [roomId]);
41
42   // Trả về dữ liệu và trạng thái để component sử dụng
43   return { messages, loading };
44 };

```

Cách hoạt động:

1. Khi component sử dụng hook này, nó sẽ đăng ký một listener với hàm *onSnapshot*.
2. Mỗi khi có tin nhắn mới được gửi, Firestore server sẽ tự động đẩy cập nhật xuống.
3. Hook cập nhật state và React tự động render lại giao diện với dữ liệu mới.

4.5. Tối ưu hóa truy vấn với Firestore Indexes

Để các truy vấn phức tạp hoạt động hiệu quả, đặc biệt là những truy vấn sử dụng *where()* và *orderBy()* cùng lúc, Firestore yêu cầu phải định nghĩa các **composite indexes** (chỉ mục tổng hợp) trước. Các chỉ mục này được định nghĩa trong tệp **firestore.indexes.json**

```
1 {
2   "indexes": [
3     // Chỉ mục cho truy vấn lấy tin nhắn theo chatId và sắp xếp theo thời gian
4     {
5       "collectionGroup": "messages",
6       "queryScope": "COLLECTION",
7       "fields": [
8         {
9           "fieldPath": "chatId",
10          "order": "ASCENDING"
11        },
12        {
13          "fieldPath": "createdAt",
14          "order": "DESCENDING"
15        }
16      ]
17    },
18    // Chỉ mục cho truy vấn lấy phòng chat theo members và sắp xếp theo thời gian
19    {
20      "collectionGroup": "rooms",
21      "queryScope": "COLLECTION",
22      "fields": [
23        {
24          "fieldPath": "members",
25          "arrayConfig": "CONTAINS"
26        },
27        {
28          "fieldPath": "lastMessageAt",
29          "order": "DESCENDING"
30        }
31      ]
32    }
33  ],
34  "fieldOverrides": []
35 }
```

Cách hoạt động:

1. Khi ứng dụng thực hiện truy vấn có sử dụng nhiều trường lọc và sắp xếp, Firestore sẽ kiểm tra xem có chỉ mục phù hợp không.
2. Nếu không có chỉ mục, truy vấn sẽ thất bại với lỗi "Missing Index".

3. Khi có chỉ mục, truy vấn sẽ hoạt động nhanh chóng vì Firestore có thể sử dụng cấu trúc cây B để tìm kiếm hiệu quả.
4. Các chỉ mục được tạo tự động thông qua Firebase CLI hoặc có thể tạo thủ công trong Firebase Console.

Ví dụ, truy vấn `getUserRooms(userId)` sử dụng chỉ mục với *members* (array-contains) và *lastMessageAt* (descending) để nhanh chóng tìm tất cả phòng chat mà người dùng là thành viên và sắp xếp theo hoạt động gần đây nhất.

4.6. Cơ chế bảo mật và xác thực trong truy vấn

Tất cả truy vấn đều phải tuân thủ các quy tắc bảo mật được định nghĩa trong `firestore.rules`:

```
1 // Quy tắc bảo mật cho phòng chat - chỉ thành viên mới có thể đọc/ghi
2 match /rooms/{roomId} {
3   allow read, write: if request.auth != null &&
4     request.auth.uid in resource.data.members;
5 }
6
7 // Quy tắc bảo mật cho tin nhắn - người dùng đã đăng nhập có thể đọc/ghi
8 match /messages/{messageId} {
9   allow read, write: if request.auth != null;
10 }
```

4.7. Kết nối và cấu hình Firebase trong ứng dụng

Cấu hình Firebase SDK:



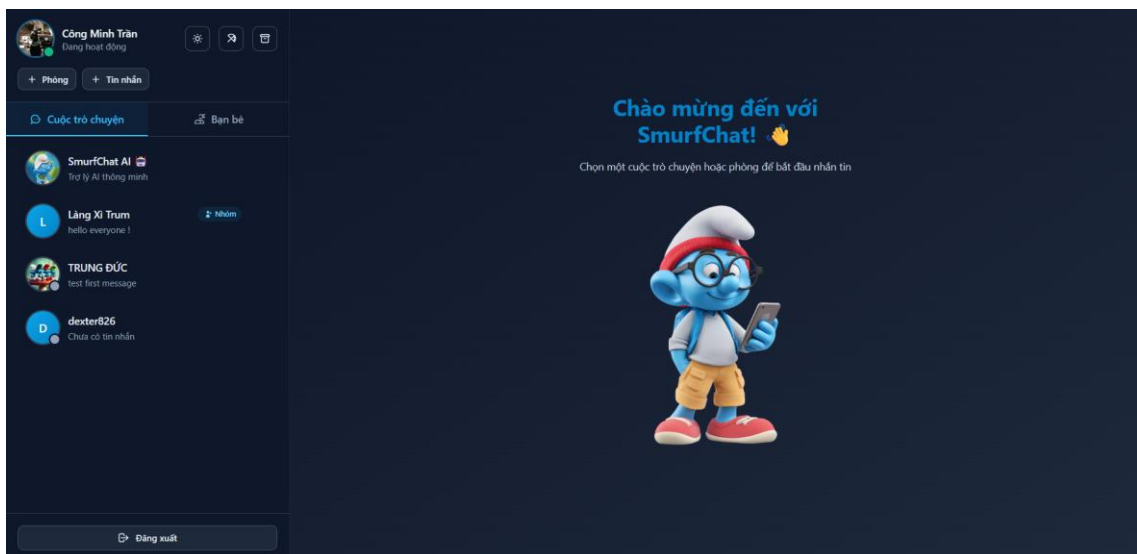
```
1  import { initializeApp } from 'firebase/app';
2  import { getAuth } from 'firebase/auth';
3  import { getFirestore } from 'firebase/firestore';
4
5  const firebaseConfig = {
6    apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
7    authDomain: process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
8    projectId: process.env.REACT_APP_FIREBASE_PROJECT_ID,
9    storageBucket: process.env.REACT_APP_FIREBASE_STORAGE_BUCKET,
10   messagingSenderId: process.env.REACT_APP_FIREBASE_MESSAGING_SENDER_ID,
11   appId: process.env.REACT_APP_FIREBASE_APP_ID,
12   measurementId: process.env.REACT_APP_FIREBASE_MEASUREMENT_ID,
13 };
14
15 const app = initializeApp(firebaseConfig);
16 const auth = getAuth(app);
17 const db = getFirestore(app);
18
19 export { db, auth };
20 export default app;
21
```

CHƯƠNG 5. DEMO ỨNG DỤNG VÀ KẾT QUẢ

5.1. Giao diện và các chức năng chính của ứng dụng

Ứng dụng có giao diện người dùng hiện đại, thân thiện, và hỗ trợ chế độ nền tối (Dark Mode) để bảo vệ mắt người dùng. Bố cục chính của ứng dụng được chia thành ba phần linh hoạt:

- **Sidebar (bên trái):** Chứa thông tin người dùng, thanh tìm kiếm, và danh sách các cuộc trò chuyện (bao gồm cả chat nhóm và chat 1-1). Người dùng có thể dễ dàng chuyển đổi giữa các cuộc trò chuyện.
- **Cửa sổ chat (ở giữa):** Là nơi hiển thị nội dung của cuộc trò chuyện đang được chọn, cho phép người dùng gửi và nhận tin nhắn.
- **Thông tin (bên phải - tùy chọn):** Hiển thị thông tin chi tiết về phòng chat hoặc người dùng đang trò chuyện, bao gồm danh sách thành viên, các tệp đã chia sẻ, và các cài đặt liên quan.

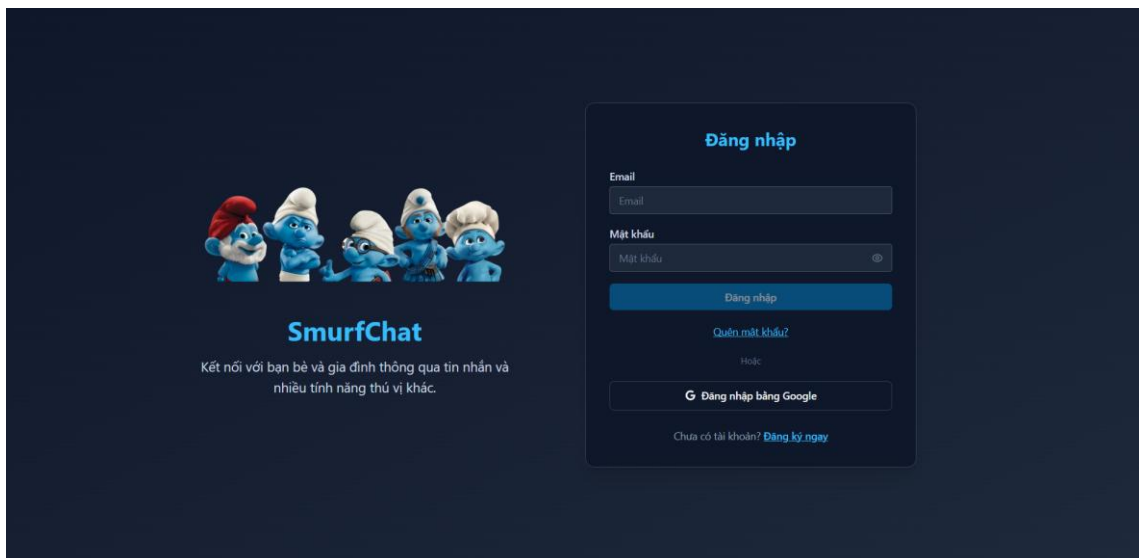


Hình 4. Giao diện chính của ứng dụng

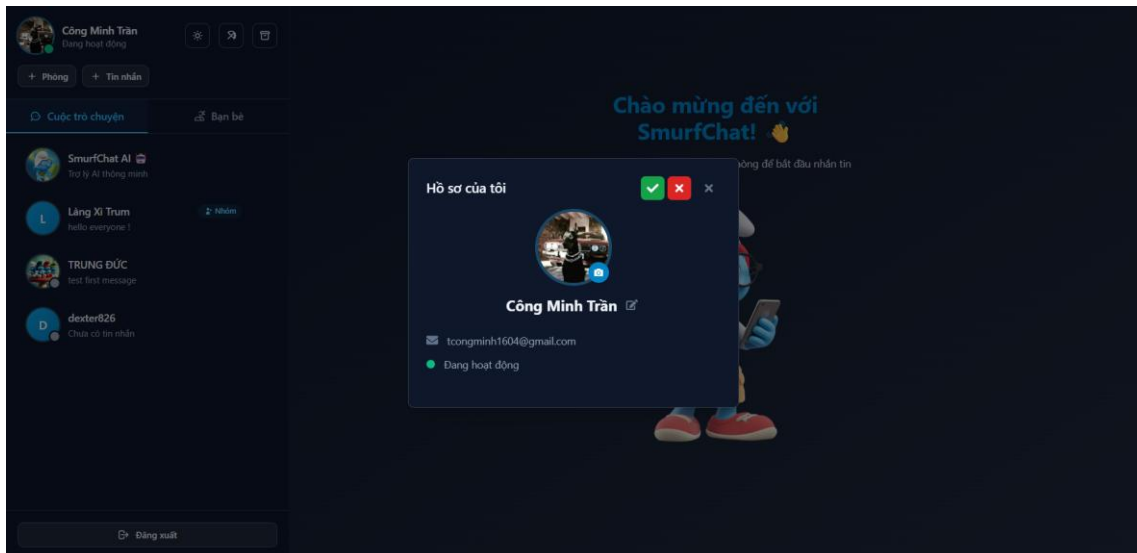
5.2. Chức năng quản trị người dùng

Việc quản lý người dùng được tích hợp chặt chẽ với dịch vụ Firebase Authentication.

- **Đăng ký và Đăng nhập:** Ứng dụng cung cấp hai phương thức xác thực linh hoạt:
 1. **Đăng nhập bằng tài khoản Google:** Một phương thức nhanh chóng, tiện lợi, chỉ với một cú nhấp chuột.
 2. **Đăng ký bằng Email và Mật khẩu:** Người dùng có thể tạo tài khoản mới với email, mật khẩu và tên hiển thị. Hệ thống có cơ chế xác thực dữ liệu đầu vào (validation) để đảm bảo tính hợp lệ.



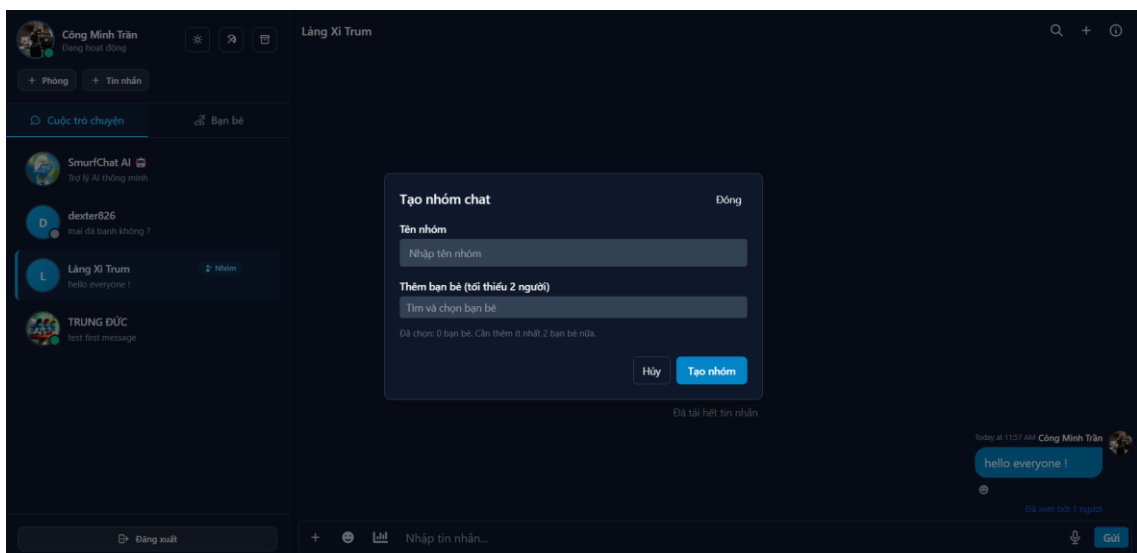
- **Đăng xuất:** Chức năng đăng xuất an toàn, có hộp thoại xác nhận để tránh người dùng vô tình thoát khỏi phiên làm việc.
- **Cập nhật thông tin cá nhân:** Người dùng có thể thay đổi tên hiển thị và ảnh đại diện của mình. Mọi thay đổi sẽ được cập nhật lên Firestore và hiển thị đồng bộ cho tất cả người dùng khác.



5.3. Chức năng thao tác trên CSDL

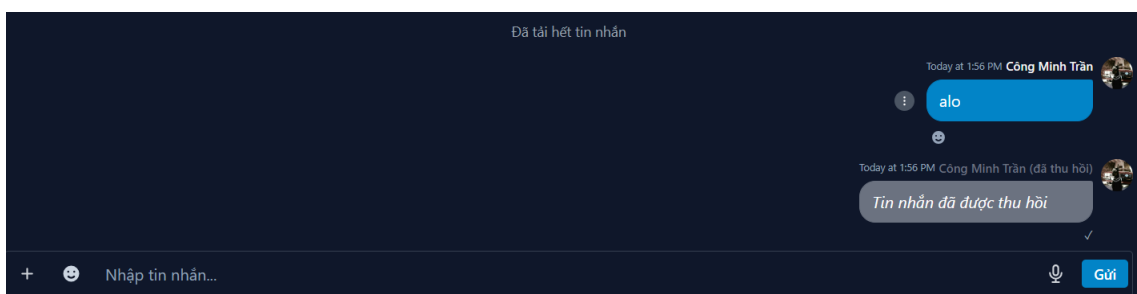
Các thao tác CSDL được người dùng thực hiện một cách trực quan thông qua các tương tác trên giao diện.

- **Tạo phòng chat mới (Thêm):** Người dùng có thể tạo một phòng chat mới bằng cách điền tên phòng, mô tả và chọn các thành viên từ danh sách bạn bè. Khi xác nhận, một document mới sẽ được tạo trong collection rooms.

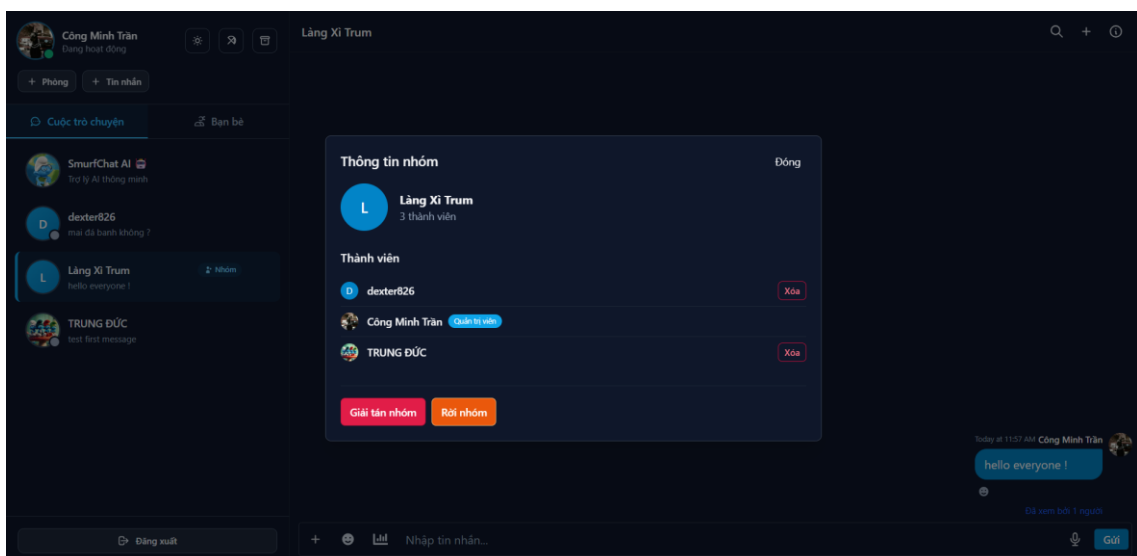


- **Gửi và Thu hồi tin nhắn:**

- **Gửi tin nhắn:** Việc gửi tin nhắn (văn bản, hình ảnh, tệp tin...) sẽ tạo một document mới trong collection messages. Nhờ cơ chế real-time, tin nhắn sẽ xuất hiện ngay lập tức trên màn hình của tất cả các thành viên trong cuộc trò chuyện.
- **Thu hồi tin nhắn:** Người dùng có thể thu hồi tin nhắn đã gửi. Về mặt kỹ thuật, hành động này không xóa document tin nhắn mà chỉ cập nhật trường isRecalled thành true, giúp giữ lại dấu vết và đảm bảo tính nhất quán.



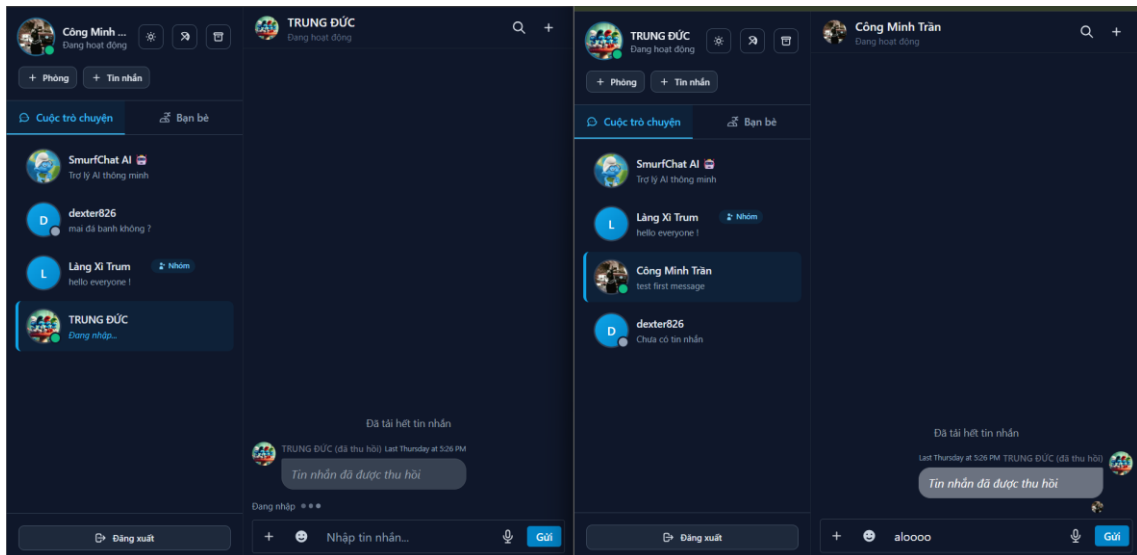
- **Thêm bạn bè và quản lý thành viên:** Người dùng có thể tìm kiếm và gửi lời mời kết bạn. Trong phòng chat, quản trị viên có quyền thêm hoặc xóa thành viên khỏi phòng. Các hành động này thực chất là cập nhật lại mảng members trong document của phòng chat tương ứng.



5.4. Chức năng quản lý chuyên biệt

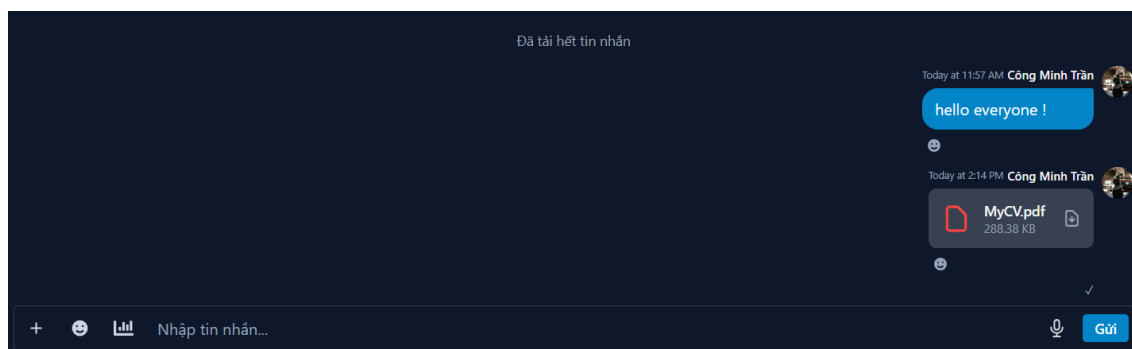
Ứng dụng được xây dựng để đáp ứng các nhu cầu đặc thù của sinh viên.

- **Hiển thị danh sách trò chuyện thông minh:** Sidebar không chỉ liệt kê các cuộc trò chuyện mà còn hiển thị tin nhắn cuối cùng, thời gian gửi, và một huy hiệu thông báo số lượng tin nhắn chưa đọc, giúp người dùng dễ dàng theo dõi.
- **Trạng thái hoạt động và tương tác thời gian thực:**
 - Ứng dụng hiển thị trạng thái online/offline chính xác của bạn bè.
 - Khi một người dùng đang gõ phím, một chỉ báo "Đang nhập..." sẽ xuất hiện, tăng cường tính tương tác cho cuộc trò chuyện.

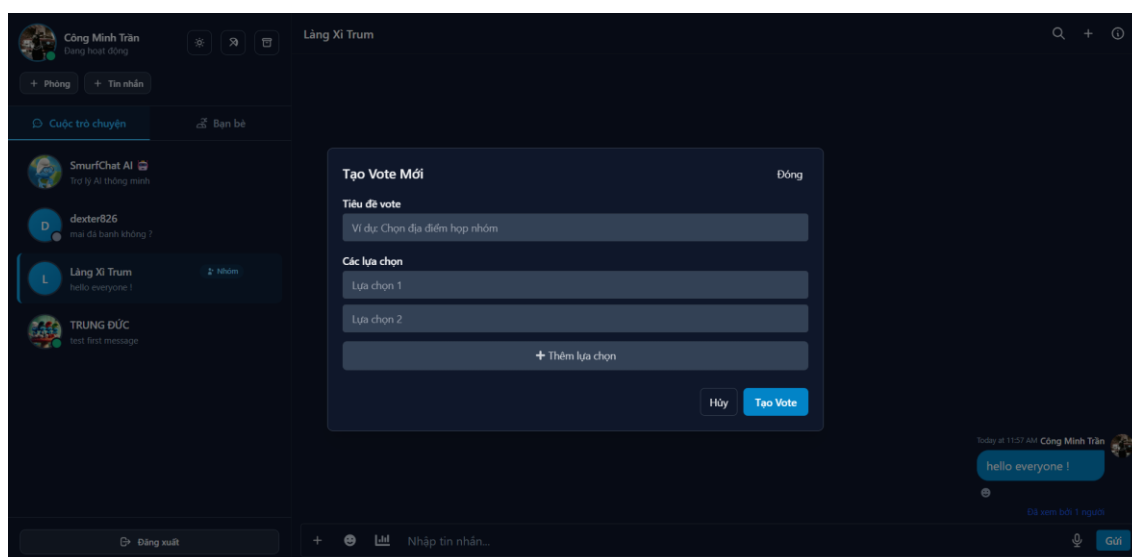


- **Chia sẻ đa phương tiện và tài liệu học tập:** Đây là một trong những tính năng mạnh mẽ nhất, được hỗ trợ bởi dịch vụ lưu trữ đám mây (Supabase Storage). Người dùng có thể:
 - Gửi nhiều loại tệp: Hình ảnh, video, tài liệu (PDF, DOCX, PPTX), và các tệp âm thanh.
 - Ghi âm trực tiếp: Gửi các tin nhắn thoại được ghi trực tiếp từ microphone.

- Chụp ảnh từ camera: Chụp và gửi ảnh ngay lập tức mà không cần rời khỏi ứng dụng.
- Chia sẻ vị trí: Gửi vị trí hiện tại, rất hữu ích cho các buổi họp nhóm hoặc gặp mặt.



- **Tạo cuộc bình chọn (Vote):** Trong các phòng chat nhóm, thành viên có thể tạo các cuộc bình chọn để nhanh chóng lấy ý kiến tập thể về một vấn đề nào đó, ví dụ như "Chọn thời gian họp nhóm".



CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Tóm tắt kết quả đạt được

Đồ án đã hoàn thành xuất sắc các mục tiêu đề ra. Bằng việc tìm hiểu sâu về CSDL NoSQL và áp dụng công cụ Firebase Firestore, đồ án đã xây dựng thành công ứng dụng chat nhóm với đầy đủ các chức năng từ cơ bản đến nâng cao, đáp ứng hiệu quả nhu cầu giao tiếp và học tập của sinh viên. Việc thiết kế một cấu trúc CSDL linh hoạt, tối ưu cho truy vấn và bảo mật đã được chứng minh qua hiệu năng và sự ổn định của ứng dụng.

6.2. Đánh giá ưu, nhược điểm của giải pháp

- **Ưu điểm:**

- Phát triển nhanh: Việc sử dụng Firebase (BaaS) đã giảm thiểu đáng kể thời gian và công sức cho việc xây dựng và quản lý backend.
- Trải nghiệm thời gian thực: Firestore cung cấp trải nghiệm người dùng mượt mà, tức thì, là yếu tố sống còn của một ứng dụng chat.
- Khả năng mở rộng: Giải pháp có khả năng mở rộng tốt, sẵn sàng đáp ứng khi số lượng người dùng tăng lên.
- Chi phí hiệu quả: Bậc miễn phí của Firebase rất hào phóng, phù hợp cho các dự án sinh viên và các sản phẩm ở giai đoạn đầu.

- **Nhược điểm:**

- Hạn chế trong truy vấn phức tạp: So với CSDL quan hệ, Firestore có những hạn chế nhất định đối với các truy vấn yêu cầu tổng hợp, phân tích dữ liệu phức tạp.
- Phụ thuộc vào nền tảng: Việc sử dụng sâu các dịch vụ của Firebase tạo ra sự phụ thuộc vào hệ sinh thái của Google.

6.3. Hướng phát triển trong tương lai

Dựa trên nền tảng vững chắc đã xây dựng, ứng dụng có thể được phát triển thêm nhiều tính năng giá trị:

1. **Tích hợp Lịch và Nhắc nhở:** Thêm tính năng tạo sự kiện, lịch biểu chung cho nhóm từ các tin nhắn (ví dụ: "Họp nhóm lúc 3h chiều mai"), và gửi thông báo nhắc nhở tự động.
2. **Video Call:** Tích hợp WebRTC hoặc một dịch vụ bên thứ ba để cho phép các cuộc gọi video trực tiếp trong ứng dụng.
3. **Mã hóa đầu cuối (End-to-End Encryption):** Nâng cao hơn nữa tính bảo mật và riêng tư bằng cách triển khai mã hóa tin nhắn ở phía client trước khi gửi lên server.
4. **Tích hợp Chatbot AI:** Xây dựng một chatbot hỗ trợ, có khả năng trả lời các câu hỏi thường gặp, tóm tắt cuộc trò chuyện, hoặc tìm kiếm tài liệu trong nhóm.

TÀI LIỆU THAM KHẢO

1. **Firestore Documentation.** (2024). *Cloud Firestore Documentation*. Lấy từ <https://firebase.google.com/docs/firestore>
2. **Firestore Documentation.** (2024). *Firestore Security Rules*. Lấy từ <https://firebase.google.com/docs/rules>
3. **Firestore Documentation.** (2024). *Firestore Authentication Documentation*. Lấy từ <https://firebase.google.com/docs/auth>
4. **MongoDB Inc.** (2024). *What is a NoSQL Database?*. Lấy từ <https://www.mongodb.com/nosql-explained>
5. **React Documentation.** (2024). *Hooks at a Glance*. Lấy từ <https://react.dev/reference/react>
6. **Supabase Documentation.** (2024). *Supabase Storage*. Lấy từ <https://supabase.com/docs/guides/storage>