Electronics and Computer Science

Faculty of Engineering and Physical Sciences

University of Southampton Malaysia

COMP3207

CLOUD APPLICATION DEVELOPMENT

Task 2: System Design

- CloudTalk

Name: Hin Jian Heng

Student ID: 33399948

Email: jhh1e22@soton.ac.uk

Lecturers	Dr. Syed Hamid Hussain Madni
	Dr. Muhamad Najib Zamri

Submission Date: 24/3/2025

0. Table of Contents

Table of Contents

1. Use Cases

Figure 1 is a CloudTalk use case diagram that visually represents the interactions between users and the system. Use cases with a blue background represent functions available only to unregistered users. Green use cases are accessible to regular users, moderators, and group admins. Yellow use cases are reserved for moderators and group admins, while red use cases are exclusively for group admins. The Firebase Realtime Database connect all the use case excepting authentication and log out part, but I didn’t connect all and focus more on real time data (message). I also not include Firebase Authentication as this directly connect to authentication part.

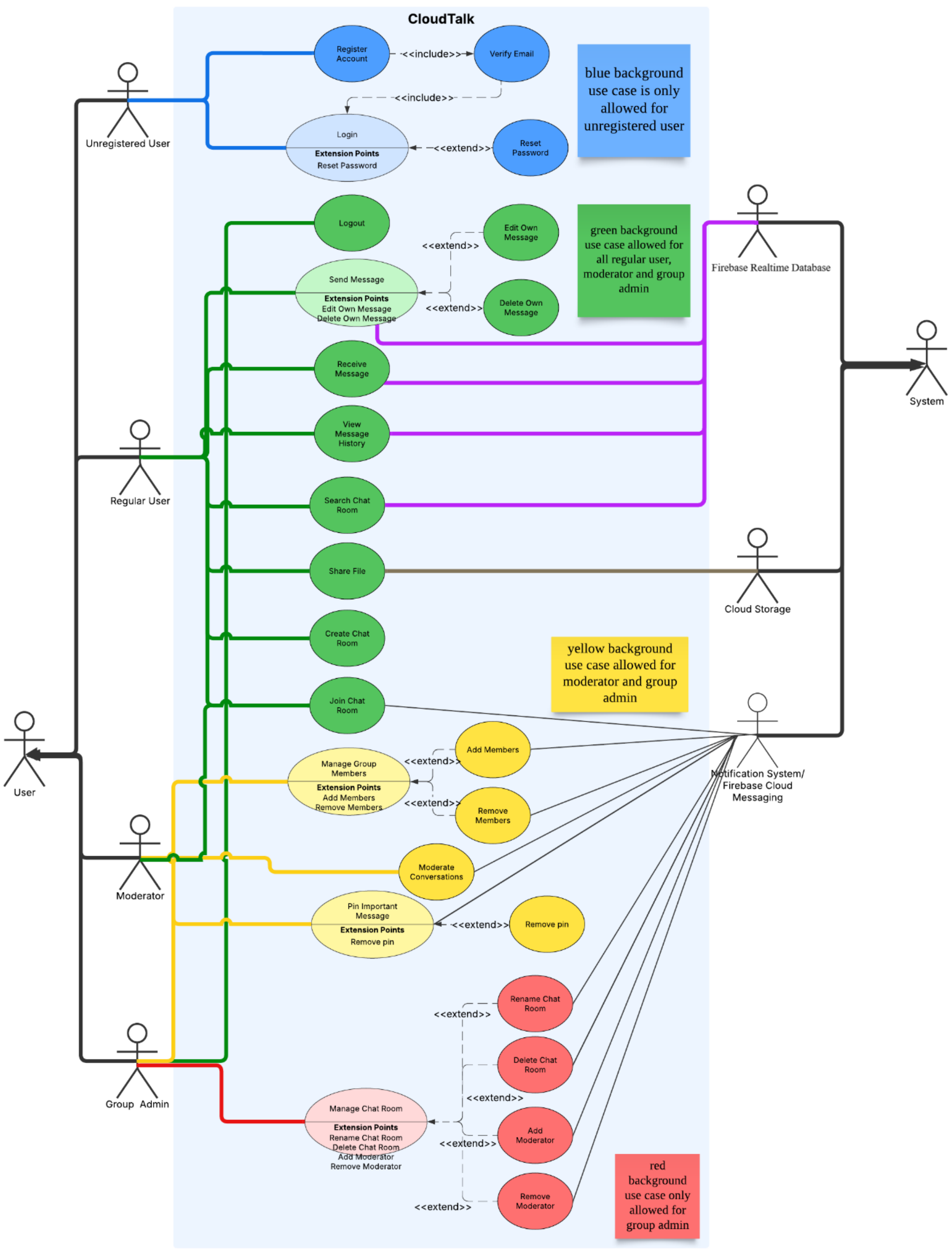


Figure 1: Use Case Diagram of CloudTalk

2. Entity-Relationship Diagram (ERD)

Figure 2 shows the entity-relationship diagram of CloudTalk, which consists of six main tables: User, Chatroom, Message, FCM tokens, Notifications and Activity Log. Since User and Chatroom have a many-to-many relationship, a Chatroom_User joint table is used. To improve clarity, yellow lines indicate Chatroom as a foreign key, while green lines indicate User as a foreign key. Password hash does not exist in Firebase Realtime Database, but in Firebase Authentication. However, I still put this attribute in it to make the user table more comprehensive.

In addition, because CloudTalk uses Firebase Realtime Database which is a non-relational database that stores data in a non-tabular format, that ERD is slightly different from SQL databases such as MySQL that store data in rule-based, relational tables. Figure 2.1 shows the SQL Entity-relationship diagram of CloudTalk and Figure 2.2 shows the NoSQL Entity-relationship diagram of CloudTalk. The dot lines in the Figure 2.2 means subcollection relationship. The template of the NoSQL ERD is get from (Datasen, 2025). The main differences between Figure 2.1 and Figure 2.2 include:

- Different data types, NoSQL can use objects or arrays. Besides that, strings and integers are not limited to the length.
- Moving the FCM tokens table to become one of the attributes of users. Because array data types are allowed, turning tokens into arrays can save computational time and memory
- Moving messages to become subcollections of chatroom. Because subcollection are allowed, messages are very dependent on chatroom as only when chatroom exists will messages exist, so I made messages one of the attributes of chatroom. Same as notifications to User.

Table Description:

- Chat Room User Joint Database: Stores basic information about users joining/leaving chat rooms and corresponding chatroom setting.
- Activity Log Database: Stores all operation logs (e.g., user joins/leaves, message deletions.)
- User: Stores user data
- FCMTokens: Stores tokens that will be used in FCM Cloud Messaging
- Notifications: Store Notifications of each user
- Messages: Store Messages of each cloud rooms
- Chat Room: Store Chat Room data

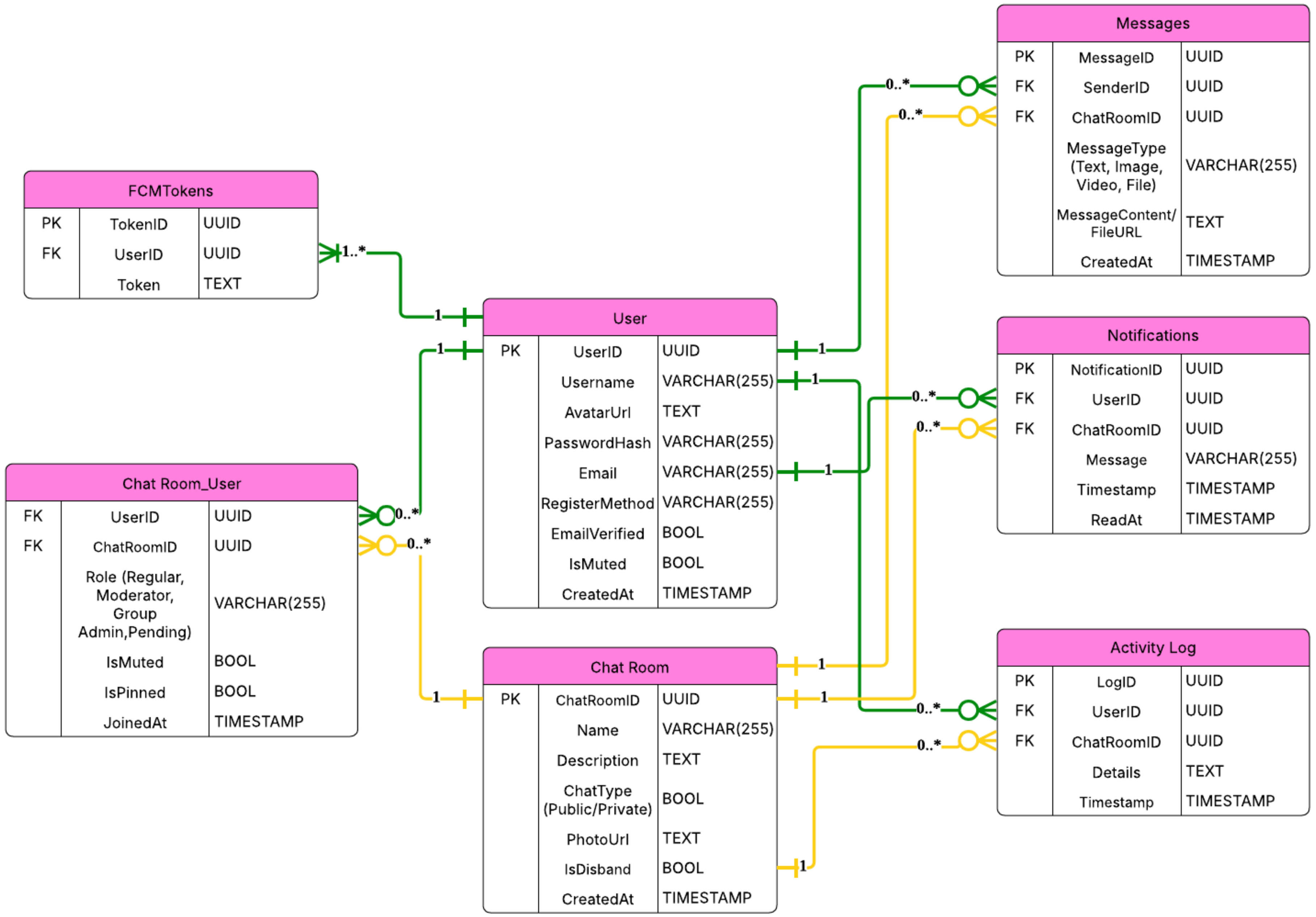


Figure 2.1: SQL Entity-relationship diagram of CloudTalk

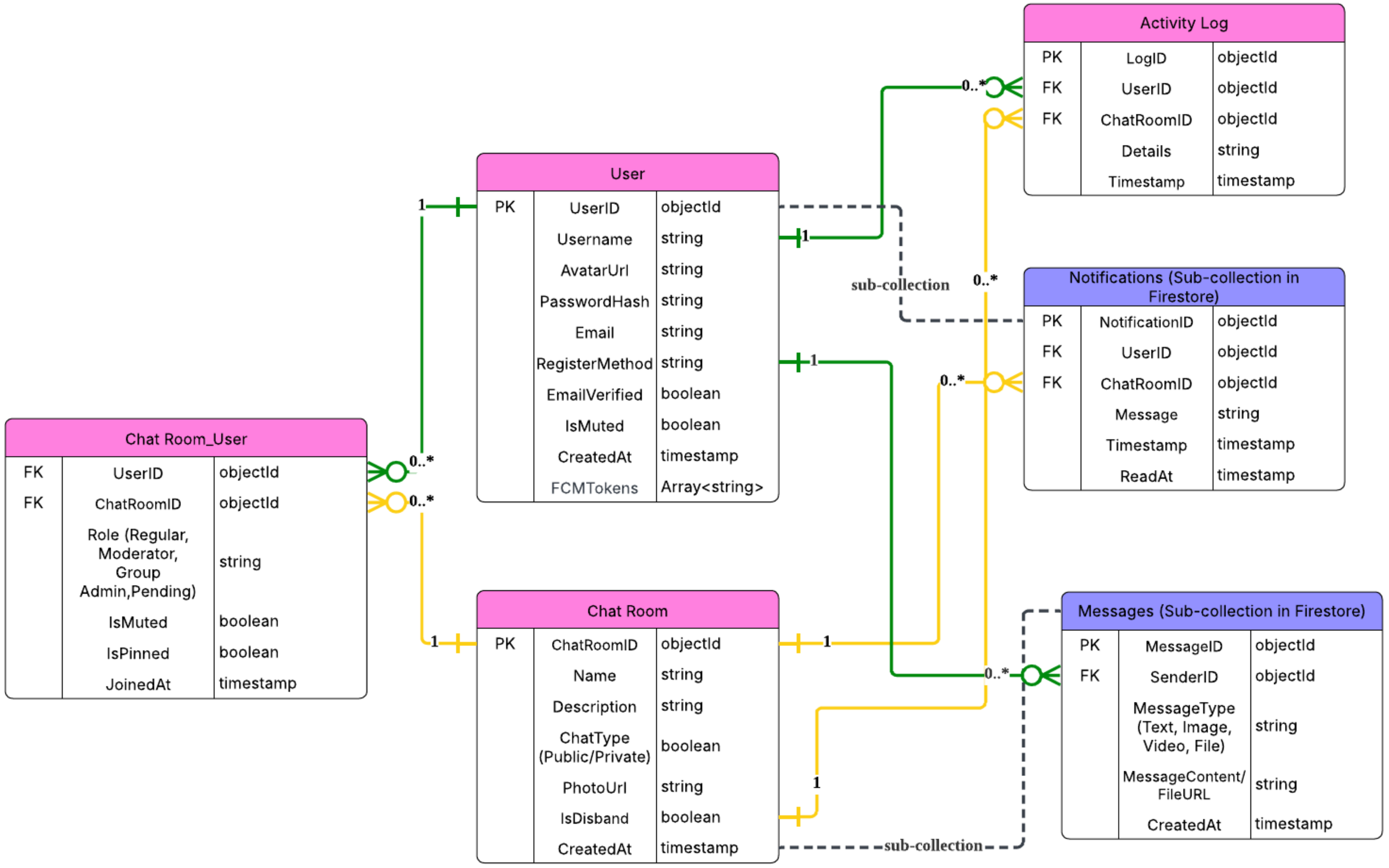


Figure 2.2: NoSQL Entity-relationship diagram of CloudTalk

3. Data Flow Diagram (DFD)

This section shows Data flow diagram which shows how data flows. So, the relationship between the users hard to be explained therefore will be explained here. Group Admin can perform all Moderator and Regular User actions. Moderator can perform all Regular User actions but can't manage rooms & group admin. Moderator handles content; Admin manages rooms & moderators.

3.1 Level 0 Data Flow Diagram

Figure 3.1 shows the Level 0 Data Flow Diagram of CloudTalk, providing an overview of the entire system. It includes three entities: Regular User, Moderator, and Group Admin, illustrating how they interact with the system.

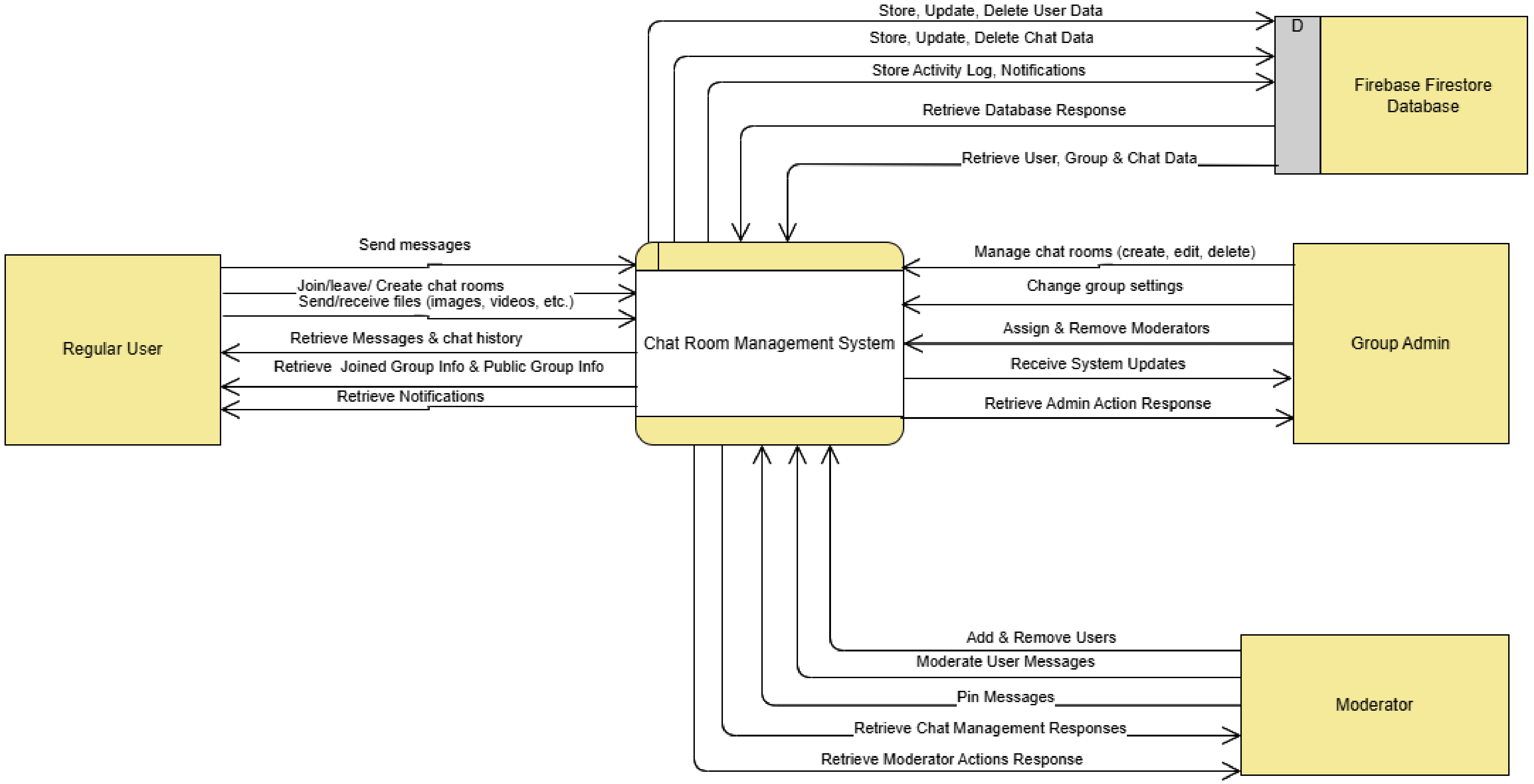

Store, Update, Delete User Data
Store, Update, Delete Chat Data
Store Activity Log, Notifications
Retrieve Database Response
Retrieve User, Group & Chat Data
D
Firebase Firestore Database
Send messages
Join/leave/ Create chat rooms
Send/receive files (images, videos, etc.)
Retrieve Messages & chat history
Retrieve Joined Group Info & Public Group Info
Retrieve Notifications
Regular User
Chat Room Management System
Manage chat rooms (create, edit, delete)
Change group settings
Assign & Remove Moderators
Receive System Updates
Retrieve Admin Action Response
Group Admin
Add & Remove Users
Moderate User Messages
Pin Messages
Retrieve Chat Management Responses
Retrieve Moderator Actions Response
Moderator

Figure 3.1: Level 0 data flow diagram of CloudTalk

3.2 Level 1 Data Flow Diagram

Figure 3.2 shows CloudTalk's Level 1 Data Flow Diagram, providing a more detailed view of the system by breaking down the main processes identified in the Level 0 DFD into sub-processes. To further illustrate the data flow in detail, Table 1 presents the Level 1 DFD in a table format. Event-driven architecture has been used in this diagram (the circle entity). The message storage data store includes Cloud Storage and Firebase Realtime Database. The orange line is chat room management data flow, the green line is user management data flow, blue line is message handling data flow, while black line is notification data flow. Dot line indicates flow to activity log database and it is less importance.

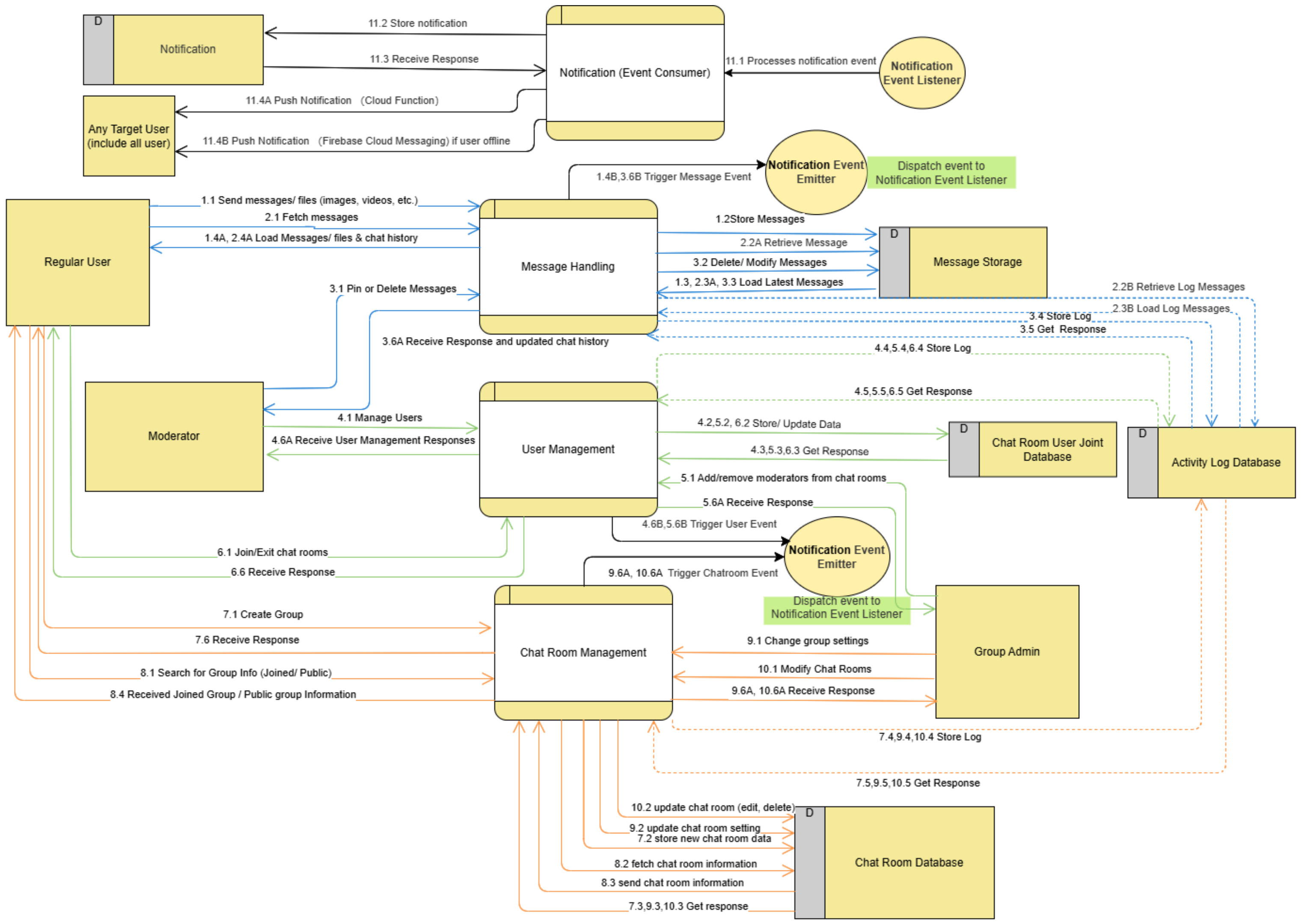

D
Notification
11.2 Store notification
11.3 Receive Response
Notification (Event Consumer)
11.1 Processes notification event
Notification Event Listener
Any Target User (include all user)
11.4A Push Notification (Cloud Function)
11.4B Push Notification (Firebase Cloud Messaging) if user offline
1.4B,3.6B Trigger Message Event
Notification Event Emitter
Dispatch event to Notification Event Listener
Regular User
1.1 Send messages/ files (images, videos, etc.)
2.1 Fetch messages
1.4A, 2.4A Load Messages/ files & chat history
3.1 Pin or Delete Messages
Message Handling
1.2Store Messages
2.2A Retrieve Message
3.2 Delete/ Modify Messages
1.3, 2.3A, 3.3 Load Latest Messages
D
Message Storage
2.2B Retrieve Log Messages
2.3B Load Log Messages
3.4 Store Log
3.5 Get Response
3.6A Receive Response and updated chat history
4.4,5.4,6.4 Store Log
4.5,5.5,6.5 Get Response
Moderator
4.1 Manage Users
4.6A Receive User Management Responses
User Management
4.2,5.2, 6.2 Store/ Update Data
4.3,5.3,6.3 Get Response
D
Chat Room User Joint Database
D
Activity Log Database
5.1 Add/remove moderators from chat rooms
5.6A Receive Response
4.6B,5.6B Trigger User Event
Notification Event Emitter
Dispatch event to Notification Event Listener
6.1 Join/Exit chat rooms
6.6 Receive Response
9.6A, 10.6A Trigger Chatroom Event
7.1 Create Group
7.6 Receive Response
8.1 Search for Group Info (Joined/ Public)
8.4 Received Joined Group / Public group Information
Chat Room Management
Group Admin
9.1 Change group settings
10.1 Modify Chat Rooms
9.6A, 10.6A Receive Response
7.4,9.4,10.4 Store Log
7.5,9.5,10.5 Get Response
10.2 update chat room (edit, delete)
9.2 update chat room setting
7.2 store new chat room data
8.2 fetch chat room information
8.3 send chat room information
7.3,9.3,10.3 Get response
D
Chat Room Database

Figure 3.2: Level 1 data flow diagram of CloudTalk

	Data Flow
1	Send messages/ files (images, videos, etc.) (Regular User → Message Handling) → Store messages (Message Handling → Message Storage) → Load latest Message (Message Storage → Message Handling) Route1 →Load Messages/ files & chat history (Message Handling → Regular User) Route 2 → Trigger Message Event (Message Handling → Notification Event Emitter → Dispatch event to Notification Event Listener
2	Fetch messages (Regular User → Message Handling Route1 → Retrieve Message (Message Handling → Message Storage) → Load Latest Messages (Message Storage → Message Handling) Route 2 → Retrieve Log Messages (Message Handling → Activity Log Database) → Load Log Messages (Activity Log Database → Message Handling) →Load Messages/ files & chat history (Message Handling → Regular User)
3	Delete/ Pin messages (Moderators → Message Handling) → Delete/Modify messages (Message Handling → Message Storage) → Load Latest Messages (Message Storage → Message Handling) → Store Log (Message Handling → Activity Log Database) → Get Response (Activity Log Database → Message Handling) Route1 → Receive response and update chat history (Message Handling → Moderators) Route 2 → Trigger Message Event (Message Handling → Notification Event Emitter → Dispatch event to Notification Event Listener
4	Add/Remove users from chat rooms (Moderator → User Management) → Store/Update data (User Management → Chat Room User Joint Database) → Get Response (Chat Room User Joint Database → User Management) → Store Log (User Management → Activity Log Database) → Receive Response (Activity Log Database→ User Management) Route1 → Receive User Management Responses (User Management → Moderator) Route 2 → Trigger Message Event (User Management → Notification Event Emitter → Dispatch event to Notification Event Listener

5	Add/Remove moderators from chat rooms (Group Admin→ User Management) → Store/Update data (User Management → Chat Room User Joint Database) → Get Response (Chat Room User Joint Database → User Management) → Store Log (User Management → Activity Log Database) → Send Response (Activity Log Database→ User Management) Route1 → Receive Responses (User Management → Group Admin) Route 2 → Trigger Message Event (User Management → Notification Event Emitter → Dispatch event to Notification Event Listener
6	Join/Exit chat rooms (Regular User → Chat Room Management) → Store/Update data (User Management → Chat Room User Joint Database) → Get Response (Chat Room User Joint Database → User Management) → Store Log (User Management → Activity Log Database) → Get Response (Activity Log Database→ User Management) → Receive Responses (Regular User → Group Admin)
7	Create Group (Regular User → Chat Room Management) → Store new chat room data (Chat Room Management → Chat Room Database) → Get Response (Chat Room Database → Chat Room Management) → Store Log (Chat Room Management → Activity Log Database) → Get Response (Activity Log Database → Chat Room Management) → Receive Responses (Chat Room Management→ Regular User)
8	Search for group information (Regular User → Chat Room Management) → Fetch chat room information (Chat Room Management → Chat Room Database) → Send chat room information (Chat Room Database → Chat Room Management) → Receive Joined Group / Public Group information (Chat Room Management → Regular User)
9	Change group settings (Group Admin → Chat Room Management) → Update chat room settings (Chat Room Management → Chat Room Database) → Get Response (Chat Room Database → Chat Room Management) → Store Log (Chat Room Management → Activity Log Database) → Get Response (Activity Log Database → Chat Room Management) Route1 → Receive Responses (Chat Room Management→ Group Admin) Route 2 → Trigger Message Event (Chat Room Management→ Notification Event Emitter → Dispatch event to Notification Event Listener
10	Modify chat rooms (edit, delete) (Group Admin → Chat Room Management) → Update chat room data (Chat Room Management → Chat Room Database) → Get Response (Chat Room Database → Chat Room Management) → Store Log (Chat Room Management → Activity Log Database) → Get Response (Activity Log Database → Chat Room Management)

	Route1 → Receive Responses (Chat Room Management→ Group Admin) Route 2 → Trigger Message Event (Chat Room Management→ Notification Event Emitter → Dispatch event to Notification Event Listener
11	Trigger notification (send to all the members in the group or related users) Processes notification event (Notification Event Listener→ Notification (Event Consumer)) → Store notification (Notification (Event Consumer) → Notification database) → Receive Response(Notification database→ Notification (Event Consumer)) Route1 Push Notification （Cloud Function） (Notification (Event Consumer) → target user) Route 2 (only if user is onle) Push Notification （Firebase Cloud Messaging) (Notification (Event Consumer) → target user)

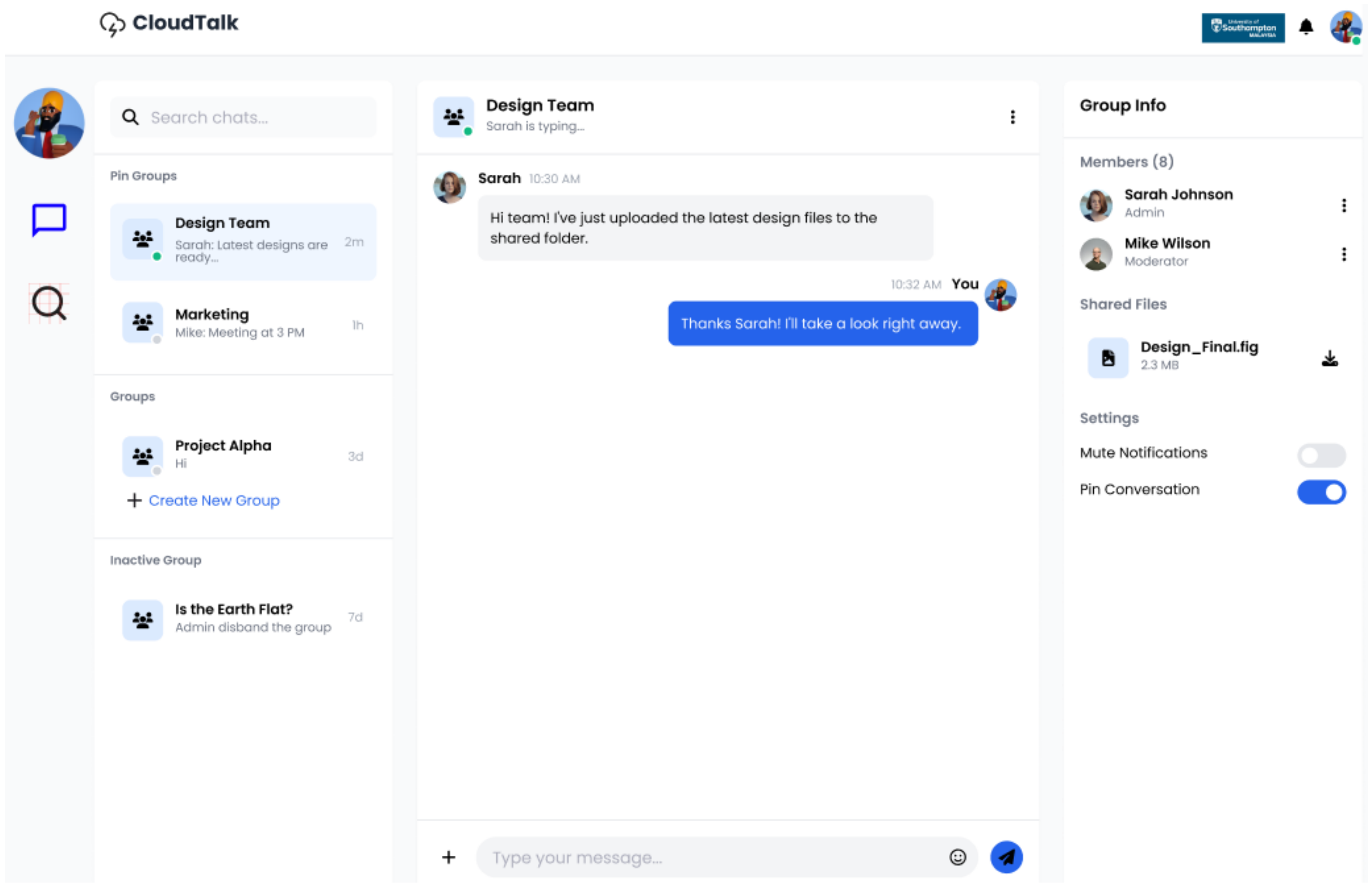
Table 1: Level 1 Data Flow Diagram Description

4. Design Patterns

4.1 Simple Frontend Prototype

4.1.1 Home Page

Figure 5 shows a prototype of the user interface for the CloudTalk home page. It includes the main functions of the application, such as sending messages, viewing messages, and accessing group information.



Design Team — Sarah: Latest designs are ready... — 2m

Marketing — Mike: Meeting at 3 PM — 1h

Project Alpha — Hi — 3d

Is the Earth Flat? — Admin disband the group — 7d

Design Team
Sarah is typing...

Sarah 10:30 AM
Hi team! I've just uploaded the latest design files to the shared folder.

10:32 AM You
Thanks Sarah! I'll take a look right away.

Sarah Johnson — Admin

Mike Wilson — Moderator

Design_Final.fig — 2.3 MB

Mute Notifications — off

Pin Conversation — on

Figure 4.1: user interface of the CloudTalk home page

4.1.2 Search Page

Figure 6 shows a prototype of the user interface for the CloudTalk search page. The main functions of the search page include searching for public groups, viewing public group information, and joining groups.

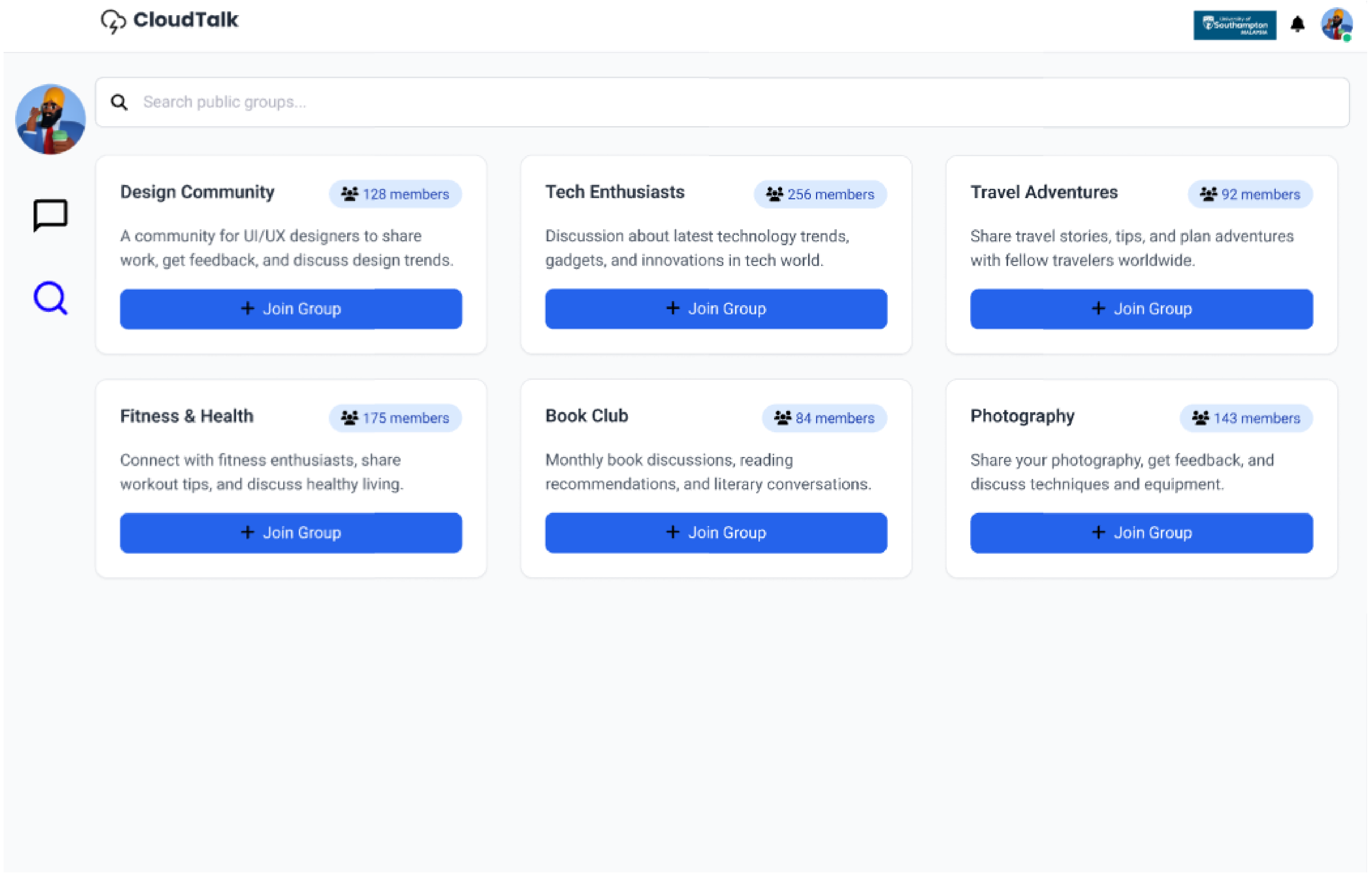
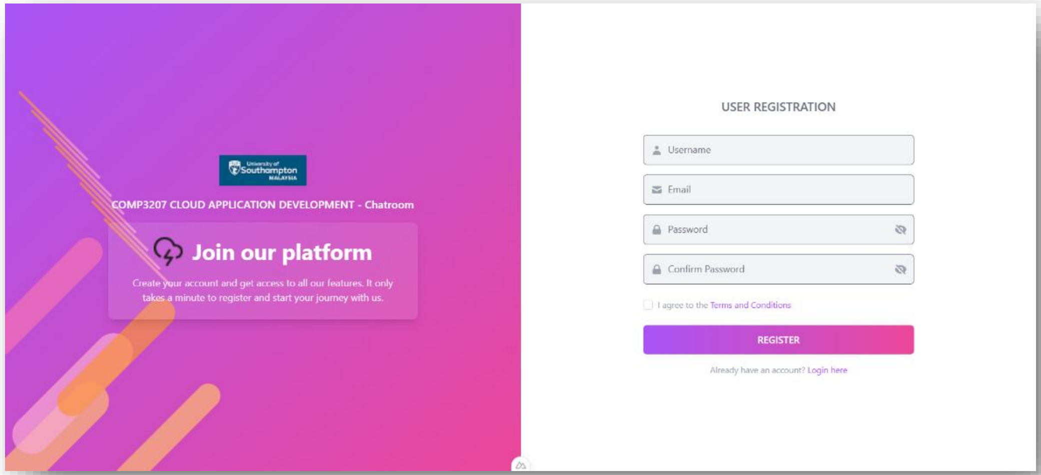

CloudTalk
Search public groups...
Design Community
128 members
A community for UI/UX designers to share work, get feedback, and discuss design trends.
+ Join Group
Tech Enthusiasts
256 members
Discussion about latest technology trends, gadgets, and innovations in tech world.
+ Join Group
Travel Adventures
92 members
Share travel stories, tips, and plan adventures with fellow travelers worldwide.
+ Join Group
Fitness & Health
175 members
Connect with fitness enthusiasts, share workout tips, and discuss healthy living.
+ Join Group
Book Club
84 members
Monthly book discussions, reading recommendations, and literary conversations.
+ Join Group
Photography
143 members
Share your photography, get feedback, and discuss techniques and equipment.
+ Join Group

Figure 4.2: user interface of the CloudTalk search page

4.1.3 Current Pages

In this section I will show my current CloudTalk main user interface. However, this will not be the final version. Figure 4.3 shows the register page, figure 4.4 shows the login page, figure 4.5 shows the home page, figure 4.6 shows the search page.



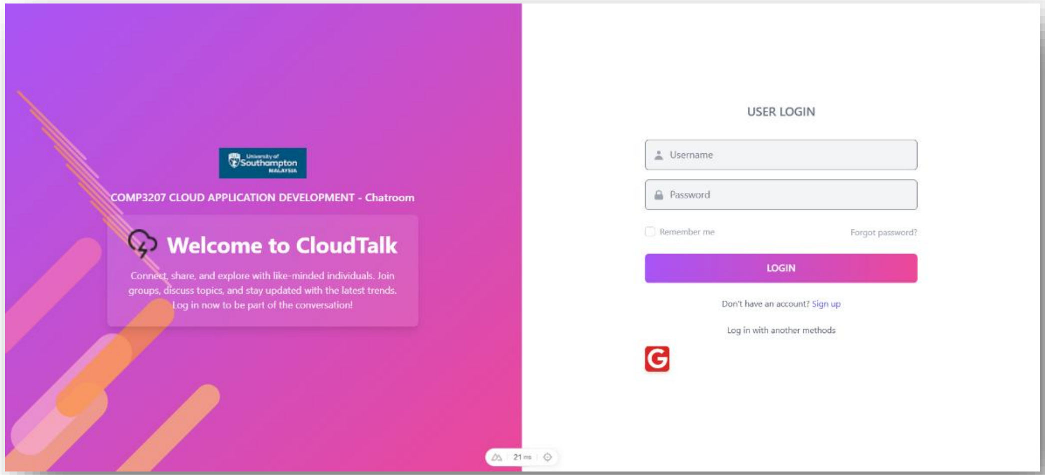
COMP3207 CLOUD APPLICATION DEVELOPMENT - Chatroom

Join our platform

Create your account and get access to all our features. It only takes a minute to register and start your journey with us.

USER REGISTRATION
Username
Email
Password
Confirm Password
I agree to the Terms and Conditions
REGISTER
Already have an account? Login here

Figure 4.3: user interface of the CloudTalk register page



COMP3207 CLOUD APPLICATION DEVELOPMENT - Chatroom

Welcome to CloudTalk

Connect, share, and explore with like-minded individuals. Join groups, discuss topics, and stay updated with the latest trends. Log in now to be part of the conversation!

Figure 4.4: user interface of the CloudTalk login page

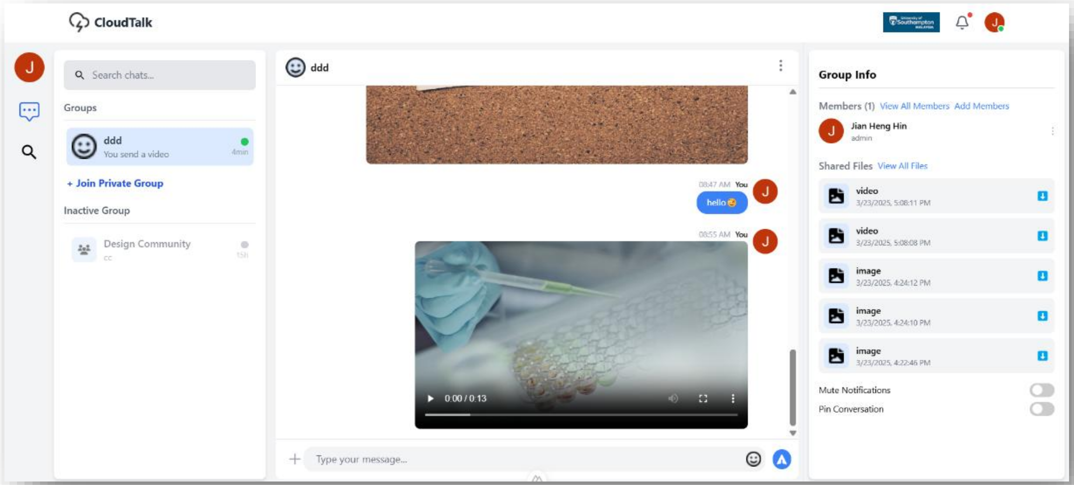


Figure 4.5: current user interface of the CloudTalk home page

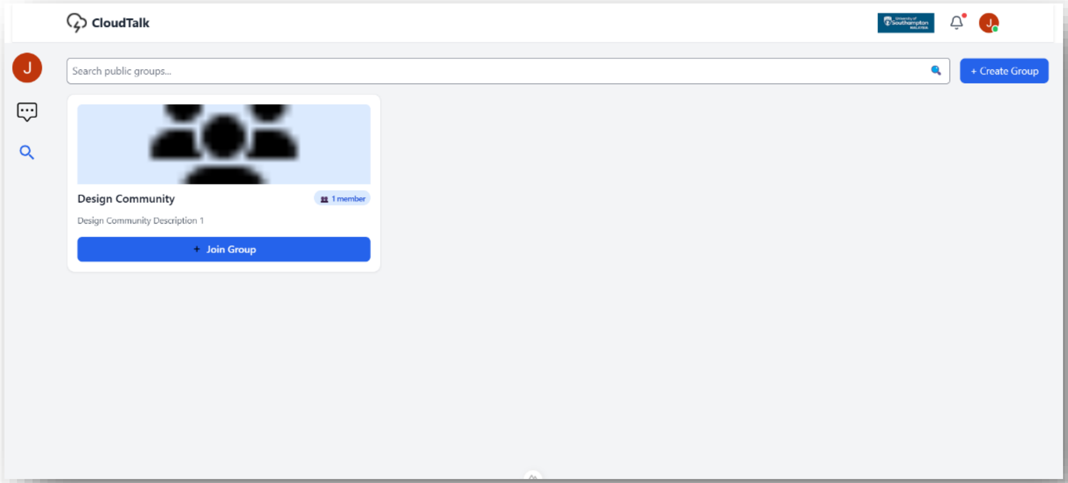


Figure 4.6: current user interface of the CloudTalk search page

4.2 Design Pattern Used in CloudTalk

Table 2 clearly presents the design patterns applied in CloudTalk, along with their purpose and application. The five design patterns used in this application include MVC (Model-View-Controller), Microservices, Serverless, Event-Driven, and Singleton, each serving different aspects of the system.

DESIGN PATTERN	PURPOSE	APPLICATION IN CLOUDTALK
MVC (MODEL-VIEW-CONTROLLER)	Separate business logic, UI logic, and input logic, so it can easily be maintained and extended. Besides that, it is also Search Engine Optimization (SEO) Friendly (GeeksforGeeks, 2024b).	**Model**: Firebase Realtime Database, Cloud Storage stores messages and user data. **View**: Nuxt.js frontend displays chat, user status, and notifications. **Controller**: Node.js backend, Cloud Functions, and Firebase Authentication handle authentication, message processing, and notifications.
MICROSERVICES	Break down the application into smaller, loosely coupled services, so that reliability and scalability can be ensured (GeeksforGeeks, 2025a).	Authentication, chat messaging, and notifications run as independent services and connect via event-driven triggers and API calls **Firebase Authentication**: User Management & Authentication **Firebase** Realtime Database: chat messaging & message storage **Firebase Cloud Messaging**: notification
SERVERLESS	Build and run applications without having to manage servers, so that it is more cost-efficient, have faster development and deployment, and automatic scaling (RedHat, 2025).	**Firebase Functions** handles most of the functionality in CloudTalk, there is no need to manage the server
EVENT-DRIVEN	By using event-driven, various microservices can communicate with each other by generating, identifying and responding to events. It allows real-time processing, greater flexibility and higher scalability (GeeksforGeeks, 2024a).	Two possible event-driven use cases New message event -> **Firebase Cloud Messaging** sends push notifications.

		User joins a chat room -> **System** automatically sends a welcome message.
SINGLETON	By ensuring a class has only one instance and provides a global access point to it. This is ideal for scenarios that require centralized control and helps reduce memory usage (GeeksforGeeks, 2025b).	Chat Manager: Ensures that only a single instance handles real-time chat operations, increasing reliability and preventing duplicate or missed messages. Notification Manager: By using a singleton design pattern for the notification manager, this can prevent duplicate notifications.

Table 2: Design Patterns applied in CloudTalk

Focus on event-driven architecture or serverless architecture?

Considering the characteristics of CloudTalk, it will focus more on **event-driven architecture**. However, **serverless architecture** still remain crucial.

Several key features of CloudTalk, such as real-time messaging, user activity logs, and notifications, rely heavily on real-time event triggering and processing. For example,

- When a user sends a message, a trigger event pushes updated message to the user.
- When a user joins or leaves a chat room, a notification update is triggered.
- When a user's role changes, a permission update is triggered.

These features are very suitable for event-driven architecture as they are executed in response to events. This reduces the need for constant listening, and enabling interaction in real time, effectively reducing the burden on the application and improving performance.

Nevertheless, when performing tasks such as authentication and message storage, serverless architecture is still required to reduce the burden of infrastructure management. For example:

- Use Firebase Authentication to simplify the user authentication process.
- Use Realtime Database and Cloud Storage to obtain auto-scaling storage capabilities.
- Use Firebase Cloud Messaging (FCM) to implement cross-platform push notifications

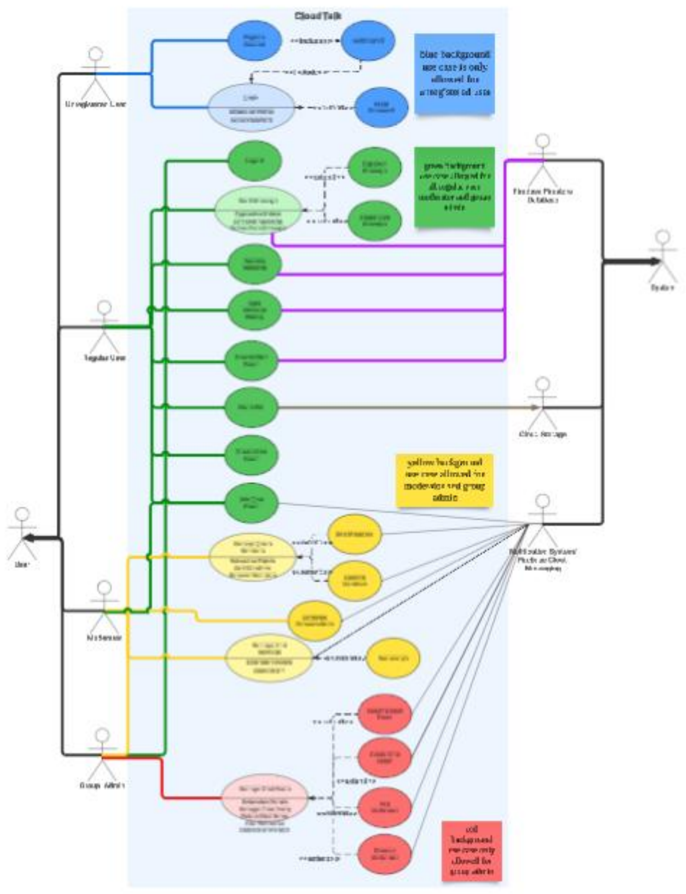
Therefore, in practical applications, combining the two will make it a better choice. For example:

- Use Firebase Cloud Functions (Serverless) to listen to Realtime Database events (Event-Driven) and trigger FCM pushes.

By having both architectures but focusing more on the event-driven architecture, the application can have better performance and be more cost-effective.

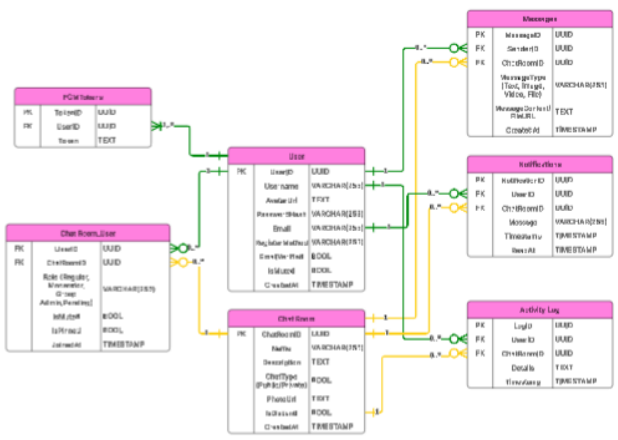
5. Appendix

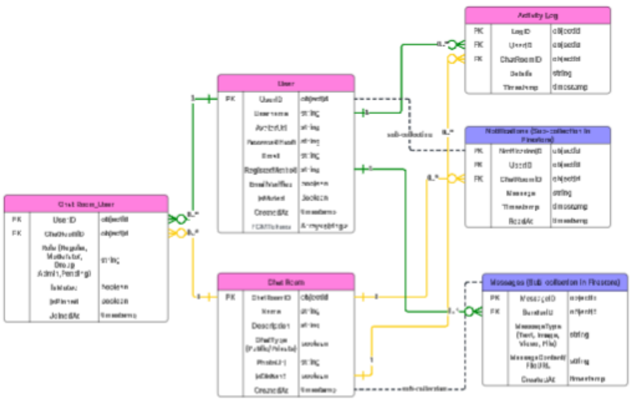
5.1 Use Case Diagram Link



Use Case Diagram link: https://lucid.app/lucidchart/c4902a71-dcba-44b7-a94a-dc842cc2950b/edit?viewport_loc=-3347%2C-114%2C8500%2C3481%2C.Q4MUjXso07N&invitationId=inv_c05485ef-ff2c-44ea-9825-473b7a2116f0

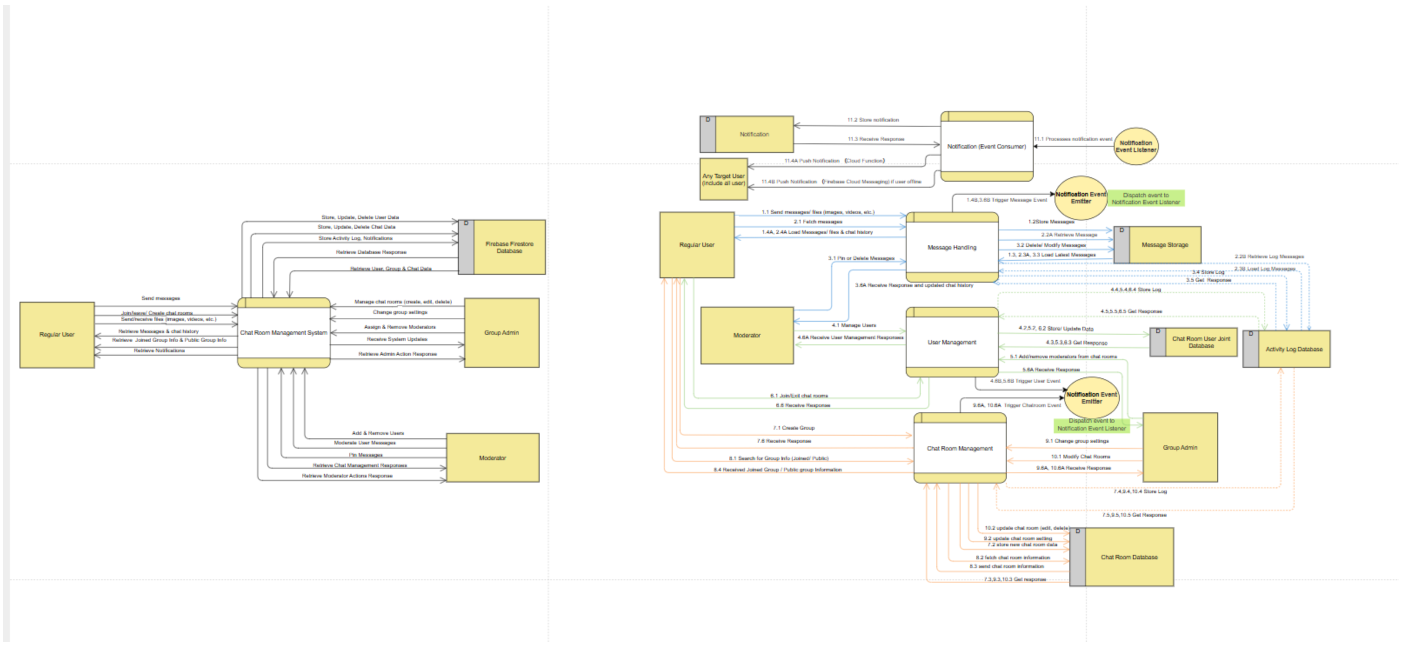
5.2 Entity-Relationship Diagram (ERD) Link




Activity Log
User
Notifications (Sub-collection in Firestore)
Chat Room_User
Chat Room
Messages (Sub collection in Firestore)

Entity-Relationship Diagram link: https://lucid.app/lucidchart/fe9c109e-ee83-44bf-983c-c8ed493bb079/edit?viewport_loc=-438%2C-471%2C1895%2C825%2C0_0&invitationId=inv_760d8a0e-165c-4e1b-8be8-49b12e577d7b

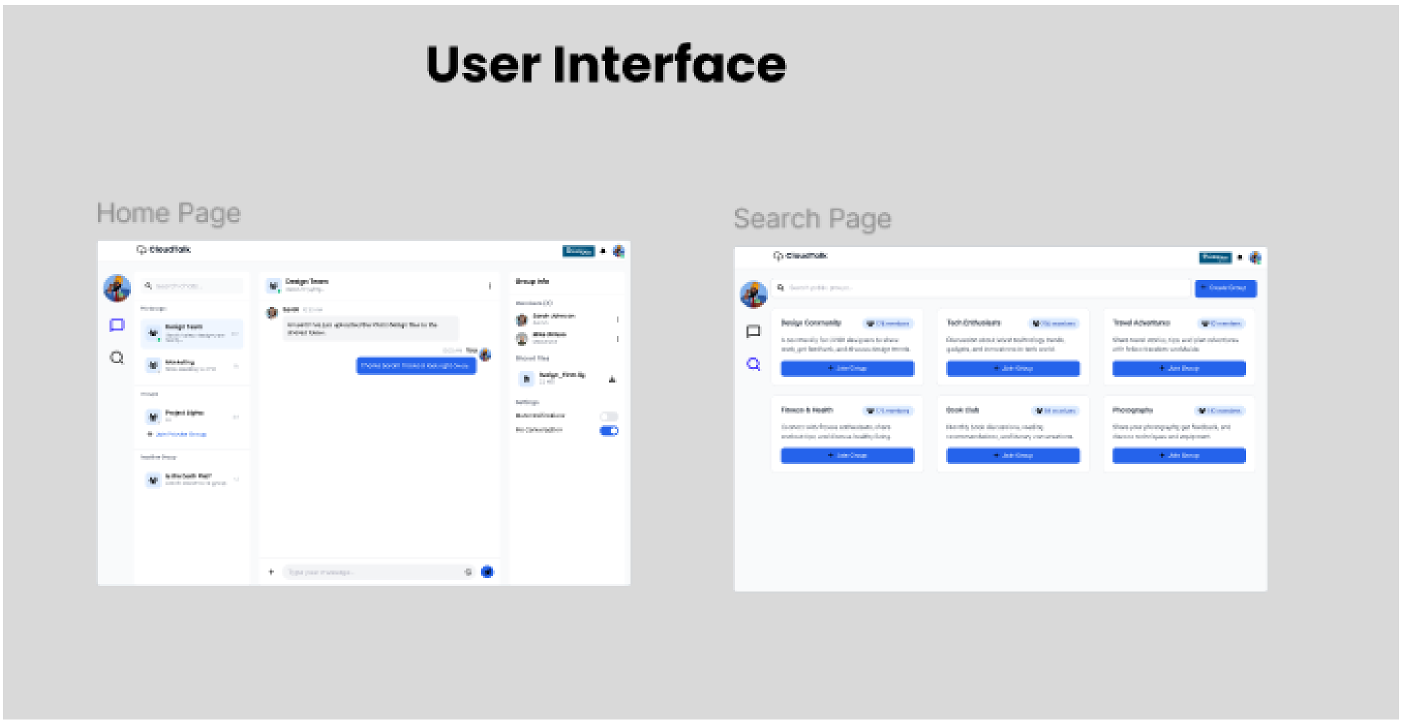
5.3 Data Flow Diagram (DFD)



Regular User
Chat Room Management System
Firebase Firestore Database
Group Admin
Moderator
Notification
Notification (Event Consumer)
Notification Event Listener
Notification Event Emitter
Any Target User (include all user)
Message Handling
Message Storage
User Management
Chat Room User Joint Database
Activity Log Database
Chat Room Management
Chat Room Database

Data Flow Diagram Link: https://online.visual-paradigm.com/share.jsp?id=333934323234362d32

5.4 Simple Frontend User Interface Prototype Link



Frontend User Interface Prototype Link:
https://www.figma.com/design/LAo3D7if8miDRgNjIFAKOy/CloudTalk-simple-prototype?node-id=0-1&t=HGkPFV1PSYblN09X-1

6. Reference

Datasen. (2025). *NoSQL Data Modeling*. https://www.datensen.com/blog/data-modeling/nosql-data-modeling/

GeeksforGeeks. (2024a). *Event-Driven Architecture - System Design*. https://www.geeksforgeeks.org/event-driven-architecture-system-design/

GeeksforGeeks. (2024b). *MVC Framework Introduction*. https://www.geeksforgeeks.org/mvc-framework-introduction/

GeeksforGeeks. (2025a). *https://www.geeksforgeeks.org/microservices/*. https://www.geeksforgeeks.org/microservices/

GeeksforGeeks. (2025b). *Singleton Method Design Pattern*. https://www.geeksforgeeks.org/singleton-design-pattern/

RedHat. (2025). *What is serverless?* https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless