SOFTWARE DESIGN DOCUMENT

CHATZEN

NOVEMBER 11, 2023

RUBIX!

CZ!

Version 1.0

Team: RUBIX!

Shreya Singh (LCS2022046)

Raghunandan Bansal (LCS2022058)

Rishi Raj Maheshwari (LCB2022005)

Table of Contents

1. Introduction:

In the ever-evolving realm of internet chat applications, ChatZen emerges as a unique social-networking tool, harnessing cutting-edge technology to provide users with a secure and private platform for seamless communication and media sharing. This application enhances connections with loved ones, offering a delightful experience through features like messaging, video calls, updates, photos, and thereby fostering local socializing.

1.1 Purpose:

This Software Design Document (SDD) is a natural extension of the Software Requirement Specification (SRS), specifically tailored for the development of the Kotlin Chat App, ChatZen. The primary purpose is to deliver a detailed and comprehensive description that acts as a reference throughout the design, development, and testing phases of the ChatZen app.

1.2 Scope:

ChatZen envisions itself as a real-time text-based communication platform, empowering users with features like message exchange, chat room creation, and contact management through user IDs. The app's exclusivity to mobile platforms is a strategic decision driven by paramount security considerations. This Software Design Document succinctly outlines the scope of ChatZen, emphasizing its commitment to mobile devices and ensuring a secure and user-friendly environment.

1.3 Definitions, Acronyms, and Abbreviations

APP	Application
API	Application Programming Interface
CHAT APP	The application being developed as per this document
CZ	ChatZen
GC	Group Chat
PC	Private Chat
SRS	Software Requirements Specification
UI	User Interface
VC	Video Chat

1.4 Overview:

The Kotlin Chat app, embodied by ChatZen, stands as a testament to mobile application engineering, facilitating real-time communication. Users can seamlessly exchange text messages and multimedia content (images) with multiple contacts through user IDs. Developed using the Kotlin framework, a leading open-source mobile app development platform by Google, the app ensures accessibility on devices. The design principles champion a smooth and intuitive user experience.

1.5 Document Contents:

The subsequent sections of this document will delve into the intricacies of ChatZen's requirements. Using a blend of diagrams, graphs, and textual guidelines, the aim is to provide a nuanced understanding of the software's architecture, design choices, and development considerations. This living document welcomes collaborative input, evolving dynamically in response to the project's nature, ensuring alignment with the shared goals of all stakeholders.

2. Design Overview

The ChatZen app adopts a client-server architecture. The client, a mobile app developed in Kotlin, caters to android. The server is an API backend coupled with a Firebase database for effective data management.

Within this design:

i. Key Components:

- User Management: Responsible for user registration, authentication, chat page, video call screen.
- Chat: Facilitates chat rooms, message exchange, and media sharing.
- Video Calling: Access to the video call using unique room ID.
- Notifications: Deploys push notifications for real-time updates.
- Media Storage: Interfaces with cloud storage (S3) for efficient media file handling.

ii. Interfaces:

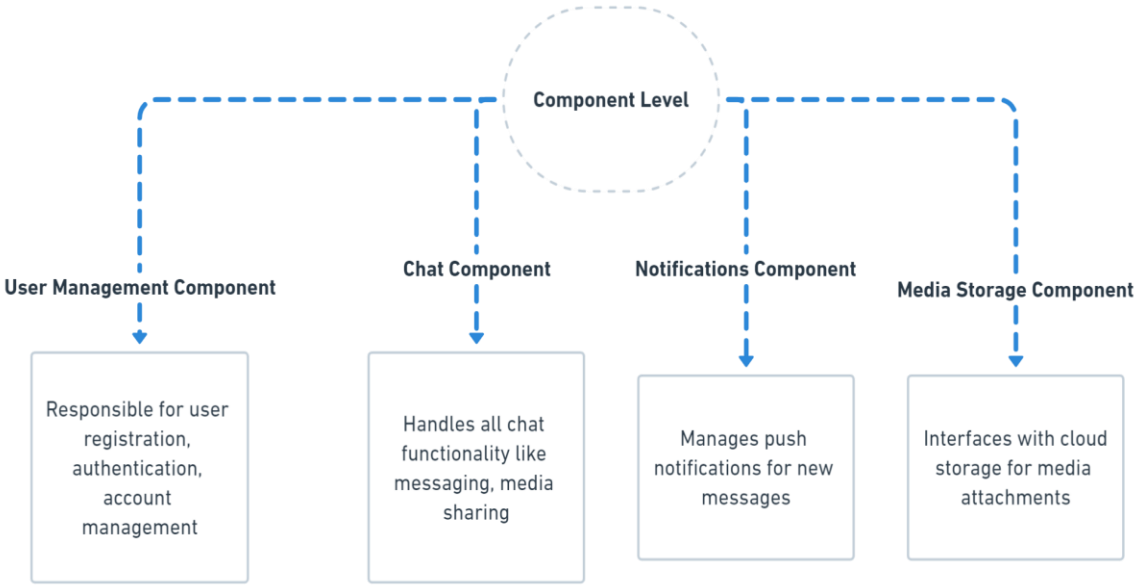
- Registration Screen: Facilitates new user signup.
- Login Screen: Manages user authentication.
- Chat Screen: Displays messages within chat rooms.
- Contact List: Provides a view of user contacts.
- Video call Screen: Exhibits user video call screen.
- Video ID room: Facilitates user video ID.

iii. Architecture:

- Client: Developed in Kotlin for both Android.
- Server: Comprises an API backend with a Firebase database.
- External Services: Utilizes S3 for media storage and Firebase for push notifications.

Communication occurs through an API calls, with real-time chat leveraging WebSockets. Media files are stored on S3, and push notifications are facilitated through Firebase.
This streamlined design overview encapsulates the ChatZen app's core structure, emphasizing its client-server model and key functional elements.

2.1 Component Level

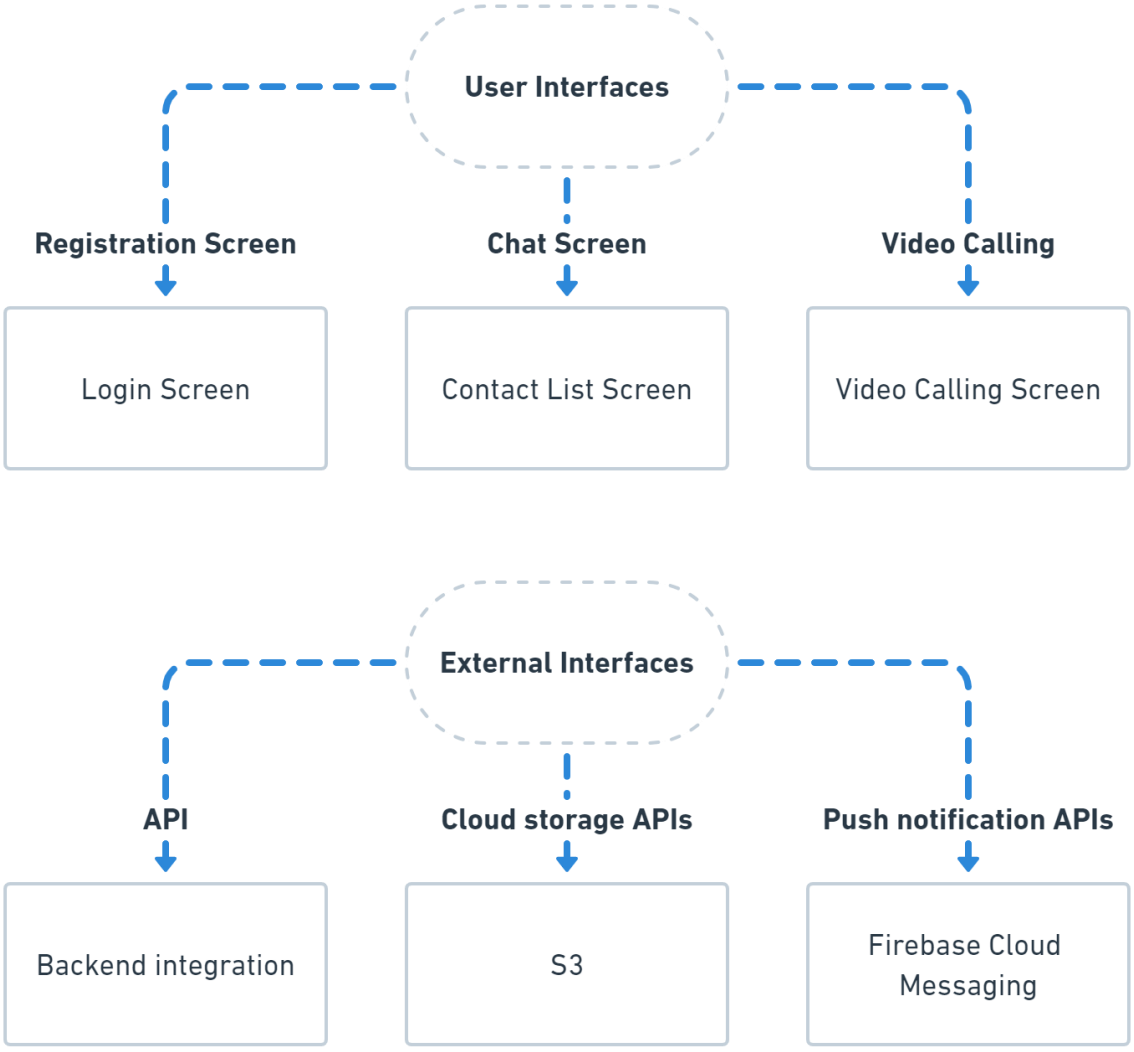


<u>The core components of the ChatZen app contribute to its seamless functionality:</u>

- <u>User Management Component:</u> Manages user registration, authentication, and account details.
- <u>Chat Component:</u> Facilitates messaging, media sharing, and overall chat functionality.
- <u>Notifications Component:</u> Orchestrates push notifications, ensuring users stay updated on new messages.
- <u>Media Storage Component:</u> Interfaces with cloud storage, streamlining the attachment of media files.

These components collectively define the operational framework of ChatZen, addressing user interaction, chat functionality, notifications, and media file management.

2.2 Interface Level



<u>The ChatZen app boasts a user-friendly interface that seamlessly integrates both user and external components.</u>

i. <u>User Interfaces:</u>

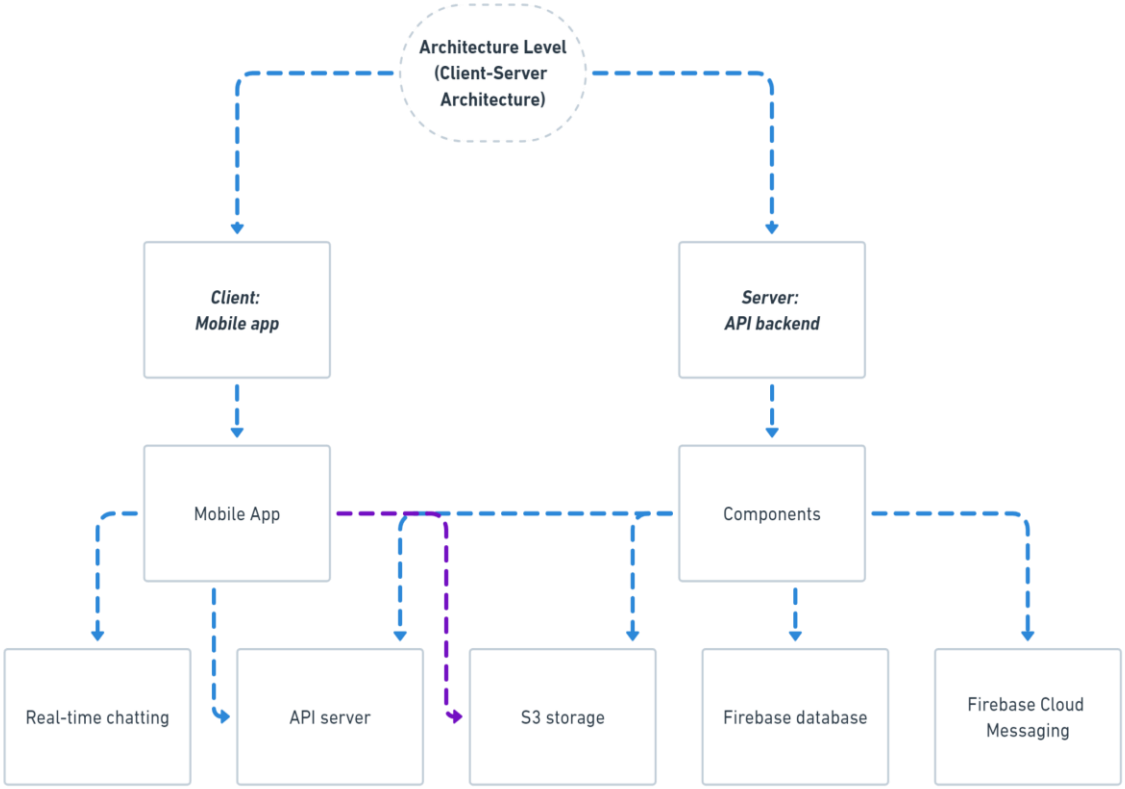
- <u>Registration Screen:</u> Facilitates seamless new user sign-up.
- <u>Login Screen:</u> Streamlines the user authentication process.
- <u>Chat Screen:</u> Provides a dynamic display of messages within chat rooms.
- <u>Contact List Screen:</u> Offers an organized view of the user's contacts.
- <u>Video Calling:</u> Displays pertinent user video calling information.
- <u>Video Calling Screen:</u> Allows users to enter the unique room ID.

ii. External Interfaces:

- API for Backend Integration: Ensures smooth communication between the client and server.
- Cloud Storage APIs like S3: Facilitates efficient integration with cloud storage for media files.
- Push Notification APIs like Firebase Cloud Messaging: Enables real-time push notifications, enhancing user engagement.

This interface overview underscores the ChatZen app's commitment to intuitive user interactions and seamless integration with external services, ensuring a robust and dynamic user experience.

2.3 Architecture Level



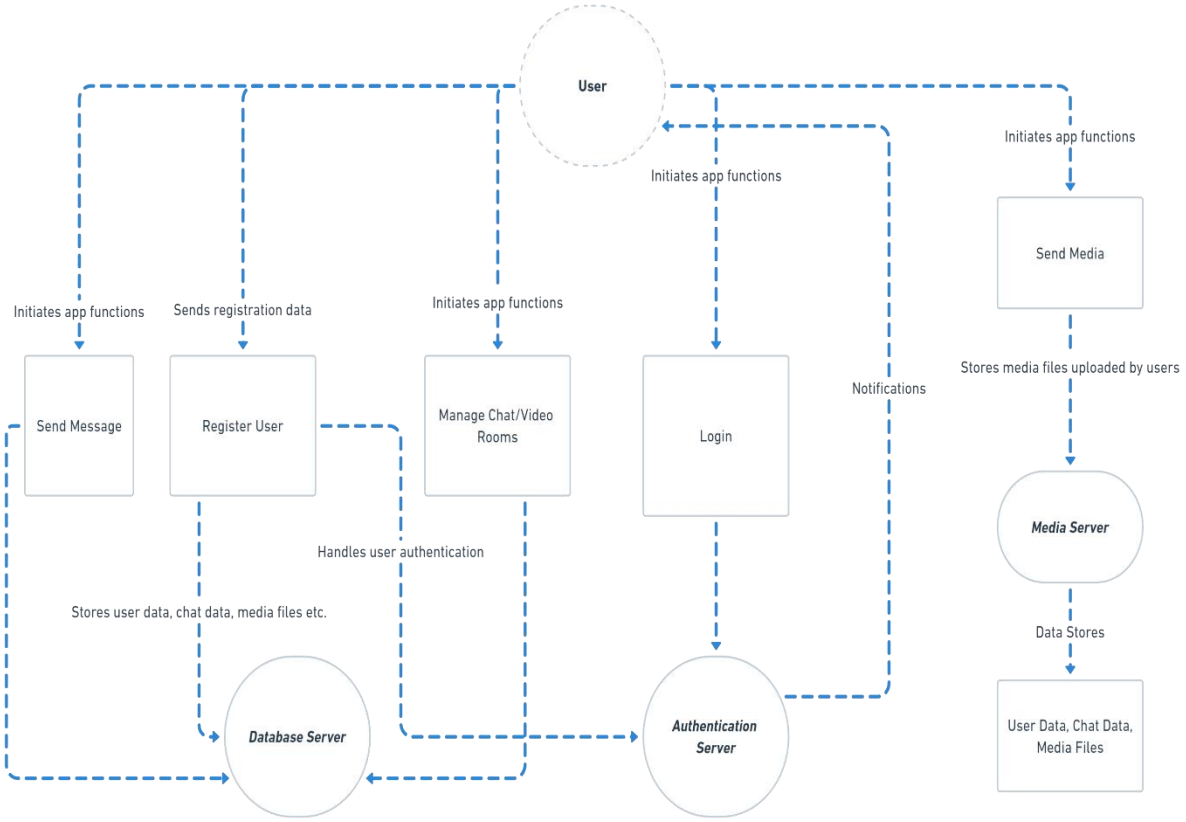
The ChatZen app operates on a client-server architecture. The client, a mobile app developed in Kotlin, communicates with the server, an API backend. The server manages crucial components, including an API server, Firebase database, S3 storage for media, and Firebase Cloud Messaging (FCM) for push notifications.

Communication between the client and the backend occurs through an API call, with real-time chat functionalities. Media files are stored on S3, and push notifications are handled seamlessly by FCM.

In summary, the high-level architecture comprises a Kotlin-based mobile app as the client, an API backend with a Firebase database as the server, and external services such as S3 for storage and Firebase for notifications. This architecture ensures efficient communication, real-time chat capabilities, and robust handling of media and notifications within the ChatZen app.

3. Data Design

3.1 Data Flow Diagram



<u>In the ChatZen ecosystem, data seamlessly navigates between external entities, processes, and data stores:</u>

i. **<u>External Entities:</u>**

- <u>User:</u> Initiates functions like registration, login, and messaging.
- <u>Database Server:</u> Stores user data, chat details, and media files.
- <u>Authentication Server:</u> Manages user authentication.
- <u>Media Server:</u> Stores user-uploaded media.
- <u>Notification Server:</u> Sends notifications to users.

ii. **<u>Processes:</u>**

- <u>Register User:</u> Captures new user data.
- <u>Login:</u> Authenticates users.
- <u>Manage Chat/Video Rooms:</u> Controls room/video creation and deletion.
- <u>Send Message:</u> Facilitates message exchange.
- <u>Send Media:</u> Manages media file transmission.

iii. Data Stores:

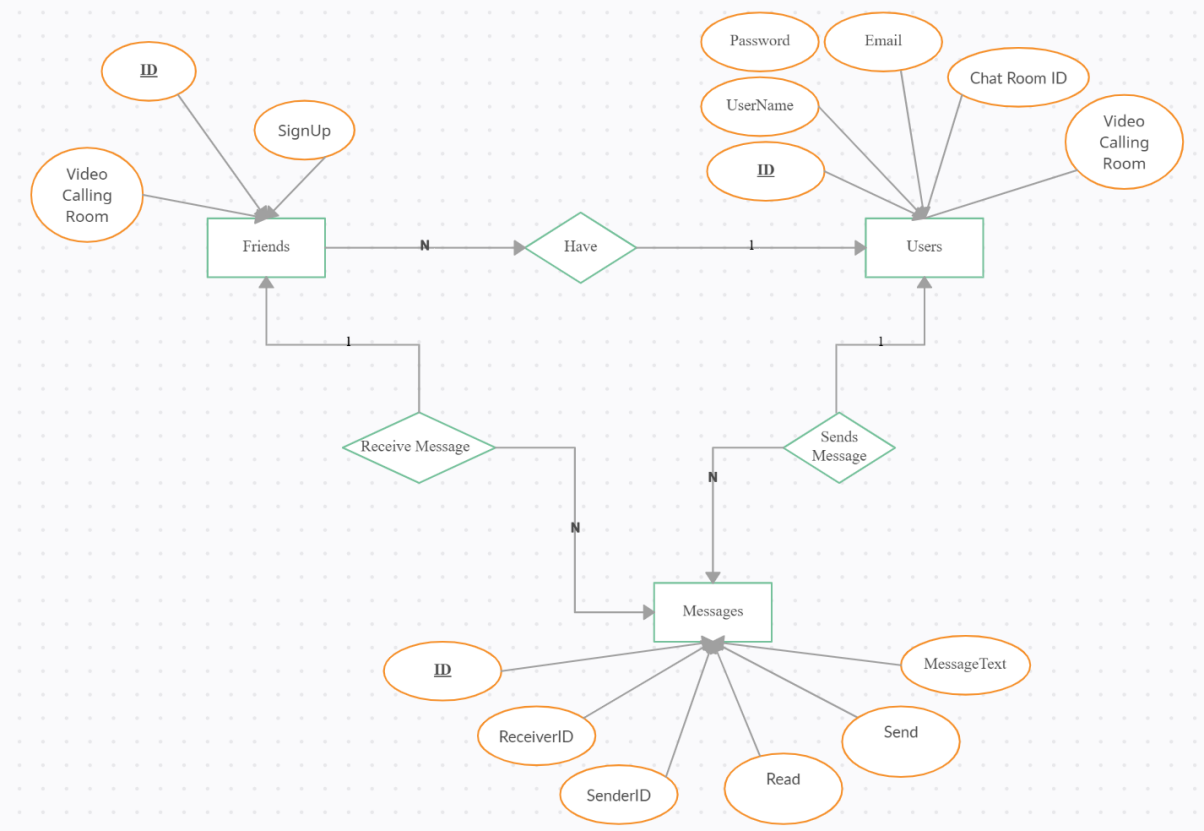
- User Data: Holds user account information.
- Chat Data: Stores chat room and message details.
- Media Files: Repository for user-uploaded media.

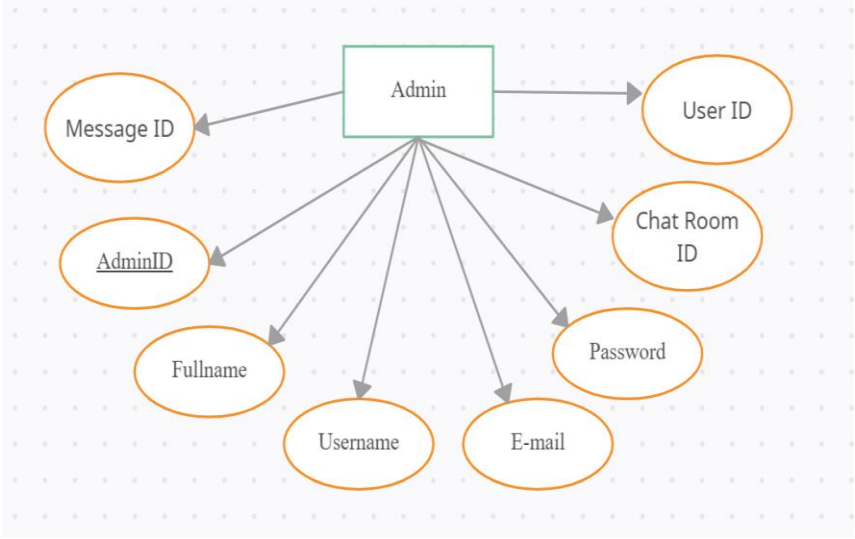
iv. Data Flows:

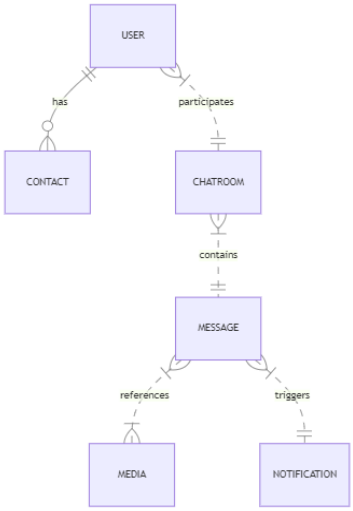
- User Registration Data: Flows from user to registration, stored in User Data.
- User Login Data: Authenticated by the Authentication Server.
- Chat/Video Room Data: Reflects room/video creation and deletion.
- Message Data: Flows between users during messaging.
- Media Files: Stored on the Media Server.
- Notifications: Sent from the Notification Server to users.

This streamlined data flow ensures efficient communication, authentication, and storage in the ChatZen application.

3.2 Entity Relationship Diagram







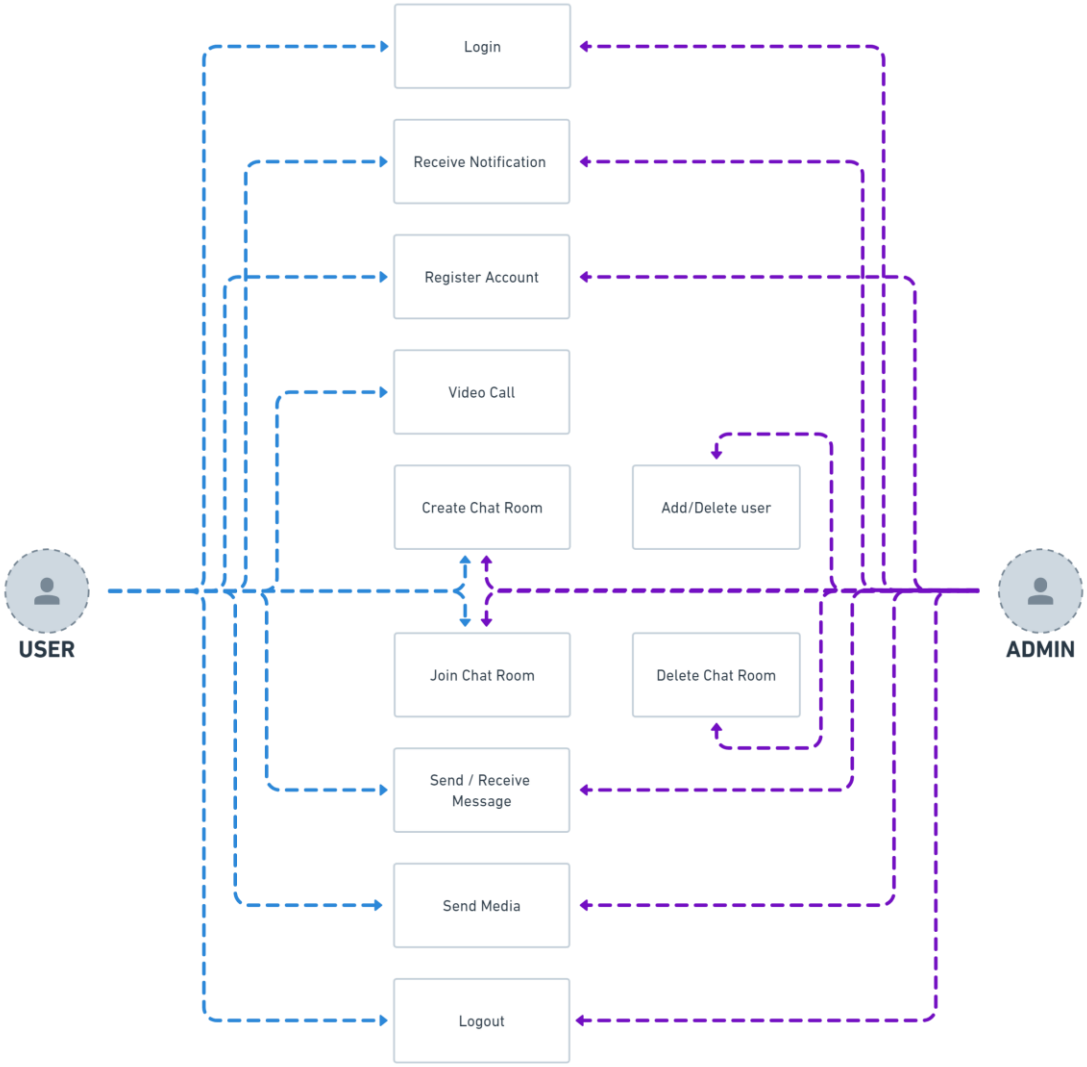
The main entities in the ER diagram are:

- User - Represents the users of the chat application. Attributes include user ID, name, email, password etc.
- Chat Room - Represents the chat rooms created in the app. Attributes include room ID, room name, admin user ID etc.
- Message - Represents the messages sent in the chat rooms. Attributes include message ID, message text, sender user ID, timestamp etc.
- Media - Represents media files like images shared in chat rooms. Attributes include media ID, media file, caption etc.
- Video Call- a user have access to make a video call using a unique room ID

The key relationships between entities are:

- User-Chat Room: A user can create and participate in multiple chat rooms (one-to-many relationship)
- ChatRoom-Message: A chat room contains multiple chat messages (one-to-many)
- User-Message: A message is sent by a user (many-to-one)
- Message-Media: A message can have zero or one media files attached (one-to-one optional)
- User-Contact: A user can have multiple contacts (one-to-many)
- Contact-User: A contact refers to a user (many-to-one)
- Video Call: a user can add any no. of participants in the video call (one to many)

3.3 Use Case Diagram



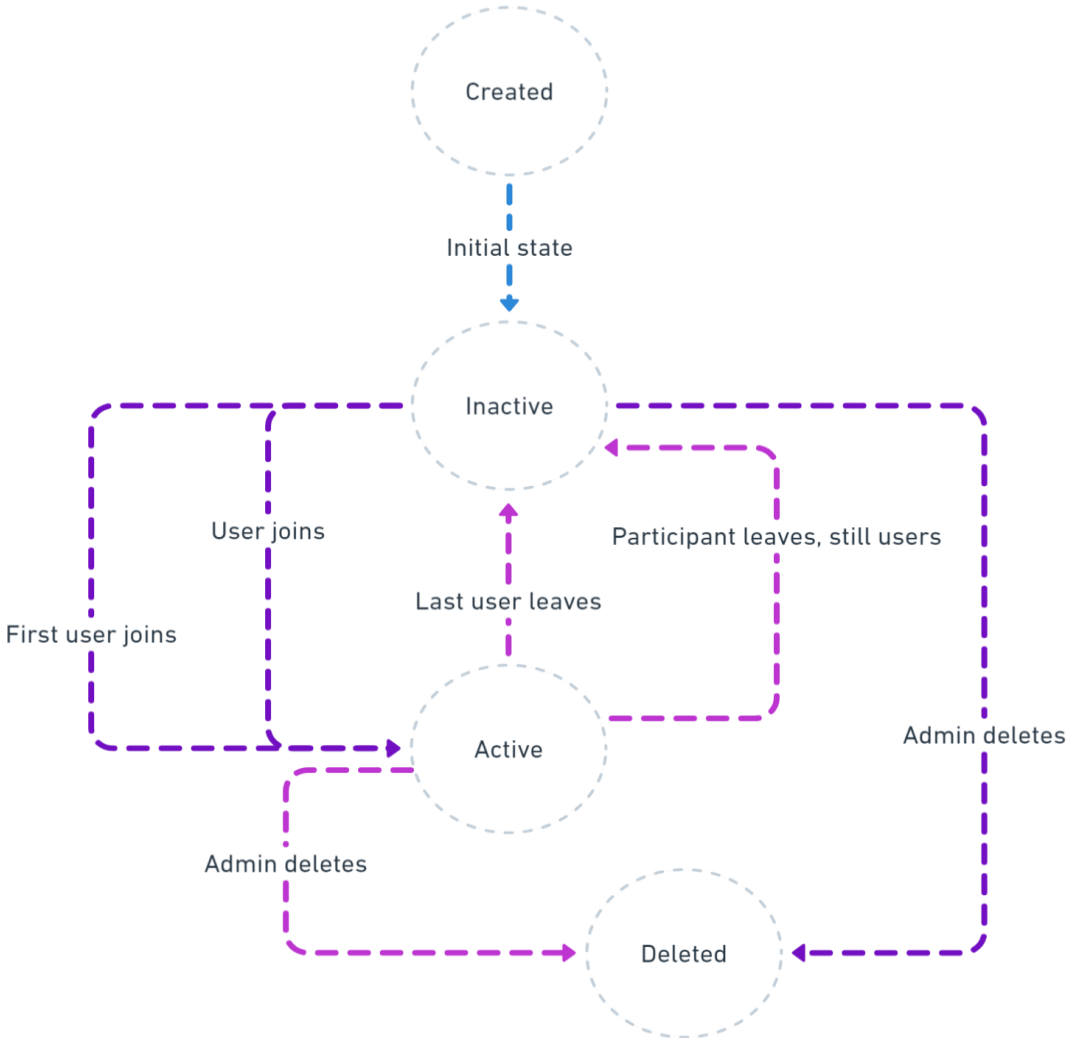
The chat application's use case diagram involves two key actors: the User and the Admin. The User, representing end-users, engages in various interactions, including registering an account, logging in, managing chat rooms, sending messages and media, handling contacts, receiving notifications, and logging out.

In contrast, the admin, with an administrative role, oversees actions like adding or banning users and deleting chat rooms. These distinct roles highlight the specific responsibilities assigned to each actor.

Relationships illustrate how user actions align with functionalities. For example, the User engages in user-specific actions, emphasizing participation in the chat environment. The Admin, in contrast, handles system-level actions like user banning and chat room deletion, showcasing overarching system management.

The diagram maintains simplicity without specific generalization, include, or extend relationships, providing a comprehensive overview of functionalities for both end-users and administrators in the chat application.

3.4 State Transition Diagram



The chat room state transition diagram involves four key states: "Created," "Active," "Inactive," and "Deleted." The initial state is "Created" when a chat room is first generated. It transitions to "Inactive" when there are no participants in the chat room. The transition from "Created" to "Inactive" represents the initial state when the chat room is established.

The "Active" state signifies that there are two or more participants in the chat room. This state is reached when the first user joins an initially "Inactive" chat room. The transitions between "Active" and "Inactive" occur based on user interactions – moving from "Active" to "Inactive" when the last participant leaves, and from "Inactive" to "Active" when a user joins an empty chat room.

The "Deleted" state is reached when an admin decides to delete the chat room, irrespective of its current state (Active or Inactive). This state transition demonstrates the admin's authority to remove chat rooms from the system.

In summary, the diagram outlines the sequential transitions between the states, reflecting the lifecycle of a chat room from creation to activation, user interactions, and eventual deletion by an administrator. This representation provides a comprehensive overview of the possible states and transitions within the chat room system.

4. Other Requirements

- **Appendix A: Project Document**

  The project documentation for an application for online commodity and delivery system

- **Appendix B: DETAILED PROJECT DOCUMENTATION**

  Candidate Name & Roll No. : Shreya Singh (LCS2022046)

  Raghunandan Bansal (LCS2022058)

  Rishi Raj Maheshwari (LCB2022005)

  Course of Study: B.Tech