DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

EXPERIMENT NO: 5

Student Name: KJ Pritpal Singh
UID: 23BCS14164
Branch: BE-CSE
Section/Group: KRG-1B
Semester: 6th
Date of Performance: 23/02/2026
Subject Name: System Design
Subject Code: 23CSH-314

Aim

To design a scalable real-time messenger application similar to WhatsApp or Facebook Messenger that supports one-to-one and group messaging with high availability, low latency, and reliable message delivery.

Objectives

- Understand the architecture of a real-time messaging system.
- Identify functional requirements such as authentication, chat, and media sharing.
- Identify non-functional requirements including availability, reliability, and latency constraints.
- Analyze CAP theorem trade-offs for messaging systems.
- Design REST and WebSocket APIs for chat communication.

Procedure

1. Studied real-world messaging systems like WhatsApp and Facebook Messenger.
2. Identified core entities such as Users, Groups, and Messages.
3. Analyzed real-time communication using WebSockets.
4. Collected functional and non-functional requirements.
5. Designed APIs for user authentication and messaging.
6. Evaluated storage and scalability requirements for large-scale systems.
7. Analyzed availability and eventual consistency trade-offs.

Functional Requirements

1. Client should be able to register and login into the system.
2. System should support one-to-one messaging.
3. System should support group messaging.
4. System should support sending textual as well as media messages.
5. System should preserve complete message history.
6. Client should be able to see delivered and read receipts.
7. Messages should be received in real-time.

Core Entities of the System

- Users
- Groups

- Messages

API Design

1. User Registration API: `POST /user/register`

2. User Login API: `POST /user/login`

One-to-One Messaging

A. Send Message (WebSocket): `WS /messages/send`

B. Get Chat List: `GET /chat/{user_id}` (Pagination Supported)

C. Get Messages of Specific Chat: `GET /messages/{user_id}/{receiver_id}`

Group Messaging

A. Create Group: `POST /groups/create`

B. Send Group Message (WebSocket): `WS /messages/send`

C. Get Group Chats: `GET /groups/{group_id}` (Pagination Supported)

D. Add Member: `POST /groups/{group_id}/add`

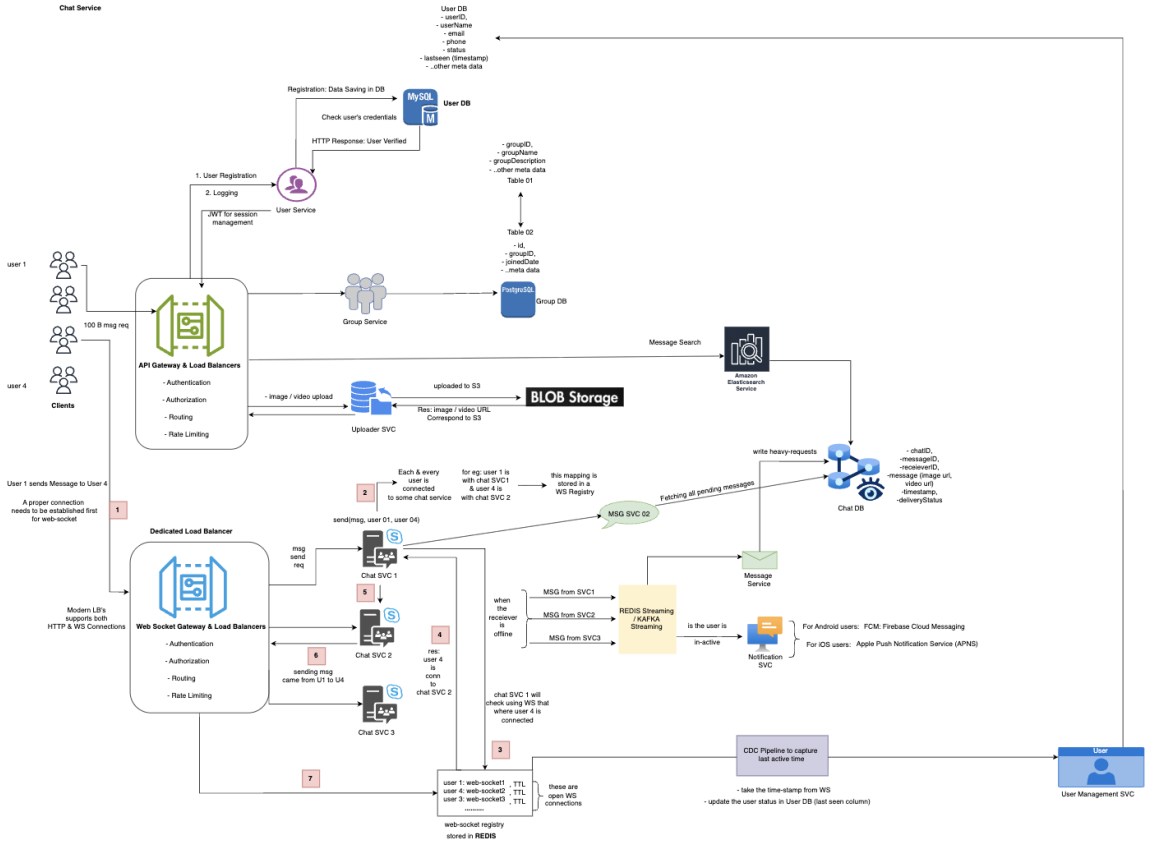
E. Remove Member: `DELETE /groups/{group_id}/remove`

Non-Functional Requirements

1. System should handle extremely large scale traffic (millions to billions of messages per day) requiring approximately 100 TB storage.
2. High Availability should be prioritized under CAP theorem.
3. System should follow Eventual Consistency model.
4. End-to-end latency should be approximately 200–300 ms.
5. System should be highly reliable with no message loss or packet drop.
6. Horizontally scalable distributed architecture.

High Level Design (HLD)

The system consists of Client Applications (Mobile/Web), API Gateway, Authentication Service, Messaging Service (WebSocket Servers), Group Service, Media Service, Message Queue, Distributed Databases, Cache Layer (Redis), and Object Storage for media files. Messages are delivered through WebSocket servers for real-time communication and stored in distributed databases for persistence.



Outcome

- Designed a scalable real-time messaging system.
- Identified functional and non-functional requirements.
- Designed REST and WebSocket APIs.
- Understood high availability and eventual consistency trade-offs in messaging systems.