Experiment 5

Student Name: Astik Joshi
UID: 23BCS10627
Branch: CSE
Section/Group: KRG -3B
Semester: 6th
Date of Performance:25/02/2026
Subject Name: System Design
Subject Code: 23CSH-314

1. 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.

2. Objective:

• Comprehend the structural design and core components of a real-time messaging architecture.
• Determine key functional capabilities such as user authentication, one-to-one and group messaging, and multimedia exchange.
• Define essential non-functional attributes including high availability, fault tolerance, scalability, and low-latency performance.
• Evaluate CAP theorem implications and consistency–availability trade-offs within distributed messaging platforms.
• Architect RESTful endpoints and WebSocket interfaces to enable real-time chat interactions.

2. Tools:

- Draw.io – For designing system architecture diagrams
- MySQL – To manage structured data such as users, subscriptions, and billing
- MongoDB (NoSQL) – For storing video-related metadata
- ElasticSearch – To implement high-performance search capabilities
- Apache Kafka – For event-driven asynchronous communication
- Redis / CDN Cache – For reducing latency through caching
- Cloud Object Storage (S3-like Blob Storage) – For storing raw and processed video files

4. System Requirements:

A. Functional Requirements:-

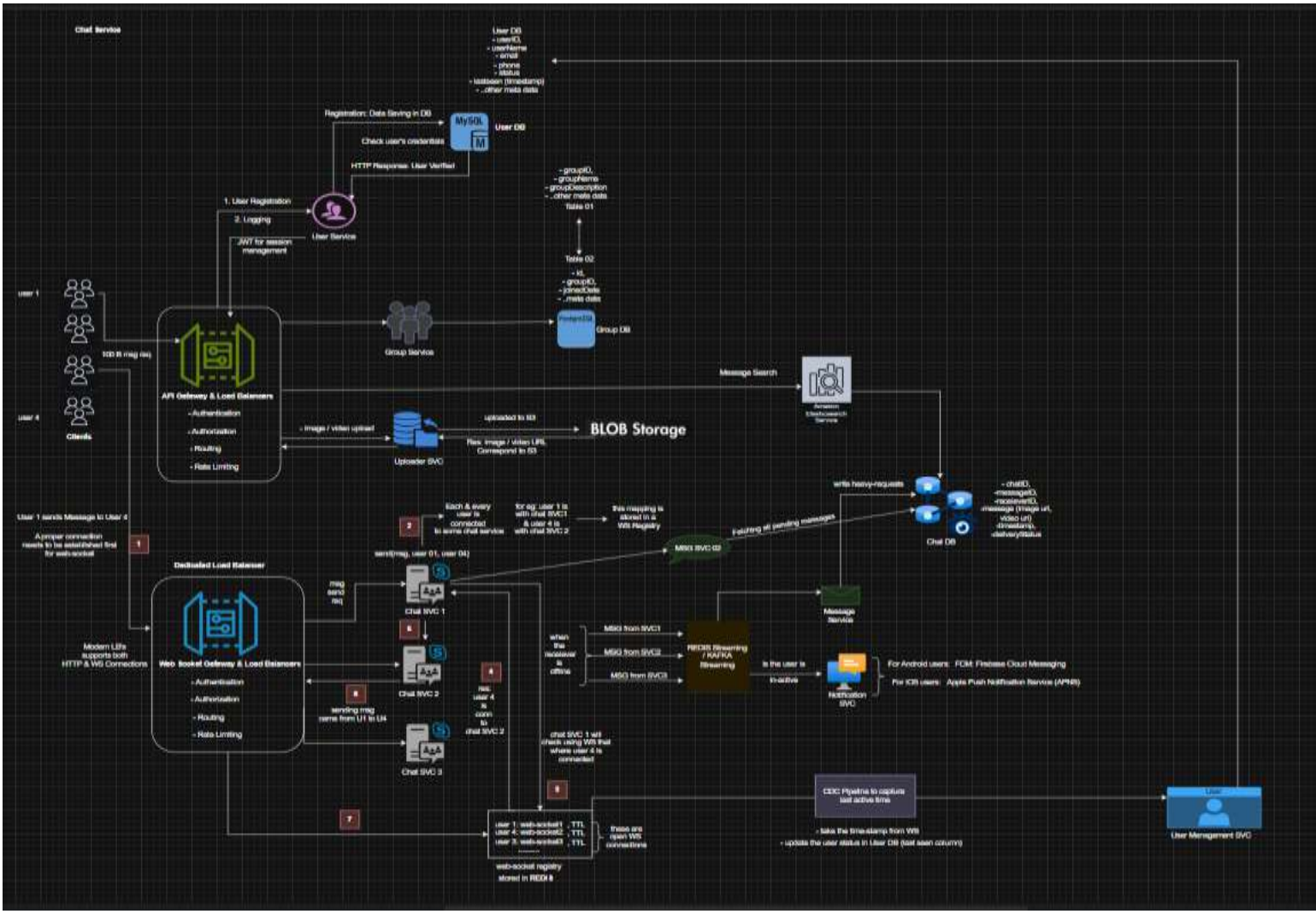
- Support direct one-to-one messaging between individual users.
- Enable group-based conversations involving multiple participants.
- Allow exchange of both text messages and multimedia content (images, videos, documents, etc.).
- Maintain a complete and persistent history of all messages.
- Provide delivery and read receipt indicators to users.
- Ensure real-time or near real-time message delivery.

B. Non-Functional Requirements:-

- Availability should be prioritized in accordance with CAP theorem trade-offs.
- The architecture should adopt an eventual consistency model across distributed components.
- End-to-end message delivery latency should remain within approximately 200–300 milliseconds.
- The platform must guarantee high reliability, ensuring zero message loss and minimal packet drops.
- The system should employ a horizontally scalable, distributed architecture to accommodate growth and load variations.

5. High Level Design (HLD):

- The system includes Client Applications (Mobile and Web interfaces) for end-user interaction.
- An API Gateway acts as the single entry point for routing client requests to backend services.
- An Authentication Service manages user identity verification and access control.
- A Messaging Service (WebSocket servers) handles real-time bidirectional communication.
- A Group Service manages group creation, membership, and group-related metadata.
- A Media Service processes and manages multimedia uploads and retrievals.
- A Message Queue enables asynchronous processing and decouples system components.
- Distributed Databases provide persistent storage for messages and system data.
- A Cache Layer (e.g., Redis) improves performance through low-latency data access.
- Object Storage is used for storing and retrieving media files.
- Messages are transmitted via WebSocket servers for real-time delivery and persisted in distributed databases for durability.



7. Learning Outcomes (What I Have Learnt):

- Architected a scalable real-time messaging platform capable of handling high traffic volumes.
- Analyzed and defined both functional and non-functional system requirements.
- Designed RESTful APIs and WebSocket-based interfaces for real-time communication.
- Evaluated high-availability priorities and eventual consistency trade-offs within distributed messaging architectures.
- Implemented a horizontally scalable distributed architecture to support fault tolerance and system resilience.