

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Student Name: Aaditya Anand
Branch: CSE
Semester: 6th
Subject Name: System Design

Experiment 5

UID: 23BCS10446
Section/Group: KRG 3-A
Date of Performance: 26/02/2026
Subject Code: 23CSH-314

1. Aim: Design a Messenger Application(Similar to WhatsApp / Facebook Messenger): To design and analyze the architecture of a scalable real-time messenger application that supports one-to-one and group communication with high availability and low latency.

2. Objective:

- To implement real-time message delivery using WebSockets.
- To support one-to-one and group messaging.
- To preserve chat history.
- To ensure high availability and reliability.
- To design APIs for user management and chat operations.
- To design the system for large-scale usage (up to 1 Billion users).

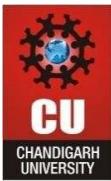
3. Tools Used:

- **Frontend (ReactJS, HTML, CSS, JavaScript, WebSocket)** ○ Built responsive UI with real-time communication support
- **Backend (Node.js / Java / Spring Boot)** ○ Developed scalable REST APIs & WebSocket server
- **Database (MongoDB / MySQL / PostgreSQL)** ○ Managed structured & unstructured data efficiently
- **Redis** ○ Implemented caching & pub-sub for high performance
- **Deployment (AWS / GCP, Load Balancer, Docker)** ○ Cloud deployment with scalability, containerization & traffic distribution

4. System Requirements:

A. Functional Requirements

- **User Registration**
 - a. POST /user/register
 - b. Fields: userName, email, phoneNumber
- **User Login**
 - a. POST /user/login



DEPARTMENT OF

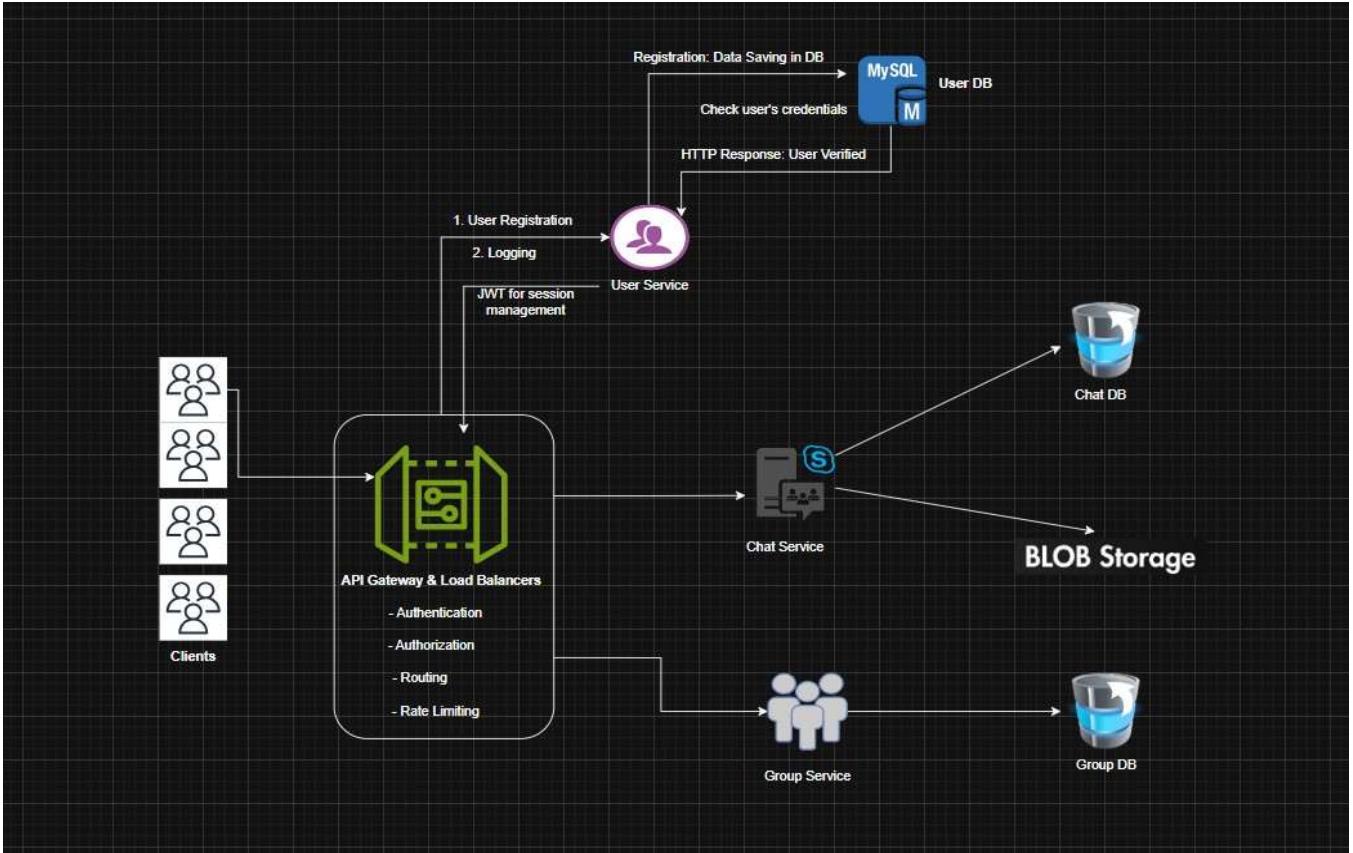
COMPUTER SCIENCE & ENGINEERING

- **One-to-One Messaging**
 - a. WS: /messages/send
 - b. GET /messages/{userId}/{receiverId}
 - c. Pagination support
- **Group Messaging**
 - a. POST /groups/create
 - b. POST /groups/{groupId}/add
 - c. DELETE /groups/{groupId}/remove
 - d. WS: /messages/send
- **Message Types**
 - a. Text messages
 - b. Media messages (images, videos, files)
- **Message History**
 - a. Persistent storage
 - b. Chat list view
- **Read Receipts**
 - a. Delivered
 - b. Seen

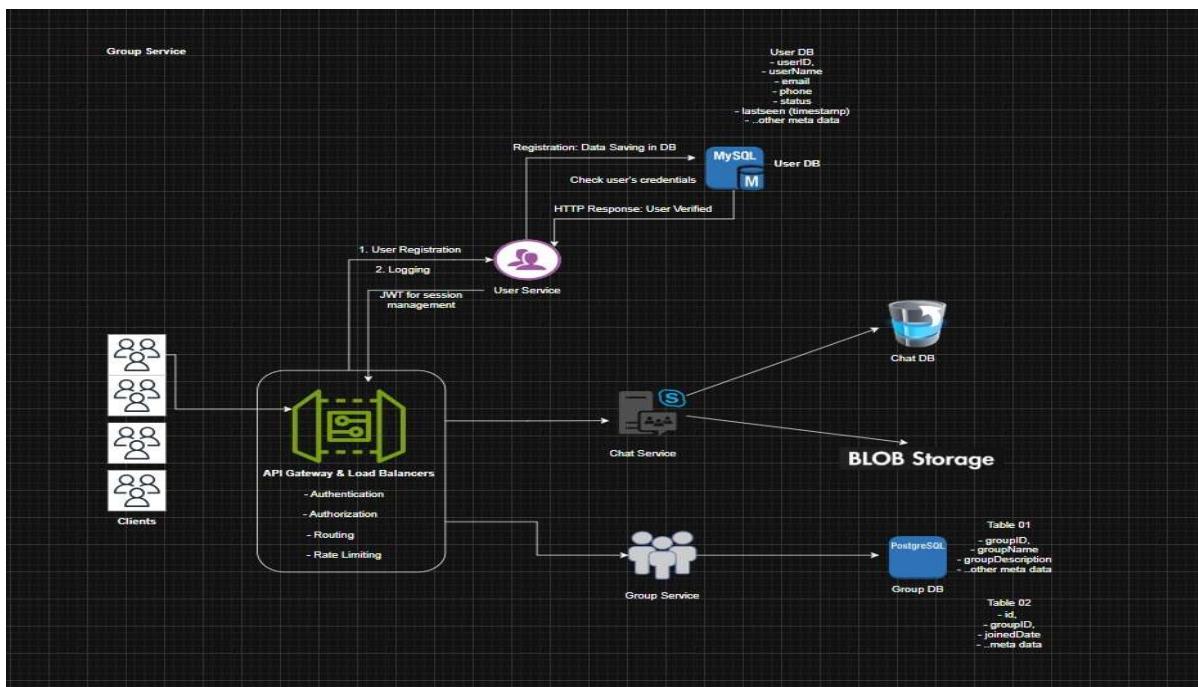
B. Non-Functional Requirements

- **Scalability** ○ Target: **1 Billion users** ○ 100 messages per user per day ○ \approx 100 Billion messages/day ○ Estimated storage: \sim 100 TB
- **CAP Theorem**
System prioritizes:
 - High Availability ○
 - Partition Tolerance ○
 - Eventual Consistency
- **Latency: 200-300ms** ○ Target message delivery time: **200–300 ms**

5. High Level Design (HLD)

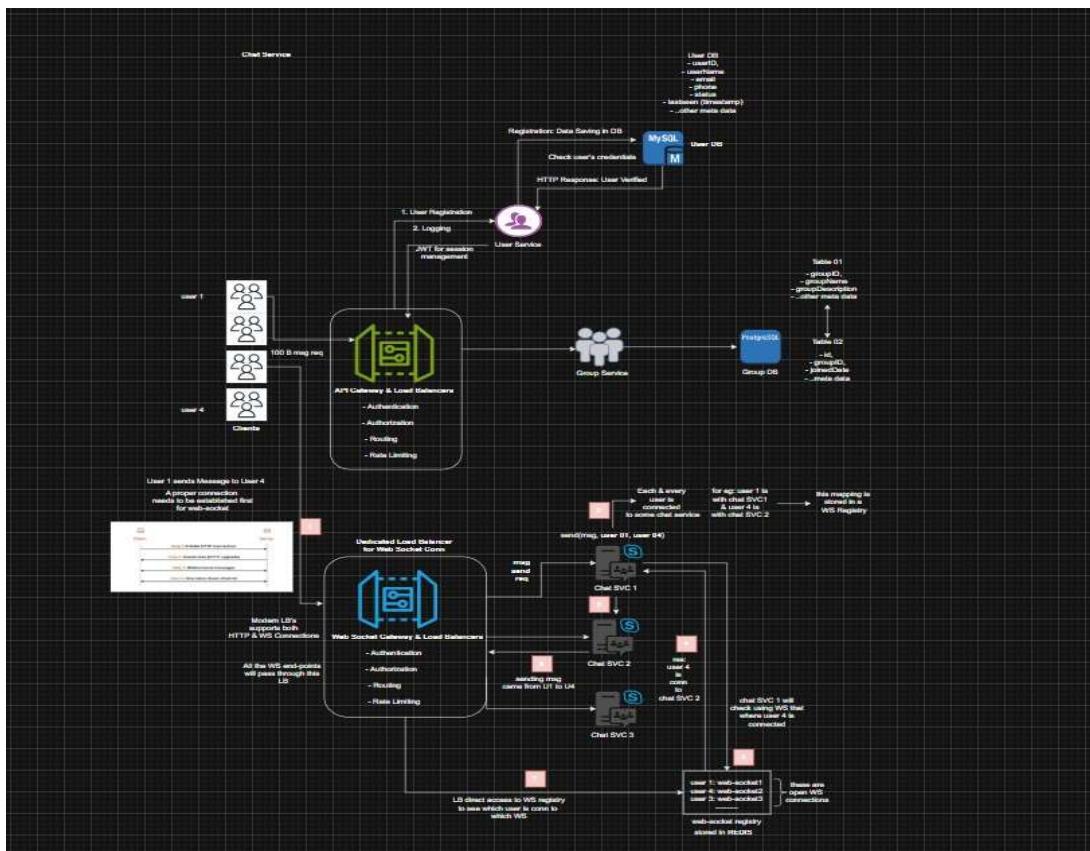
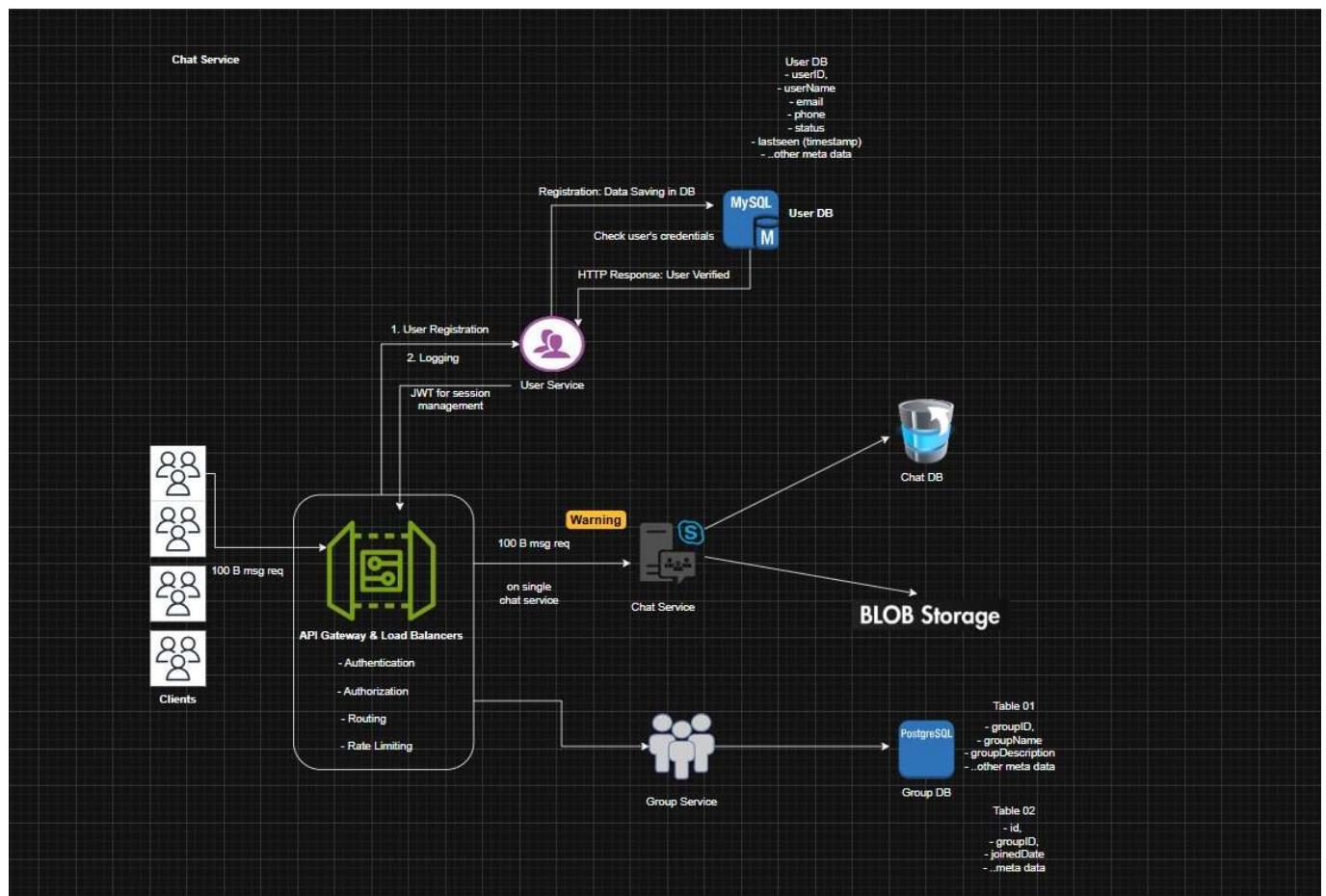


6. Low Level Design (LLD):



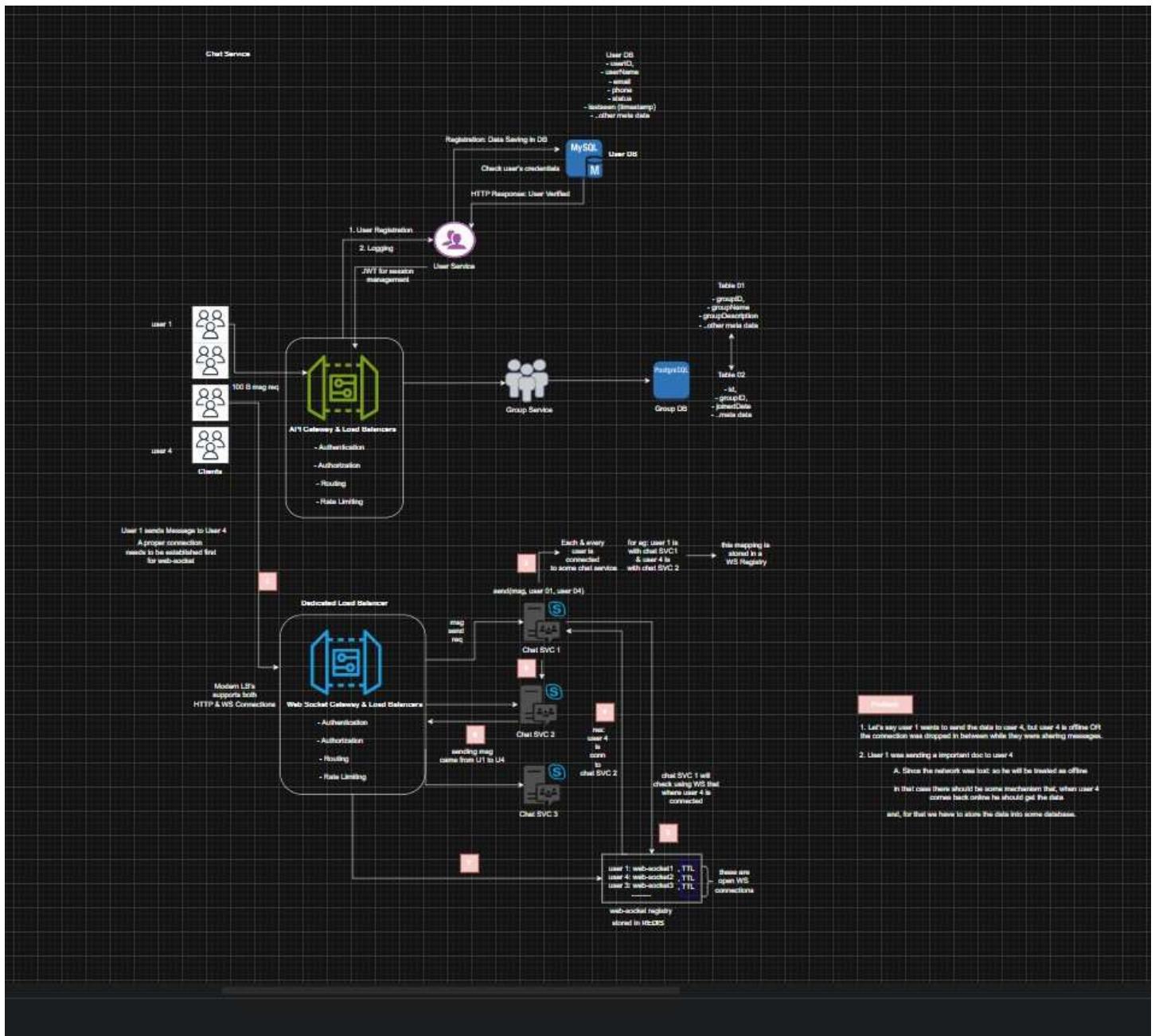


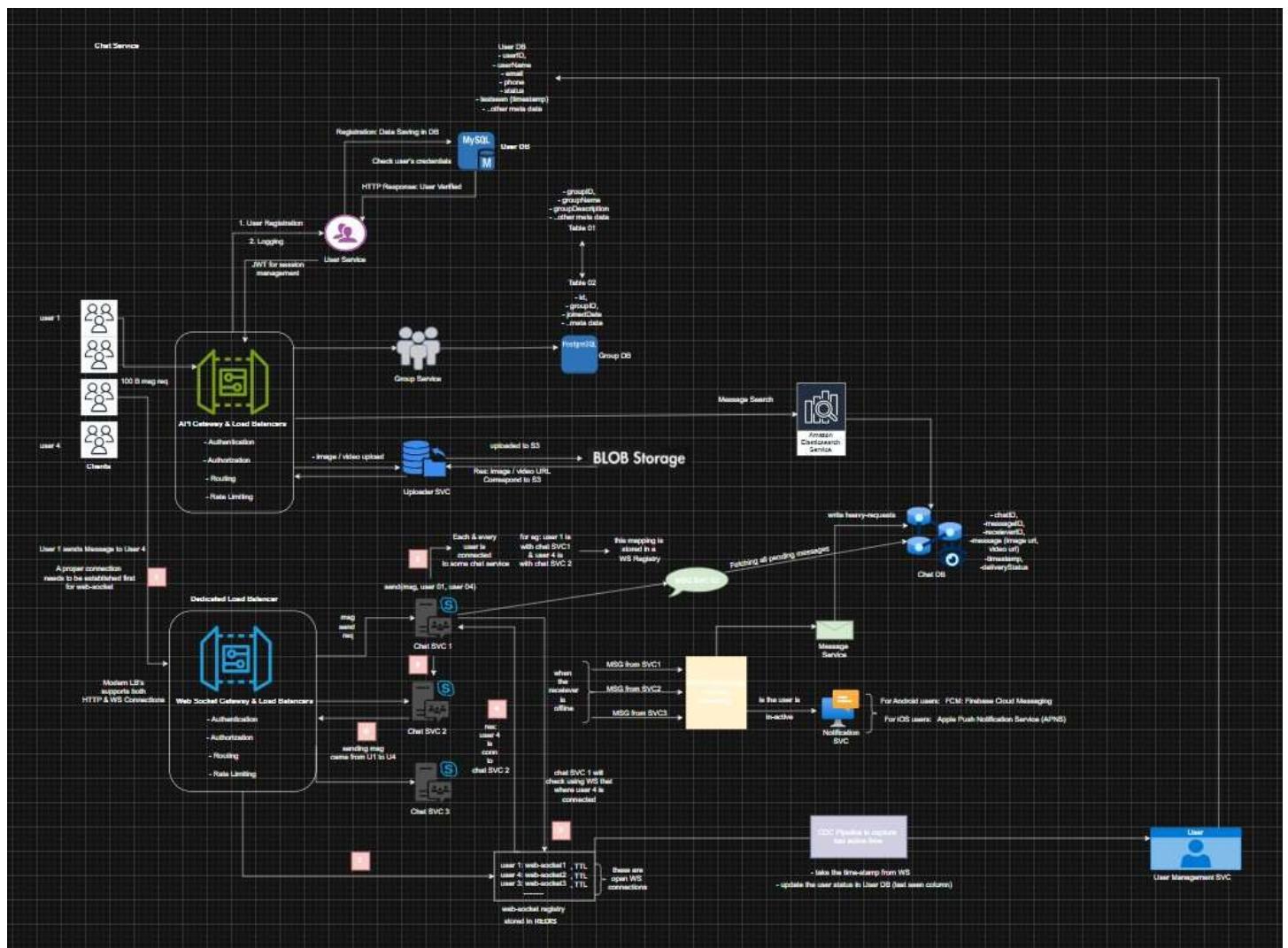
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING





7. Scalability Solution

- Horizontal Scaling**

Add multiple application servers behind a load balancer to distribute traffic and handle billions of requests efficiently.

- Database Sharding**

Partition data based on userID or chatID so that message load is distributed across multiple database servers.

- Caching with Redis**

Store frequently accessed chats and recent messages in Redis to reduce database load and improve response time.

- Message Queue System**

Use Kafka/RabbitMQ for asynchronous message processing and reliable message delivery.

- CDN for Media Files**

Use Content Delivery Network to efficiently deliver images, videos, and files globally with low latency.



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

8. Learning Outcomes

- Understand real-time communication using WebSockets.
- Learn how to design scalable distributed systems.
- Apply CAP Theorem concepts in real-world system design.
- Design RESTful APIs and WebSocket endpoints.
- Analyze trade-offs between consistency, availability, and latency.

.