High Availability Requirements

To ensure that the messaging system remains accessible and operational even under failures, high traffic or network issues the system must adhere to several additional constraints These constraints are designed to maintain continuity of service and prevent data loss.

1. Server Redundancy

The system must deploy multiple server instances to eliminate a single point of failure. A load balancer should distribute traffic across all active servers, ensuring that if one instance becomes unavailable others can continue handling user requests without interruption

---

2. Database Replication

All critical data, including user information, room details, and queued offline messages, must be stored in replicated databases. replication ensures that data remains available and consistent even if one database node fails.

---

3. Fault-Tolerant Messaging Queue

Offline messages and room broadcasts should utilize a persistent, fault-tolerant messaging queue. Replicated queues (e.g., using Kafka or RabbitMQ with durability) guarantee that messages are not lost during server or network failures, maintaining reliability in message delivery.

---

4. Session Persistence

User sessions must be maintained across multiple server instances. Storing session tokens or authentication states in a distributed cache (such as Redis with replication) ensures that users remain logged in even if their initial server instance fails.

---

5. Network and Geographic Redundancy

Servers should be deployed across multiple geographic regions using regional clusters or cloud availability zones. Content delivery networks (CDNs) can also be employed to reduce latency. This ensures system continuity in the event of regional outages.

---

6. Health Monitoring and Automatic Failover

The system must continuously monitor server and service health. Automated failover mechanisms should reroute traffic or restart failed instances without manual intervention, minimizing downtime and service disruption.

---

7. Rate Limiting and Resource Management

To maintain stable performance during peak traffic the system should implement request rate limiting, message throttling, and connection management. These measures prevent resource exhaustion and reduce the risk of server crashes.
