Messaging App

→ Designing Whatsapp

Functional Requirement [core functionality of our app]

(i) send and recieve message (1:1) / (Group msg)

(ii) Showing Sent, Delivered, Read reciept

(iii) Sharing multimedia files [image and video] + docs etc.

Non-functional Requirement

1. High consistency → Sequence of message matters.
2. Low latency → as fast ~~as~~ possible
3. CAP Theorem

↓

[Consistency, > Availability] Partitioning

Capacity Estimation

2B users.

100B msg/day

100 Bytes/msg

} 100Bytes X 100B msg/day *

= 10TB/day (Storage)

API Details

/message → POST

Send message(sender_id, reciever_id, text=None, media=None) ⇒ msg id.
હોઈ શકાય

/getMessage → GET ⇒ getmessage(user_id)

/media → POST ⇒ uploadfile(file, filetype) ⇒ file id.

/getmedia ⇒ GET ⇒ Download media(user_id, file_id)

Long Polling:- Simplest way of persistent connection with server
[Not use protocol like web-socket / TCP]
→ message passed after a wait.

Web Socket → Both side connection
→ from Server Side also
↓
We can use it for sending Notification from Server Side

~~Sbe~~ Polling

Short polling.
↓
Client ⟶ Server
On Certain time gap, request મોકલતા છે. [for info]

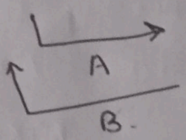
Thus high No. of request and chance of {} reponse.

long Polling
↓
Chance ની Real interaction ના હોવાથી
↓
Connection open હોતા છે {when {} send}.
Thus when server has data, it respond back.

server holds connection
↓
Thus, No. of request ↓↓

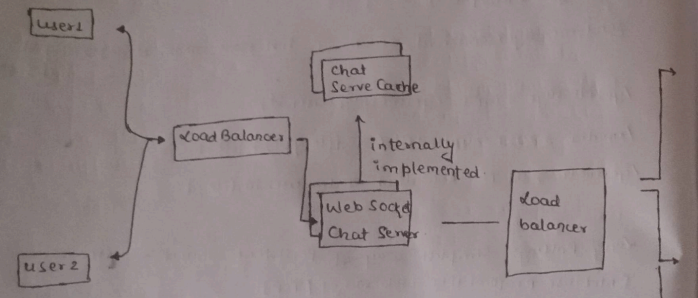
But we can't open connection indefinitely

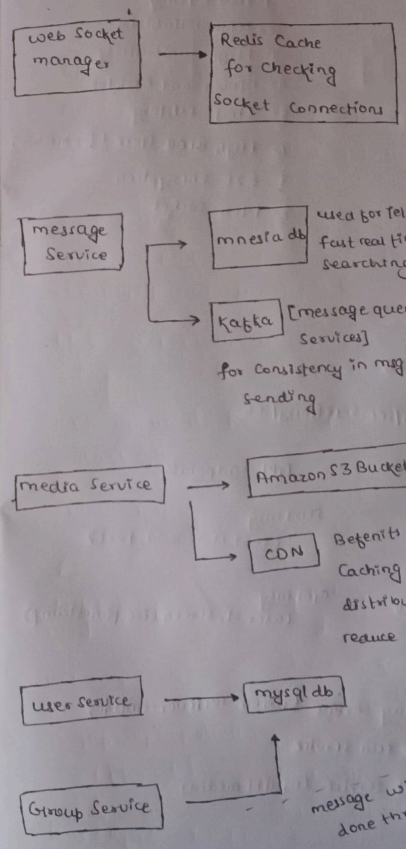
[Websockets] ↔ Bidirectional connection



} બંને msg મોકલ શકતા છે.

Eg. ⇒ Chat System
Game (Real time) [more than 1 player]


User1
User2
Load Balancer
Chat Serve Cache
internally implemented
Web Socket Chat Server
Load balancer


web Socket manager
Redis Cache for checking Socket Connections
message Service
mnesia db
used for Telecomm. fast real time data searching
Kafka
[message queue Services]
for consistency in msg sending
media Service
Amazon S3 Bucket
CDN
Benefits for sending img, video, static sheet
Caching of static content
distribute incoming traffic
reduce latency
User Service
mysql db
Group Service
message will be done through it.

Chatting APP

Functional Requirement

1. one-one chat
2. Group chat
3. Read Reciept
4. Online status
5. Notification
6. Sharing multi-media

System Requirement

1. Low latency
2. High reliablity
3. High availability
4. Mobile and Desktop
5. Chat History
6. High BLOB Store (for media)
7. E2E Encryption
8. Web socket

Capacity planning

- Total active user 500M
- 30 message per day. → 1user

Total msg/day = 500X30M = 1.5B

$$= \frac{500 \times 30 \times 1M}{3600 \times 24} \text{ ms/sec} = 18K \text{ msg/sec}$$

Storage Estimation

→ Total msg per day = ~~1B~~ 1.5B

Each msg size = 30/50 kB

= 50X1.5 = 75Pb

API Endpoints

Send msg
(send_id, recv_id, text)

Get msg
(user_id, screen_size, timestamp)

Services

Messaging Service

Group Service

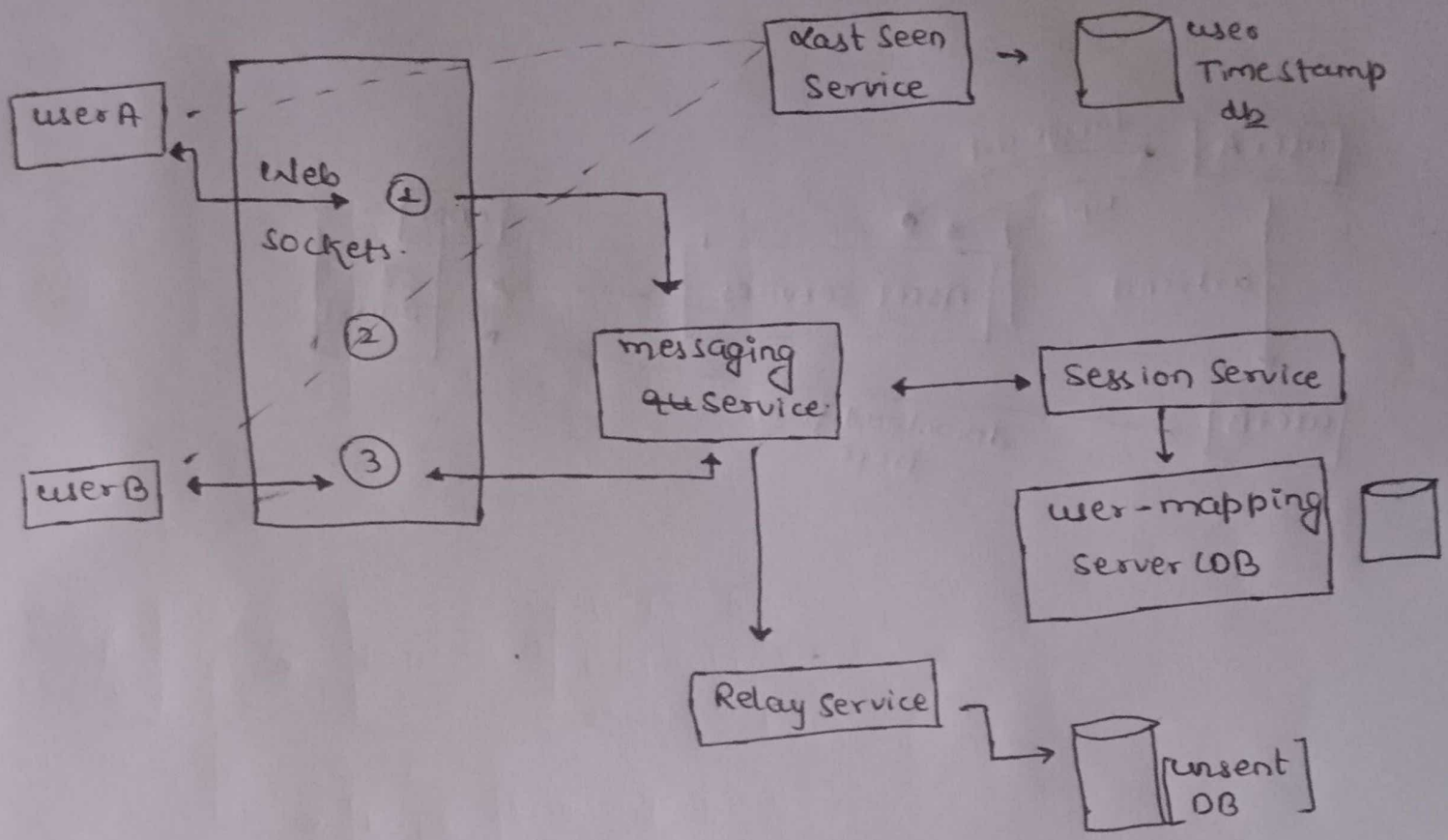
Session Service

Relay Service (when reciever=online)

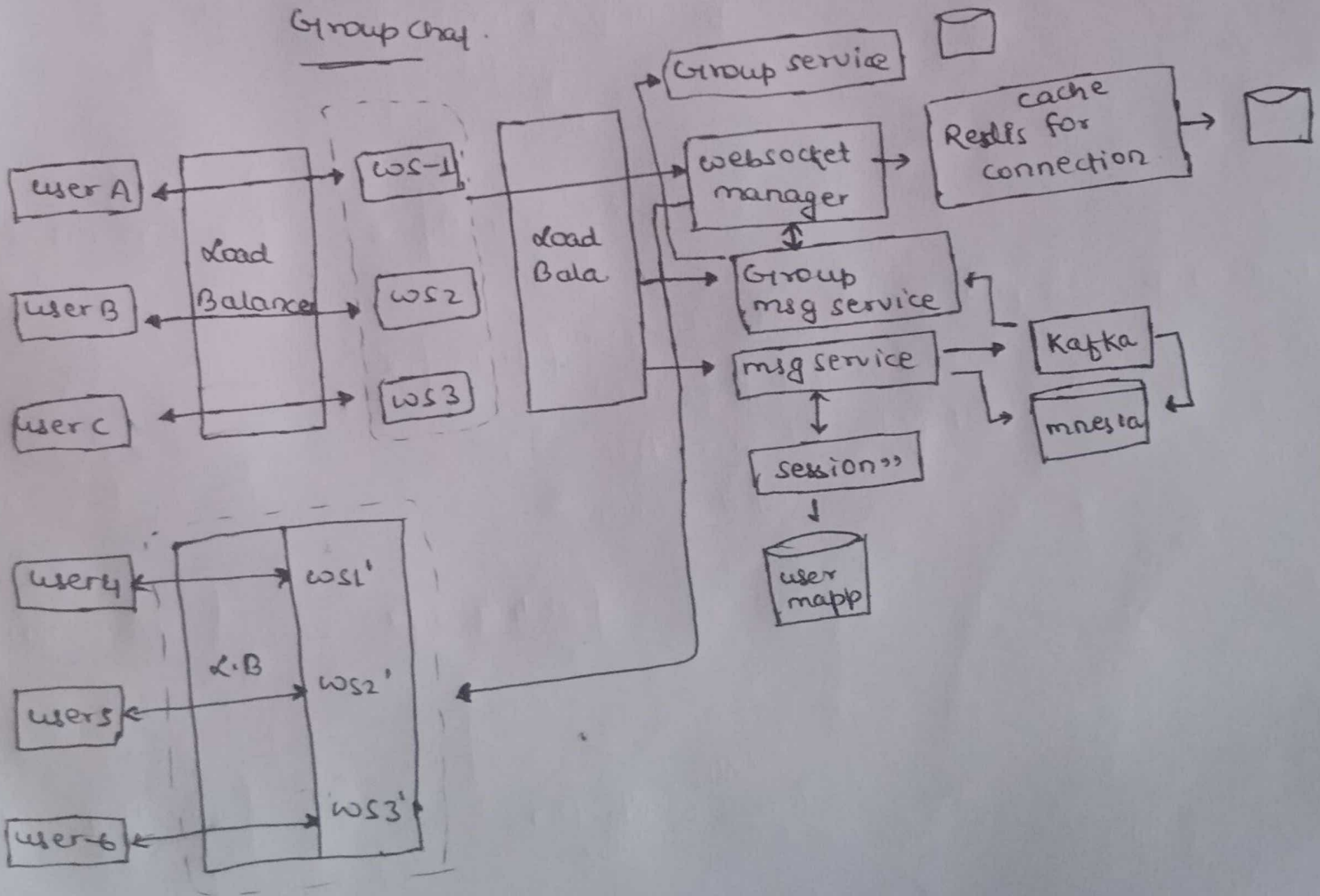
Last seen Service

Asset service (Multi-media)

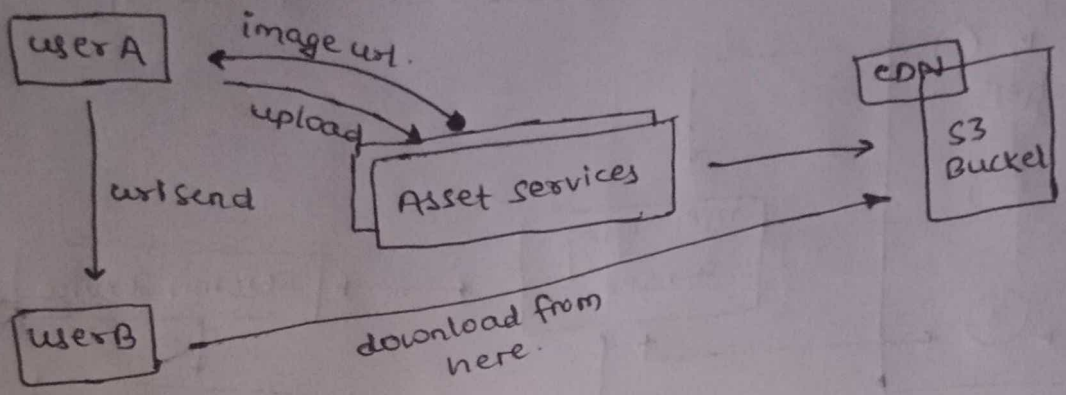
1-1 Chat.



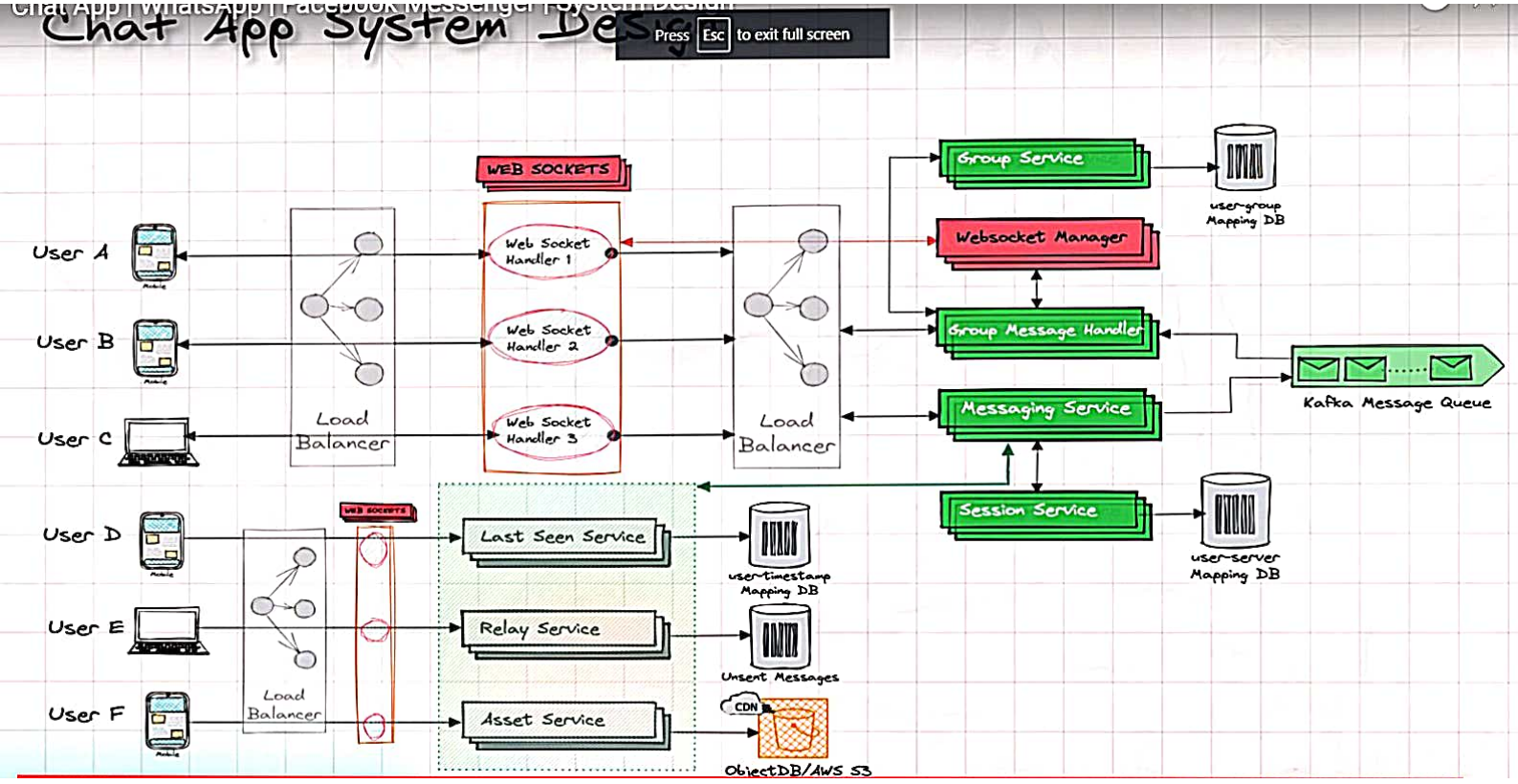
Group Chat.



Asset Services




T_Users
userId
username
contact
T_Groups
groupId
userId
T_LastSeen
userId
timestamp
T_UnsentMessages
messageId
sent_to_id
sent_from_id
content
media_url
timestamp
T_Sessions
userId
serverId


Chat App System Design
Press Esc to exit full screen
WEB SOCKETS
User A
User B
User C
User D
User E
User F
Load Balancer
Web Socket Handler 1
Web Socket Handler 2
Web Socket Handler 3
Load Balancer
Group Service
user-group Mapping DB
Websocket Manager
Group Message Handler
Messaging Service
Kafka Message Queue
Session Service
user-server Mapping DB
Last Seen Service
user-timestamp Mapping DB
Relay Service
Unsent Messages
Asset Service
CDN
ObjectDB/AWS S3