SOFTWARE REQUIREMENT SPECIFICATIONS

WEB BASED CHAT APPLICATION

1. Introduction

- The purpose of this project is to build a chat application that will allow users with an internet connection to engage in private and public conversations.
- The development of this project centered on the development of a message protocol that would allow the application to properly log in users , send messages, and perform system maintenance.

1.1 Scope of the project:

- ❖ Companies would like to have a communication software wherein they can communicate instantly within their organization.

- ❖ The fact that the software uses a internal network setup within the organization makes it very secure from outside attacks.

1.2 Objective:

- **Communication:** To develop an instant messaging solution to enable users to seamlessly communicate with each other.
- **User Friendliness:** The project should be very easy enabling even a novice person to use it.

2. Features and Requirements

Functional requirements

- One-one chat
- Group chat
- Read Receipt
- Online status
- Push notifications
- Share multimedia
- Multi device support

Non-functional requirements

- Low latency
- Highly available

- Highly scalable

3. Assumptions and Dependencies

Capacity Planning

Traffic estimations:

Total active users: 500M

On average a user sends 30 messages per day.

Total messages per day= 500M*30=1500M=1.5B

msgs per day= 1.5B/3600*24=18k msgs per sec

Storage estimations:

Total messages per day= 1.5B

Considering each message is on an average of 50 KB.

Total storage required to store all the messages=1.5B*50KB= 75PB

Messages are not going to be stored.

Let’s say 1:10 of the above data is for undelivered messages and we are going to store undelivered messages only for 30 days.

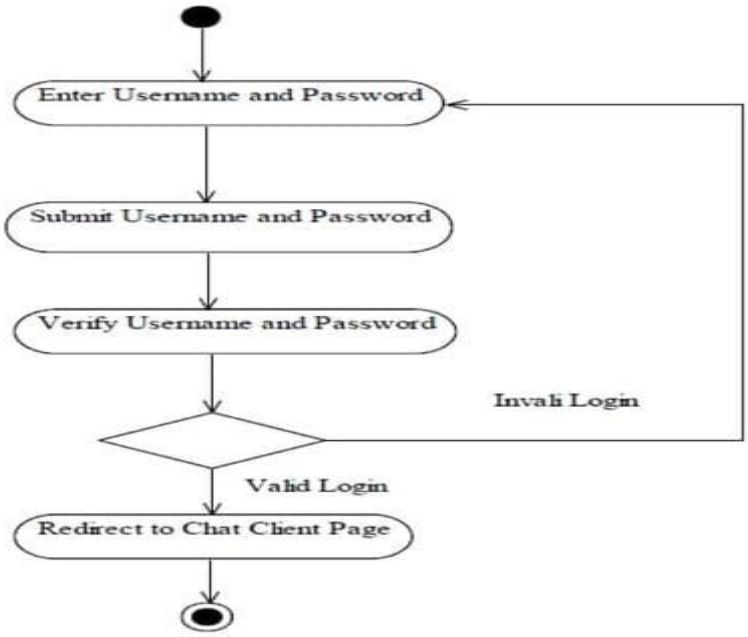
Storage required for undelivered messages for one day=75PB/10= 7.5 PB

Messages in 30 days= 225PB

Basic API Services

- Send_Message(sender_userID, receiver_user ID, text)
- Get_Message(user_ID,screen_size, before_timestamp)

Activity Diagram for Login



4. Chat Workflow

Messaging service :

When user A wants to send message to User B, he sends a request to the messaging service with the ID of User B. Before this, User A establishes a persistent connection to the

messaging service via web socket protocol because it is bidirectional connection.

In a traditional HTTP protocol the client needs to send a request every time when it requires some response but in web socket protocol, the server can respond without any client request to be made.

Messaging service identifies the User B via session service and sends the message accordingly.

Session service :

So how does session service works? Whenever a user connects to the messaging service, the messaging service tells session service in which server the user has established the connection, which is stored in a database more likely a 'No SQL' database since we don't have any scope for relations between the data. Later this information is used to send messages to the other end.

Relay service :

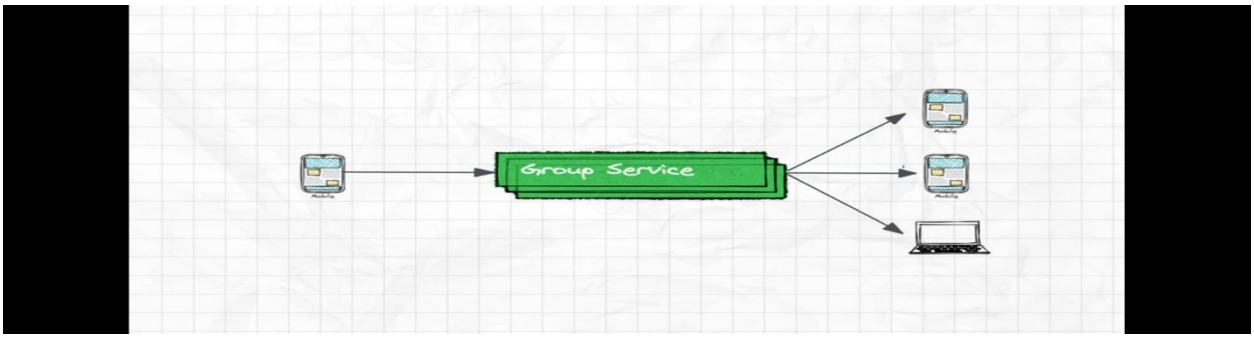
What if the User B is offline .In such cases we need to temporarily store the message to deliver it lately. Message service forwards this message to the relay service which will store the unsent messages with the from and to user ID in a database like Cassandra.

Last seen service :

The last seen service is used to store the timestamp for each user. This information is based on logging of each user activity. The client side application should be intelligent enough to identify the difference between the user activity and the application activity itself and sends a signal to the app server. This information can also be used to show the online status of the users.

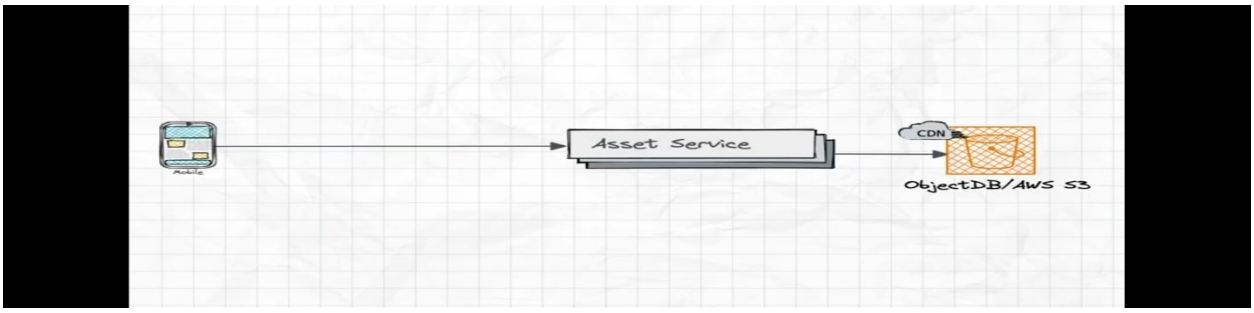
Group messaging service :

This is more like the messaging service except we need to publish the message to all the users associated with the same group ID. This service will rely on the session service to identify the server to which each user is connected to.

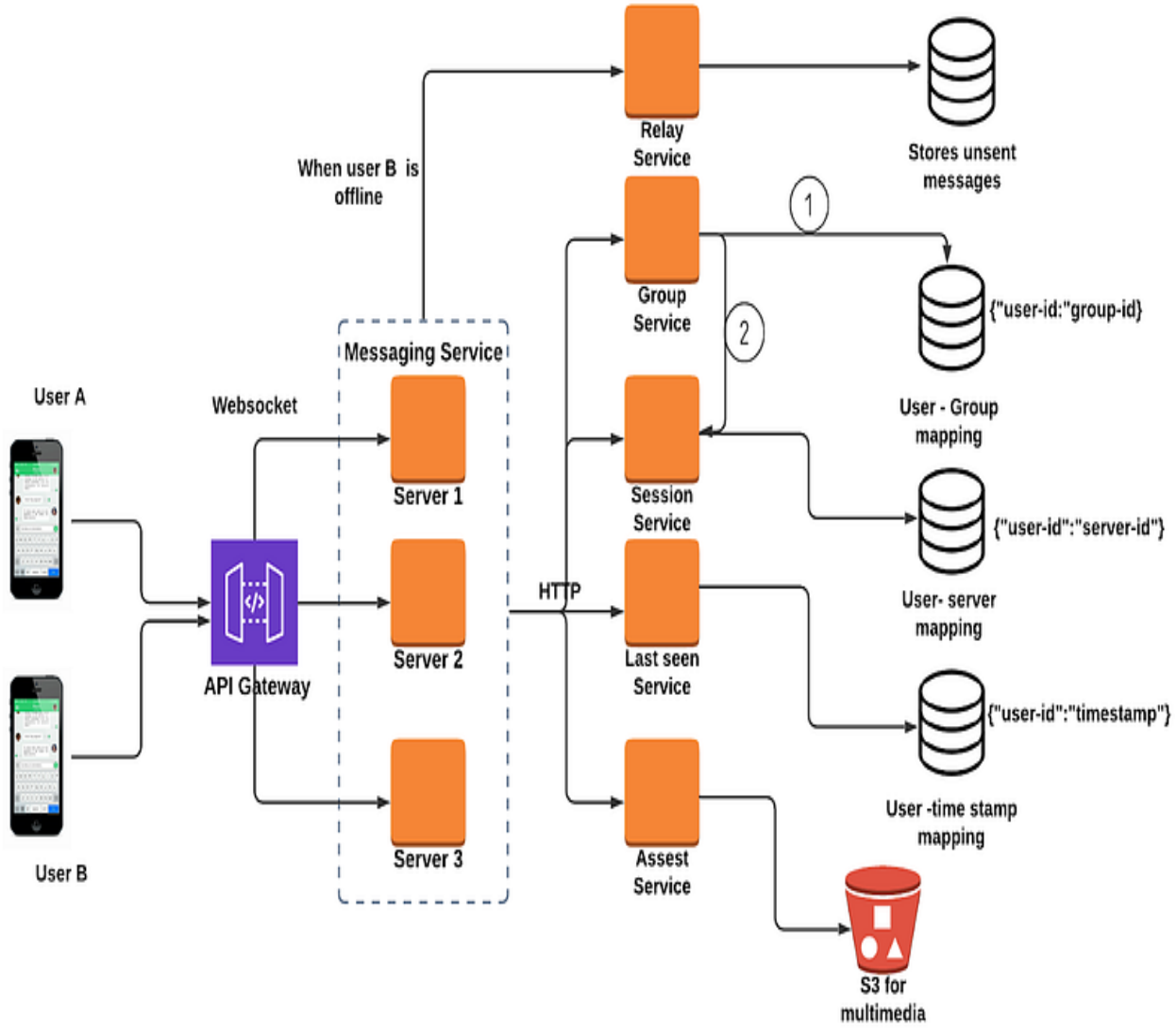


Asset service :

Asset service is used to store and retrieve multimedia files in an object based storage like AWS S3 bucket.

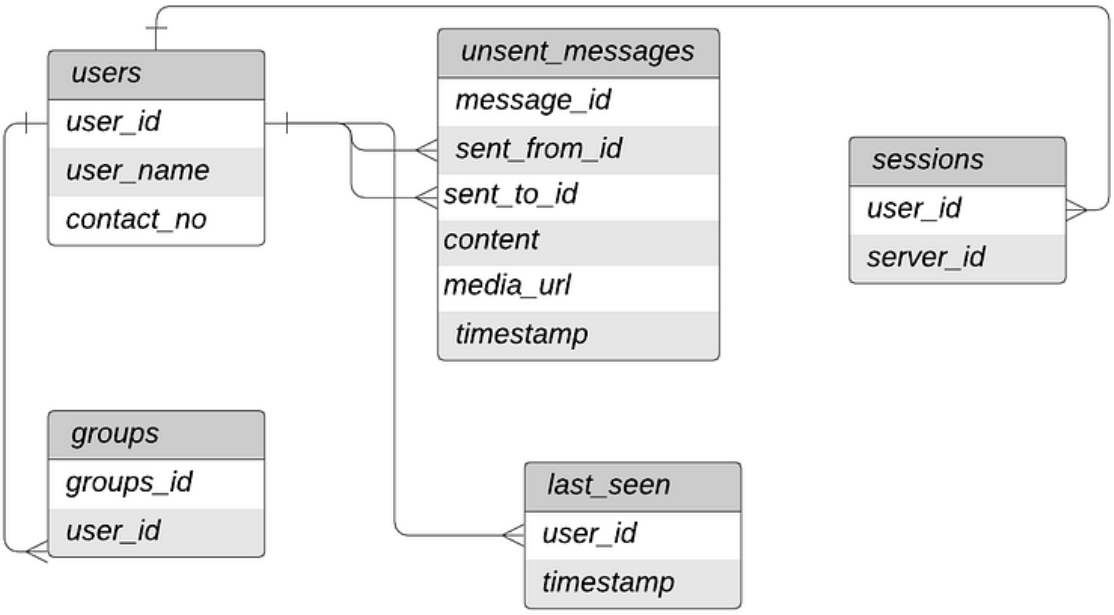


High Level Design



Database Schema

The choice of database can be RDBMS since we are dealing with more relations here.



SUBMITTED BY:

BHUVANESHWARI.S

HARISTHA.S

BHAVADHARANI.D

GRACELIN.J