System Design Ginger-Chat_App

1. Introduction

This is the system design of a messaging service prototype named Ginger.

1.1. Requirements

So, the functional requirements of this app are:

- Registration and Login Authentication
- One-one chat: two users can chat with each other
- Group chat: users can participate in group conversations
- Join/leave groups: (add/delete users)
- Typing indicator: when typing, the recipient gets notified
- Notifications
- Share texts messages

Now, the non-functional requirements of this app are:

- Low Latency
- Available
- Scalable

1.2. Traffic Estimation

For the traffic estimation, I have roughly estimated the number of users.

A research shows that Express.js can handle approx 15,000 requests per second and the basic HTTP module, 70K requests per second.

1.3. Storage Estimation

I have used MongoDB database which is a nonSQL database. For my app the chat history is being stored in the database.

The information of the registered user is stored in the database, which is used later on when they try to login. The get authenticated as registered user and gets successfully logged in.

The maximum size an individual document can be in MongoDB is 16MB with a nested depth of 100 levels.

2. High-Level Design

2.1. Basic API services

Send_Message(sender_userID, reciever_userID,message)

Get_Messages(user_Id1, user_Id2, latest message)

Join_Group(user_Id, group_Id)

Leave_Group(user_Id, group_Id)

Login credentials of registered user.

2.2. Database Design

Read Operations

- Given user A and user B, retrieve messages, find user ID
- Given a group G, retrieve all messages, find all member user ID’s
- Given user joining different groups, so, group ID

Write Operations

- Save a message between user A and user B
- Save a new message by user A in group G
- Add/delete user A to/from group G

2.2. Architecture

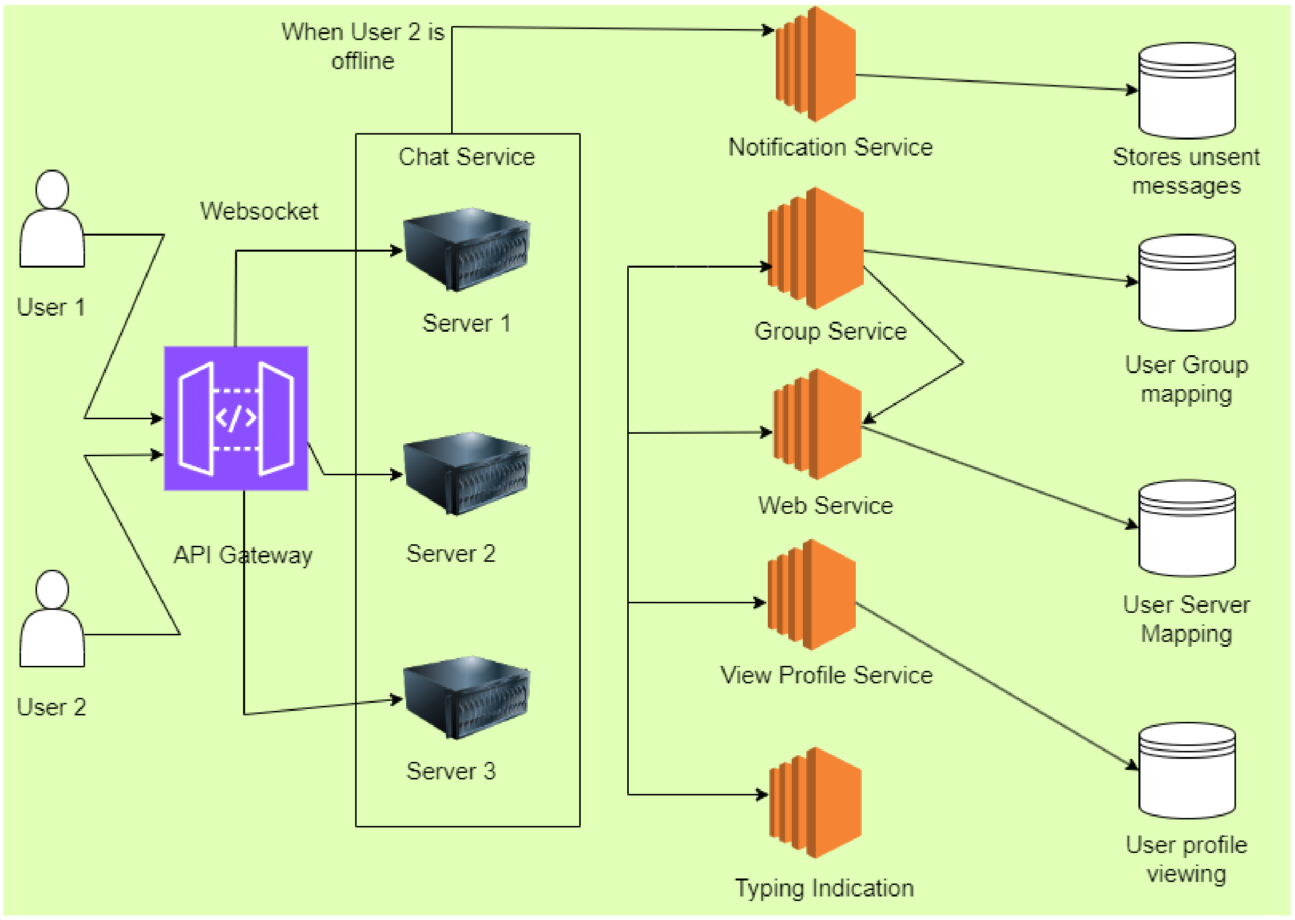
- **Chat Service:** Each online user maintains a WebSocket connection with a WebSocket server in the Chat Service. Outgoing and incoming chat messages are exchanged here.

- **Web Service:** Users talk to this service for authentication, join/leave groups, etc.
- **Notification Service:** When the user is not present in the one-one chat box or group chat-box, then it receives notification, notifying him about the latest message.
- **View Profile Service:** A user can view his/her profile. Also it has gives the facility of viewing other users profile and email.
- **Group Messaging Service:** In this the message is published to all the users associated with the same group ID.
- **Typing Indication:** This helps in telling that the screen-side user is messaging in real-time.

3. Summary

In this system design of Ginger, A chat web-app messaging service prototype. I discussed about the high-level architecture for the app, defined system API's as well as explained how different functionalities are working.

4. Diagrams:


When User 2 is offline
Notification Service
Stores unsent messages
Chat Service
Websocket
User 1
Server 1
Group Service
User Group mapping
Web Service
API Gateway
Server 2
User Server Mapping
User 2
View Profile Service
Server 3
User profile viewing
Typing Indication