A Project Report

on

CHAT APPLICATION CHATTERLY USING NODE.JS, SOCKET.IO, EXPRESS

Submitted in partial fulfillment of the Requirement for the award of the degree of

B.TECH
COMPUTER SCIENCE
AND ENGINEERING

Under The Supervision of
Dr.kirti Shukla
ASSISTANT PROFESSOR
Submitted By

RUDRAKSH YADAV
18021012030

SOMDUTT SHARMA
18021011694
SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING / DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"CHATTERLY"** in partial fulfillment of the requirements for the award of the Bachelor of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of September2021 to December 2021, under the supervision of Dr.KIRTI SHUKLA Assistant Professor, Department of Computer Science and Engineering, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

RUDRAKSH YADAV 18021012030

SOMDUTT SHARMA 18021011694

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr.KIRTI SHUKLA

Assistant Professor

CERTIFICATE

The Final Project Viva-Voce examination of RUDRAKSH YADAV 18SCSE1010434 and SOMDUTT SHARMA 18SCSE1010466 has been held on _____ and the work is recommended for the award of Bachelor of Technology.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

Abstract

In this project, one can do live chat with others. For this we use server-client architecture. In this, we used to have a server and with this server different clients can connect. Now, in order to establish connection with server with clients we use the concept of socket programming. Now let me throw some light on socket programming. Since for a machine to connect with other machine there is need of socket. So with the help of sockets server, can also connect to different clients and thus data transfer can take place in very efficient way. In this server program will run on server and during the time server program is running, client will also run its program from respective computer and matching the IP address and port number of server connection will take place and data transmission can occur. Similarly, various clients can connect to server through server's port number. Thus, client will send request to server for connection and server will respond to request and thus live chat will take place on server between various clients connected to server.

Table of Contents

List of Figures

Acronyms

NLP	Natural Language Processing
ML	Machine Learning
AI	Artificial Intelligent
DL	Deep Learning
SVM	Support Vector Machine
LR	Logistic Regression
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
NB	Naïve Bayes
DT	Decision Tree
ANN	Artificial Neural Network

Chapter 1

INTRODUCTION

In this project, one can do live chat with others. For this we use server-client architecture. In this, we used to have a server and with this server different clients can connect. Now, in order to establish connection with server with clients we use the concept of socket programming. Now let me throw some light on socket programming. Since for a machine to connect with other machine there is need of socket. So with the help of sockets server, can also connect to different clients and thus data transfer can take place in very efficient way. In this server program will run on server and during the time server program is running, client will also run its program from respective computer and matching the IP address and port number of server connection will take place and data transmission can occur. Similarly, various clients can connect to server through server's port number. Thus, client will send request to server for connection and server will respond to request and thus live chat will take place on server between various clients connected to server.

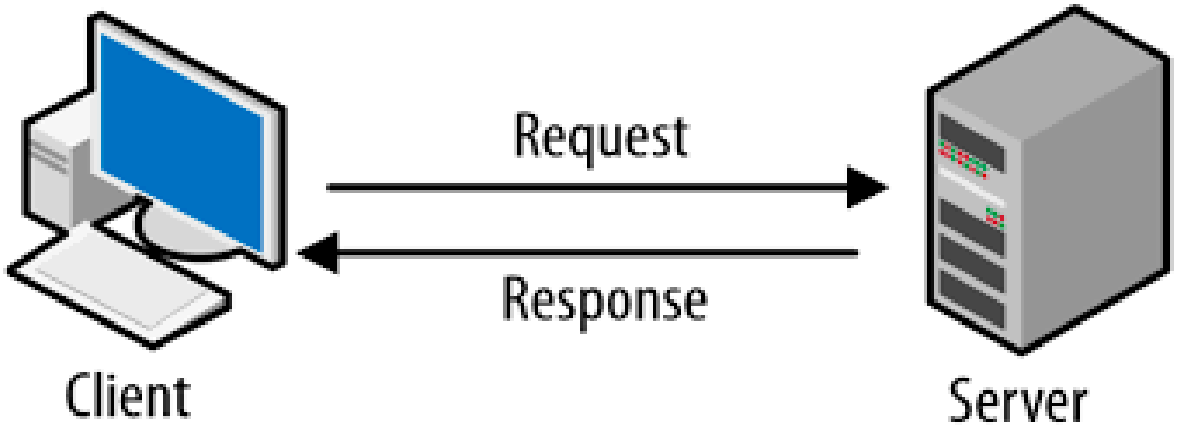

Figure 1

This project is about how different users can communicate in real time with each. With real time it can said to communicate live with each other on a single platform. So to make this thing possible we have a communication platform for chatting which is client server architecture. In this application, we will have various clients who can join a private or public server and can communicate with other clients using that server. In this world we have different platforms which uses this concept like you tube. In you tube, when someone do live streaming you will see various people are chatting side by side of video in real time. So here there is a server of you tube and we have various users as clients who are connected to that server and it is server-client architecture through which it was possible that all users were able to have a communication with other clients in present time on a single platform.

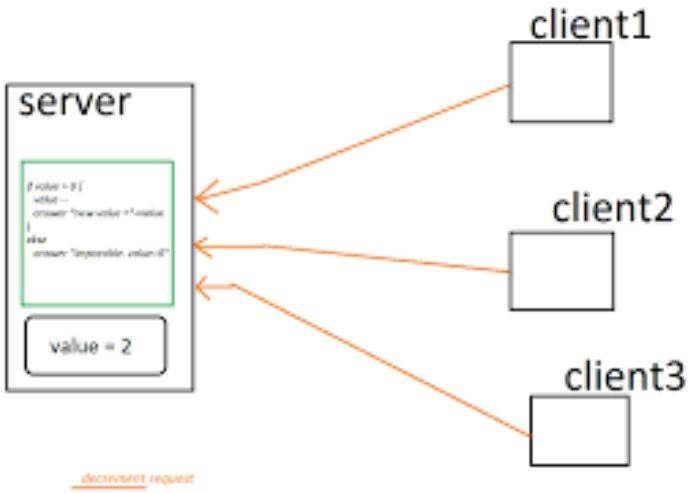

Figure 2

Now in this chat app to communicate through machines of individuals and server we have a thing in each machine known as sockets. These are only sockets through which it was possible for different machines to establish connection with each other. This establishment of different machines is possible only due to socket programming. Due to socket programming it is possible for machines to have end to end communication and send data from one device to other. Now we will see about sockets in detail in system development section.

NEED OF SENTIMENTAL ANALYSIS

Sentiment analysis is one of the most recent applications of natural language processing. With the advent of mobile chat applications, sentiment analysis can be done on the textual data that is generated by the application. The proposed sentiment analysis system consists of two phase – Recognition Phase and Distribution Phase. Recognition phase uses as set of features to perform sentiment classification. Logistic regression algorithm is employed to train the classifiers. The training set

required for system Recognition comprises of the text messages sent by the user. In the distribution phase, the application server sends out the updated contact lists of all other nodes with the latest evaluated sentiment .This second phase of the system uses PUSH Technology to distribute the result of the sentiment analysis over the network**.**

SENTIMENT CLASSIFICATION

The sentiment analysis system comprises of two phases. Recognition and distribution phase. Architecture of the system is described according to the phases as follows.

A. Recognition Phase

The Recognition phase of the system is implemented on every single device. Every user has a device from which he/she uses the chat application. Ex. Mobile phones, Tablets. The Recognition phase module runs after a certain period of time to predict the sentiment of the user. It transforms the textual data into the selected features and using logistic regression, recognizes the emotion.

Sentiments are measured on two different scales - Emotion and Intensity. The system classifies emotion meter in five classes – Very Happy, Happy, Neutral, Sad and Very Sad. The intensity meter is classified similarly – Very Excited, Excited, Neutral, Muted and Very Muted. The output of the system will be the combination of the two scales. The sentiment will be predicted as a combination of the two scales. This module will run after every time period and calculate the sentiment by analyzing all the text messages sent in this period on every single chat.

B. Distribution Phase

The distribution phase of the system has three components. First is the device on which the Recognition phase will be running. Second is the application server through which all the interactions take place. Third is the device whose status or sentiments list has to be updated.

Formulation of Problem

The evolution of the internet technologies had benefit people to accessing to the web easily. More and more services provide by this internet All of this can be virtualize thank to the technologies. Communication between people using the internet becomes part of their daily life. People used to communicate with each other's using the online chat system to transfer their messages. Traditionally, when people need to communicate with others they will have a face to face conversation to deliver the message, same goes to the education field. It is strongly encourage that student seeking for academic assistance from the lecturer when they face difficulties. Most often happening is when the exam are in the corner and or assignment due date. The traditional way to have a consultation is student make an email appointment with the lecturer and the lecturer accepted the appointment or lecturer is free and available at his room or lecturer consultation hour. However, this communication solution might be not convenient and not efficient due to some issue that happen before the consultation started. The consultation session can be realized in another similar way using an online solution. There are few issue might be arise when the student want to have consultation with the lecturer using the traditional method which is making appointment and meet at a lecturer room to having the communication.

1. Student or lecturer are not available in the school

Sometime, the lecturer and student might have some personal issue and cannot come to school. For example the lecturer has to outstation for some important works and the outstation will take a period of time. Also student cannot come to school because they are not available in the area that near to the school. During this period of time, if a student wanted to have a consultation session with their teacher, they cannot meet the respective teacher in the university which causes the student delay their studies progress.

2 Fail to recall about the consultation

It is human nature that forgets some stuff from time to time. It is possible that either student or lecturer have totally forgotten about the consultation. If it is happen, these will caused another parties to wasting time and wait. Besides that, it is possible that either student or lecturer have forgotten what is the context or topic will be discuss in the later appointment. This is because student or lecturer has totally forgotten about there will be a consultation session later.

TOOLS AND TECHNOLOGY USED

For the proper working of the system we can list our assumptions and dependencies as follows:

- **VS Code**

visual studio code is an integrated Development environment made by a Microsoft for windows, Linux and macOS. Feature include support for debugging, syntax, highlighting, intelligent code completion, snippets ,code refactoring, and embedded.

• Node.js

It is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside the browser. The most important advantage of using Node is that we can use JavaScript as both a front-end and back-end language.

• NPM Modules

Nodejs allows the modules of libraries to be included in the application. These modules can be user-defined or third party modules.

• **Express .JS**

Express.js, or simply Express, is a web application framework for Node.js. Express provides a robust set of features for web and mobile applications. Express provides a thin layer of fundamental web application features, without obscuring Node.js features.

Chapter 2
Literature Review

Introduction

Messaging apps now have more global users than traditional social networks—which mean they will play an increasingly important role in the distribution of digital information in the future. In 2016, over 2.5 billion people used at least one messaging app. That's one-third of the world's entire population, with users ranging from various age grades. Today, it's common place for offices to use a messaging app for internal communication in order to coordinate meetings, share pitch decks, and plan happy hours. And with the latest bot technology, chat apps are becoming a hub for employees to do work in their apps without leaving the chat console. For many people, chat apps are a given part of their workday. But how did these chat apps become so popular?

Historical Overview

2.1 INSTANT MESSAGING:

CHILD OF THE 90'S Chat apps (and their subsets, chat rooms) bring to remembrance images of the 1990s, with its dial-up internet and classic sitcoms, however, commercial chat apps date back to the 1980s. CompuServe released CB Simulator in 1980, and 1985 brought the launch of Commodore's Quantum Link (also known as Q-Link). An online service, it allowed multiuser chat, email, file sharing, and games. If Q-Link sounds familiar, that's because it is: in 1991, the company changed its name to America Online (AOL). But AOL wouldn't launch its signature product, AOL Instant Messenger (AIM), until 1997. In the meantime, the Vodafone GSM network enabled the first SMS in 1992. And in 1996, ICQ launched as the first widely-adopted instant messaging platform. 6 The late 1990s

brought dramatic changes in the chat app market. Both Yahoo! And MSN launched their own instant messengers (in 1998 and 1999, respectively), and AOL bought out competitor ICQ's parent company, Mirabilis, for a fee around $287 million upfront, with an additional $120 million paid out later. That's about $612.7 million today! As AIM pioneered Chabot, like SmarterChild, it also increased its market share. Its rivalry with MSN Messenger began almost as soon as the competitor launched in 1999.

Figure 3

By 2006, AIM controlled 52% of the IM market. Its dominance, however, was

short lived. Struggling to monetize and facing increasing competition from new apps like Skype and Google Talk, AIM fell out of favour, eventually eliminating its entire development team in 2012.

2.2. THE SLOW GROWTH OF ENTERPRISE CHAT APPS

Despite developing around the same time, the history of enterprise chat apps is markedly different than the story of their consumer-facing counterparts. The very first enterprise chat apps do not enjoy the same place in our collective memory as AIM and ICQ. Early contender Yammer launched in 2008; Microsoft acquired the platform in 2012. Clearspace began in 2006, rebranding several times until its rebirth as Jive six years later. Other enterprise chat programs, usually integrated with other social features like blogs and wikis, blipped in and out of existence. None of this initial crop of enterprise apps proved a runaway success, and many theories exist as to why later programs have overshadowed them. One thing is certain: these programs predated the rise of smart phones, and mobility certainly fomented the creation of secondgeneration commercial apps like WhatsApp and Snapchat.

2.3 ENTERPRISE CHAT: THE NEXT GENERATION

As the first decade of the new millennium closed, an enterprise chat app renaissance slowly began. Though email had been widely used for the previous twenty years, companies soon began looking for a better way to communicate quickly; email-based workflows are slower and do not allow for many business functions that are now critical to work, like screensharing or video calls. Some one-to-one chat applications existed, like GChat and Outlook Messenger, but group messaging applications had yet to take off. In January 2010, three graduates of Rensselaer Polytechnic Institute—Chris Rivers, Garret Heaton, and Pete

Curley—launched Hipchat, a web-based chat and instant messaging service. Shortly thereafter, Atlassian acquired it in March 2012. Its premium version addressed several enterprise concerns by adding screens haring, history retention controls, and the ability to run within corporate firewalls.

In August 2013, the enterprise chat space exploded. Slack, which stands for "Searchable Log of All Conversation and Knowledge," grew out of an internal tool used during the development of Glitch, a defunct online game. Though not the first (or even the most revolutionary) office chat app, Slack's growth—it now has four million users—has dwarfed most other programs. A mere 1.25 years after launching, it reached a valuation of $1 billion. By April 2015, Slack was worth almost three times that. Slack–with its channels, DMs, and private groups–is reminiscent of IRC, which might help explain its popularity. A host of similar apps have since incorporated these features. 2015 saw the launch of Cisco Spark, which has since evolved into an entire ecosystem of SDKs, APIs, and a developer site. Microsoft designed MS Teams to compete directly with Slack; it launched in early 2017 as an integrated component of Office 365. As businesses prioritize digital transformation, enterprise chat apps continue to enjoy overwhelming popularity. What's to come? Chatbots, machine learning, and increased integration are all popular with both enterprise and commercial users, who discover new ways to chat every day.

2.4 INDUSTRY CHALLENGES

1. FRAGMENTATION: The social media landscape is entering a period of hyperfragmentation that may be a challenge to publishers: Facebook, Twitter, and Instagram continue to loom large, but social media managers can now launch official channels on roughly 10 chat apps with over 50 million monthly, active

users each.

2. ANALYTICS: For organizations accustomed to robust, real-time data, the lack of good analytics tools for messaging apps remains a major deterrent to adoption. The challenge is twofold: Strong analytics dashboards take time to build, and many messengers are privacycentric by nature.

2.2.6 INDUSTRY OPPORTUNITIES

1. HIGHER ENGAGEMENT: Since many chat apps provide publishers with push notifications or chatbot experiences (programmable robots that converse with users), they can deliver significantly higher engagement rates.

2. AUDIENCE DEVELOPMENT: With billions of active users across multiple major chat apps, there is the opportunity in building large audiences fairly quickly on several platforms.

3. A CHANCE TO CONNECT WITH USERS IN A NEW WAY: Messaging apps offer a host of features not unavailable on social networks or other platforms. Programmers can creatively leverage these tools to socialize in new ways.

2.5 Related Work

2.5.1 WHAT THE FUTURE HOLDS

Predictions are always prone to inevitable ridicule and failure, but there are some that are worth making.

2.5.1.1 THE GROWING IMPORTANCE OF SECURITY, CIRCUMVENTION, AND DATA RESTRICTIONS

As government snooping, personal privacy, and security become issues for many people globally, those living in countries where these are particular concerns will increasingly look for platforms that enable them to both communicate securely and

receive accurate information, unfiltered by government censors.

2.5.1.2 THE EMERGENCE OF REGIONALIZED AND LOCAL MESSAGINGAPP ECOSYSTEMS

This is the era to launch CHATTERLY as it focuses on specific target audience in Africa**.**

2.5.1.3 MESSAGING WILL BECOME LIKE ELECTRICITY

While messaging is currently a clearly defined function of specific apps, the future is likely to be one wherein the capability is baked-in to nearly all digital technologies and services. The point where a messaging app begins and ends will begin to blur. Already, app classification is getting trickier, especially as social media platforms update their in-app messaging capabilities, moving them closer to chat app experience.

Chapter 3
Working Model

Understand the problem and establish design scope

It is vital to agree on the type of chat app to design. In the marketplace, there are one-on-one chat apps like Facebook Messenger, WeChat, and WhatsApp, office chat apps that focus on group chat like Slack, or game chat apps, like Discord, that focus on large group interaction and low voice chat latency.

The first set of clarification questions should nail down what the interviewer has in mind exactly when she asks you to design a chat system. At the very least, figure out if you should focus on a one-on-one chat or group chat app. Some questions you might ask are as follows:

Candidate: What kind of chat app shall we design? 1 on 1 or group based?
Interviewer: It should support both 1 on 1 and group chat.

Candidate: Is this a mobile app? Or a web app? Or both?
Interviewer: Both.

Candidate: What is the scale of this app? A startup app or massive scale?
Interviewer: It should support 50 million daily active users (DAU).

Candidate: For group chat, what is the group member limit?
Interviewer: A maximum of 100 people

Candidate: What features are important for the chat app? Can it support attachment?
Interviewer: 1 on 1 chat, group chat, online indicator. The system only supports text messages.

Candidate: Is there a message size limit?
Interviewer: Yes, text length should be less than 100,000 characters long.

Candidate: Is end-to-end encryption required?
Interviewer: Not required for now but we will discuss that if time allows.

In the chapter, we focus on designing a chat app like Facebook messenger, with an emphasis on the following features:

- A one-on-one chat with low delivery latency
- Small group chat (max of 100 people)
- Online presence
- Multiple device support. The same account can be logged in to multiple accounts at the same time.
- Push notifications

Propose high-level design

To develop a high-quality design, we should have a basic knowledge of how clients and servers communicate. In a chat system, clients can be either mobile applications or web applications. Clients do not communicate directly with each other. Instead, each client connects to a chat service, which supports all the features mentioned above. Let us focus on fundamental operations. The chat service must support the following functions:

- Receive messages from other clients.
- Find the right recipients for each message and relay the message to the recipients.
- If a recipient is not online, hold the messages for that recipient on the server until she is online.

Figure.4

When a client intends to start a chat, it connects the chats service using one or more network protocols. For a chat service, the choice of network protocols is important. Let us discuss this with the interviewer.

Requests are initiated by the client for most client/server applications. This is also true for the sender side of a chat application. when the sender sends a message to the receiver via the chat service, it uses the time-tested HTTP protocol, which is the most common web protocol. In this scenario, the client opens a HTTP connection with the chat service and sends the message, informing the service to send the message to the receiver. The keep-alive is efficient for this because the keep-alive header allows a client to maintain a persistent connection with the chat service. It also reduces the number of TCP handshakes. HTTP is a fine option on the sender side, and many popular chat applications such as Facebook [1] used HTTP initially to send messages.

However, the receiver side is a bit more complicated. Since HTTP is client-initiated, it is not trivial to send messages from the server. Over the years, many techniques are used to simulate a server-initiated connection: polling, long polling, and WebSocket. Those are important techniques widely used in system design interviews so let us examine each of them.

Polling

Polling is a technique that the client periodically asks the server if there are messages available. Depending on polling frequency, polling could be costly. It could consume precious server resources to answer a question that offers no as an answer most of the time.

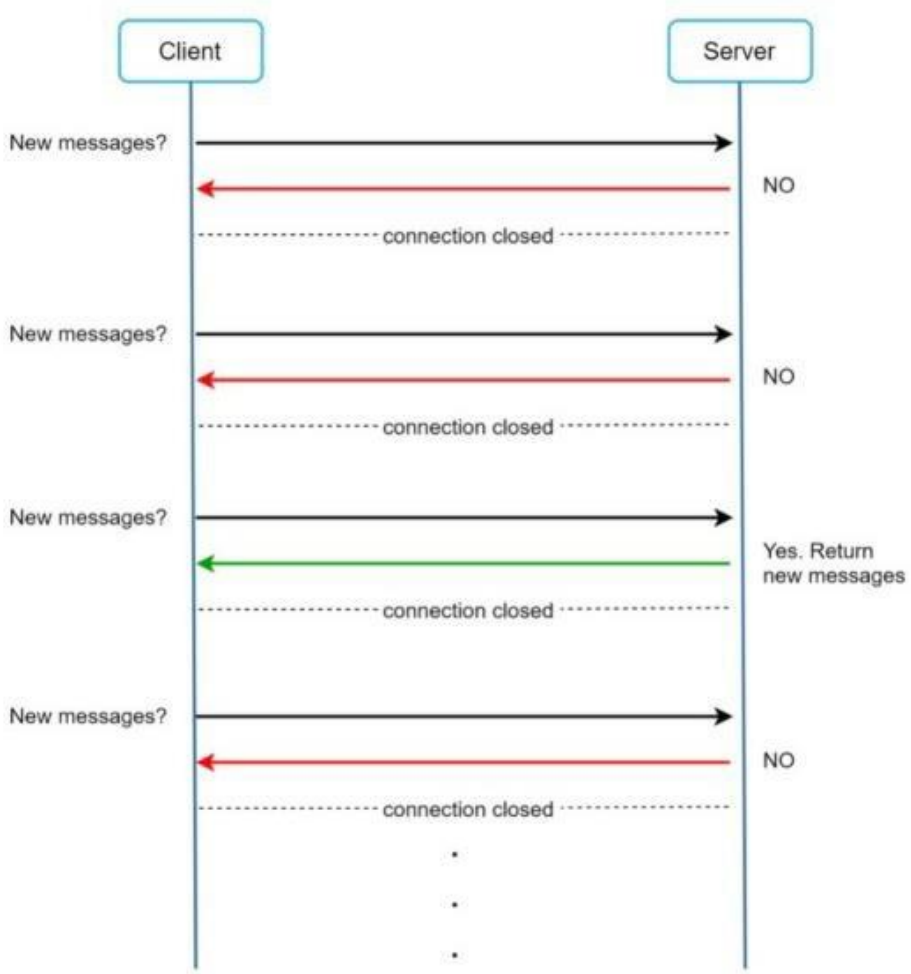

FIGURE 5

Long polling

Because polling could be inefficient, the next progression is long polling.

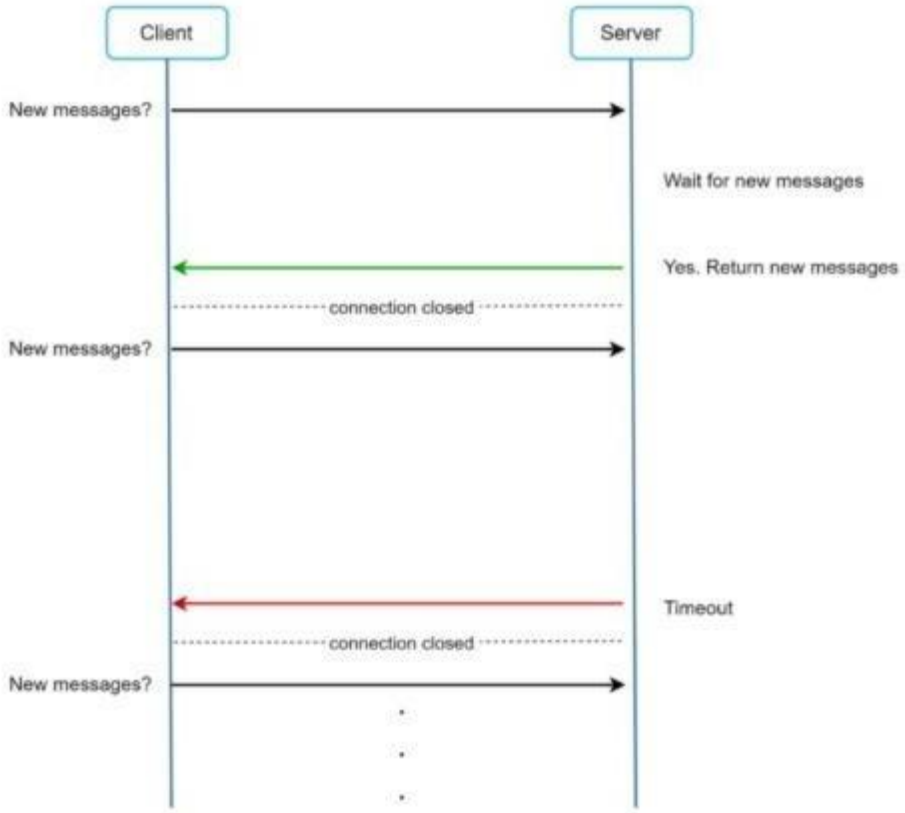

FIGURE.6

In long polling, a client holds the connection open until there are actually new messages available or a timeout threshold has been reached. Once the client receives new messages, it immediately sends another request to the server, restarting the process. Long polling has a few drawbacks:

- Sender and receiver may not connect to the same chat server. HTTP based servers are usually stateless. If you use round robin for load balancing, the server that receives the message might not have a long-polling connection with the client who receives the message.
- A server has no good way to tell if a client is disconnected.
- It is inefficient. If a user does not chat much, long polling still makes periodic connections after timeouts.

WebSocket

WebSocket is the most common solution for sending asynchronous updates from server to client. Figure 7 shows how it works.

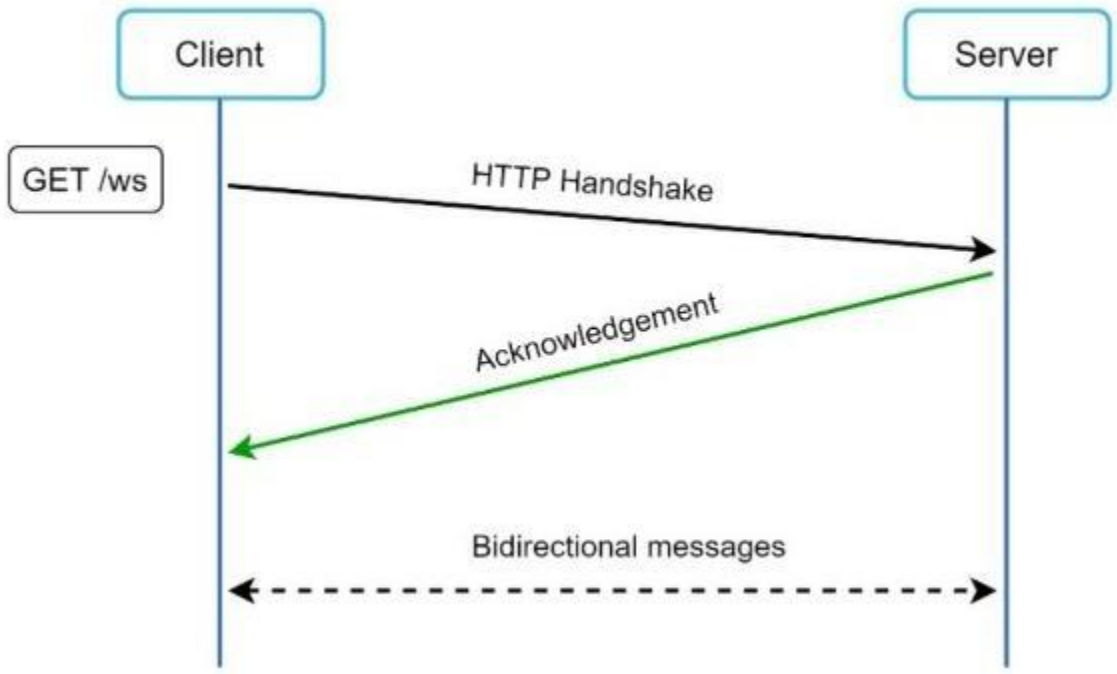

FIGURE.7

WebSocket connection is initiated by the client. It is bi-directional and persistent. It starts its life as a HTTP connection and could be “upgraded” via some well-defined handshake to a WebSocket connection. Through this persistent connection, a server could send updates to a client. WebSocket connections generally work even if a firewall is in place. This is because they use port 80 or 443 which are also used by HTTP/HTTPS connections.

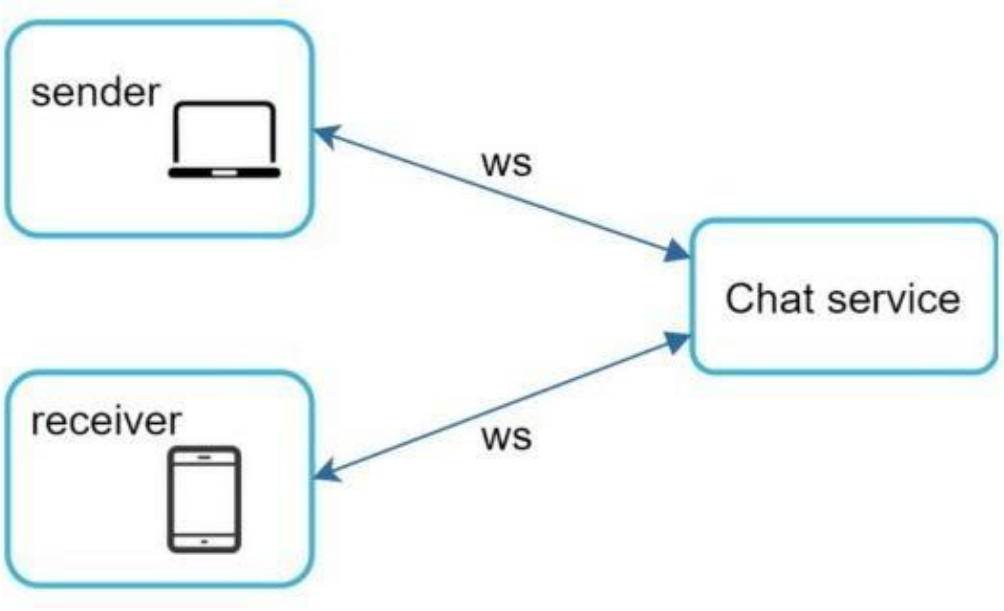

FIGURE.8

By using WebSocket for both sending and receiving, it simplifies the design and makes implementation on both client and server more straightforward. Since WebSocket connections are persistent, efficient connection management is critical on the server-side.

High-level design

Just now we mentioned that WebSocket was chosen as the main communication protocol between the client and server for its bidirectional communication, it is

important to note that everything else does not have to be WebSocket. In fact, most features (sign up, login, user profile, etc) of a chat application could use the traditional request/response method over HTTP. Let us drill in a bit and look at the high-level components of the system.

the chat system is broken down into three major categories: stateless services, stateful services, and third-party integration.

Stateless Services

Stateless services are traditional public-facing request/response services, used to manage the login, signup, user profile, etc. These are common features among many websites and apps.

Stateless services sit behind a load balancer whose job is to route requests to the correct services based on the request paths. These services can be monolithic or individual microservices. We do not need to build many of these stateless services by ourselves as there are services in the market that can be integrated easily. The one service that we will discuss more in deep dive is the service discovery. Its primary job is to give the client a list of DNS host names of chat servers that the client could connect to.

Stateful Service

The only stateful service is the chat service. The service is stateful because each client maintains a persistent network connection to a chat server. In this service, a client normally does not switch to another chat server as long as the server is still available. The service discovery coordinates closely with the chat service to avoid server overloading. We will go into detail in deep dive.

Third-party integration

For a chat app, push notification is the most important third-party integration. It is a way to inform users when new messages have arrived, even when the app is not running. Proper integration of push notification is crucial. Refer to Chapter 10 Design a notification system for more information.

Scalability

On a small scale, all services listed above could fit in one server. Even at the scale we design for, it is in theory possible to fit all user connections in one modern cloud server. The number of concurrent connections that a server can handle will most likely be the limiting factor. In our scenario, at 1M concurrent users, assuming each user connection needs 10K of memory on the server (this is a very rough figure and very dependent on the language choice), it only needs about 10GB of memory to hold all the connections on one box.

If we propose a design where everything fits in one server, this may raise a big red flag in the interviewer's mind. No technologist would design such a scale in a single server. Single server design is a deal breaker due to many factors. The single point of failure is the biggest among them.

However, it is perfectly fine to start with a single server design. Just make sure the interviewer knows this is a starting point. Putting everything we mentioned together, Figure 8 shows the adjusted high-level design.

In Figure 8, the client maintains a persistent WebSocket connection to a chat server for real-time messaging.

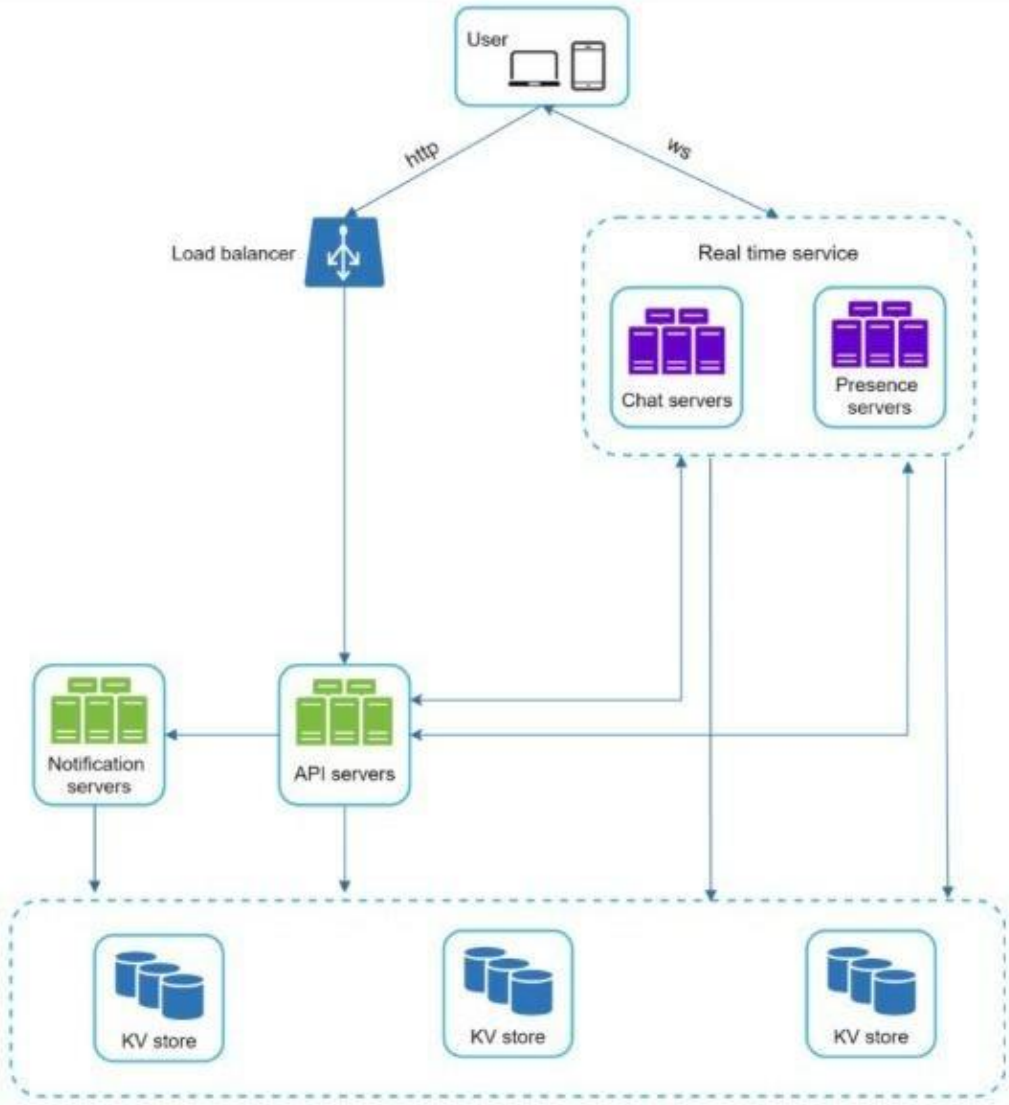

FIGURE 9

- Chat servers facilitate message sending/receiving.
- Presence servers manage online/offline status.
- API servers handle everything including user login, signup, change profile, etc.
- Notification servers send push notifications.
- Finally, the key-value store is used to store chat history. When an offline user comes online, she will see all her previous chat history.

Storage

At this point, we have servers ready, services up running and third-party integrations complete. Deep down the technical stack is the data layer. Data layer usually requires some effort to get it correct. An important decision we must make is to decide on the right type of database to use: relational databases or NoSQL databases? To make an informed decision, we will examine the data types and read/write patterns.

Two types of data exist in a typical chat system. The first is generic data, such as user profile, setting, user friends list. These data are stored in robust and reliable relational databases. Replication and sharding are common techniques to satisfy availability and scalability requirements.

The second is unique to chat systems: chat history data. It is important to understand the read/write pattern.

- The amount of data is enormous for chat systems. A previous study [2] reveals that Facebook messenger and Whatsapp process 60 billion messages a day.
- Only recent chats are accessed frequently. Users do not usually look up for old chats.
- Although very recent chat history is viewed in most cases, users might use features that require random access of data, such as search, view your mentions, jump to specific messages, etc. These cases should be supported by the data access layer.
- The read to write ratio is about 1:1 for 1 on 1 chat apps.

Selecting the correct storage system that supports all of our use cases is crucial. We recommend key-value stores for the following reasons:

- Key-value stores allow easy horizontal scaling.
- Key-value stores provide very low latency to access data.
- Relational databases do not handle long tail [3] of data well. When the indexes grow large, random access is expensive.
- Key-value stores are adopted by other proven reliable chat applications. For example, both Facebook messenger and Discord use key-value stores. Facebook messenger uses HBase [4], and Discord uses Cassandra [5].

Data models

Just now, we talked about using key-value stores as our storage layer. The most important data is message data. Let us take a close look.

Message table for 1 on 1 chat

Figure 9 shows the message table for 1 on 1 chat. The primary key is message_id, which helps to decide message sequence. We cannot rely on created_at to decide the message sequence because two messages can be created at the same time.

Message table for group chat

The composite primary key is (channel_id, message_id). Channel and group represent the same meaning here. channel_id is the partition key because all queries in a group chat operate in a channel.

Message ID

How to generate message_id is an interesting topic worth exploring. Message_id carries the responsibility of ensuring the order of messages. To ascertain the order of messages, message_id must satisfy the following two requirements:

- IDs must be unique.
- IDs should be sortable by time, meaning new rows have higher IDs than old ones.

How can we achieve those two guarantees? The first idea that comes to mind is the "auto_increment" keyword in MySql. However, NoSQL databases usually do not provide such a feature.

The second approach is to use a global 64-bit sequence number generator like Snowflake [6]. This is discussed in "Chapter 7: Design a unique ID generator in a distributed system".

The final approach is to use local sequence number generator. Local means IDs are only unique within a group. The reason why local IDs work is that maintaining message sequence within one-on-one channel or a group channel is sufficient. This approach is easier to implement in comparison to the global ID implementation.

Step 3 - Design deep dive

In a system design interview, usually you are expected to dive deep into some of the components in the high-level design. For the chat system, service discovery, messaging flows, and online/offline indicators worth deeper exploration.

Service discovery
The primary role of service discovery is to recommend the best chat server for a client based on the criteria like geographical location, server capacity, etc. Apache Zookeeper [7] is a popular open-source solution for service discovery. It registers all the available chat servers and picks the best chat server for a client based on predefined criteria.

1. User A tries to log in to the app.
2. The load balancer sends the login request to API servers.
3. After the backend authenticates the user, service discovery finds the best chat server for User A. In this example, server 2 is chosen and the server info is returned back to User A.
4. User A connects to chat server 2 through WebSocket.

Message flows

It is interesting to understand the end-to-end flow of a chat system. In this section, we will explore 1 on 1 chat flow, message synchronization across multiple devices and group chat flow.

Wrap up

In this chapter, we presented a chat system architecture that supports both 1-to-1 chat and small group chat. WebSocket is used for real-time communication between the client and server. The chat system contains the following components: chat servers for real-time messaging, presence servers for managing online presence, push notification servers for sending push notifications, key-value stores for chat history persistence and API servers for other functionalities.

If you have extra time at the end of the interview, here are additional talking points:

- Extend the chat app to support media files such as photos and videos. Media files are significantly larger than text in size. Compression, cloud storage, and thumbnails are interesting topics to talk about.
- End-to-end encryption. Whatsapp supports end-to-end encryption for messages. Only the sender and the recipient can read messages.

- Caching messages on the client-side is effective to reduce the data transfer between the client and server.
- Improve load time. Slack built a geographically distributed network to cache users' data, channels, etc. for better load time [10].
- Error handling.
    - The chat server error. There might be hundreds of thousands, or even more persistent connections to a chat server. If a chat server goes offline, service discovery (Zookeeper) will provide a new chat server for clients to establish new connections with.
    - Message resent mechanism. Retry and queueing are common techniques for resending messages.

CHAPTER-5
CONCLUSION

This project is talking about developing a chat system for the teacher and student. The motivation to develop this project is because communication through the internet has become common in daily life. Communication using the internet in the academic field might help the student and teacher communicate more effectively and efficiently. The transitional consultation method may give several problems. The first problem is that a student or lecturer is not available in the school. With the online consultation solution, location is not a concern as long as you have internet. Furthermore, if the student needs to show their work on their computer they can show it here because some students might only have a desktop, not a laptop. It is not convenient to bring the entire desktop to the school. The second issue is that bunches of email have to be filtered by the lecturer. A systematic appointment system will help in managing the appointment more effectively and efficiently. The student or teacher can manage their appointment easily using the appointment system. The third issue is the arrangement issue regarding the appointment. Lecturer might mistakenly schedule the appointment. The appointment system will detect if the time slot is available and ensure that no 2 consultation sessions are scheduled at same time. If the time slot either lecturer or student have clashing the appointment will not be made. The last issue is failing to recall about the consultation. It is human nature that forgets some stuff from time to time. SMS reminder system developed helped to remind the lecturer about the consultation session. In order to develop the WebRTC, RTCMulticonnection has been implemented. Thanks to Muaz Khan for developing this awesome API. The documentation also explained in very detail about how to use the library.

FutureScope

Sentiment analysis is a uniquely powerful tool for businesses that are looking to measure attitudes, feelings and emotions regarding their brand. To date, the majority of sentiment analysis projects have been conducted almost exclusively by companies and brands through the use of social media data, survey responses and other hubs of user-generated content. By investigating and analyzing customer sentiments, these brands are able to get an inside look at consumer behaviors and, ultimately, better serve their audiences with the products, services and experiences they offer.

As time goes on, the system should be improved to a bigger scope. There are few recommendations that can be used in future development. Firstly, the online consultation solution which is the WebRTC can be built on the mobile browser since today a lot of people can afford to have a smartphone. Secondly, the appointment system can add 1 more module which makes the teacher set a period of availability time. When the student makes the appointment with the teacher, if the system detects the time is not in range of the availability time, it will display an error message. Thirdly, if a group of students have the same topic of question. The lecturer manages to edit the appointment and can be joined by how many people. After that the online consultation will perform one to many broadcasts for the group of students. Last but not least, the entire online consultation session can be recorded as the reference for the student. If the student wants to watch the consultation session, he can replay the video.

References

[1] KarenChurch , What's up with WhatsApp? Comparing Mobile Instant Messaging Behaviors with Traditional SMS, AUGUST 30th, 2013 – MUNICH, GERMANY.

[2] Sushant A. Patinge, Pravin D. Soni, A Survey on Instant Message and Location Sharing System for Android, International Journal of Application or Innovation in Engineering & Management (IJAIEM) , Volume 2, Issue 10, October 2013.

[3] Margaret Butler, "Android: Changing the Mobile Landscape", 2011 PERVASIVE Computing, Available: www.computer.org/pervasive, [Online; accessed 24-Sep-2013].

[4] Zhaohui Wang, Rahul Murmuria, Angelos Stavrou, "Implementing and Optimizing an Encryption File System on Android", 2012 13th IEEE International Conference on Mobile Data Management, pp. 52-62.

[5] Davis, F.D., Bagozzi, R.P. and Warshaw, P.R. (1989), "User acceptance of computer technology: comparison of two theoretical models", Management Science, Vol. 35 No. 8, p. 982-1003.

[6] Skog, B. (2002), "Mobiles and the Norwegian teen: identity, gender and class". In Katz J.E. and Aakhus, M. (eds.), Perpetual contact, Cambridge University Press, New York.

[7] Ramesh Shrestha, Yao Aihong, "Design of Secure Location and Message Sharing System for Android Platform",2012 IEEE International Conference on Computer Science and Automation Engineering, pp. 117-121.

[8] Jashandeep Singh , Mobile messaging through android phones: an empirical study to unveil the reasons behind the most preferred mobile messaging application used by college going students , 25 April 2014, Vol.2 (March/April 2014 issue). [9]

Lee, W.J., Kim, T.U. and Chung, J.-Y. (2002), “User acceptance of the mobile internet”, working paper, IT Management Research Center, Graduate School of Business, Sungkyunkwan University, Seoul.