REAL TIME CHAT APPLICATION BETWEEN MULTIPLE CLIENTS

A COURSE PROJECT REPORT

By

JOESAM DINESH C [RA2111003011485]

MOHAMED ZIYADH M[RA2111003011487]

SHRINIDHI S [RA2111003011488]

HARISHKUMAR S [RA2111003011491]

RITULPRIYADARSHINI[RA2111003011495]

Under the guidance of

Dr.J.D.Dorathi Jayaseeli

Associate Professor

Computing Technologies

In partial fulfilment for the Course

of

18CSS202J - COMPUTER NETWORKS

In CTECH

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chenpalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "**REAL TIME CHAT APPLICATION BETWEEN MULTIPLE CLIENTS"** is the bonafide work of JOESAM DINESH C [RA2111003011485], MOHAMED ZIYADH M [RA2111003011487], SHRINIDHI S [RA2111003011488], HARISHKUMAR S [RA2111003011491] and RITUL PRIYADARSHINI [RA2111003011495] who carried out the project work under my supervision.

SIGNATURE

Dr.J.D.Dorathi Jayaseeli

Associate Professor

Computing Technologies

SRM Institute of Science and Technology

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honourable Vice Chancellor Dr. C. MUTHAMIZHCHELVAN, for being the beacon in all our endeavours.

We would like to express my warmth of gratitude to our Registrar Dr. S. Ponnusamy, for his encouragement.

We express our profound gratitude to our Dean (College of Engineering and Technology) Dr. T.V.Gopal, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing Dr. Revathi Venkataraman, for imparting confidence to complete my course project.

We wish to express my sincere thanks to Course Audit Professor Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications and Course Coordinators for their constant encouragement and support.

We are highly thankful to our Course project Faculty Dr. J.D.Dorathi Jayaseeli Associate Professor, CTECH, for her assistance, timely suggestionand guidance throughout the duration of this course project.

We extend my gratitude to our HOD Dr. M.Pushpalatha, Professor and Head, Department of Computing Technologies and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1. ABSTRACT

The Real-Time Chat Application between Multiple Clients is a software system designedto facilitate communication among users in real-time. It allows multiple users to interact with each other using text messages, voice calls, or video calls over the internet. The system provides a user-friendly interface that enables users to send and receive messages and media files. The Real-Time Chat Application is built on a client-server architecture where multiple clients connect to a central server. The server acts as a mediator between the clients, enabling them to communicate with each other. The server also handles the storage and retrieval of messages and media files. The system uses modern web technologies such as HTML, CSS, and JavaScript on the front end and Node.js on the back end. Socket.io, a real-time communication library, is used to establish a real-time, bidirectional connection between the server and clients. The system also employs variousencryption techniques to secure the communication between the clients and the server. The Real-Time Chat Application offers features such as group chat, private messaging, user profiles, and media sharing. Users can create and join multiple chat rooms, invite other users to join them, and participate in group discussions. They can also send privatemessages to other users, view their profiles, and add them as friends. Overall, the Real- Time Chat Application between Multiple Clients is an efficient and secure solution for users to communicate and collaborate with each other in real-time.

2. INTRODUCTION

A real-time chat application is a software system that enables multiple users to communicate with each other via text messages in real-time. This type of application hasbecome increasingly popular in recent years due to the rise of online collaboration and the need for remote communication. In this section, we will discuss the components and features of a real-time chat application and the technologies used to build it.

Components of a real-time chat application

A real-time chat application typically consists of four main components:

1. Client-side interface: This is the part of the application that users interact with. It includes the graphical user interface (GUI), which allows users to send and receive messages, view chat history, and manage their contacts.

2. Server: This is the backend component that manages the communication between clients. It stores chat history, manages user authentication.

3. Database: A database is used to store user information, chat history, and other data related to the chat application.

4. Network: The network component allows clients to connect to the server and communicate with each other in real-time.

Features of a real-time chat application

A real-time chat application typically includes the following features:

1. Private messaging: Users can send private messages to one or more users, which arenot visible to others in the chat group.

2. Group chat: Users can create or join group chats, where multiple users can communicate with each other simultaneously.

3. Multimedia sharing: Users can share multimedia files, such as images, videos, and documents.

4. Emojis and stickers: Users can express their emotions and add more fun to the chat byusing emojis and stickers.

5. Read receipts: Users can see whether their messages have been read by other users ornot.

6. Notifications: Users can receive notifications when a new message is received, even ifthey are not actively using the application.

3. REQUIREMENTS

Requirement Analysis

Building a real-time chat application between multiple clients requires careful planning and consideration of various requirements. In this section, we will discuss some of the key requirements that must be taken into account when building a real-time chat application.

1. Real-time communication: The primary requirement for a real-time chat application is the ability to provide real-time communication between multiple clients.

2. Scalability: A real-time chat application must be able to handle large volumes of traffic, including a large number of users and messages. This requires the application to be designed to scale horizontally and vertically.

3. Security: The security of a real-time chat application is critical, as users may be sharing sensitive information or personal details..

4. User authentication: A real-time chat application must have a robust user authentication system to ensure that only authorized users can access the application.

5. Privacy: Users of a real-time chat application may have different privacy requirements. The application must provide features that allow users to control their privacy settings, such as the ability to block other users, hide their online status, and delete chat history.

6. User interface: The user interface of a real-time chat application must be easy to use and intuitive, with features such as message notifications, emoticons, and file sharing.

7. Compatibility: The real-time chat application must be compatible with different browsers and operating systems to ensure that users can access the application from any device.

8. Data persistence: A real-time chat application must be able to store data such as chat history, user information, and messages. This requires the application to have a reliable and scalable database system that can handle large volumes of data.

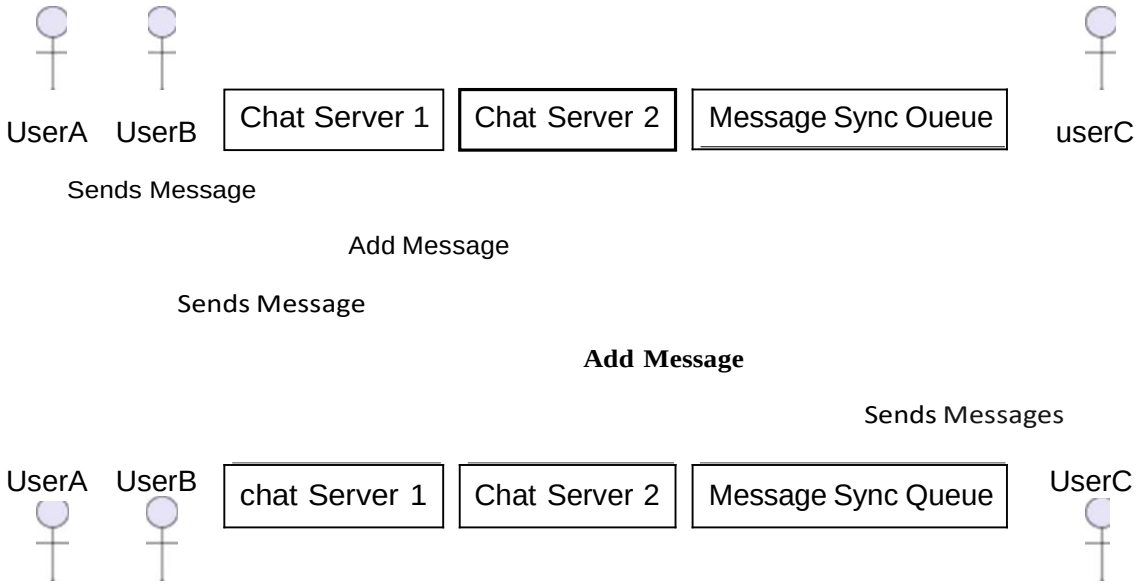
9. Monitoring and analytics: A real-time chat application must be monitored and analyzed to ensure that it is performing well and to identify any issues or bottlenecks. This requires the application to have a monitoring and analytics system that can provide real-time insights into the performance of the application.

4. ARCHITECTURE AND DESIGN

This is a sequence diagram that shows the timeline of events. It's important to understandthe flow before we start coding so make sure you understand the diagram completely.

Read through the points below if you are not familiar with Sequence Diagrams.

- Time flows downward.
- Arrows represent events. The start of an arrow denotes the emitter , the end of an arrow denotes the listener. For example, Player1 emits the Create Game Event andthe Socket Server listens to this event.
- A rectangular box denotes waiting/processing time.



This architecture provides a scalable and highly available infrastructure for a real-time chat application between multiple clients. By distributing traffic across multiple application servers and storing data in a scalable database, the application can handle largevolumes of traffic and provide a seamless communication experience for users.

5. IMPLEMENTATION

Implementing a real-time chat application between multiple clients can be a complextask, but there are several tools and technologies available to simplify the process. In this section, we will discuss some of the key steps involved in implementing a real- time chat application and the technologies that can be used.

1. Choose a programming language and framework: There are several programming languages and frameworks available to build real-time chat applications. Some popularchoices include Node.js with Socket.io, Ruby on Rails with ActionCable, and Django with Channels.

2. Set up a server: To implement a real-time chat application, you will need to set up aserver to handle incoming connections from clients. This server can be hosted on a cloud platform such as Amazon Web Services or Microsoft Azure, or on your own server.

3. Implement a database: A database is required to store user information, chat history,and other data related to the chat application. You can use a database management system such as MySQL and MongoDB.

4. Implement user authentication: User authentication is critical to ensure that only authorized users can access the chat application. You can implement user authentication using a third-party authentication provider such as Auth0 or by buildingyour own authentication system.

5. Implement real-time communication: The core functionality of a real-time chat application is real-time communication between clients. You can implement this usingWeb-Socket or HTTP long-polling, depending on your chosen programming language and framework.

6. Implement user interface: The user interface is what users will interact with, so it's important to create an intuitive and easy-to-use interface. You can use a front-end framework such as React, Vue, or Angular to build the user interface.

7. Implement privacy settings: To provide users with greater control over their privacy,you can implement features such as blocking other users, hiding online status, and deleting chat history.

8. Implement monitoring and analytics: Monitoring and analytics are critical to ensurethat the chat application is performing well and to identify any issues or bottlenecks. You can use a monitoring and analytics tool such as New Relic or Datadog to monitor the performance of your application.

In conclusion, implementing a real-time chat application between multiple clients requires careful planning and consideration of various factors. By choosing the right programming language and framework, implementing a database, user authentication,real-time communication, user interface, privacy settings, and monitoring and analytics, you can create a high-quality real-time chat application that meets the needsof your users and provides a seamless communication experience..

CODE

SERVER SIDE —

```
import socket
import threading
class Server:
  def  init  (self, host, port):
    self.host = host
    self.port = port
    self.server_socket = None
    self.client_sockets = []
    self.client_names = []

  def start(self):
    self.server socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.server_socket.bind((self.host, self.port))
    self.server socket.listen(5)
    print(fi'Server started on (self.host}:(self.port}")
    while True:
      client_socket, client_address = self.server_socket.accept()
      self.client_sockets.append(client_socket
      client_thread — threading.Thread(target=se1f.hand1e_client, args=(c1ient_socket,))
      client_thread.start()

  def handle_client(self, client_socket):
    client name = client socket.recv(l024).decode()
    self.clent names.append(client name)
    print(f'New client connected: (client name}")
    self.broadcast(I'{client_name} has joined the chat.")
    while True:
      try:
        message = client_socket.recv(1024).decode()
        if message:
          self.broadcast(f'{client_name}: {message}")
      except ConnectionResetError:
        self.c1ient_sockets.remove(c1ient_socket)
        self.c1ient_names.remove(client_name)
        self.broadcast(f"(client_name} has left the chat.")
        print(f'Client disconnected: {client_name}")
        break

  def broadcast(self, message):
    for client socket in self.client sockets:
      client_socket.send(message.encode())

if  name  —=' main
  server = Server('localhost', 12345)
  server.start()
```

CLIENT SIDE -

```
import socket
import threading

class Client:
    def     init    (self, host, port):
        self.host = host
        self.port — port
        self.client socket = None
        self.username = ""

    def connect(self):
        self.client socket = socket.socket(socket.AF INET, socket.SOCK_STREAM)
        self.client_socket.connect((self.host, self.port))

        self.username = input("Enter your username: ")
        self.client_socket.send(self.username.encode())

        receive thread = threading.Thread(target=self.receive_messages)
        receive thread.start()

        self.send_messages()

    def receive_messages(self):
        while True:
            try:
                message — self.client_socket.recv(1024).decode()
                print(message)
            except ConnectionResetError:
                print("Connection to the server was closed.")
                break

    def send_messages(self):
        while True:
            message = input()
            self.client_socket.send(message.encode())

            if message.lower() == "exit":
                self.c1ient_socket.close()
                break

if     name     == '    main
    client = Client('localhost', 12345)
    client.connect()
```

7. EXPERIMENT RESULT & OUTPUTS

SERVER OUTPUT —

```
Server started on localhost:12345
New client connected: Divya
New client connected: Aamuktha
Client disconnected: Divya
Client disconnected: Aamuktha
```

CLIENT 1 OUTPUT —

```
Enter your username: Divya
Divya has joined the chat.
Aamuktha  has joined the chat.
Hello
Divya : Hello
Aamuktha : hello
Aamuktha : cc mini project
realtime client server char application
Divya : realtime client server chat application
Aamuktha : okay
bye
Divya : bye
Aamuktha : bye
exit
Exctptior. ir. t..read Th:ead-I
```

CLIENT 2 OUTPUT —

```
Enter your username: Aamuktha
Aamuktha has joined the chat.
Divya : Hello
hello
Aamuktha : hello
cc mini project
Aamuktha : cc mini project
Divya : realtime client server chat application
okay
Aamuktha : okay
Divya : bye
bye
Aamuktha : bye
Divya has left the chat.
exit
Exception in tk:ead Th:tad'i
```

8. CONCLUSION & FUTURE ENHANCEMENT

In conclusion, a real-time chat application between multiple clients can be a complex yet rewarding project to undertake. By implementing the appropriate tools and technologies,you can create a highly scalable and reliable infrastructure that provides users with a seamless communication experience.

It is important to consider the requirements of the chat application, such as the number ofconcurrent users, the expected traffic, and the privacy and security needs. Choosing the right programming language and framework, implementing a database, user authentication, real-time communication, user interface, privacy settings, and monitoring and analytics are all critical steps in developing a high-quality real-time chat application.

Ultimately, a real-time chat application between multiple clients can be a valuable tool for businesses, social networks, or any platform that requires real-time communication between users. By creating a robust infrastructure and implementing the necessary features, you can provide a high-quality communication experience that meets the needs of your users.

9. REFERENCES

Here are some references related to real-time chat application between multipleclients:

1. "Building a Real-time Chat App with Node.js, Express.js, and Socket.io" by Maximilian Schwarzmüller- https://www.freecodecamp.org/news/how-to-bui1d-a-real-time-chat-app-with-node]s-expressls-and-socket-io/

2. "Real-time Chat Application using Django Channels" by James Loh - https://www.twi1io.com/blog/how-to-build-a-real-time-chat-application-with-dlango-channels

3. "Creating a Realtime Chat Application with Ruby on Rails" by Abhishek Verma - https://medium.com/swlh/creating-a-realtime-chat-application-with-ruby-on-rails-dcfe898a0b29

4. "Real-Time Chat Application with React and Firebase" by Raja Tamil - https://www.sitepoint.com/build-a-real-time-chat-application-with-react-and- firebase/

5. "Real-Time Chat Application with Socket.io" by Brad Traversy - https://www.voutube.com/watch?v=rxzO$P9YwmM

6. "Real-Time Chat Application with ASP.NET Core SignalR" by Anthony Chu - https://anthonychu.ca/post/real-time-chat-app-aspnet-core-signalr-angular/

These resources provide valuable insights and tutorials on how to implement a real-time chat application between multiple clients using various programming languages and frameworks.