DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' grade,

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-111

Mini Project Report

on

"ChitChat - WebRTC Video Calling App"

Submitted by

Xitiz Verma (1DS19CS193)

Yashaswini Shree (1DS19CS196)

Mohammed Shagil Khan (1DS20CS415)

Soumya Kumari (1DS19CS161)

Sixth Semester B.E (CSE)

Mini Project

19CS6DCMIP

Under the guidance of

Prof. Sahana MP
Assistant Professor
Dept. of CSE
DSCE, Bangalore

Department of Computer Science and Engineering

Dayananda Sagar College of Engineering

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

DAYANANDA SAGAR COLLEGE OF ENGINEERING

Shavige Malleshwara Hills, Kumaraswamy Layout, Bangalore - 560078

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled **ChitChat - WebRTC Video Calling App** is a bonafide work carried out by **Xitiz Verma (1DS19CS193), Yashaswini Shree (1DS19CS196), Mohammed Shagil Khan (1DS20CS415), Soumya Kumari (1DS19CS161)** in fulfilment of 6th semester, Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during the year 2022-23.

Prof. Sahana MP	**Dr.Ramesh Babu**	**Dr.CPS Prakash**
(Internal Guide)	Vice principal & Head	Principal,
Asst. Professor,	Department of CSE,	DSCE
Department of CSE,	DSCE	
DSCE		
Signature.....	Signature.....	Signature.....

Name of the Examiners:	Signature with date:
1.	.....
2.	.....

ACKNOWLEDGEMENT

We are pleased to have successfully completed the Mini project **"ChitChat - WebRTC Video Calling App"**. We thoroughly enjoyed the process of working on this projectand gained a lot of knowledge doing so.

We would like to take this opportunity to express our gratitude to **Dr. C P S Prakash**, Principal of DSCE, for permitting us to utilize all the necessary facilities of the institution.

We also thank our respected Vice Principal, HOD of Computer Science & Engineering, DSCE, Bangalore, **Dr. Ramesh Babu D R**, for his support and encouragement throughout the process.

We are immensely grateful to our respected and learned guide, **Prof. Sahana MP**, Asst. Professor CSE, DSCE for her valuable help and guidance. We are indebted to them for their invaluable guidance throughout the process and their useful inputs at all stages of the process.

We also thank all the faculty and support staff of Department of Computer Science, DSCE. Without their support over the years, this work would not have been possible.

Lastly, we would like to express our deep appreciation towards our classmates and our family for providing us with constant moral support and encouragement. They have stood by usin the most difficult of times.

Xitiz Verma (1DS19CS193)

Yashaswini Shree (1DS19CS196)

Mohammed Shagil Khan (1DS20CS415)

Soumya Kumari (1DS19CS161)

CONTENTS

ABSTRACT

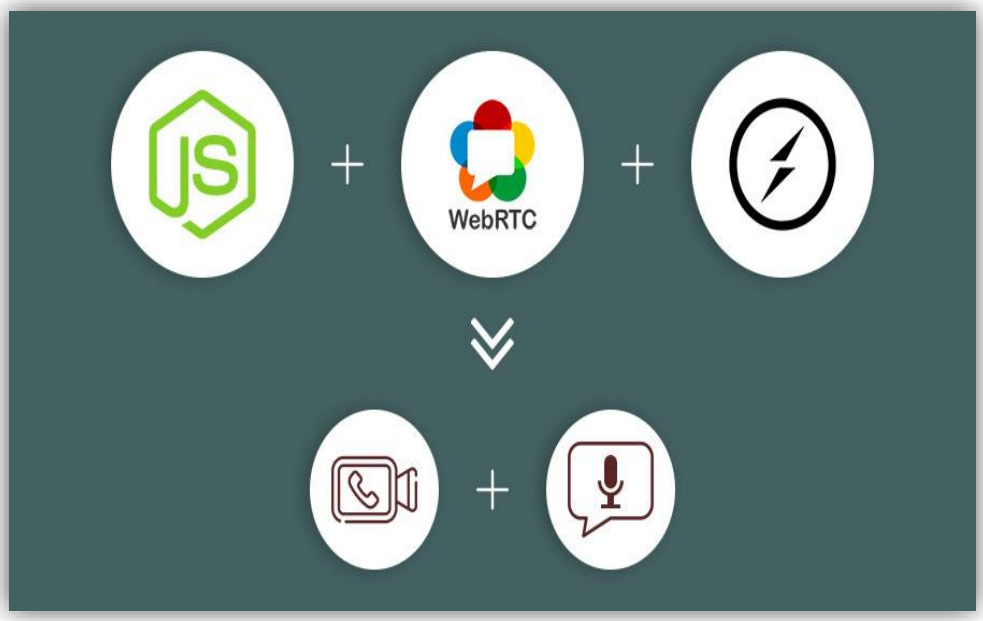
In this age of social distancing, Let's Chitchat.

WebRTC (Web Real-Time Communications) is an open source project which enables real-time communication of audio, video and data in Web and native apps. And we'll use Socket.IO and Node Js Development Services for setting up signaling server.

WebRTC is a new standard for enabling Real Time Communication (RTC) within a web browser. A web browser that has support for WebRTC includes the necessary technology to build a two-way video chat client directly in the browser without requiring the user to download any software.

The WebRTC project was initiated by Google and standardization is being performed both at W3C and the IETF. WebRTC is being rapidly adopted by numerous technology companies.

With the advent of WebRTC and the increasing capacity of browsers to handle peer-to-peer communications in real time, it's easier than ever to build real-time applications. In this article, we'll take a look at SimpleWeb RTC and how we can use the platform in implementing WebRTC technology. We'll also look at other alternatives that could help us achieve the same goal.



INTRODUCTION

To set up WebRTC signaling, the ICE framework requires you to provide two types of servers, detailed below.

1. STUN Server

The **STUN (Session Traversal Utilities for NAT)** server does exactly what I've just described above. It simply provides a meeting space for computers to exchange contact information. Once the information is exchanged, a connection is established between the peer computers and then the STUN server is left out of the rest of the conversation.

Here's an example script that runs on the client, which allows the browser to initiate connection through a STUN server. The script allows for multiple STUN server URLs to be provided in case one fails.

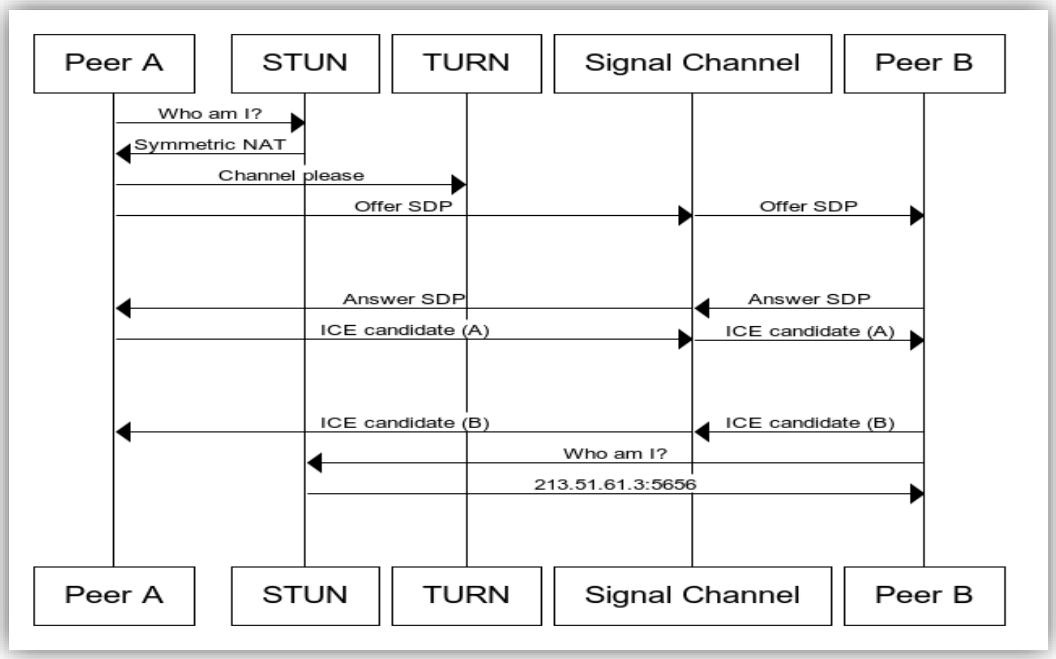
Connections established via STUN servers are the most ideal and cost-effective type of WebRTC communication. There's hardly any running cost incurred by the users. Unfortunately, the connection may fail to establish for some users due to the type of NAT device each peer is using. In such a situation, the ICE protocol requires you to provide a fallback, which is a different type of signaling server known as a **TURN** server.

2. TURN Server

A **TURN (Traversal Using Relay NAT)** server is an extension of the STUN server. Where it differs from its predecessor is that it handles the entire communication session.

Basically, after establishing a connection between the peers, it receives streams from one peer and relays it to the other, and vice versa. This type of communication is more expensive and the host has to pay for the processing and bandwidth load required to operate a TURN server.

Below is a graphical depiction of the entire signaling process involving first the STUN server and then the TURN server as fallback a complete Architectural Diagram showing the entire WebRTC.



LITERATURE SURVEY

- For the project, **ChitChat - WebRTC Video Calling App**, the following research papers were referred before the implementation process. These research papers are studied and tabulated as follows having the title, algorithm/technique used, as well as their performance and the dataset used in the implementation.

1. Peer to Peer Multimedia Real-Time Communication System based on WebRTC Technology
   Link: https://ieeexplore.ieee.org/abstract/document/7160422

2. WebRTC role in real-time communication and video conferencing
   Link: https://ieeexplore.ieee.org/abstract/document/9119656
3. WebRTC Security Architecture
   Link: https://ftp.ripe.net/rfc/v3test/testing_dl_v25.pdf
4. Using Socket.io Approach for Many-to-Many Bi-Directional Video Conferencing
   Link: https://csmj.mosuljournals.com/article_174411.html
5. Developing Video Chat Application with ReactJs And WebRTC
   Link: https://www.theseus.fi/handle/10024/500565

SYSTEM DESIGN & METHODOLOGY

Let us understand how our Video Calling App Works?
It helps to first understand the steps involved in a WebRTC video chat app from the perspective of the caller and callee.

Caller (Peer 1)

1. Start a webcam feed
2. Create an 'RTCPeerConnection` connection
3. Call **createOffer()** and write the offer to the database
4. Listen to the database for an answer
5. Share ICE candidates with other peer
6. Show remote video feed

Callee (Peer 2)

1. Start a webcam feed
2. Create an 'RTCPeerConnection` connection
3. Fetch database document with the offer.
4. Call **createAnswer()**, then write answer to database.
5. Share ICE candidates with other peer
6. Show remote video feed

Signaling

To connect two or more peers via WebRTC, each clientside app needs to provide an ICE (Internet Connectivity Establishment) server configuration. The apps must signal to each other how they should connect, which requires a backend server or database (like Firebase). The database allows the peers to relay the required information to establish a connection.

STEPS & PROCEDURES

Step by step procedure upon how we implemented our Chitchat App are -

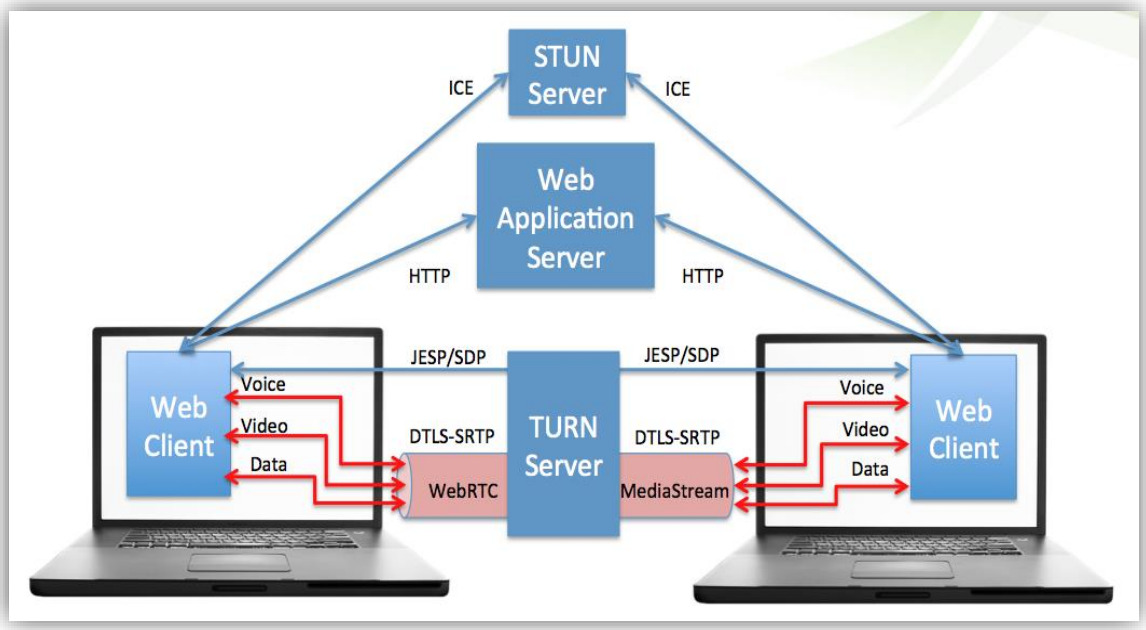
1. Display a **MediaStream** video of yourself on your computer
2. Display a **MediaStream** video of your friend on his computer
3. Create a **PeerConnection** on your computer
4. Create a **PeerConnection** on your friend's computer
5. Create an **Offer** on your computer
6. Add that **Offer** to the **PeerConnection** on your computer

7. Send that **Offer** to your friend's computer
8. Add that **Offer** to the **PeerConnection** on your friend's computer
9. Generate **ICE Candidates** on your computer
10. Send those **ICE Candidates** to your friend's computer
11. Add **ICE Candidates** to the **PeerConnection** on your friend's computer
12. Create an **Answer** on your friend's computer
13. Add that **Answer** to the **PeerConnection** on your friend's computer
14. Send that **Answer** to your computer
15. Add that **Answer** to the **PeerConnection** on your computer
16. Generate **ICE Candidates** on your friend's computer
17. Send those **ICE Candidates** to your computer
18. Add **ICE Candidates** to the **PeerConnection** on your computer
19. Display a **MediaStream** video of your friend on your computer
20. Display a **MediaStream** video of yourself on your friend's computer

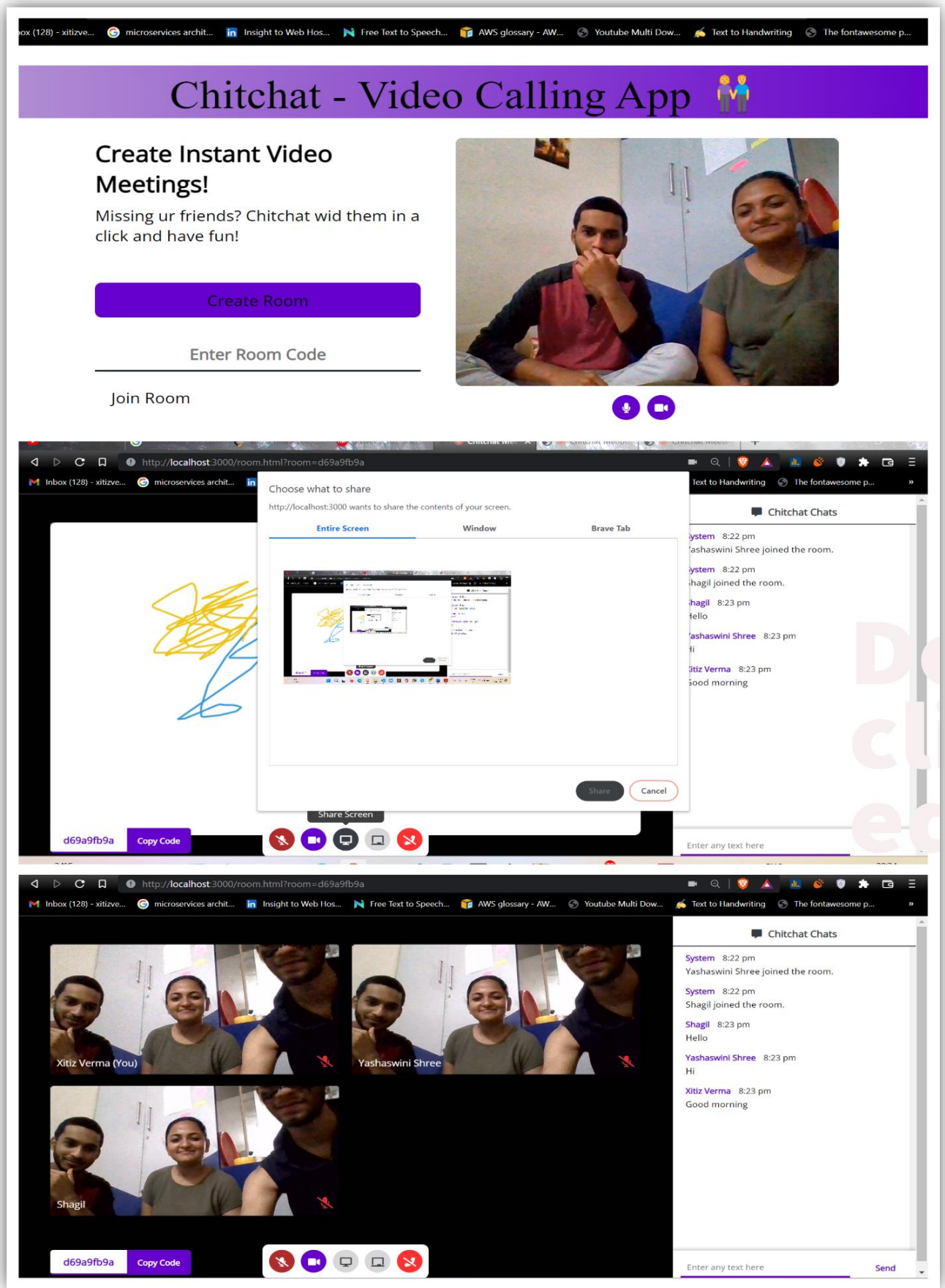
FEATURES & FUNCTIONALITIES

We successfully implemented the video calling App wherein more than two people can currently video call together using the web app. We also implemented:

1. Switch audio on/off
2. Switch video on/off
3. Create your own meeting url
4. Join an existing meeting with more than two people
5. Adding own name to profile with which you join the meeting
6. Update whenever somebody joins or leaves the meeting
7. Copy URL in-meeting
8. Chatbox
9. Screenshare
10. Collaborative whiteboard
11. Leave Meeting



UI SNAPSHOTS AND RESULTS



CONCLUSION & FUTURE ENHANCEMENTS

The project has a very vast scope in future. The project can be implemented on intranet in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner. The following are the future scope for the project.

- Deploy Chitchat over Heroku so that our fellow Community can use it.
- Individual Attendance system with photo using Student login
- To extend Support Components for even more versatile funtionality.

REFERENCES

1. Getting Started with WebRTC By Rob Manson
2. Developing Video Chat Application with ReactJs And WebRTC
3. Blog : https://fireship.io/lessons/webrtc-firebase-video-chat/
4. Youtube : https://youtu.be/_3exOT53faw
5. Google Scholar Articles

PROJECT SOURCE CODE LINK:

https://github.com/XitizVerma/Chitchat---Video-Calling-WebRTC-App

Thank You.