MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.)
A Govt. Aided UGC Autonomous & NAAC Accredited Institute, Affiliated to R.G.P.V, Bhopal

(DEPARTMENT OF INFORMATION TECHNOLOGY)

A Minor Project Report

On

"VIDEO CONFERENCING"

SUBMITTED TO:

Prof. Vikas Sejwar

Dr. Yogeshwar Singh

SUBMITTED BY:

Amrit Kaur (EC-20)

Aniket Sahu (IT-10)

UNDERTAKING

I declare that the work presented in this project titled "**VIDEO CONFERENCING**", submitted to the DEPT OF CSE AND IT, Faculty of Madhav institute of technology and science, Gwalior, M.P for the award of the **Bachelor of Technology** degree in **INFORMATION TECHNOLOGY**, is our original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

JULY – DEC 2021

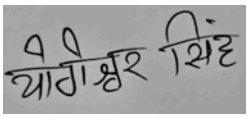
AMRIT KAUR & ANIKET SAHU

CERTIFICATE

Certified that the work contained in the project titled "**VIDEO CONFERENCING**", by **Amrit Kaur** and **Aniket Sahu** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Prof.Vikas Sejwar)

Mentor



(Prof. YOGESHWAR SINGH)

Mentor

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility and played crucial role in completing this project. And provides best guidance and keep me motivated and invest their full effort in helping us. A special gratitude I give to our minor project supervisor **Prof. Vikas Sejwar, Dr. Yogeshwar Singh**, whose contribution in stimulating suggestions and encouragement, helped us to coordinate in our project.

Aniket Sahu & Amrit Kaur

TABLE OF CONTENTS

1. INTRODUCTION
2. NEED OF WEBSITE
3. THEORY
    - 3.1. Why “WebRTC”
    - 3.2. What is WebRTC
    - 3.3. WebRTC APIs
4. WORKING
    - 4.1. Signaling
    - 4.2. NAT Traversal
    - 4.3. Firewall
5. SCREENSHOTS OF WEBSITE
6. REFERENCES

7.

Introduction

This Video conferencing website is aimed at developing a platform where real-time communication can be done between the browsers without requiring the user to download any software and without the use of any additional plugins.

To achieve this objective, we have used **WebRTC** — an open-source framework providing web browsers and mobile applications with real-time communication via simple APIs.

It enables peer-to-peer communication without any server in between and allows the exchange of audio, video, and data between the connected peers.

Need of website

Remote communication has become an indispensable part of our lives. It allows people from different locations to communicate with each other in real-time.

Video conferencing boosts productivity, saves time, reduces travel expenses, and overall promotes collaboration. The advantage of video conferencing is the ability to facilitate all of those benefits without requiring constant travel for face-to-face communication.

The ongoing COVID-19 situation has many more people working from home than ever before and companies are using web conferencing as their primary means of communication between employees and clients.

And with much of the world on lockdown with social distancing measures enforced, even spending time with family and friends is now largely restricted to online video calls. From parties to business meetings, these video conferencing platforms are now commonplace for virtually everyone.

Theory

▪ Why “WebRTC”

A question may arise, how WebRTC is better than any other online communication web applications?

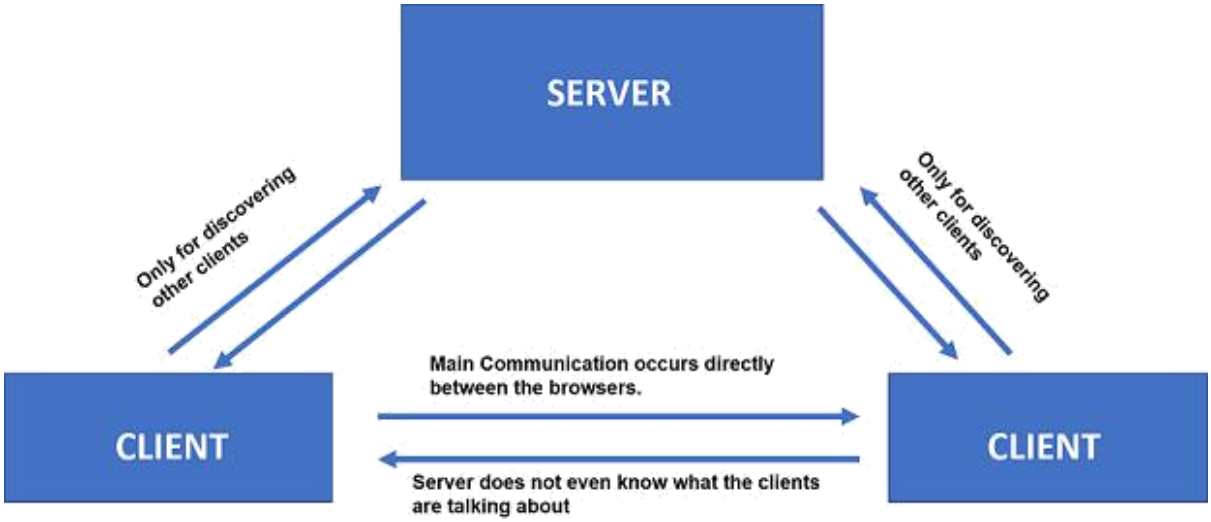
In a simple chatting web app, when two browsers need to send messages to each other, they typically need a server in between for coordination and passing the messages. But having a server in the middle results in a delay in communication between the browsers. This delay hardly affects the utility of the chatting app. Even if this delay is (say) 6 secs, we would still be able to use this chatting application.

However, in the case of a video conferencing application, this delay is significant. It will be extremely difficult to talk to someone using such an application.

Hence, for video conferencing, we require **Real-Time Communication** between the browsers. Such communication is possible if we eliminate the server from between. This is why we will have to use **WebRTC.**

▪ What is WebRTC

WebRTC stands for Web Real-Time Communication. It enables peer-to-peer communication without any server in between and allows the exchange of audio, video, and data between the connected peers. With WebRTC, the role of the server is limited to just helping the two peers discover each other and set up a direct connection.



The set of standards that comprise WebRTC makes it possible to share data and perform teleconferencing peer-to-peer, without requiring that the user install plug-ins or any other third-party software.

▪ WebRTC APIs

WebRTC consists of several interrelated APIs and protocols which work together to achieve Real Time Communication. The most important APIs are:

- getUserMedia(): capture audio and video.
- MediaRecorder: record audio and video.
- RTCPeerConnection: stream audio and video between users.
- RTCDataChannel: stream data between users.

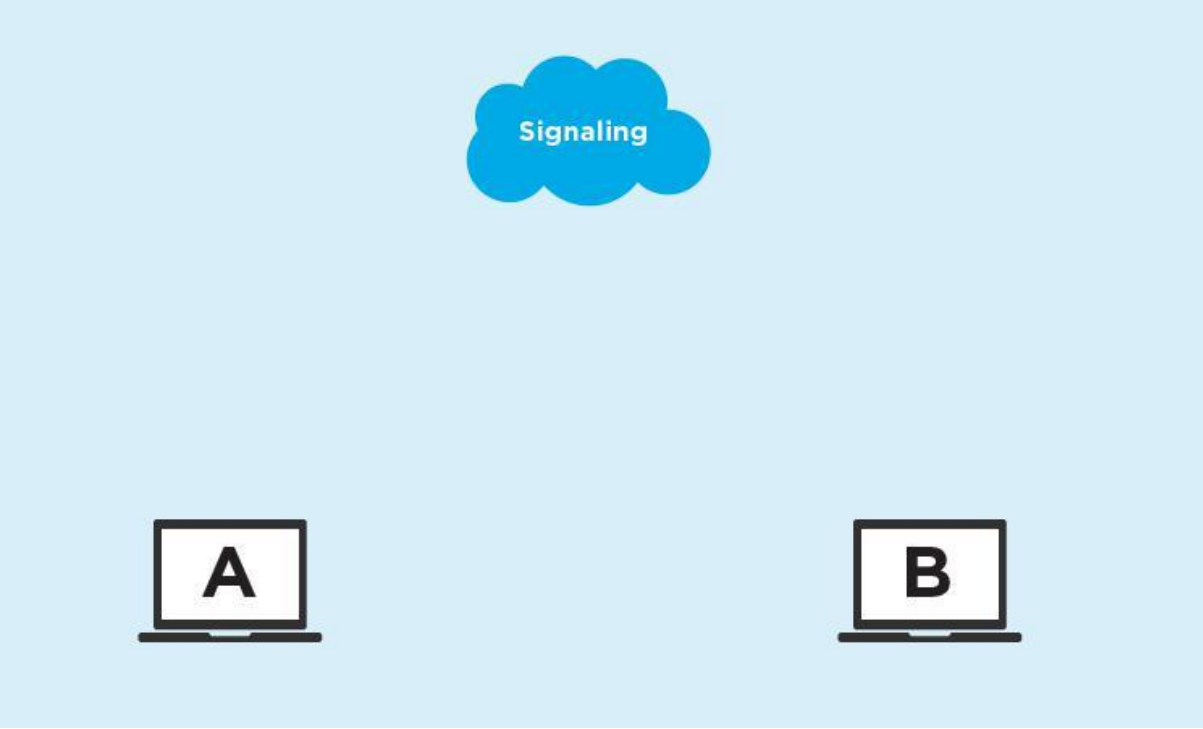
Working

- **Signaling**

Before a peer-to-peer video call can begin, a connection between the two clients needs to be established. This is accomplished through signaling.

Signaling allows two endpoints (senders, receivers, or both) to exchange metadata to coordinate communication in order to set up a call. For example, before two endpoints can start a video call, one side has to call the other, and the called side has to respond. This call-and-response message flow (also known as offer-answer message flow) contains critical details about the streaming that will take place - the number and types of streams, how the media will be encoded, etc. - and is often formatted using the Session

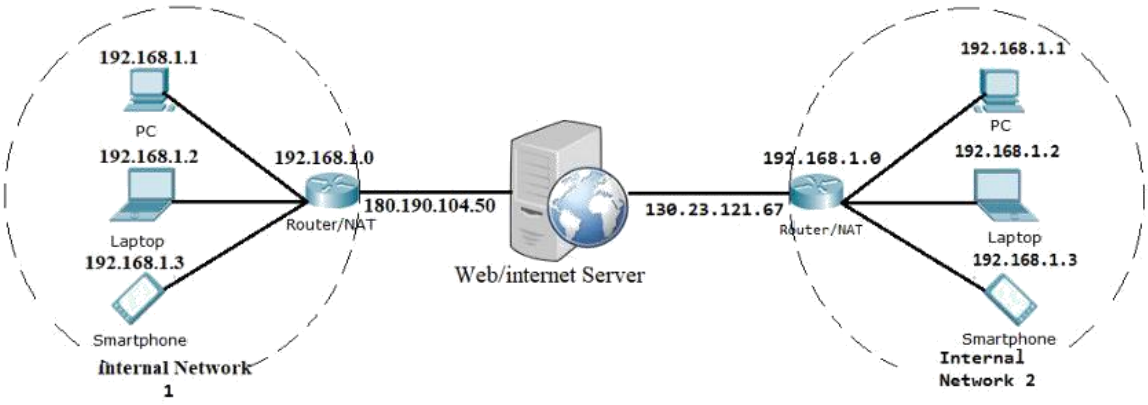
Description Protocol (SDP), a standard format used by many real-world systems, including VoIP and WebRTC.



- **NAT Traversal**

Once the initial signaling for a streaming connection has taken place, the two endpoints need to begin the process of NAT (Network Address Translation) traversal. When NAT assigns a public address to a computer inside a private network it can cause difficulties for setting up a real-time video connection. NAT Traversal is a method for getting around the issues associated with IP address translation.

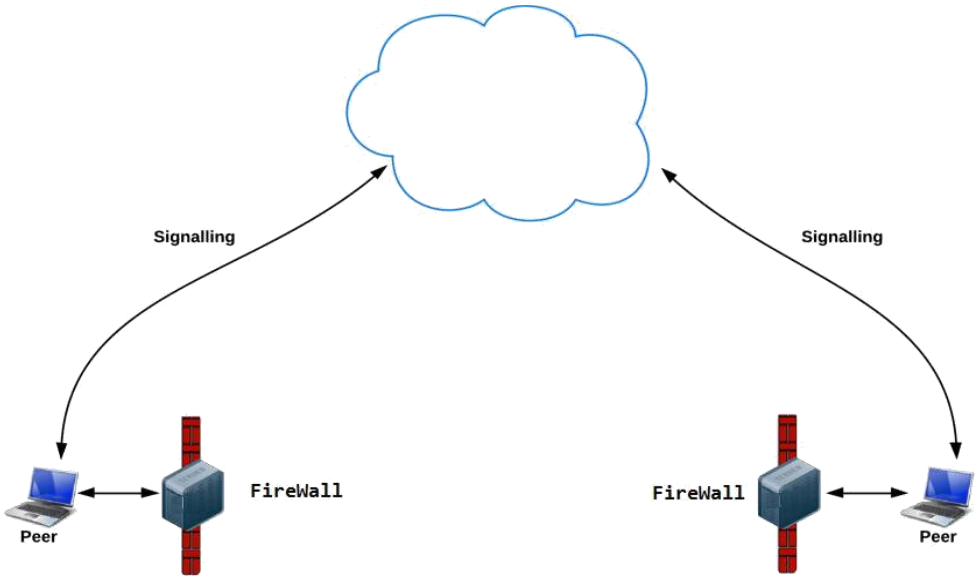
each device has two IP addresses — a Private IP address (assigned to the device) and the Public IP address (assigned to the router to which the device is connected to).



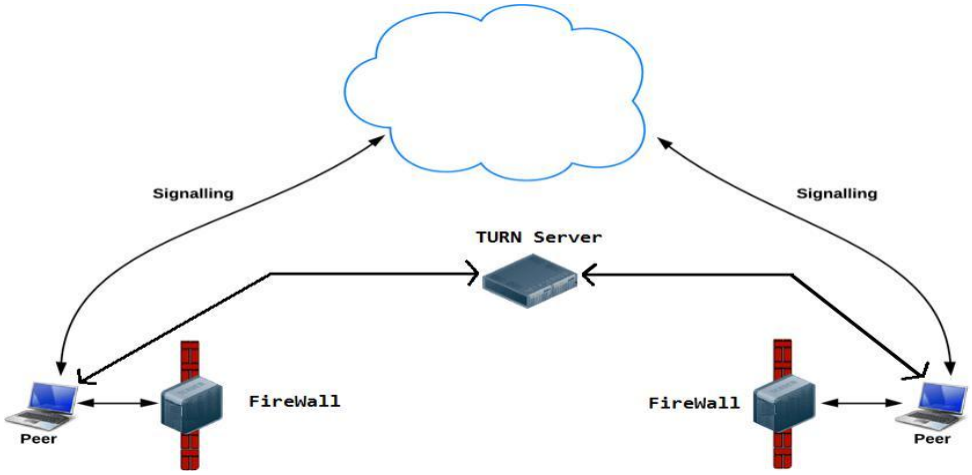
This can cause problem for WebRTC as the network ICE candidates (generated by the browser) contain the private IP address and not the public IP address of the device. Hence, we must find a way for the browser to know the Public IP Address so that it can create candidates containing the Public IP Address. The solution is a STUN (Session Traversal Utilities for NAT) server. When a device makes a request to the STUN server, the STUN server responds back with a message containing the public IP of the router to which the device is connected to. In this way, the STUN server helps the browser to generate candidates.

Firewall

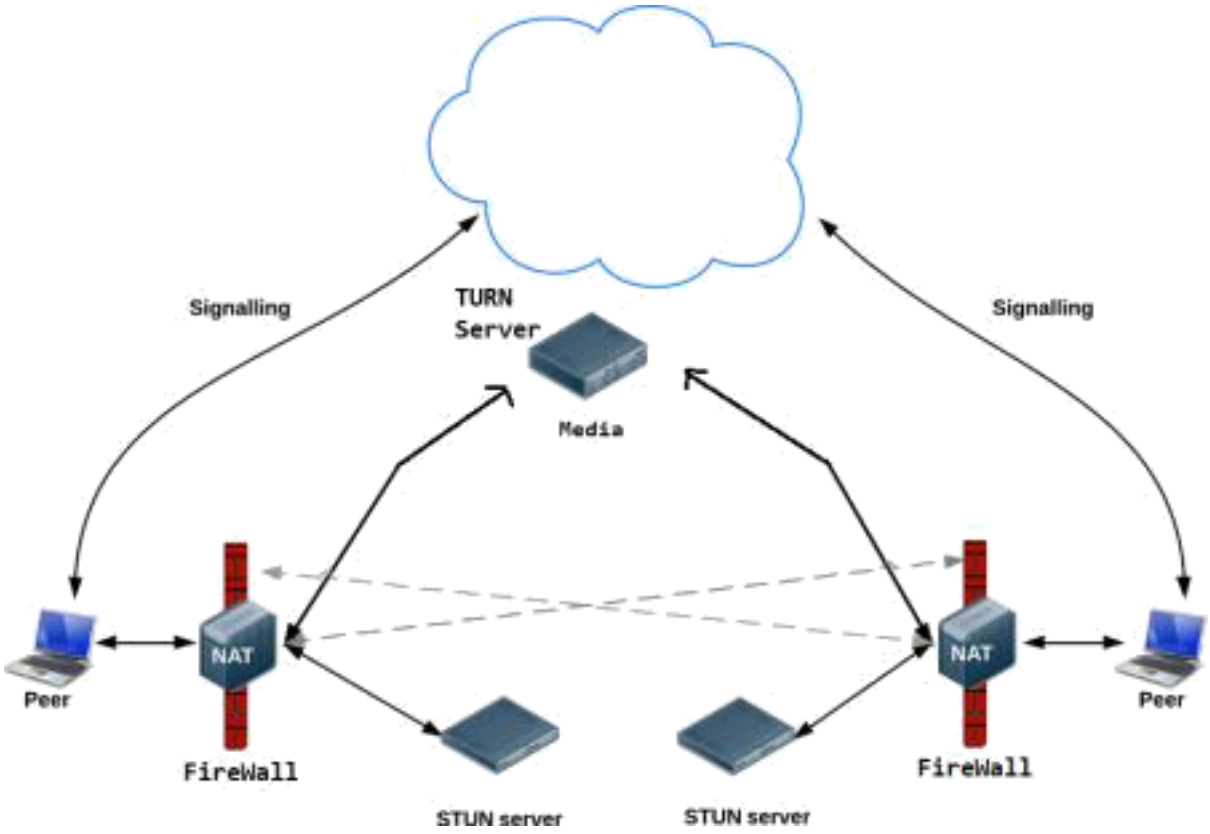
In reality, most devices live behind one or more layers of firewalls which are like antivirus softwares that blocks certain ports and protocols. A firewall and a NAT may in fact be implemented by the same device, such as a home WIFI router. Since WebRTC uses a number of non-standard Ports, some Firewalls do not allow a direct connection to be made between the two browsers.



Hence, to solve this, we need a TURN (Traversal Using Relay NAT) server. TURN server basically acts a Relay Server i.e. the relay traffic directly between the two peers if direct (peer-to-peer) connection fails. Following image illustrates:-



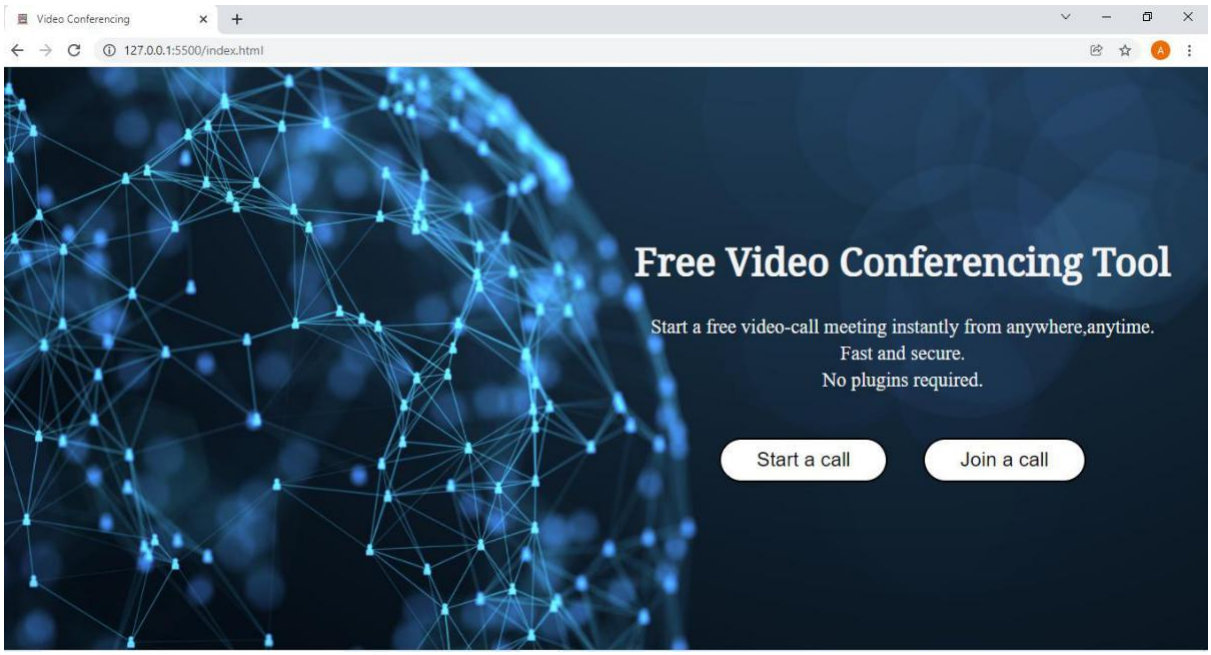
Following diagram illustrates all the connections made during a WebRTC call



- Most of the times, a successful connection will be made using a STUN server only and without the need of a TURN server. Only for a few times, you will require a TURN server for a successful call.

Screenshots of website

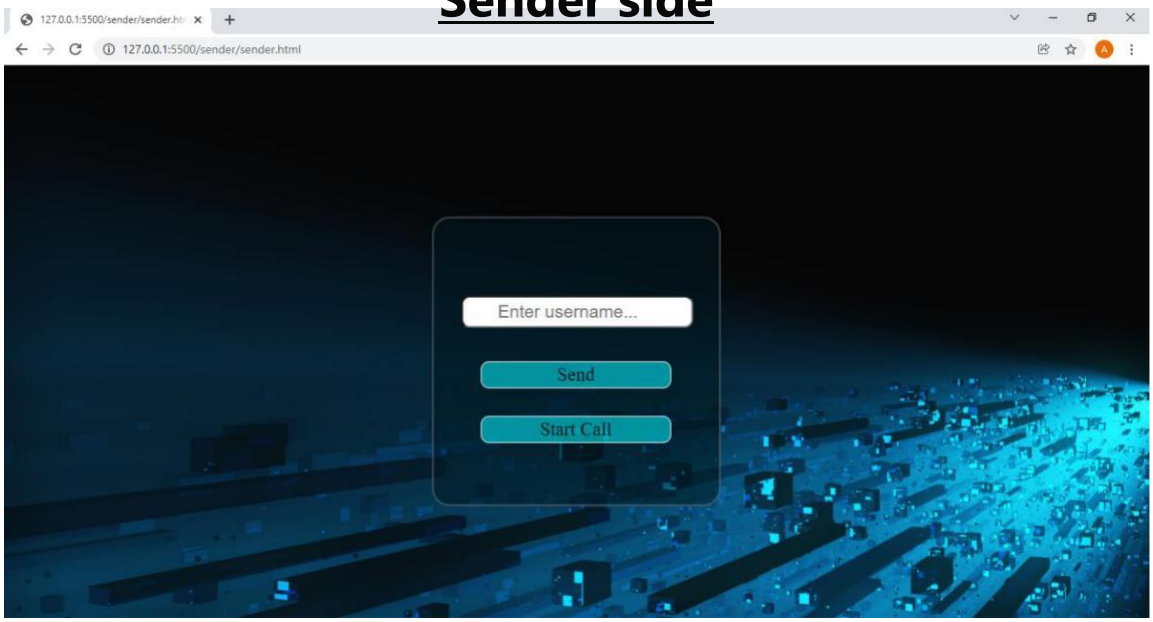
Home page



Above is the home page of our website. The user can start a call or join a call. This website can be directly made available online thus the user doesn't have to install any third-party software.

If the user clicks on send call then the below page will open.

Sender side



To start a call user just have to enter its username ,say Raj.

On entering the username , the user has to click on the send button.

After hitting send button the website will send an offer to the server which indicates that user named raj wants to start the call and it will send the network information of the caller.

Now the sever has the information of the caller side.

When the caller hits the start a call button , firstly the website will ask permission to access the microphone and the webcam.

After giving permission the video stream of the caller will be displayed on the top-left corner of site

Now the caller will wait for the callee to join the call.

Now on the receiver side.

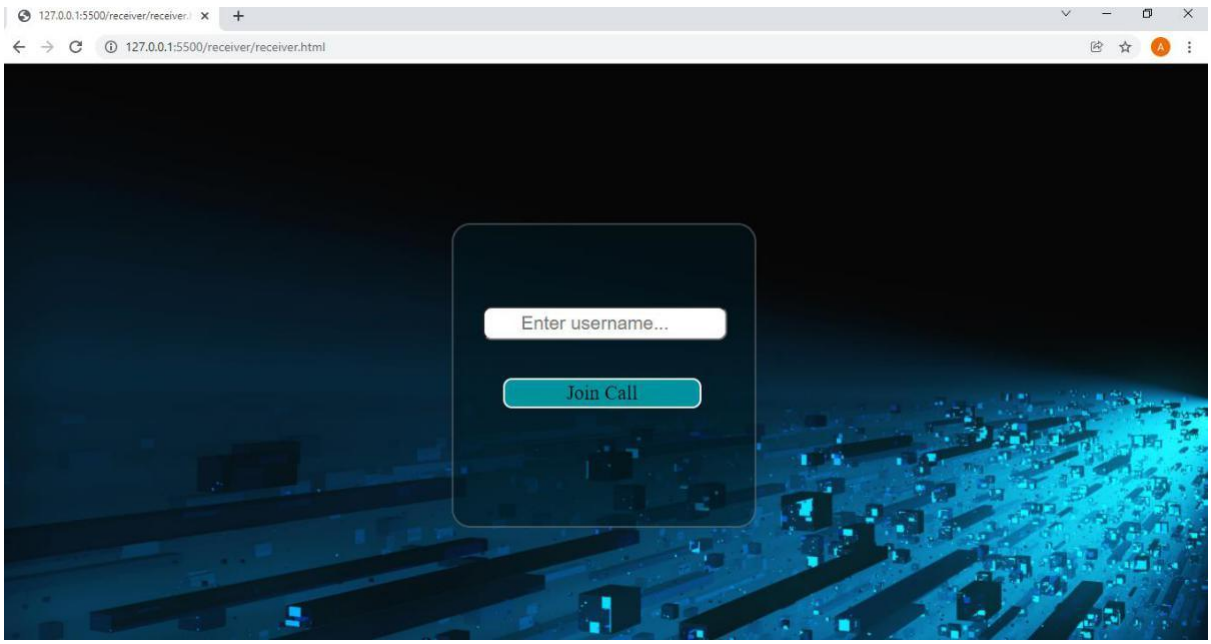
The person who wants to join the call has to enter the username of the caller. In this case, caller has to write ‘Raj’. When it hits join call button then the server will receive the network information of the receiver also.

In this way server will send information of sender to receiver and information of receiver to sender.

Thus now there is a peer-to-peer connection between sender and receiver.

Now there is no need of the server for the peers to communicate and the two peers can communicate directly.

Receiver side



References

https://webrtc.org/

https://www.liveswitch.io/

https://developer.mozilla.org/

https://dev.to/

https://simplecoding.dev/

https://www.canva.com/en_in/