ABSTRACT

The Covid-19 pandemic has made people to move onto using the live video sharing platforms. So there is huge surge in number of people using the video chatting applications and websites in order to stay connected with the people of their respective institutions or working communities. The difficulties people facing in initiating a video call and using the websites lies in the complexities and number of options that particular website provides. This is my small attempt to make video calling easy and user friendly with minimal options and no authentication. This website has been created in point o view of a person with zero knowledge of how to make a video call, all he needs is basic knowledge of operating a computer. Having a video chat is no more a night mare for a common man. Now a website named ChatVideo which is not loaded with options and very simple to use is available for everyone for free at https://chat-video-008.netlify.app.

TABLE OF CONTENTS

1. Introduction (3-5)
2. Existing Method (6-7)
3. Proposed Method (8-9)
4. Features (10-12)
5. Implementations (13-27)
6. Conclusion (28)
7. Libraries (29)

INTRODUCTION

What is Online Chat? Online Chat is communicating, whether by text, sound or video, with people over the Internet. With more and more people accessing the internet on their phones or smart devices, a number of services have become available that allow for conversations to take place between anywhere in the world, without paying hefty international call rates. These services still use the internet, so are not truly ‘free’, but are often far cheaper, faster and better than traditional alternatives like mail and long distance calls. Communication comes in many forms. • Text-based messaging refers to sending simple, short messages similar to that of a SMS. New services allow us to do this without paying the cost of an SMS message, and even organize large groups to talk together. • Voice Over Internet Protocol (VOIP) calls refer to voice conversations that take place over the internet, rather than a regular phone call. These are often far cheaper than regular calls, and are most commonly used to stay in contact with friends and family overseas. • Video calls use cameras and microphones to allow people to have face to face conversations over the internet. With cameras coming built-in to devices such as smartphones and tablets, and the rise of free, easy-to-use services such as Skype and FaceTime, this is becoming an increasingly common way to communicate. This class will explore some of the services that can be used to chat online. These include Skype, Messenger, FaceTime, Google Hangouts and Viber. All the services have their own advantages and disadvantages, with some being more suited to the devices you own.

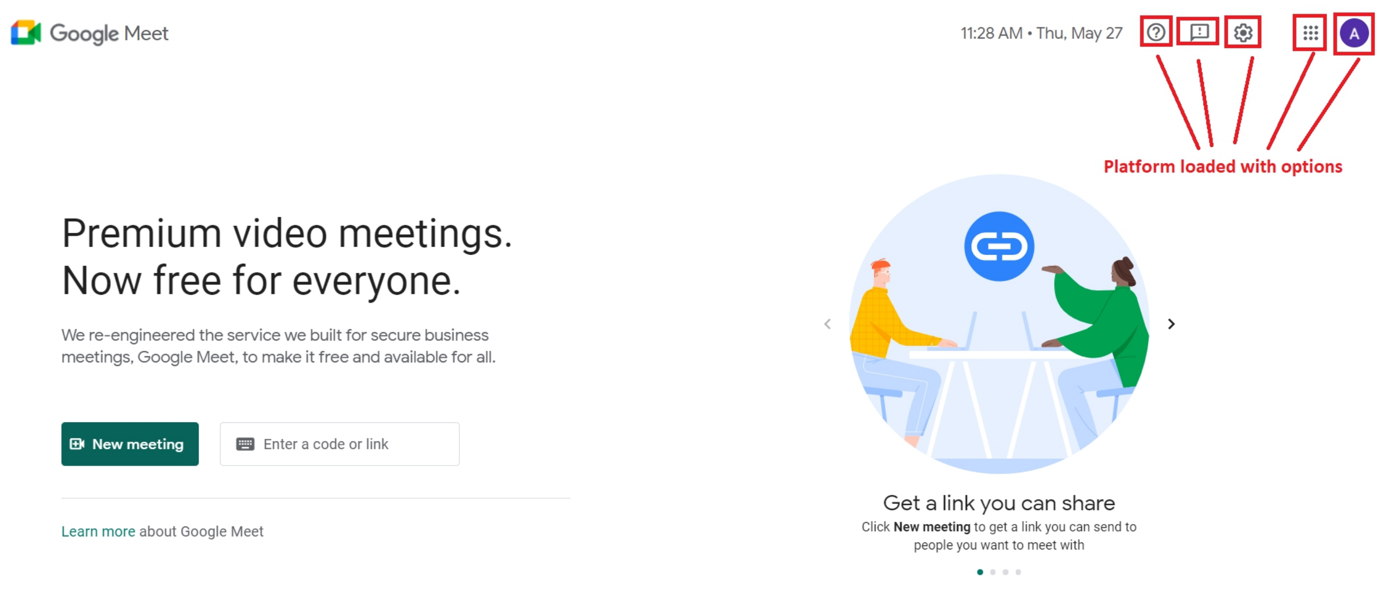
What do I need? To get started, there are a few things you'll need; • An internet connection. o For simple text messaging, a basic internet connection with a small data allowance is all that is needed. o For audio and video calls, a larger data allowance is highly recommended for frequent users. The library offers free Wi-Fi access for members to use with their wireless-capable devices, such as phones, laptops and tablets. • A device to access the internet. o For simple text messaging, this can be anything ranging from laptops, desktop computers, smartphones to tablets. o For voice calls, this device must also have a microphone. On laptops and computers, the built-in microphone, or microphone input, can usually be identified by the small symbol next to it. Smartphones will always have a microphone, as do the majority of tablets. o For video calls, the device must also have a camera. On laptops, this is usually located at the top, just above the screen. Not all laptops have built in cameras. Phones & Tablets will require a camera on the front, aimed at the user.

What can I do? Chat using Video or Voice Chat using Video or Voice Video Chat services allow people to have face-to-face conversations with people over the internet. Not only do you get to see the person you are talking to, but often it is more efficient than talking by text or email. People often use these services to catch up with friends or family overseas or interstate. They are typically much cheaper than long distance phone calls, using only the data in your existing internet plan. Chat using Text Chat using Text Online Messaging services allow people to communicate with short, instant messages over the internet. These services use only data: offering a far cheaper solution than SMS. Over the years, online messaging has grown, expanding beyond simple one-to-one text conversation. You can now send images and audio, providing a quick way to share moments with friends that is again much cheaper than MMS. You can set up group conversations: allowing you to have an instant conversation with a small group of people, making the planning of events much easier. These services essentially combine traditional SMS messaging with social networking; fast, cheap, simple and fun.

EXISTING METHOD

As mentioned above the existing methods include google meet , skype etc. The problem with these platforms are it is not common man friendly. They are packed with a lot of options which eventually confuses or creates chaos for a non computer person handling the platform.

It is very difficult to find a website where a person can easily experience the delight of a video chat without authentication. There is no need of authentication if the caller and receiver has unique id which changes for every reload.


Google Meet

11:28 AM • Thu, May 27

Platform loaded with options

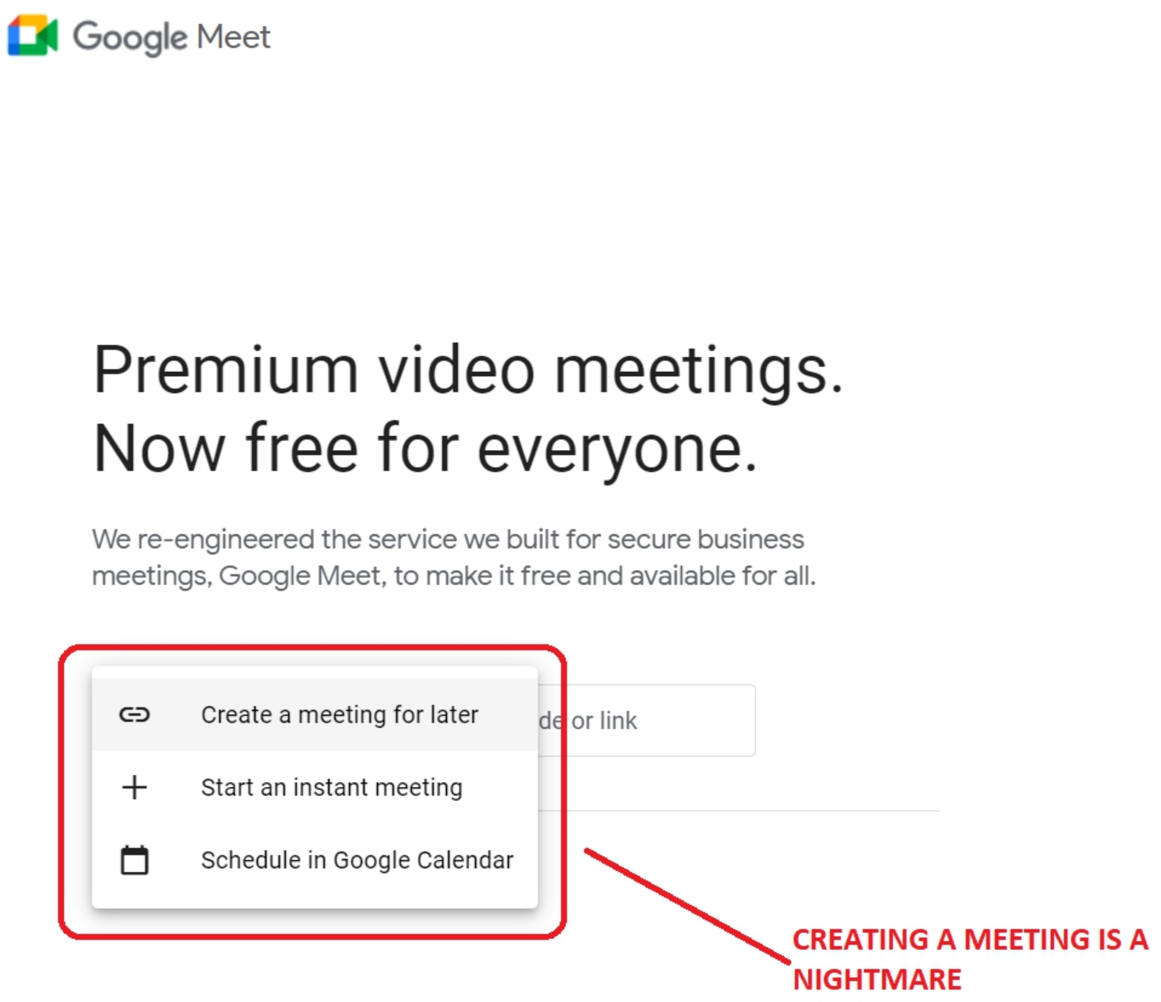
Premium video meetings.
Now free for everyone.

We re-engineered the service we built for secure business meetings, Google Meet, to make it free and available for all.

Learn more about Google Meet

Get a link you can share

Click **New meeting** to get a link you can send to people you want to meet with



Google Meet

Premium video meetings.
Now free for everyone.

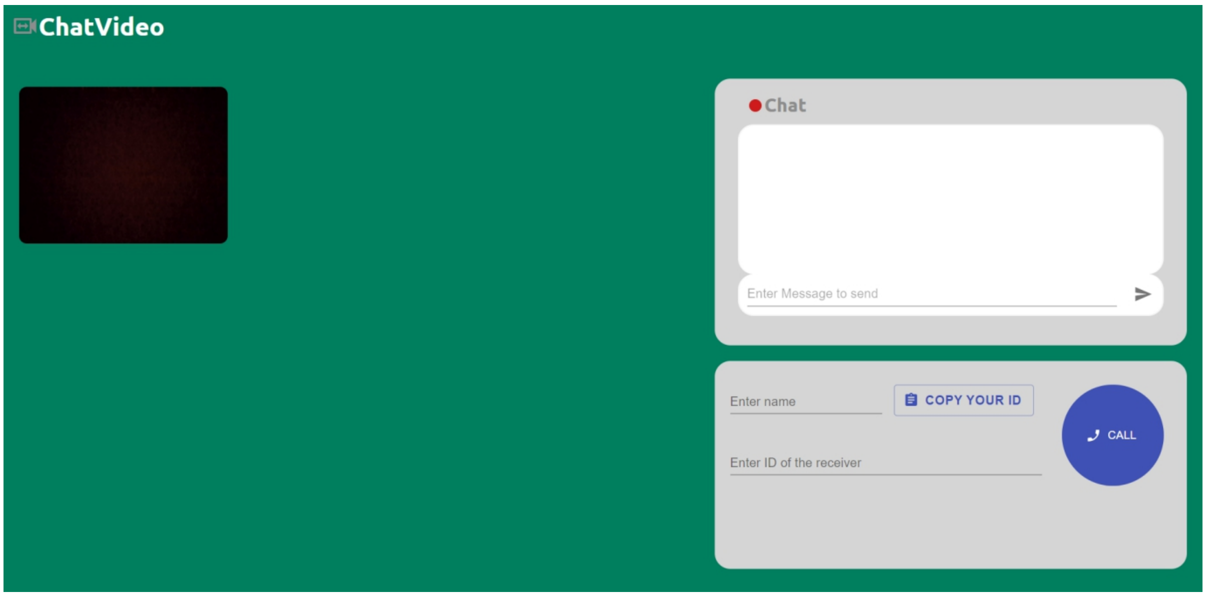
We re-engineered the service we built for secure business meetings, Google Meet, to make it free and available for all.

Create a meeting for later
Start an instant meeting
Schedule in Google Calendar
de or link
CREATING A MEETING IS A NIGHTMARE

Seeing all this a non computer guy will be in chaos. There is a need of a simpler platforms for simple one-one calls.

PROPOSED METHODS

ChatVideo :



ChatVideo is one such web-app built using React.js as its front-end framework and Node.js as its back-end framework. Here, a user can have a video chat and a text chat alongside. The web-app is built in POV of a non computer user such that he should not face any issues while initiating a call or answering a call.

The user interface is also very simple and entertaining with very basic graphical buttons created using Material-UI (a framework for React). As you can see all the options are crystal clear.

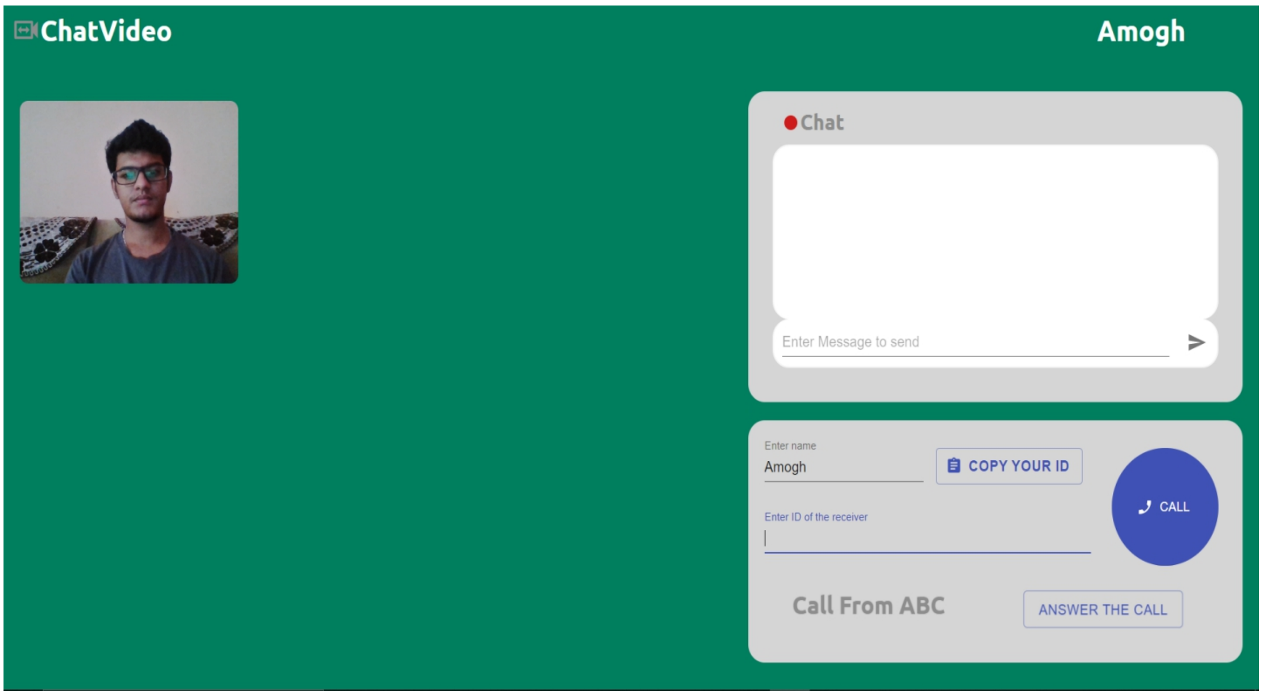
Websites like this can be very easily handled by anyone across the world who knows English.

Some of the main frameworks used for the following web-RTC video chat app are :

- Simple-peer
- react-copy-to-clipboard
- Socket.io
- Socket.io-client
- Material-ui/Core
- Material-ui/icons
- Express

FEATURES

Catchy UI:

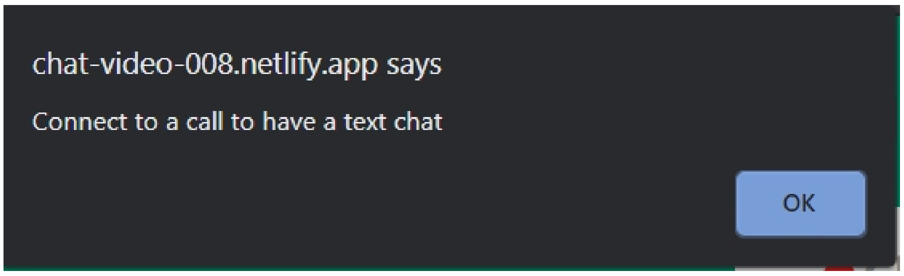


The call from a user is notified by a animation frame which pulsates the notification on the screen so that it is not missed by the receiver.

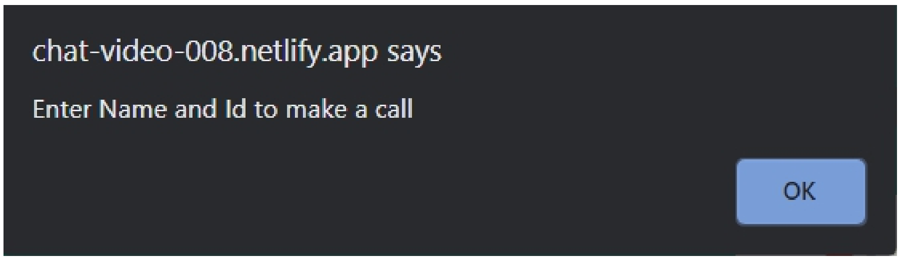
Even when a user wants to call the call button is clearly placed with the circular shape. All a user needs to connect to the other user is his ID. This ID could be obtained by just clicking on the ‘copy your id’ button on the right bottom segment of the web -app.

Alerts:

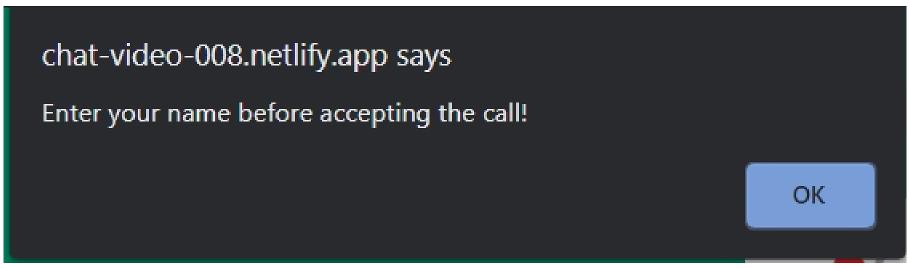
If a user tries to anything which is not in accordance with the working principles of the web-app a alert alerts the person what to do .


chat-video-008.netlify.app says

Connect to a call to have a text chat


chat-video-008.netlify.app says

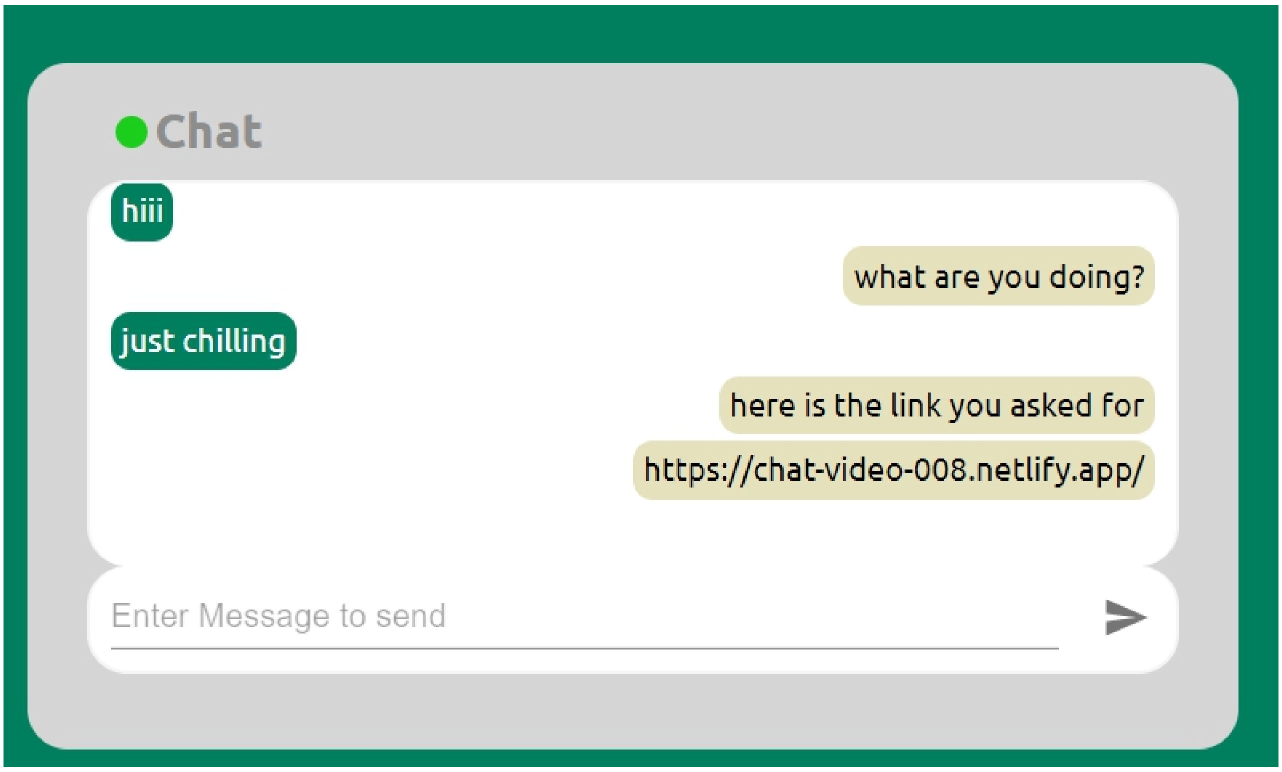
Enter Name and Id to make a call


chat-video-008.netlify.app says

Enter your name before accepting the call!

These alerts will indicate when what to do for a user.

Messages:


Chat

hiii

what are you doing?

just chilling

here is the link you asked for

https://chat-video-008.netlify.app/

The messages are equipped with scroll to bottom functionality. And different color for both the sender and receiver. The color of the chat online icon before the ‘chat text’ changes to green after the call is connected and will be red before the call is connected.

IMPLEMENTATION

Code:

- **index.html**

```
        <!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link  rel="icon" href="./video-camera (2).png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta


    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>ChatVideo</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>

  </body>
</html>
```

- index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import {ContextProvider} from "./SocketContext";

ReactDOM.render(
    <ContextProvider>
      <App />
    </ContextProvider>, document.getElementById('root')
);
```

- index.css

```css
*{
margin:0;
padding: 0;
}
@import url('https://fonts.googleapis.com/css2?family=Ubuntu:wght@300;400;500;700&display=swap');
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background-color:#007F5F;
  color:rgb(141, 141, 141);
  font-family: 'Ubuntu', sans-serif;

}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

- App.js

```javascript
import './App.css';
import React from "react";


import VideoPlayer from './components/VideoPlayer';
import TopBar from './components/TopBar';
import Messages from './components/Messages';
import CallOptions from './components/CallOptions';
function App() {


  return (

      <div className="App">
       <div className="">
        <TopBar/>
       </div>
       <div className="App__body">
          <div className="App_bodyVideo">
            <VideoPlayer/>
          </div>

 <div  className="sidebar">
            <div>
              <Messages className="messages"/>
            </div>
            <div>
```

```
              <CallOptions/>
            </div>
          </div>
       </div>
      </div>  );export default App;
```

- App.css

```
.App{
    display: flex;
    flex-direction: column;
    min-width: 1350px;
    height: 100vh;
    width: 100%;

    overflow: hidden;
}

.App__body{
    display: flex;

    height: 100%;
}
.sidebar{
    padding: 0px 20px;
    position: relative;
    width: 42vw;
    right: 0;
    top: 0;
    height: 700px;
}
.messages{

    position: relative;

}
```

- SocketContext.js

```
import React, { createContext, useRef, useState, useEffect } from "react";
import {io} from "socket.io-client"
import Peer from "simple-peer"

const SocketContext = createContext();
const socket = io("https://chat-video-app.herokuapp.com/");
const ContextProvider = ({children})=>{

    const callerVideo = useRef();
    const connectionRef = useRef();
    const [incommingCall,setIncomming]=useState(false);
    const [myId,setMyId] = useState('');
    const [stream,setStream]=useState();
    const [userName,setUserName]=useState('');
    const [callEnded,setCallEnded]=useState(true);
    const [call,setCall]=useState({});
    const [callAccepted,setCallAccepted]=useState(false);
    const Video=useRef();
    const [messages,setMessages]=useState([]);
    const [targetId,setTargetId]=useState("");

    useEffect(() => {
        navigator.mediaDevices.getUserMedia({ video:true , audio: true })
        .then((currentStream) => {
          setStream(currentStream);

          Video.current.srcObject = currentStream;

      })
      .catch((err)=>{
          console.log(err);
      });

       socket.on('successfulConnection',(id)=>{
        setMyId(socket.id);
       });

       socket.on('incommingCall',({from,signal,callerName})=>{

           setCall({from:from,signal:signal,name:callerName});

           setIncomming(true);
       });
       socket.on('incommingMessage',(msg)=>{
        setMessages((messages)=>{
            return (
                [...messages,msg]
            )
        })
    })
    socket.on('callEnded',()=>{
        alert('Call Ended');
        setCallEnded(true);
        connectionRef.current.destroy();
        window.location.reload();

    })

    },[]);
```

```
const answerCall =() =>{
    setCallAccepted(true);
    setIncomming(false);
    const peer =new Peer({initiator:false,trickle:false,stream:stream})
    peer.on('signal',(data)=>{
        socket.emit('callAnswered',({signal:data,to:call.from,name:userName}));
    });
    setTargetId(call.from);
    peer.on('stream',(callerStream)=>{
        callerVideo.current.srcObject=callerStream;
    });
    peer.signal(call.signal);
    connectionRef.current = peer;
};
const callUser = (id) =>{
    setTargetId(id);
    const peer =new Peer({initiator:true,trickle:false,stream:stream});
    peer.on('signal',(data)=>{
        socket.emit('callUser',({idToCall:id,signalData:data,from:myId,myName:userName}))
    });
    peer.on('stream',(userStream)=>{
        callerVideo.current.srcObject=userStream;
    });
    socket.on('callAccepted',(signal)=>{
        setCallAccepted(true);
        peer.signal(signal.signal);
        setCall({name:signal.name})
    });
    connectionRef.current = peer;
}
const sendMessage=(message)=>{
    const msgObj={
        id:myId,
        text:message,
        to:targetId
    }
    socket.emit('sendMessage',(msgObj));
}

const leaveCall=()=>{

   socket.emit('endCall',(targetId));

    setCallEnded(true);
    connectionRef.current.destroy();
    window.location.reload();

}

return (        <SocketContext.Provider value={{
            call,
            userName,
            callAccepted,
            Video,
            callEnded,
            setCallEnded,
            myId,
            incommingCall,
            callerVideo,
            setUserName,
            answerCall,
            callUser,
            sendMessage,
            setMessages,
            messages,
            leaveCall
            }
        }>
            {children}
        </SocketContext.Provider>
    )
}
export {ContextProvider,SocketContext}
```

- ## CallNotification.js

```
import React, { useContext } from 'react'
import {SocketContext} from "../SocketContext";
import { Button } from '@material-ui/core'
import "./callNotifi.css"
function CallNotification() {
    const {call,answerCall,userName} =  useContext (SocketContext);
    const handleAnswer = ()=>{
      if(userName===""){
        alert('Enter your name before accepting the call!');
      }else{
        answerCall();
      }
    }
    return (
        <div className="notification">
        <div className="name">
           <h2>Call From {call.name}</h2>
        </div>
        <div className="button">
          <Button variant="outlined" className="setW" color="primary" onClick={handleAnswer}>Answer the call</Button>
        </div>

        </div>
    )
}

export default CallNotification
```

- ## callNotifi.css

```
.notification{
    margin-left: 70px;
    margin-top: 30px;
    display: flex;
    animation-duration: 1s;
    animation-name: scale;
    animation-iteration-count: infinite;
}
@keyframes scale {
    0% {transform:scale(1);}
    50% {transform:scale(1.2);}
    100% {transform:scale(1);}
}

.name{
    width:100%;
}
```

- CallOptions.js

```
import React, { useContext, useState } from 'react'
import { Button, TextField } from '@material-ui/core';
import { Assignment, PhoneDisabled, PhoneEnabled } from '@material-ui/icons';
import {SocketContext} from "../SocketContext"
import "./callOptions.css"
import {CopyToClipboard} from "react-copy-to-clipboard";
import CallNotification from './CallNotification';
function CallOptions() {
    const {setUserName,userName,myId,callAccepted,callEnded,callUser,incommingCall,leaveCall} = useContext(SocketConte
xt);
    const [idToCall,setIdToCall]=useState("");
    const handleClick=(id)=>{
        if(userName==="" && id===""){
            alert("Enter Name and Id to make a call")
        }
        else if(userName===""){
             alert("Enter name to make a call")
        }else if(id==="")
        {
            alert("Enter Id to make a call")
        }else{
            callUser(id);
            setIdToCall("")
        }

    }

    const handleEndCall=()=>{
      leaveCall();
    }
    return (
        <div className="callOptions">
            <div className="default">
             <div className="left">
             <div className="nameAndId">
             <div className="textField">
             <TextField onChange={(e)=> setUserName(e.target.value)} label="Enter name" className="topBar__nameField"
value={userName}  />
             </div>
             <div className="copyBoard">
             <CopyToClipboard text={myId} >

             <Button className="copyButton" variant="outlined" color="primary"  startIcon={<Assignment fontSize="large
" />}>
                 <h3  className="copyText" > Copy Your ID</h3>
                </Button>


              </CopyToClipboard>
              </div>
             </div>
              <div className="toCallInfo">
             <TextField onChange={(e)=> setIdToCall(e.target.value)} label="Enter ID of the receiver" fullWidth classN
ame="topBar__nameField" value={idToCall} />

             </div>
              </div>
```

```
               <div className="bigButton">
               {!callAccepted && callEnded && (<Button  variant="contained" className="button" color="primary" onClick
={()=>handleClick(idToCall)} startIcon={<PhoneEnabled fontSize="large"/>} >Call</Button>)}
               {callAccepted && callEnded && (<Button  variant="contained" className="button" color="secondary" onClic
k={handleEndCall} startIcon={<PhoneDisabled fontSize="large"/>} >Hang up</Button>)}
               </div>

        </div>
            {incommingCall && !callAccepted && (<CallNotification/>)}

        </div>

    )
}
export default CallOptions
```

- callOptions.css

```
.callOptions{

    padding: 10px;
    background-color: #D5D5D5;
    height: 30vh;
    padding: 20px;
    border-radius: 20px;
    margin-top: 20px;
    min-width: 500px;

}
.default{
    display: flex;
    justify-content: space-between;

}
.letf{

   display: flex;
   flex-direction: row;

}
.textField{

   width: 200px;
   margin-right: 10px;
   margin-bottom: 10px;

}
.copyBoard{
    margin-top: 10px;
    padding: 0px;
    width: 100%;
    flex: 1;

}
.button{
    height: 100%;
    width: 100%;
   margin-bottom: 10px;
}
.bigButton{
    margin-top: 10px;
    margin-right: 10px;
    margin-left: 10px;
    width: 130px;
    height:130px;
    border-radius: 50%;
    overflow: hidden;


}

.toCallInfo{
    margin-top: 20px;
}
.nameAndId{
    display: flex;
    width: 100%;
    margin-right: 10px;
}
```

- Messages.js

```
import React, { useContext, useEffect, useState } from 'react'
import "./messges.css"
import FiberManualRecordIcon from '@material-ui/icons/FiberManualRecord';
import { IconButton, Input } from '@material-ui/core';
import { Send } from '@material-ui/icons';
import {SocketContext} from "../SocketContext"


function Messages() {
  const {sendMessage,setMessages,messages,myId,callAccepted}=useContext(SocketContext);
  const [message,setMessage]=useState("");
  const handleClick=()=>{
    if(!callAccepted){
      alert('Connect to a call to have a text chat')
    }else{
      if(message!==""){
        sendMessage(message);
      setMessages((prev)=>{
        return(
          [...prev,{id:myId,text:message}]
        )
      })
      setMessage('');




      }
    }

  }
```

```
  useEffect(()=>{
    var myDiv = document.getElementById("myDiv");
    myDiv.scrollTop = myDiv.scrollHeight;
  },[messages])
```

```
    return (
        <div onKeyDown={(e)=>{return(e.key==='Enter'? handleClick() : null)} }>
            <div className="messages">
              <div className="heading">

              {callAccepted?<FiberManualRecordIcon className="iconOnline"/>:<FiberManualRecordIcon className="iconOfli
ne"/>}
             <h2> Chat</h2>
```

```
              </div>
              <div className="chatLog">

              <div className="messageLog" id="myDiv">

                   {messages.map((msg,i)=>
                      <p key={i} className={msg.id===myId?"myMessage":"receivedMessage"}>{msg.text}</p>
                    )}

                </div>


                <div className="inp_Chat">
                   <div className="inputField">
                   <Input  onChange={(e)=>setMessage(e.target.value)} placeholder="Enter Message to send" fullWidth va
lue={message}/>
                   </div>
                   <div className="sendButton">
                     <IconButton onClick={handleClick}  >

                     <Send />
                     </IconButton>
                   </div>
                </div>
              </div>
            </div>
        </div>
    )
}
export default Messages;
```

- messages.css

```
.messages{

    background-color: #D5D5D5;
    height: 40vh;
    padding: 20px;
    border-radius: 20px;
    margin-top: 10px;
    display: flex;
    flex-direction: column;
    min-width: 500px;
    max-width: 605px;
}


.heading{
    display: flex;
    /* justify-content: center; */
    align-items: center;
    margin-left: 20px;
}
.heading .iconOnline{
    color:rgb(28, 206, 28);
}
.heading .iconOfline{
    color:rgb(206, 28, 28);
}
.heading h2{
    flex:1
}
```

```
.chatLog{
    padding: 10px;
    position: relative;
    /* border:2px solid black; */
    flex:1;
    border-radius: 20px;
    display: flex;
    flex-direction: column;

}
.messageLog{
    background-color:white;
    flex: 0.8;

    /* border:1px solid black; */
    border-radius: 20px;
    padding:10px;
    border: 2px solid whitesmoke;
    border-bottom: none;
    overflow: scroll;
    /* height: 7%; */
    max-height: 150px;

    padding-bottom: 30px;
}
.messageLog::-webkit-scrollbar {
    display: none;
  }

  /* Hide scrollbar for IE, Edge and Firefox */
  .messageLog {
    -ms-overflow-style: none;  /* IE and Edge */
    scrollbar-width: none;  /* Firefox */
  }

.inp_Chat{
    background-color: white;
    /* border:1px solid black; */
    flex:0.2;
    border-radius: 20px;
    display: flex;
    align-items: center;
    border: 2px solid whitesmoke;
    border-top: none;
}
.inputField{
    padding: 10px;
    flex: 1;
}


.myMessage{
    position: relative;
    background-color: rgb(230, 225, 189);
    width: fit-content;
```

```css
    padding: 5px;
    border-radius: 10px;
    color: black;
    /* width: 100%; */
    margin-left: auto;
    margin-bottom: 3px;
    max-width: 70%;
}
.receivedMessage{
    position: relative;
 background-color: #007F5F;
 padding: 5px;
 border-radius: 10px;
 color: white;
 width: fit-content;
 max-width: 70%;
 margin-bottom: 3px;
}
```

- TopBar.js

```javascript
import SwitchVideoOutlinedIcon from '@material-ui/icons/SwitchVideoOutlined';

import React, { useContext} from 'react'
import {SocketContext} from "../SocketContext"
import "./topBar.css";
function TopBar() {
    const {userName} = useContext(SocketContext);


    return (
        <div className="topBar">
                <div className="topBar__Icon">
                    <SwitchVideoOutlinedIcon fontSize="large"/>
                </div>
                <h1 className="brand">ChatVideo</h1>

                <h1 className="username">{userName}</h1>
        </div>
    )
}
export default TopBar
```

- topBar.css

```css
.topBar{
    height:10vh;
    background-color:#007F5F;
    width: 100%;
    min-height: 60px;
    display: flex;
    margin-bottom: 10px;
}

.brand{
    padding:10px 0px 10px;
    flex:1;
    color:white;
}
.topBar__Icon{
    padding:10px 0px 0px 10px;
}
.topBar__nameField{
    margin-right: 20px;
}
.topBar__button{
    padding: 10px;

}
.copyBoard{
    padding: 5px;
}
```

```
.username{
    padding: 10px 50px;
    color: white;
    height: 100%;
    margin-right: 40px;
}
```

- VideoPlayer.js

```
import React, { useContext } from 'react'
import { SocketContext } from '../SocketContext'
import "./videoPlayer.css"
function VideoPlayer() {
    const {Video,callerVideo,callAccepted,callEnded,call}=useContext(SocketContext);
    return (

        <div className="videoplayer">

           {callAccepted && callEnded &&
                (
                    <div className="videoplayer__callerVideo">
                    <video playsInline ref={callerVideo} autoPlay className="videoplayer__callerVideo__video">

                    </video>
                    <h3 className="text">{call.name? call.name : "caller"}</h3>
                    </div>
                )
            }

            <div className="videoplayer__myVideo">
                <video playsInline muted ref={Video} autoPlay className="videoplayer__myVideo__video">

                </video>

            </div>


        </div>
    )

}
export default VideoPlayer
```

- videoPlayer.css

```css
.videoplayer{
    position: relative;
    min-width: 700px;
    width: 58vw;
    height: 90vh;
    /* border: 2px solid white; */
    overflow: hidden;
    display: flex;


}
.videoplayer__myVideo{
    position: absolute;
    top:0;
    left: 0;
    width: 100%;

    /* overflow: hidden; */

    /* border-radius: 20px; */

}
.videoplayer__myVideo__video{
    /* padding: 30px; */
    margin-top: 20px;
    margin-left: 20px;

    border-radius: 10px;
    width: 30%;
    /* z-index: 1; */

}
.videoplayer__callerVideo{
    position:relative;
    /* place-items: center;
    align-items: center; */

}
.videoplayer__callerVideo__video{

    height: 90vh;
    width: 59vw;
    min-width: 700px;
    object-fit: contain;
    z-index: -1;
    border: 0px solid black;
    border-top-right-radius: 20px;
}
.text{
    padding: 5px 10px;
    position: absolute;
    top:20px;
    right: 40px;
    color:#D5D5D5 ;

    border-radius: 10px;
    /* z-index: 1; */
}
```

BACKEND:

- **index.js**

```
const app =require('express')();
const server = require('http').createServer(app);
const cors = require("cors");

const io = require("socket.io")(server,{
    cors:{
        origin:"*",
        methods:['GET','POST']
    }
});
app.use(cors());
const PORT= process.env.PORT || 5000;
app.get("/",(req,res)=>{
    res.status(200).send("Server running...");
});

io.on('connection',(socket)=>{

    socket.emit('successfulConnection',(socket.id));


    socket.on('callAnswered',(data)=>{

        io.to(data.to).emit('callAccepted',({signal:data.signal,name:data.name}));
    });
    socket.on('endCall',(id)=>{
        io.to(id).emit('callEnded');

    });
   socket.on('callUser',(data)=>{

        io.to(data.idToCall).emit('incommingCall',({from:data.from,signal:data.signalData,callerName:data.myName}));
    });
    socket.on('sendMessage',(msgObj)=>{
        const msg={
            id:msgObj.id,
            text:msgObj.text
        }
            io.to(msgObj.to).emit('incommingMessage',(msg));
    })
    socket.on('disconnect',()=>{


    });
});
server.listen(PORT,()=>{
    console.log(`Server is up and running on port ${PORT}`);
});
```

CONCLUSION

This project thus concludes the usage of a peer to peer (P2P) technology to exchange video stream data and a real time communication between two clients. It is clear from the above written codes that React.js has been used to build the web-app. The web-app is built dynamically. The website is code is divided into several parts. Every part of the web-app has its own functionality which is finally been put into a single page. Frameworks like simple-peer and socket.io have been helpful in establishing the connection between two clients. The Node.js with the express (server-end technology) have been helpful in building the server and implementing the required frameworks.

Finally the back-end was hosted on heroku and frontend on netlify after the build (using $npm run build). Link of the website is :

https://chat-video-008.netlify.app/

LIBRARIES USED AND THEIR VERSIONS

- **FRONTEND**

```
 "@material-ui/core": "^4.11.4",
"@material-ui/icons": "^4.11.2",
"@testing-library/jest-dom": "^5.12.0",
"@testing-library/react": "^11.2.6",
"@testing-library/user-event": "^12.8.3",
"react": "^17.0.2",
"react-copy-to-clipboard": "^5.0.3",
"react-dom": "^17.0.2",
"react-router-dom": "^5.2.0",
"react-scripts": "4.0.3",
"simple-peer": "^9.11.0",
"socket.io-client": "^4.1.1",
"web-vitals": "^1.1.2"
```

- **BACKEND**

```
 "cors": "^2.8.5",
"express": "^4.17.1",
"socket.io": "^4.1.1"
```