CODE

- **index.html**

```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

```javascript
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

```
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

```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;
```

```css
    /* 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;
    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

```js
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

```
.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

```
.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}`);
});
```

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"
```